Juridiska institutionen

Examensarbete vårterminen 2015
30 HP

# How liability for intellectual property defects in computer software transactions affect an efficient allocation of resources

A journey in the protection, transaction, infringement and uncertainty of copyright protected software

Johan Corell

Handledare: Kristoffer Schollin

Examinator: Claes Martinson

GÖTEBORGS UNIVERSITET
HANDELSHÖGSKOLAN

## Summary

Copyright has been used for over a hundred years to provide authors and artists with incentive to create and share their works, and the system of copyright is a flexible legal phenomenon that has adapted to new forms of creations as society has developed. When software emerged and was recognized as valuable to society, copyright was chosen as the mean for protecting the incentives for further innovation and development in this field. This thesis investigates, from a legal constructivist approach, how this form of protection creates uncertainty and how this uncertainty is transferred in a simplified value chain, from the software developer or licensor as seller to the end-user or licensee as buyer.

As seen in the thesis, developing modern software is quite different from (the illusion of) the sole author creating a work of art, which is the classic matter that copyright would be suitable for. Thus when applying principles of copyright as a legal phenomenon to protect interest behind software this creates uncertainty for the entities that act in an environment of software transactions. In legal constructivist terms I conclude that copyright protection of software is a phenomenon not reified to a satisfactory level in society.[1] This is combined with the uncertainty of *intellectual property defects*,[2] as shown in this thesis, another phenomenon not satisfactory reified in the Swedish or European jurisdictions. I have investigated how one can argue regarding these uncertainties with and without agreements regulating the matter.

When transacting software the buyer must commonly agree to an end-user license agreement that often stipulates whether the seller will be responsible or not for the properties of the software, for example, who should bear the risk if the software infringes a third party intellectual property right. However if the parties in a transaction of software have not regulated these matters in an agreement it is also uncertain to whom this risk is allocated in the transaction. When applying economic theory to these matters, I have found that there are transaction costs hindering successful bargaining between the parties and I argue that if the seller is liable for intellectual property defects, resources will be more efficiently allocated. Because of this, changes in the copyright legislation should be made to increase the possibility of an efficient allocation of resources.


---

[1] For an explanation of reification see the section on theory.

[2] My translation of ”immaterialrättsligt fel”.

## Abbreviations

| | |
|---|---|
| CA | The Swedish Copyright Act (SFS 1960:729) |
| CISG | United Nations Convention On Contracts For The International Sale Of Goods |
| COTS | Commercial off-the-shelf |
| DRM | Digital Rights Management |
| EULA | End-user license agreement |
| GPL | General public license |
| GUI | Graphical user interface |
| IP | Intellectual property |
| IPR | Intellectual property right |
| NJA | Nytt Juridiskt Arkiv, avdelning I |
| NPE | Non-practicing entity |
| OS | Operating system |
| OSS | Open-source software |
| WIPO | World Intellectual Property Organization, United Nations |

# Table of contents

# 1 Introduction

> *"Abraham Lincoln told a story about a lawyer who tried to establish that a calf had five legs by calling its tail a leg. But the calf had only four legs, Lincoln observed, because calling a tail a leg does not make it so.*
>
> *Before us is a case about a lawyer who tried to establish that a company owned a copyright by drafting a contract calling the company the copyright owner, even though the company lacked the rights associated with copyright ownership. Heeding Lincoln's wisdom, and the requirements of the Copyright Act, we conclude that merely calling someone a copyright owner does not make it so."*
>
> - Judge Clifton in *Righthaven LLC v. Hoehn*, 716 F. 3d 1166 - Court of Appeals, (9th Cir. 2013).

Even if Judge Clifton was referring to the specific case at hand I have lately thought about the same statement – *that merely calling someone a copyright owner does not make it so* - and found it to be applicable elsewhere. I have asked myself - What does it mean when someone is calling himself or herself an owner of copyright protected software? What consequences can arise if someone claims to have the exclusive right of making copies of a specific piece of software?

## 1.1 The dangers of intellectual property defects in software

For the reader to understand why the intellectual property defect[3] is a complex and interesting issue I will provide a couple of illustrations throughout the thesis when this issue have been present, starting in this chapter.

### 1.1.1 Illustration 1: *The case of Versata Software, Ameriprise and XimpelWare*

Versata Software Inc. ("Versata"), an American company focused on software development, had developed a software program named DCM – Distribution Channel Management software. They licensed the DCM software to companies in the financial sector, among others Ameriprise Financial Services ("Ameriprise"). Ameriprise, also an American based company, is active in the field of financial advice regarding financial

[3] My translation of "immaterialrättsligt fel".

planning, investments and insurance among others, and licensed the DCM software to use it to manage their independent financial advisors.[4] A dispute arose between the two companies concerning Ameriprise use of third party contractors to modify the DCM software. One of Ameriprise contractors had allegedly decompiled[5] the DCM software to make a competing product, something prohibited in the software license agreement between the parties. Ameriprise opposed the allegations why Versata filed suit against Ameriprise for breach of contract. During the discovery process in court Ameriprise received documents showing that parts of the source code used by Versata in the DCM software was originally open source code developed by a third party, a software company named XimpelWare Corporation ("XimpelWare"). XimpelWare licensed out their source code under the copyleft license GPL v.2, which requires the licensee to include the text of the GPL v.2 license, the required copyright notices and a copy of the source code towards their licensees.[6] The problem however was that Versata had not included the text of the GPL v.2 license, the required copyright notices and a copy of the source code of XimpleWare to their licensees, why Versata themselves were in breach of the GPL v.2 license. Due to this matter Versata did not have the rights to include the XimpleWare source code in its DCM software in the first place.[7]

Knowing these facts Ameriprise counterclaimed that the DCM software was in its whole licensed under the GPL v.2 terms why Ameriprise actions, including any decompilations, where consistent with the software license agreement, i.e. the GPL v.2. Under the GPL v.2 Versata were also obliged to make the DCM source code freely available to all users including Ameriprise and its contractors. In addition to this Ameriprise informed the

---

[4] For more information see, https://www.ameriprise.com/about-ameriprise-financial/.

[5] "Decompiling" a software program is similar to reverse engineering a technical solution, see Margaret Rouse. decompile, *WhatIs.com*. 2005. Available at: [http://whatis.techtarget.com/definition/decompile] Accessed 2015-07-17.

[6] For the GPL v.2, see Free Software Foundation. GPL v.2. *GNU Operating System*. 1991. Available at: [https://www.gnu.org/licenses/old-licenses/gpl-2.0.en.html] Accesses 2015-07-17.

[7] Aaron Williamsson. Lawsuit threatens to break new ground on the GPL and software licensing issues, *opensource.com*. 2014. Available at: [http://opensource.com/law/14/7/lawsuit-threatens-break-new-ground-gpl-and-software-licensing-issues] Accessed 2015-07-17; and Mark Radcliffe. GPLv2 goes to court: More decisions from the Versata tarpit, *opensource.com*. 2014. Available at: [http://opensource.com/law/14/12/gplv2-court-decisions-versata] Accessed 2015-07-17.

original right holder XimpelWare about Versatas breach of the GPL v.2. When XimpelWare received the information they decided to sue both Versata and Versatas customers, including Ameriprise, for patent infringement and copyright infringement since they had made a profit on XimpelWares source code. This was in 2013 and after several sessions in court the parties decided to settle the remaining claims that were still ongoing in February of 2015.[8]

In this first example, it becomes evident that there are large effects when one company has incorporated proprietary source code into their own commercial product without permission. The first illustration raises several questions, for example why did Versata copy a piece of open source code and paste it in their commercial software? How can one determine which parts of the software program that are protected by copyright? And an obvious issue for the costumers of Versata, who got sued by XimpelWare for copyright and patent infringement, could they claim the costs incurred by these lawsuits from Versata, i.e. what responsibility did the licensor have for intellectual property defects in the transacted software? To fully understand the width of these problems I will provide the reader with yet another example closer to the Swedish jurisdiction.

### 1.1.2 Illustration 2: *The case of Swish Payment System*

Swish® is an innovative solution that let its users pay each other in real time by the use of an application in their mobile phones.[9] One of the revolutionary things about their solution is that the user does not need to know the recipients bank account number, only their telephone number, making transactions faster and easier. Swish® was developed by HiQ for Sweden's six major banks (Danske Bank, Handelsbanken, Länsförsäkringar Bank, Nordea, SEB, Swedbank and Sparbankerna) and has 3.001.950 users.[10] In the fall of 2014 a software developer decompiled and analysed the source code of the Android version of the application Swish® and found that Swish® contained pieces of open source code developed by a third party and licensed under the GPL v.3. Due to this, the company Swish® is required by the GPL v.3 to make the source code freely available to

---

[8] Y. Peter Kang. XimpleWare, Versata Settle Insurance Software IP Dispute, *LAW360*. 2015. Available at: [http://www.law360.com/articles/620898/ximpleware-versata-settle-insurance-software-ip-dispute] Accessed 2015-07-17.

[9] For more information see Swish. Available at: [https://www.getswish.se/] Accessed 2015-07-17.

[10] As of 2015-07-17, see Swish. Available at: [https://www.getswish.se/] Accessed 2015-07-17

all users.[11] When the rumour was spread on the Internet developers started contacting Swish®, asking for the source code, but the company chose not to comply with the requests and decided not to redistribute source code, which means that they did not meet criteria of the GPL v.3 license.[12] The claimed original right holder to the source code, Moxie Marlinspike, has yet to decide if he wants to take legal action against the company.[13]

Imagine if the original right holder to Swish® sued all the end-users of the application for copyright infringement? As I will discuss in my thesis, copyright gives the right holder the exclusive right to make copies of a protected work and a user of a software program that is copyright protected makes a copy of the protected work in the software program both when they download the work to a hard drive and each time they run the software program. If you got sued by the right holder when you used the Swish® application would the company behind Swish®, often your bank, be liable for your costs?

## 1.2 Purpose and research questions

The purpose of this thesis is to critically analyse the copyright protection of computer software. To be able to critically analyse the copyright protection of computer software I have to put this subject into a context, and the contexts chosen for this thesis is the risk for intellectual property defects in transactions of software between a software developer as seller and an end-user as buyer, and the objective of an efficient allocation of resources in society.
This is conducted by using three perspectives:

---

[11] Nullbyte. Open curtains in swish payments service, *Nullbyte*. 2014. Available at: [http://blog.nullbyte.eu/open-curtains-in-swish-payments-service/] Accessed 2015-07-17, and Free Software Foundation. GNU General Public License version 3. *GNU Operating System*. 2007. Available at: [https://www.gnu.org/licenses/gpl.html] Accessed 2015-07-17.

[12] Beppe Bergqvist. *Hej, ni verkar använda ett GPLat bibliotek, så jag skulle vilja ha tillgång till källkoden för er Androidapp. Detta är licensvillkor ni godkänt när ni valde att använda det GPLade biblioteket,* [Facebook discussion with Swish], Facebook. 2014. Available at: [https://www.facebook.com/getswish/posts/10152347980456949?comment_id=10152349476451949&offset=0&total_comments=3] Accessed 2015-07-17.

[13] Jonas Ryberg. Säkerhetsgurun: Swish har stulit min kod, *IDG.se*. 2014. Available at: [http://www.idg.se/2.1085/1.593454/sakerhetsgurun-swish-har-stulit-min-kod] Accessed 2015-07-17.

1. How uncertainty enfolds when interests behind software development are protected by copyright,
2. How the economic risk, especially regarding claims made by third parties based on copyright, enfolding from the uncertainty in (1.) can be distributed in transactions between seller and buyer, as well as,
3. How the distribution in (2.) affects an efficient allocation of resources.

To be able to fulfil the purpose of this thesis I will answer the following research questions.

### 1.2.1 Research questions

1. How does uncertainty emerge in the intersection of copyright and development of software?
2. When a seller transacts the software to a buyer, what responsibility do the seller have for the uncertainty found in question 1, especially regarding claims made by third parties based on copyright, in relation to the buyer if this is not regulated in an agreement?
3. How can the parties in a transaction handle the uncertainty found under question 1 and 2, with contractual tools, especially in end-user license agreements?
4. Is it so that the uncertainty associated with our current copyright and contractual system risks an inefficient allocation of resources, and although I conclude that it is possible to manage such uncertainty (as demonstrated in the previous question) with the contractual tools, the ability of the parties to be informed and take precaution still risks an efficient allocation of resources?
5. If I under question 4. will find that there is a risk of an inefficient allocation of resources, what could be a suitable legislative measure to increase the efficient allocation of resources?

## 1.3 Method

This thesis is based on a legal constructivist theory, as explained in the section 1.4 below. The **first question** was answered by using a qualitative method where I both analysed how a piece of software (which should be differed from the legal definition of a *computer program*) can be argued to be protected by copyright law and how software development techniques correspond to the copyright protection. To find suitable software development techniques I have got myself an idea of modern trends in software

development, which causes friction with the model of creation that is embedded within the copyright phenomenon. Since software development is done in a similar manner throughout the world, I have looked for development techniques that were similar over the world. As I constructed a protection of software by copyright I applied a legal constructivist approach where I considered the concept of software development, searched for sources of law, and then designed normative claims, i.e. arguments, of how software is protected by copyright.

The **second question** was also answered by applying a legal constructivist approach to the law where I analysed and constructed what an intellectual property defect is and designed normative claims of how the responsibility of intellectual property defects are allocated between the licensor and the licensee in absence of an explicit agreement regulating this matter. The **third question** was answered by searching for end-user license agreements with clauses that govern a party's responsibility if the transacted object infringes in an intellectual property right. These clauses are used as examples or as inspiration of clauses that regulate the risk and liability of intellectual property defects between the seller and the buyer. I also analysed how different clauses affect the responsibility of intellectual property defects. When I have analysed the risk in the relationship between seller and buyer I have examined clauses in the following categories: warranties, warranty disclaimers, limitation of liability and intellectual property indemnifications.

The **fourth question** was answered by applying economic theory to construct an argument as to which distribution of liability for intellectual property defects generate the most efficient allocation of resources. I compared the result from this question with my conclusions in the second and third question and I used economic theories explaining the intersection between law and economics relating to liability, transactions and allocation of resources. I applied the economic principles found to support my analysis. The **fifth question** was answered by analysing my conclusions from the first four questions, after which I constructed legislative measures that affect the allocation of resources towards an increased efficiency.

### 1.3.1 The legal phenomena

When I have deconstructed and reconstructed the legal phenomena I have used sources of information that create a notion of permanence and collective belief regarding the

legal phenomena.[14] For this thesis I have used Swedish and European legislation, case law from the Swedish, European and U.S. jurisdictions, preparatory works for the Swedish legislation, legal doctrine (the authors communicate normative claims, and thus they affect the belief of lawyers and society by the logic of the their argument), in chapter 4 I have also used commercial license agreements as normative claims for inspiration to construct possible distributions of liability. I conducted a qualitative interview with one actor in the market of software development to get myself an opinion of how such an actor regarded the legal phenomenon and to get myself an understanding of the interests behind software development.[15]

These sources were applied by interpretation. The interpretation was done in several ways, e.g. by literary interpretation, by teleological interpretation or interpretation according to the logic of the legislation. The priority that I have attributed to these sources affected the construction of this thesis. When prioritizing the sources I have considered the values *permanence* and *collective belief* i.e. the sources that created the strongest notion of permanence and collective belief regarding a legal phenomenon received a higher priority than a source that created less notion of permanence and collective belief regarding a legal phenomenon. This priority should not be confused with the dogmatic legal method were a source of higher priority is seen as being a more valid indication of "the law" than a source of lower priority. The priority of sources of this thesis only indicates which sources create a stronger or lesser notion of permanence and collective belief, and thus affecting the reification process accordingly.

For this thesis, the priority of the sources of information that create a notion of permanence and collective belief regarding the legal phenomena is as follows:

1. European and international legislation
2. Swedish legislation
3. European case law
4. Swedish case law
5. Swedish preparatory work
6. U.S. case law

---

[14] For an explanation of permanence and collective belief, see the section 1.4.1, about legal constructivist theory below.

[15] The low number of interviews is due to the limited time constraint of this thesis.

7. Literature
8. Commercial agreements
9. Opinions from actors in the market

## 1.4 Theory

This thesis is built upon several theories that are used for different purposes in the thesis. The first theory, the legal constructivist theory, is the base of the thesis and the theory that explains how I view the law. This theory is used throughout the whole thesis when deconstructing and constructing normative claims regarding the legal phenomena. The second theories presented are economic theories that are related to transactions, liability and allocation of resources. These theories were used in chapter 5 and chapter 6 when I analysed the effects of the legal phenomena. When I conducted the interview I used the theory on interview methodology during the interview process, which is presented last.

### 1.4.1 Legal constructivist theory

This thesis is based on social constructivist theory, especially the legal constructivist theory.[16] As explained by Glavå & Petrusson (2002): "*The foundation of a legal constructivist approach is the recognition of the need for, on the one hand, the ever-present idea of deconstruction and on the other hand, the ever-present idea of design* [my translation]".[17] The ever-present idea of deconstruction means that the lawyer must always translate a (constructed) legal norm into its real underlying interests and relate it to the specific context. With this perspective the lawyer can analyse whether there is wide or narrow normative space for argumentation (see section 1.4.2 below). In the ever-present idea of design the lawyer recognizes that he or she is *always* part of the construction process of the legal phenomena through communicative actions that lead to normative experiences in society. This means that the individual lawyer has a responsibility for the legal construction he or she constructs while performing a task, why it is unsatisfactory if the lawyer hides behind the argument that they have only *found* the existing law.

As a legal constructivist, I do not believe in a one autonomous law (commonly known as *de lege lata*), which the lawyer can find by applying a correct legal method. Instead of one

---

[16] Mats Glavå & Ulf Petrusson. Illusionen om rätten! – juristprofessionen och ansvaret för rättskonstruktionerna. In *Erkjennelse og engasjement : minnesseminar for David Roland Doublet [1954-2000],* Askeland, B & Bernt, J.F. (eds.). Bergen: Fagbokforlaget, 2002, p. 25.

[17] Ibid., p. 23.

autonomous law I regard the law an institutional fact in the social world created by collective learning experiences, externalisation and internalization processes among human beings.[18] Through these processes the law is experienced as a concrete fact and not something created by human beings. This occurrence is called reification.[19] Reification comes from the word *reify,* which means to regard something abstract as a concrete existing thing.[20] Reification is therefore the process of turning something abstract into an impression of something concrete. The law is a set of norms whose purpose is to regulate behaviours and control conflicting interests in society by creating accepted institutional facts such as *limited liability companies, courts, transactions* and *liabilities*. These institutional facts become accepted in society by reification. The objective reality has to be translated to fit these norms. For example, we can perceive through a "search process" that a human being has helped another human being in exchange for a physical object valued by other human beings. This act might be translated, through a "justification process" into an "employment" or a "consultancy service".[21] Depending on what norm the reality is translated into, and communicated as, it creates different normative experiences for the human beings in this example, an illustration of such normative experience would be the tax authorities (another institutional fact) demand of payments for social security taxes or VAT. For the legal construction to give rise to normative experiences they have to be internalized (accepted) and externalized (communicated) by the actors in the legal profession and ultimately by society, this creates collective belief in the legal phenomenon and an impression of the legal phenomenon as something permanent. Sources of information about the law such as legislations, preparatory works, case law and doctrine have a central role in maintaining the legal construction by documenting and communicating legal norms.[22] These sources contribute to a notion of permanence and collective belief towards the legal

---

[18] Peter Berger & Thomas Luckmann. *The social construction of reality - A Treatise in the Sociology of Knowledge.* London: Penguin Books, 1991, pp. 78 - 79. For "institutional fact" see John Searle. *The construction of social reality.* New York: Simon and Schuster, 1995, p. 27.

[19] Berger & Luckmann, pp. 106 - 109.

[20] See the definition of "*reify"* at Merriam-Webster. Available at: [http://www.merriam-webster.com/] Accessed 2015-07-17. Search for "reify".

[21] Glavå & Petrusson, p. 29.

[22] Ibid., p. 27.

phenomenon and they are used during the justification process when the lawyer justifies his or her creation of a legal construction.[23]

### 1.4.2 Normative space, uncertainty and risk

In this thesis I have used the concept of "normative space".[24] Normative space relates to what level of difference may exist between two different communicative actions (normative claims or arguments) about what a specific legal constructions is and how it should function etc., while both communicative actions still is acceptable by the legal profession and society. A legal construction that has been completely reified in society has a narrow normative space since it would only exist one communicative action, i.e. argument, which is accepted by the legal profession and society. For example, a court will most likely ridicule a claim that the legal concept of *physical persons* includes *cats*. It is highly certain that society will act as if cats are not physical persons, e.g. not enforcing cats as owners of property, why it´s easy for a lawyer to perceive how such a communicative action would be received in society.[25] In such case the normative space is narrow. In comparison if a constructed legal norm has only been reified in society to a small extent there are more alternative communicative actions that can be accepted by the legal profession and by the society. For example, when analysing whether a small excerpt of a specific computer source code is protected by copyright one will find that this matter is only subject to a low level of reification in society since there are several competing communicative actions available depending on what outcome the acting entity wants to achieve, i.e. the normative space is wide. In such circumstances it is harder for a lawyer to predict how a specific communicative action would be received by society. In other words, if there exists one communicative action with a high level of collective belief to that action the normative space is narrow, while if there exists several possible communicative actions with low levels of collective belief, the normative space is wide.

---

[23] Ibid., pp. 25 – 31.

[24] Ulf Petrusson. *Intellectual Property & Entreprenuership – Creating Wealth in an Intellectual Value Chain.* Gothenburg: Center for Intellectual Property Studies (CIP), 2004, pp. 120 - 121 and pp. 166 – 169.

[25] This might of course change, for example there are lawyers arguing that animals should not be treated as *property*, see for example Cass R. Sunstein & Martha C. Nussbaum. *Animal rights: current debates and new directions.* USA: Oxford University Press, 2005.

From each communicative action follows a set of normative experiences, i.e. consequences. For example a claim of infringement in an intellectual property right might create a claim of liability, which might result in an obligation to pay a certain amount of money to another entity etc. Where there are several alternative communicative actions, each of these actions have different normative experiences for entities in society, and depending on the setting and market these normative experiences might result in additional costs for different entities. Thus, if the normative space is wide, i.e. several arguments can be accepted and several consequences might occur, it is severely difficult for entities in the market to predict what their cost might be in a particular situation or transaction, thus if the normative space is wide this gives rise to uncertainty and risk.

### 1.4.3 Economic theory

In this thesis I discuss uncertainty in relation to protecting, transacting and infringing copyright protected software. Uncertainty as a phenomenon is nothing undesirable per se, for example it might provide for flexibility on a case-by-case basis, why it must be put into relation to something else to be determined as something desirable or not. For this thesis I discuss uncertainty in relation to an efficient allocation of resources in society. One purpose of economic theory is to find models for an efficient allocation of resources and explain behaviours of actors in the market.[26] According the economic theory, resources should be allocated were they provide most efficiency.[27] For this thesis the efficient allocation of resources is one where wealth is maximized, i.e. the entity that values a good the most should have that good. As I show, the distribution liability for intellectual property defects relate to the uncertainty and to an efficient allocation of resources. Since intellectual property, distribution of liability and allocation of resources are all built upon economic theory, I used economic theory to support my argument in chapter 5 and chapter 6.

According to economic theory the law can help to increase to efficient allocation of resource in two different means. First according to Coase theorem the law can be used to lower transaction costs to increase successful bargaining between parties, the parties is then able to allocate the resources efficiently depending on who values a transactable

---

[26] Robert Cooter & Thomas Ulen. *Law and Economics.* 6th Ed. Boston: Pearson Education, 2012, chapter 2.

[27] Ibid., p. 7.

object or right the most. An example of this would be if the law would assign clear and precise ownership rights that make it less difficult to transact objects or rights in a market economy. Secondly, Hobbes theorem states that if the parties cannot bargain to reach an efficient allocation or resources the law should allocate the resources efficiently by distributing transactable objects or rights to the party who values them the most.[28]

#### 1.4.3.1 Coase theorem, transaction costs and externalities

According to Coase theorem, rational parties will allocate legal entitlements between themselves in the most efficient distribution, given zero transaction costs.[29] Transaction costs is a term that encompass all impediments to bargaining and can be divided into three subcategories: *search and information costs*, *bargaining and decision costs*, *policing and enforcement costs*.[30] *Search and information costs* relate to imperfect information about the supply and location of trading opportunities and/or the characteristics of the items accessible for transactions, such as quality. *Bargaining and decision costs* relate to costs for finding out the demand of other parties, i.e. at which prices and conditions are they willing to trade. *Policing and enforcement costs* relate to enforcement of contracts and these are necessary since the parties don´t possess the information as to whether the other party will be perform their obligations in the agreement.[31] All these relate to imperfect information. As stated by Dahlman "*it is really necessary to talk only about one type of transaction cost: resource losses incurred due to imperfect information*".[32]

In the perfect transaction, the parties share both all benefits and all costs in a transaction. However zero transaction costs is not a realistic goal and sometimes when a company produces goods they create externalities.[33] An externality is an effect, positive or negative, that affects other parties than the ones in the transaction. For example a factory might emit smoke while producing products, which contaminates the surrounding air of the factory. If such contamination is not a cost priced in the transaction between the seller

---

[28] Ibid., p. 93 and p. 103.

[29] Ibid., p. 85 and p. 291.

[30] Cooter & Ulen, p. 85. See also, Carl J. Dahlman. The Problem of Externality, *Journal of Law and Economics,* Vol. 22, No. 1, The University of Chicago Press for The Booth School of Business of the University of Chicago and The University of Chicago Law School (1979), p. 148.

[31] Dahlman, p. 148.

[32] Ibid.

[33] Cooter & Ulen, p. 39.

and buyer it will be carried by third parties e.g. the inhabitants living close to the factory, who might need to invest in air filtration to coup with the smoke. The cost of this smoke is an externality. The total cost to society, i.e. the *social cost,* in the transaction is the private cost plus the externality. Prices are used to reflect the actual cost of a product or service in relation to competing goods, thus enable the buyer to make an informed decision about their purchase.[34] Resources are misallocated if goods are produced which the buyer would not want if he or she had to pay the full extent of the cost of that good to society, i.e. the social cost.[35]

#### 1.4.3.1.1 Intellectual property infringements as a negative externality

If one firm, A, is out-competing another firm, B, by copying their copyright protected source code, it can be argued that the cost of the firm B´s lost sales is purely private and it may be difficult to identify an externality at all. However since the purpose of copyright protected computer programs is to incentivize investments and creation of quality computer programs I have assumed that such intellectual property infringement lead to decreased incentives for investments and creation of quality computer programs why loss of incentive for these subjects is the externality of infringements in copyrights.

### 1.4.3.2 Intellectual property defects as consumer product injuries

Since the transacted software might cause additional costs for the buyer when using the software program, due to claims made by third parties based on intellectual property rights, this resembles the situation were a buyer becomes injured by the use of a bought product, i.e. *consumer product injuries.*[36] There are two important aspects of liability for consumer product injuries that can be applied to liability for intellectual property defects in software transactions. The first is whether the buyer is *perfectly or imperfectly informed*,[37] the second is whether precaution for reducing the likelihood and severity of a damaging occurrence is *unilateral or bilateral.*[38] For this thesis I have assumed that this theory is applicable on allocation of liability for intellectual property defects.

---

[34] Guido Calabresi. Some Thoughts on Risk Distributions and the Law of Torts, The Yale Law Journal, Vol. 70, No. 4, (1961). Available at: [http://digitalcommons.law.yale.edu/fss_papers/1979] Accessed 2015-07-17. p. 502.

[35] Ibid., p. 503.

[36] Cooter & Ulen, p. 225.

[37] Ibid., p. 226.

[38] Ibid., pp. 251 - 252.

#### 1.4.3.2.1 The perfectly or imperfectly informed buyer

As said above, according to economic theory a goal is efficient allocation of resources. The allocation of resources regarding consumer product injury depend on the behaviour of the buyer, which is determined by the information accessible to the buyer, the construction of liability law and the market for the product.[39]

| **Cost of product** | **A** | **B** | **C** | **D** | **E** |
|---|---|---|---|---|---|
| **Behaviour of firm** | **Firms cost of production per unit** | **Probability of accident to buyer** | **Loss if accident** | **Expected accident loss** | **Full cost per unit** |
| Use Battery 1 | 20 Euro | 1/10 000 | 200 000 Euro | 20 Euro | 40 Euro |
| Use Battery 2 | 25 Euro | 1/20 000 | 100 000 Euro | 5 Euro | 30 Euro |

Table 1.

For example, a buyer has the decision to purchase either Battery 1 or Battery 2 for their computer, see Table 1.[40] Battery 1 is cheaper to produce than Battery 2 as seen in column A, however Battery 1 is twice as likely to cause an accident (indicated by column B), e.g. explode and cause a fire, and the expected loss is more severe than with Battery 2 (column C). The expected loss in column D is calculated by multiplying the probability of accident to buyer (column B) with the loss if an accident occurs (column C). The full cost per unit (column E) is calculated by adding the firm's cost of production per unit (column A) and the expected accident loss (column D). As can be seen the full cost per unit is lower for Battery 2 than for Battery 1, as indicated in column E, why an efficient use of resources would be for the buyer to purchase Battery 2. The question is if a customer would actually buy Battery 2.

We assume perfect competition, why the price of a unit in the market equals the production cost plus the cost of the manufacturers liability.[41] Therefore under a rule of no liability (for the manufacturer) the cost of each unit will equal the firm's production

[39] Ibid., p. 226.

[40] This example is inspired by the example in Cooter & Ulen, p. 225.

[41] Cooter & Ulen, p. 226.

cost, i.e. 20 Euros for Battery 1 and 25 Euros for Battery 2. If the buyer would be perfectly informed they would know that they must bear the costs during an accident and they would also know the probability of an accident and the likely cost of such accident. The rational buyer would buy Battery 2 since the total cost is lower for them than if they would buy Battery 1. The buyer would buy the *most efficient product*. However if the buyer would be imperfectly informed (not knowing about the probability or costs of an accident), still under a rule of no liability for product injuries, there is a risk that the buyer would underestimate or disregard the probability or cost for an accident associated with Battery 1, choosing to buy the *less efficient product* Battery 1.[42] If we include a rule of strict liability for the manufacturer/seller the cost of each unit, as said above, will equal the firm's production cost plus the cost of the manufacturers liability, i.e. 40 Euros for Battery 1 and 30 Euros for Battery 2. In such circumstances even the imperfectly informed buyer will choose Battery 2 even though they could overestimate, underestimate or disregard the probability or the cost of an accident with Battery 1.

This theory shows that if a buyer is not informed about probability or cost of an accident by a bought product, the most efficient resource allocation is still reached when the seller has a strict liability. The model ignores shortcomings of the system such as administrative costs, the lack of incentives for precaution by the injured party, and overinsurance of consumers by the producers.[43] However even when considering these issues, they don´t necessarily affect the accuracy of the model.[44]

#### 1.4.3.2.2 Unilateral or bilateral precaution

According to economic theory, the liability standard for consumer product injuries can be determined by regarding whether precaution for reducing the likelihood and severity of a harming event is *unilateral or bilateral*.[45] If the precaution lies unilaterally, i.e. only one party can reduce the likelihood and/or severity of harm, economic theory states that this party should be the one to carry the costs and the liability of such accident. Most often this is the producer of the good, since they are in control of the design- and production process and they are also the ones who can assess whether the product has any special

[42] The customer could also overestimate the probability or cost for an accident associated with Battery A, still choosing to buy Battery A.

[43] Cooter & Ulen, p. 226.

[44] Ibid., pp. 230 - 244.

[45] Ibid., pp. 251 - 252.

dangers towards the user, why they can most efficiently inform to buyer of such dangers through warnings.[46]

If the precaution lies bilaterally, i.e. both parties can reduce the likelihood and/or severity of harm, economic theory states that the liability standard should be based on a rule of negligence. In such a solution the producer might have liability for the design and the manufacturing process, while the user might have liability for misuse of the product. This incentivizes the producer to provide the buyer with information about proper use to avoid liability.[47]

### 1.4.4 Theory on interview methodology

When I have conducted interviews to gather knowledge I have used inspiration from interview methodology.[48] To be coherent with the social and legal constructivist approach I have applied to the legal phenomenon, I have applied a constructivist approach to the knowledge gathered through interviews. I therefore regard an interview as *knowledge construction* rather than *knowledge collection*.[49]

The interviewer as traveller is a good metaphor for this perspective. Brinkmann & Kvale (2015) explains this metaphor:

> "The interviewer-traveler [sic] wanders through the landscape and enters into conversations with the people he or she encounters. The traveler explores many domains of the country, as unknown terrain or with maps, roaming freely around the territory. […] The potentialities of meanings in the original stories are differentiated and unfolded through the traveler's interpretations of the narratives he or she brings back to home audiences. The journey may not only lead to new knowledge; the traveler might change as well. The journey might instigate a process of reflection that leads the traveler to new ways of self-understanding, as

---

[46] Ibid., p. 252.

[47] Cooter & Ulen, p. 252.

[48] Brinkmann, Svend. & Kvale, Steinar. *Interviews – Learning the craft of qualitative research interviewing*. 3rd Ed. Thousand Oaks, California: SAGE Publications, Inc., 2015. See also, Annika Lantz. *Intervjumetodik*. 2d Ed. Pozkal: Studentlitteratur, 2007

[49] Brinkmann & Kvale, p. 57.

> well as uncovering previously taken-for-granted values and customs in the traveler's home country".[50]

An alternative perspective on interviews contrast to the interviewer as a traveller is the perspective that the interviewer is rather like a miner. In the interviewer-miner metaphor the interviewer searches for pieces of knowledge waiting to be uncovered by the interviewer. The pieces of knowledge found are viewed as objective facts that remain constant during the interview process.[51] I have applied the interviewer as a traveller approach by thinking of the interview more as a conversation rather than a strict interview, and also by letting the interview subject answer as freely as possible during the conversation. After conducting the interview I have translated the material relevant for this thesis into English and the interview subject has approved of the translation used.

## 1.5 Delimitations

This thesis is not focused on the question whether intellectual property rights should protect software or not. I have assumed that there is an importance in having an efficient protection of the interests behind software development and this thesis only critically analyses how such protection is designed. And as explained in the section on theory I use the aim of *an efficient allocation of resources*, and this thesis do not question whether the objective of wealth maximization is fair or not.

In this thesis, I do investigate software development and software transactions from a copyright perspective, focused on the Swedish jurisdiction, although I do provide arguments from other jurisdictions that may affect the notion of permanence and collective belief regarding the legal phenomena in Sweden as well. Under certain circumstances software programs can be protected with patents as well as copyright, however in this thesis I do not investigate the patent protection of software. Due to this, when I refer to *intellectual property defects*, these are based on third party claims based on copyright.

Regarding copyright; this thesis focuses on the economic rights granted to the right holder by copyright, especially the right to make reproductions of a protected work. I do

---

[50] Ibid., pp. 56 - 57.

[51] Ibid., p. 57.

not focus on the right to make it available to the public, mainly because the right to make reproductions of the work is the most common right infringed when it comes to software programs.

I also delimit this thesis to commercial transactions of software programs where both parties in the exchange are commercial actors. To delimit this thesis I have also used a simplified value chain existing of a seller and a buyer of the software program and the seller is also the developer of the software program. I have used the terms "software developer", "licensor" or "seller" to indicate the party who has the obligation to supply the software program in the transaction and the words "customer", "licensee" or "buyer" as to indicate the party who has the obligation to pay for the software. The seller is also mentioned as *employer,* having employees and consultants that develop software programs. I do also mention a "third party"; this entity is not part of the value chain mentioned above but has copyright to a source code or another element that can be incorporated in a software program. I have used the terms "software", "computer software", "software program" or "piece of software" regarding the same type of transacted object. These terms should not be confused with the term "computer program" which is used as the legal definition of a category of elements protectable with copyright. Although as shown in this thesis a "computer program" is often one of several elements in a "software program".

The thesis is also focused on *commercial off-the-shelf* ("COTS") software, as opposed to *customised software* (where the buyer orders customised software from a software developer). In the third research question I do not discuss in depth whether the clauses mentioned are legitimate or not, i.e. I do not analyse whether they can be changed or invalidated in court, for example by art. 36 in the Swedish Contract act. I have used the word "risk" and the word "uncertainty" for the same phenomenon even though there might exist differences between these words.

## 1.6 Disposition

In chapter 2, I critically analyse how the legal phenomenon of copyright protection of software creates uncertainty. I touch upon the history of software programs including the incentives for protecting software programs with copyright and the conducts in modern software development. I relate these matters to the criteria for protecting, transacting and infringing copyright protected software. If the reader would like to understand more

about the complexities in the intersection of software programs and copyright I would suggest reading the second chapter.

In chapter 3, I discuss the transactions of software and intellectual property defects. This part is centred on the transaction of software programs between two entities and is focused on intellectual property defects in such transactions. This chapter is suitable for the reader who would like to know more about how to define a transaction of computer software and how one can define an intellectual property defect.

In chapter 4, I have provided the reader with some examples of clauses that distribute the responsibility for intellectual property defects between seller and buyer. If the reader is interested in examples of commercial contract clauses that distribute liability for intellectual property defects, this is a suitable chapter to read.

Economic theory is applied to the distribution of liability for intellectual property defects in chapter 5. This chapter presupposes that the reader has a basic understanding of intellectual property law and economic theory why it might be challenging for the novice reader. The chapter analyses how the liability for intellectual property defects affect an efficient allocation of resources.

Chapter 6 provides the reader with a suggestion of changes in the Swedish Copyright act (SFS 1960:729) that would allow for an increased efficiency regarding the allocation of resources. This chapter is suited for a reader who wants to analyse my conclusions of the subject.

# 2 UNCERTAINTY – How the legal construction of copyright protected software creates uncertainty

## 2.1 Introduction

An interesting aspect with computer programs compared to other copyright protectable creations is that when an audience is looking at a performance they can perceive the protected expression of an idea, the performance. The reader of a book can perceive the protected expression of an idea, the text. The observer of a painting may as well perceive the protected expression of an idea, the painting. However the user of the computer program will most often not be able to perceive the protected expression of the idea, the source code. The user of the computer program may often experience the actual function or idea behind the computer program through the graphical user interface, which can be an own protected work, but they won´t see the underlying source code which is "hidden" through compilation.[52] This is one of several features of computer programs that makes them different compared to other works protected by copyright. In this chapter I will discuss computer programs and the broader concept "software programs" related to protection, transactions and infringement of copyright.

To be able to explain why copyright protection of software creates uncertainty I will start with a presentation of the historic background of copyright protection of software programs. I will also present some trends in software development that influences the decision on what elements of software is protectable by copyright. I will further deconstruct software programs into three different elements, source code, databases and GUIs and investigate their individual protection before discussing the transactions and infringement of copyright relating to software programs. This will show that the current protection of software creates uncertainty.

## 2.2 The birth of the copyright protected software

The notion of protecting instructions to a machine is a rather new phenomenon. In the first half of the twentieth century the focus among computer scientists was to develop hardware, i.e. machines, and the software, i.e. the instructions to those machines, was only seen as a necessity to make the hardware function. As the hardware developed and

[52] Jon Bing. Vurdering av opphavsrettslig krenkelse av datamaskinprogrammer et praktisk perspektiv, *Nordisk immateriellt rättsskydd*, NIR 2/1999, (1999), p. 283.

became more advanced the need for more complex instructions to control the hardware increased. Because of the increased demand for instructions some people started to specialize and work solely with developing such instructions for the hardware and the profession of the software engineer emerged. In 1968 the importance of software was discussed during the NATO Software Engineering Conference, which was also the first time "software engineer" was mentioned as a term for the programmer. The main focus of the conference was to raise awareness of software and the problems in software development.[53]

Up until that time software had been seen as an instruction to assist the hardware to perform its function and the hardware was the main commodity of the two. However during the conference they discussed whether software should be priced separately from hardware. The importance of this subject should not be underestimated; if software would be priced separately from hardware you would create a new market and software itself would become a commodity. The arguments in favour for price separation, which represented the large majority, were among others that if software would have a price and its own market competition would arise and the software quality would increase. They also noted that software, at that time, was treated as if it were of no financial value why there was no effective market, and to be able to build a market, the knowledge built in the software would need to be protected in some way.[54] There were also arguments against a separate market for software, among others, the now classic free software argument that software belong to the world of ideas and should be free for all.[55]

A couple of years later, in the 1970's, WIPO initiated an investigation regarding the protection of software programs under copyright.[56] The work on these matters began by means of an advisory group of individuals who established a document containing recommended standard regulations for national legislation of the legal protection of

---

[53] This period was called the software crises. The cause of the crises was the difficulty of developing useful and efficient software in required time. Peter Naur & Brian Randell. *Software Engineering Report on a conference sponsored by the NATO Science Committee Garmisch, Germany, 7th to 11th October 1968*. NATO Scientific Affairs Division, 1969. p. 8 Available at: [http://www.scrummanager.net/files/nato1968e.pdf] Accessed 2015-07-17.

[54] Ibid., p. 76.

[55] Ibid., pp.76 - 77.

[56] SOU 1985:51. *Upphovsrätt och datorteknik*, p. 38.

computer software.[57] Four reasons for the protection of software were explicitly mentioned: The first reason was to protect the financial investment and the time that must be invested in the development of software. As a second reason they mentioned that software is likely to form an increasing part of the computer systems in the future. As a third reason it was stated that an effective protection would encourage the software developers to make their programs public. As a fourth, and final, reason it is mentioned that software is vulnerable, because a program developed by investments and then released in the mass market is very easy and cheap to copy, whereby a considerable damage is caused to the developer.[58]

In the year 1980 the U.S jurisdiction chose to explicitly protect computer programs under copyright and under the years following, Australia, Germany, Great Britain, France and Japan made the same decision.[59] The trend among these countries was recognised in the European Community at large including the Swedish jurisdiction, and in Sweden the copyright system was proposed as the mean to protect the interests behind software development, since it was concluded that "*Sweden can not in such an economic key area as* [the protection of computer programs] *go their own way without regarding how the other industrialized countries handle the legal issues for computer programs*".[60] I the preparatory works prior to the explicit regulation of computer programs in the copyright legislation it was determined that protecting computer programs under copyright was unproblematic since the copyright system was adaptable and prepared to handle new forms of intellectual creation.[61] The proposal succeeded and computer programs became explicitly mentioned as protected works in the Swedish Copyright Act (SFS 1960:729).

## 2.3 Modern software development

If someone creates a piece of art, writes a book, or performs a dance there are few who would question whether those acts would be creations subject to copyright. The legal construction of copyright in such works has become reified in society to such extent that they are taken for granted. The alike cannot be said about software. Copyright is built

---

[57] WIPO. (1978). *Model Provisions on the Protection of Computer Software, WIPO publication ; no. 814*.

[58] SOU 1985:51. *Upphovsrätt och datorteknik*. p. 44.

[59] Prop. 1988/89:85. *Om upphovsrätt och datorer*, p. 10.

[60] Ibid.

[61] Ibid., p. 11.

upon the idea of the sole genius, the lone artist who creates a piece of art as an expression of his or her intellect. At most this perspective suits a situation were a lone software engineer sits by himself or herself and writes source code in a closed environment without having any inspiration from other software. Such developer would write all software from the start without using pre-existing libraries of source code available. In such a situation it would be rather easy to apply the criteria for copyright protection and determine what should be protected as a computer program. However in the modern environment of software development there is often several people involved, inspiration is everywhere, both when searching the Internet and when working for an employer. And the notion of "free software", were source code could be used without the obligation to pay a license fee under certain circumstances, is increasing.

### 2.3.1 Incentives during development

A matter interesting for this thesis is the incentives while developing the work. While the classic artist strives to achieve something original and unique when creating a work, a software developer often strives towards other values when developing software than creating the software in the most unique way. For a software developer values such as efficiency, readability, comprehensibility and maintainability are common in the development of software.[62] This is natural since the purpose of software development is to create function rather than a unique expression. This difference in values is crucial and as seen below in section 3.4 the criteria for protecting a piece of art with copyright might not be as suitable for protecting a piece of source code.

Another issue is the concept of "code reuse". Already during the software crisis in the 1960's it was recognized that developing useful and efficient software takes time. You would have to understand the problem, come up with a solution and write the solution in a way that is efficient, and every time you start a new project you would have to start from the beginning. It was noted that if software development could be more like an industry, were the developer could use already constructed components that they knew solved the problem efficiently, the final software would have a higher reliability and performance while more effort could be put into solving critical parts of the software

[62] See Chad Perrin. Why clean code is more important than efficient code, *TechRepublic*. 2011. Available at: [http://www.techrepublic.com/blog/software-engineer/why-clean-code-is-more-important-than-efficient-code/] Accessed 2015-07-17.

system.[63] The difficulty in the 1960's was that that the software developer had no catalogue of such components to order pieces of software from, why all components were developed from start each time. A major difference today is that sharing of information and data has become much easier through technical developments and the Internet. Today *code reuse* or *software reuse* is the concept of incorporating existing code into development of new software.[64] The developer does this by finding source code that performs a function, copies that source code and implements the code in their own software program. Depending on the circumstances the code might need some adjustments to be implemented, however this might still take less time than writing the source code from start. There are several motivators for code reuse.[65] Among others it increases productivity and shortens time to market since the time spent writing code can be decreased. It is also claimed to improve software quality and reduce risk of malfunction since the source code is known to encompass the functionality and interoperability.[66]

---

[63] Douglas McIlroy. Mass produced software components. In *Software Engineering Report on a conference sponsored by the NATO Science Committee Garmisch, Germany, 7th to 11th October 1968*. Naur, Peter. & Randell, Brian. (eds.) NATO Scientific Affairs Division, 1969.

[64] William W. Agresti. Software Reuse: Developers' Experiences and Perceptions, *Journal of Software Engineering and Applications, 2011, 1, 48-58,* (2011), p. 1. I have yet to find any Swedish articles discussion the legality of code reuse. Per Matson discusses negotiations regarding the ownership of the copyright in component built software, see Matson, Per. Förhandling om upphovsrätten till komponentutvecklade applikationer, *Juridisk Tidskrift vid Stockholms universitet. Årg. 10 (1998-99), nr. 3, p. 788-793*. However he does not consider projects were the components in the software are developed externally, outside the company.

[65] I have got the notion that code reuse seems to be motivated were the source code that needs to be written would take time and effort to *produce*, which means that the code would either take long time to write or it might take long time to develop because of its complexity. Both of these are types of investments that are or have been protected by intellectual property. Developing complex software is usually an expression of the author's own intellectual creation. Writing code that does not fulfil the criteria of originality might still require skill and labour. Under the *sweat-of-the-brow doctrine* an author acquires copyright protection to their work not on the basis of *originality*, but on the basis of *skill and labour* put into the work. This was first established by the House of Lords in Walter v. Lane [1900] AC 539 H.L, but is less used today were the authors own intellectual creation is used as criteria for protection.

[66] Lombard Hill Group. Software Reuse 101: What Is Software Reuse?, *Lombard Hill Group.* Available at: [http://lombardhill.com/articles/software-reuse-101-what-is-software-reuse/] Accessed 2015-07-17.

Another trend combined with code reuse is that developers are using code from the web.[67] Code retrieval on the web is the practice of using search tools to find source code on the Internet that the developer might reuse or learn from.[68] The reuse of source code found on the Internet has also increased in commercial software development because of the increase of open source software distributed under licenses allowing commercial use, which makes it highly attractive for commercial software developers. Some companies have begun to *systematically* reuse source code found on the Internet by having a systematic approach consisting of several step such as identification, evaluation, and integration of suitable code into their own software system to ensure compliance. In comparison to systematic reuse, it is more uncertain whether third party rights are respected or not during *ad-hoc reuse.* Ad-hoc reuse is when a software developer searches for source code, copies source code and implements it in software development, on his or her own and often without telling anybody.[69]

### 2.3.2 A study in software development

In a study made by Sojer and Henkel, they conducted an empirical investigation into whether professional software developers respected license obligations when they conducted ad-hoc code reuse from the Internet.[70] They found that more than half of the software developers in the study regarded ad-hoc reuse of code from the Internet at least "somewhat important" for their work.[71] Regarding the source from which the software developers had learnt about code reuse from the Internet almost one quarter had not received any training or information on this type of code reuse at all and only 37% had received training on this matter from a firm or an educational institution. What I myself consider the most interesting finding in the study is the software developers' answers to a

---

[67] Sim, et al. How Well Do Search Engines Support Code Retrieval on the Web?. *ACM Transactions on Software Engineering and Methodology (TOSEM),* Volume 21 Issue 1, Article 4 (2011).

[68] Susan Elliott Sim & Rosario E. Gallardo-Valencia. Introduction: Remixing Snippets and Reusing Components. In *Finding Source Code on the Web for Remix and Reuse*, Sim, Susan Elliott. & Gallardo-Valencia, Rosario E (eds.). Pages 1 – 14, New York: Springer-Verlag, 2013.

[69] Manuel Sojer & Joachim Henkel. License Risks from Ad-Hoc Reuse of Code from the Internet. *Communications of the ACM*, Vol. 54, No. 12, pp. 74-81, (2011), p. 2.

[70] See Sojer, Manuel. & Henkel, Joachim. License Risks from Ad-Hoc Reuse of Code from the Internet. *Communications of the ACM*, Vol. 54, No. 12, pp. 74-81, (2011). They used a survey to conduct the investigation in 2009 and the aim of the survey was towards current or former professional software developers. They received 869 answers from people across the globe with such profession.

[71] 59%, see Sojer & Henkel, p. 5.

question in a quiz regarding their knowledge about open source licenses and source code reuse. The question was (with alternatives):

"*Somebody posts a code snippet in the newsgroups or on a tutorial website. Under which conditions is it completely safe to integrate this snippet?*

1. *If the poster does not mention any obligations that come with the snippet*
2. *If the poster explicitly declares that he does not demand any obligations from using the snippet*
3. *If the snippet is not part of any program*
4. *If any one of the conditions above mentioned is true, integration would be safe*
5. *None of the conditions mentioned above would be enough*
6. *Do not know*"[72]

Even if the question regards a relatively common situation found on the Internet, only one fifth of the software developers answered correctly; that none of the conditions mentioned would be enough. The ones that had answered that they were "very familiar" with source code license obligations on the Internet, failed an average of two out of five questions in the questionnaire.[73] One interesting conclusions that is drawn in the study is that many software developers lack detailed information about Internet code reuse, i.e. using source code found on webpages on the Internet. 14 – 21% of the software developers have either not looked thoroughly for the license obligations or knowingly ignored such obligations while reusing code from the Internet. One conclusion drawn from these findings is that software firms should consider that some of their software, which incorporates reused code, might infringe in third party copyright, since there is a likelihood that they don´t fulfill the license obligations.[74] According to Elliot and Stenberg: "*Information and computer scientists and engineers who are trained in programming, and have engaged in software reuse and remix have a different perspective* [than legislators] *on what counts*

[72] See Sojer & Henkel, Appendix 4.

[73] Ibid., p. 11.

[74] Ibid., p. 15.

*as an idea that is worth protecting, the value of effort that goes into an idea, and status of copying in computer systems".*[75]

### 2.3.3 One software experience - several protectable elements

Another matter in the intersection of law and software is that while software, from a user perspective, is seen as *one* product enabling the computer to perform an entertaining experience or one or several functions for the user, it is seen from a legal perspective as consisting of several elements, some protectable by different categories of copyright, e.g. the instructions to the computer, the compilations of data, the graphical user interface are different elements that can be protected as a *computer program*, a *database*, or an *artistic work* etc.[76] The difficulties arise when different categories of copyright create different normative experiences, i.e. different categories are handled differently by the law and thus creates different consequences, especially when it relates to how the copyright is transacted. In the next part I will discuss further how the copyright protection of software creates uncertainty.

## 2.4 The protected work

Uncertainty emerges when trying to figure out how to translate a software program to fit into the legal phenomenon, i.e. what can be argued to be a *work* in a software program and thus be protected by copyright? As I will show in this chapter, a software program often contains several elements from the legal perspective and not all protected by the same category of copyrights. In this section I will look at the criterion for protection under Swedish and European law.

### 2.4.1 The law

According to art. 4 WIPO Copyright Treaty, which Sweden has ratified, computer programs should be held to be protected works under art. 2 Bern Convention, and in Sweden computer programs have been protected expressively since 1989 in the Swedish Act on Copyright as a literary work.[77] Although it is important to note that there is no

---

[75] Susan Elliott Sim & Erik B. Stenberg. Intellectual Property Law in Source Code Reuse and Remix. In *Finding Source Code on the Web for Remix and Reuse*, Sim, Susan Elliott. & Gallardo-Valencia, Rosario E (eds.). Pages 311 – 322, New York: Springer-Verlag, 2013.

[76] Jan Rosén. *Upphovsrättens avtal : regler för upphovsmäns, artisters, fonogram-, film- och databasproducenters, radio- och TV-bolags samt fotografers avtal.* 3rd Ed. Stockholm: Norstedts Juridik AB, 2006, p. 264.

[77] According to the preparatory work preceding the change in the Swedish Copyright act (SFS 1960:729) computer programs were protected under Swedish copyright inexpressively before 1989, as a literary work

legal definition of a computer program in the Swedish law.[78] In the European Union, copyright protection is divided among several directives depending on the category of protection. Protecting computer programs with copyright is the same construction used by all member states in the EU, which is currently regulated by Directive 2009/24/EC. However, a software program often contains several elements and not all of them are protected as computer programs but as other categories of copyright. If an element in the software does not fulfil the criteria for one category, e.g. the criteria to be protected as a computer program under the Directive or art. 1 in the Swedish Copyright act (SFS 1960:729), it can still be protected in another category, e.g. either under the database directive or under the Infosoc directive. All of these elements need to address certain basic criteria to be protectable by copyright. I will now touch upon these basic copyright criteria in relation to software.

#### 2.4.1.1 Novelty criterion

Copyright does not protect *priority*, i.e. that the work is a *novel* creation from an objective perspective. Copyright protects a work against unauthorised copying why it is enough with *subjective* novelty to receive protection.[79] This means that a situation might occur where two pieces of source code is exactly alike and both may be granted protection if they both fulfil the originality criterion.[80] This situation, that two pieces of source code is similar, is not just a theoretical example but might occur since there are only a limited number of ways to express a function with source code. This is so due to constraints while developing code, i.e. standardised language, efficiency and readability aims etc., which limits the freedom of the software developer. This can be compared to a writer of a book who has a larger freedom when creating the expressions due to fewer constraints. Thus if both software developers have developed the code without being aware of the other developer's code, the criterion of subjective novelty is fulfilled why both can be granted copyright protection.

---

stating that "*It is also evident that a computer program is an expression of human intellectual creativity, for which it requires a high degree of knowledge, capacity for logical thinking and intuition in order to in the simplest and most effective manner reach the best result* [my translation]", see Prop. 1988/89:85. *Om upphovsrätt och datorer*, pp. 10 - 11.

[78] Rosén, p. 261.

[79] Michael Plogell. Upphovsrätt till datorprogram ur ett EG-perspektiv. *Juridisk Tidskrift vid Stockholms Universitet. (1993-94), nr. 1,* p. 64.

[80] Marianne Levin. *Lärobok i immaterialrätt – Upphovsrätt, patenträtt, mönsterrätt, kännetecknesrätt – i Sverige, EU och internationellt*. 10 Ed. Stockholm: Norstedts Juridik AB, 2011, pp. 80 - 82.

I regard the subjective novelty criterion as troublesome when applied to computer programs since it provides incentives to reinvent the wheel. The novice software developer who has little knowledge of third party source code may fulfil the subjective novelty criterion without effort when developing new software, while an experienced software developer who has much knowledge of third party source code would thus risk infringing in the copyright to such source code during his or her own development of software. Normally this is solved with a *double creation criterion*[81] that states if a work has been created twice, independently from each other, it is a sign indicating that no one of the works fulfils the criteria of originality. Although due to the EU legislation among others, the criteria of originality is set to a low threshold why this might still occur. I will discuss this further below under the section on *infringement*.

#### 2.4.1.2 Originality criterion v. Threshold of originality

According to art. 1 in the Swedish Copyright act (SFS 1960:729) "Anyone who has created a literary or artistic work shall have copyright in that work",[82] however the copyright act does not mention or describe any further criterion regarding what should be fulfilled for something to constitute a "work". In the States Official Report preceding the current copyright law they discuss what establishes a "work". They write that a work is constituted by a mental effort with a degree of independence and originality, and that it need at least be the expression of the author's own individuality. They expressively write that *a mere mechanical production* is not sufficient for copyright protection.[83] These criteria for copyright protection have been called collectively *the threshold of originality*.[84]

---

[81] My translation of "dubbelskapandekriteriet".

[82] Unofficial translation of the Swedish Copyright act (SFS 1960:729). WIPO. Available at: [http://www.wipo.int/wipolex/en/text.jsp?file_id=290912] Accessed 2015-07-17.

[83] SOU 1956:25. *Upphovsmannarätt till litterära och konstnärliga verk*, p. 66 with a reference to the report of 1914 (*förslag till lag om rätt till litterära och musikaliska verk, förslag till lag om rätt till verk av bildande konst samt förslag till lag om rätt till fotografiska bilder, avgivet den 28 juli 1914*). These criteria are again recognized in SOU 2010:24. *Avtalad upphovsrätt*, p. 64.

[84] My translation of "verkshöjd".

EU does not apply the *threshold of originality* criterion but instead they refer to a work as *the author's own intellectual creation*, which is often called the *originality criterion*.[85] In the case Infopaq International A/S v Danske Dagblades Forening, CJEU determined that a work should be the *author's own intellectual creation* to acquire copyright protection under the Infosoc Directive.[86] In Painer, a case regarding the originality criterion for a portrait photograph under Directive 93/98/EEC, the court provided the originality criterion with more precise content stating that for the work to be the authors intellectual creation it should reflect the author's *personality* and express the author's *free and creative choices* in the production of the work, and thus stamp the work with the author's *personal touch*.[87] The criteria of that it should be the author's *free and creative choices* has been used in the case Football Dataco v. Yahoo! to determine the originality criteria in relation to databases.[88] It is important to note that these criteria was used regarding a photograph and a database, and that they have not been used to determine copyright protection in a case regarding computer programs, why it is uncertain and up for argumentation whether these criteria should apply in the protection of computer programs as well.

Another uncertainty also arises since it is not clear how the Swedish and European criteria for protection relate to each other. In the Directives art. 1(3), it is stated that no other criterion (than the originality criterion) should be applied in the assessment, why Swedish courts are not allowed to use the Swedish *threshold of originality* criteria when determining if a computer program is protected by copyright.[89] However as seen above the CJEU, in the Painer case, created further criteria to assess whether the originality criterion was fulfilled and Swedish courts have still used the threshold of originality to

---

[85] Art. 1(3) in Directive 2009/24/EC of the European Parliament and of the Council of 23 April 2009 on the legal protection of computer programs OJ L 111/16, art. 3(1) in Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the legal protection of databases OJ L 77/20, and art. 6 in Directive 2006/116/EC of the European Parliament and of the Council of 12 December 2006 on the term of protection of copyright and certain related rights OJ L 372/12.

[86] Case C-5/08 *Infopaq International A/S v Danske Dagblades Forening* [2009] para. 37.

[87] Case C-145/10 *Eva-Maria Painer v Standard VerlagsGmbH and Others* [2011] para. 85 - 99.

[88] Case C-604/10 *Football Dataco Ltd and Others v Yahoo! UK Ltd and Others* [2012].

[89] Several authors argue that the only change is that the national courts uses another language but that they still use the national criteria to determine the protection, see for example Gunnar W. G Karnell. European Originality: A Copyright Chimera. In *Intellectual Property and Information Law, Essays in Honour of Herman Cohen Jehoram,* Jan J.C. Kabel, Gerard J.H.M. Mom (eds.). Kluwer Law International, 1998.

determine if the originality criterion is fulfilled.[90] In Sweden a *double creation criteria* has been applied to decide whether a work has originality. The rule states that it should be close to precluded that a similar effort in creation should result in an identical form.[91] The rule can be explained as follows; if two identical forms arise independently from each other this should be considered a sign that the creations lack originality.[92]

#### 2.4.1.3 General common form[93]

A banal computer program cannot receive protection since simple creations does not reflect the originality of the author. A computer program is banal if it is just a *general common form* of a computer program since such simple code is not an expression of the author's creativity.[94] However even if a creation is banal it might receive protection if it is combined with other creations and the combination as such is a sign of individuality, see the Swedish Supreme Court in NJA 1990 s. 499, which seems to manifestation of the expression that the whole is greater than the some of its parts. If we apply this argument on computer programs, even simple and banal piece of source code might receive protection if it is combined with other snippets of source code in such a manner that the combination is a sign of individuality and thus worthy of copyright protection.

#### 2.4.1.4 Arguments from the US jurisdiction

According to Olsson, the principles developed in the US jurisdiction through case law has been internationally excepted as regulating whether a piece of software is protected by copyright or not.[95] Although US case law is not binding in Swedish courts, it might still affect the collective belief and thus the reification of copyright protection of software programs. I will therefore briefly present some principles used in the US jurisdiction.

---

[90] An example is Attunda District Court's argument in case T 2714-07.

[91] My translation of "*det får betraktas som nära nog uteslutet att resultatet av en likartad arbetsprestation skulle ha fått samma form*", see the argumentation of the Swedish Supreme Court in NJA 1995 s. 256.

[92] Agne Lindberg & Daniel Westman. *Praktisk IT-rätt.* 3rd Ed. Stockholm: Norstedts Juridik AB, 2004, p. 233.

[93] My translation of "allmänna formförrådet".

[94] Prop. 1992/93:48. *Om ändringar i de immaterialrättsliga lagarna med anledning av EES-avtalet m.m.* p. 109 and Michael Plogell. Upphovsrätt till datorprogram ur ett EG-perspektiv. *Juridisk Tidskrift vid Stockholms Universitet. (1993-94), nr. 1,* p. 62 ff.

[95] Henry Olsson. *Copyright : svensk och internationell upphovsrätt.* 9 Ed. Stockholm: Norstedts Juridik AB, 2015, p. 66.

#### 2.4.1.4.1 Scènes à faire doctrine

The scènes à faire doctrine has been established in the US jurisdiction through case law.[96] The doctrine denies copyright protection "to those expressions that are standard, stock, or common to a particular topic, or that necessarily follow from a common theme or setting".[97] When adjusted for the computer software context, the doctrine would deny copyright protection for parts of a computer program that are "dictated by practical realities – e.g., by hardware standards and mechanical specifications, software standards and compatibility requirements, computer manufacturer design standards, target industry practices, and standard computer programming practices".[98] The underlying logic of the scènes à faire doctrine is that if an expression becomes a common technique to express the idea that expression is not original, i.e. it would lack the criteria for copyright protection.[99]

#### 2.4.1.4.2 Merger doctrine

The merger doctrine is based on the idea/expression dichotomy. It states that if there is only one way to express an idea, the expression will merge with the idea itself, and since one can´t be granted copyright on an idea the creator won´t have copyright in that expression. It does not need to be physically impossible to make another expression of the idea; it is enough for the merger doctrine that there only exists one reasonable approach to express the functionality. According to the court in Computer Associates International, Inc. v. Altai, Inc, "*It follows that in order to determine whether the merger doctrine precludes copyright protection to an aspect of a program's structure that is so oriented, a court must inquire 'whether the use of this particular set of modules is necessary efficiently to implement that part of the program's process' being implemented*".[100] An example is the equation 1 + 2 = 3. Lets say you have one apple and two pears and you would like to add these together, the easiest way for you to express this would be the equation: 1 + 2 = 3. You could create an alternative expression with the same sum by writing 1 + (1 + 1) = 3 or 1 * 1 + 1 * 2 = 3. However the only reasonable approach to write this would be to write 1 + 2 = 3 since the other suggested equations are just inefficient ways to express the same idea.

---

[96] It was first mentioned in *Cain v. Universal Pictures, Co.*, 47 F. Supp. 1013 (S.D. Cal. 1942).

[97] *Gates Rubber Co. v. Bando Chem. Indus., Ltd.*, 9 F.3d 823 (10th Cir. 1993).

[98] *Lexmark International Inc. v. Static Control Components, Inc.*, 387 F.3d 522 (6th Cir. 2004).

[99] *Gates Rubber Co. v. Bando Chem. Indus., Ltd.*, 9 F.3d 823 (10th Cir. 1993).

[100] *Computer Associates International, Inc. v. Altai, Inc.*, 982 F.2d 693 (2d Cir. 1992).

The difficulty is how to ensure whether something is written in the one reasonable approach. For example there might exist more efficient ways to create a function but the parties lack knowledge of such function at the time of litigation or such more efficient function might be developed first after the litigation. It is not reasonable that the copyright protection will be dependant on whether someone has created, or will create, a more efficient way to solve a particular function. This results in that efficient computer program structures might not receive copyright protection while inefficient structures will.[101]

#### 2.4.1.4.3 Are these principles already applied in the Swedish jurisdiction?

It can be argued that both of these principles can be seen in the case NJA 1996 s. 79 and I will now present my opinion on this matter. In the case NJA 1996 s.79 the Court of Appeals mentions the following motivation under the assessment of copyright protection for the computer programs:

> "The investigation in the case shows that the programs are intended for word processing, presentation, graphic design etc. *These uses include great opportunities for different solutions in terms of program design*. As the District Court concludes all the programs include a large number of program instructions. Already those facts may indicate that the lower boundary for the threshold of originality is passed. Although the programs are not presented in operation in front of the Court the fact that there are several different programs for the same purpose but with *different structure* indicate sufficiently for the Court of Appeals, like for the District Court, to concluded that *none of the program designs are technically given*. The above stated does not mean that there cannot be great similarities between certain computer programs, but these should primarily be attributable to the function of the computer program when in use and this does not, in the Court of Appeal's perspective, result in lack of protection. With reference to the above stated the Court of Appeals concludes in an overall assessment of the programs that […] all

[101] Rick Sanders. Copyright Protection of APIs after Oracle v. Google: Poppin a Whelan, *IP breakdown*. 2012. Available at: [http://ipbreakdown.com/blog/copyright-protection-of-apis-after-oracle-v-google-poppin-a-whelan/] Accessed 2015-07-17.

> meet the criterion of the threshold of originality, and are thus subject to copyright protection [My translation and emphasis added]".[102]

As can be seen the Court of Appeal discuss several criteria that seem similar to the criteria used in the U.S jurisdiction. For example when the court discusses whether the *"uses include great opportunities for different solutions in terms of program design",* this resembles the criterion assessed under the merger doctrine regarding if there is only one reasonable approach to express the functionality. And when the Court concludes that "*none of the program designs are technically given",* this is similar to the criterion evaluated under the scènes à faire doctrine whether if the expression is dictated by mere practical realities, such as a standard computer programming practices. The Swedish Supreme Court does not mention these criteria explicitly in their verdict; they only state that the computer programs in question fulfil the criteria for copyright protection. However since the Supreme Court do not express a dissenting opinion regarding the motivation made by the Court of Appeals, these criteria, mentioned by the Court of Appeals, can be argued to be applicable in the Swedish jurisdiction in the assessment of copyright. On the other hand, one could also argue that since the Supreme Court does not explicitly mention these criteria, or that they concur with the motivation for protection given by the Court of Appeals, they use other criteria for protection that are not explicitly mentioned. From the legal constructivist approach I can only conclude the criteria would have been reified to a larger extent if the Supreme Court had communicated the criteria explicitly in their verdict, instead of only concluding that the computer programs are subject to copyright protection, which now opens up the normative space and uncertainty when determining which criteria to apply in the assessment of copyright protection.

### 2.4.2 The code

Computer programs are usually written in a human readable programming language, e.g. C++ or JAVA, and this is called the source code.[103] When the programmer has written a series of commands he or she uses a compilation software that translates the source code into object code, i.e. into a set of high or low voltages representing ones and zeros, which is machine-readable. This process creates an executable computer program that

---

[102] See the Court of Appeals motivation in NJA 1996 s. 79.

[103] Lindberg & Westman, p. 226. See also Mads Bryde Andersen. *IT-Retten.* Publisher: Author, 2001, chapter 3.2.

can be uploaded or downloaded to different storage devices such as an USB-stick or a hard drive. The source code can be deciphered from the object code by *decompiling* the program, however decompiling object code will only give a close resemblance of the original source code.[104]

The code in the software will acquire copyright protection as a computer program under Directive 2009/24/EC when it fulfils two criteria. Firstly it need to be an expression in any form of a computer program (including preparatory design material) and secondly a computer program shall be protected if it is original in the sense that it is the author's own intellectual creation.[105] According to art. 1(3) in Directive 2009/24/EC no other criterion should be applied in the assessment why Swedish courts are not allowed to use the Swedish *threshold of originality* criterion when determining if a computer program is protected by copyright. As a computer program it will be governed by some of the general rules of copyright and some special rules, i.e. the computer program will be protected for seventy years after the author has died however an anomaly is that the copyright will automatically be transferred to an employer under certain circumstances, in accordance with art. 40a in the Swedish Copyright act (SFS 1960:729), see section on employers right below.[106]

The Directive states that "Protection in accordance with this Directive shall apply to the expression in any form of a computer program" and both the source code and object code of a computer program has been seen as such protectable expressions under the directive by the CJEU.[107] According to the directive the protection extends further than the source and object code and includes the preparatory design material for the software. However the protection does not include the ideas and principles behind the computer program.[108] This relates to the general principle in copyright that *the protection extends to*

---

[104] Bing, p. 284.

[105] Art 1(2) – 1(3) Directive 2009/24/EC.

[106] Michael Plogell. Upphovsmannen, *Nordisk immateriellt rättsskydd*, NIR 1/99, (1999), p. 13.

[107] Art. 1(2) Directive 2009/24/EC.

[108] Art. 1(1-2) Directive 2009/24/EC, and Case C-393/09 *Bezpečnostní softwarová asociace* [2010]. Since the dispute arose in 2001 the courts decision makes references to the old Council Directive 91/250/EEC of 14 May 1991 on the legal protection of computer programs, however these references should still be valid under the amendments that has led up to the current legislation, Directive 2009/24/EC.

*expressions and not to ideas, procedures, methods of operation or mathematical concepts as such.*[109] In SAS Institute, Inc. v. World Programming Ltd. the CJEU withheld this principle stating that the functionality of a computer program is not subject to copyright protection.[110]

As seen under section 2.4.1 above, for the work to be the author's own intellectual creation it should reflect the author's *personality* and express the author's *free and creative choices* in the production of the work, and thus stamp the work with the author's *personal touch*. If we consider the above criteria, how do we determine what parts of a piece of software are protected as a computer program? I can be determined that on a scale, the abstract idea of the program is on one end, and the object code, i.e. the positive or negative charges, as a physical expression of the program is on the other. To illustrate the problem and make my argument I will use simple examples of source code. For the first example of C++ source code see *Source code 1* below.

```
#include <iostream>

int main()
{
        std::cout << "Hello thesis reader!";
}
```

Source code 1

This piece of source code will generate a software program that writes "Hello thesis reader!" on the screen. The purpose of the software program when I designed it was to communicate a sentence to the user of the software. According to the Directive the first criteria is that the element must constitute an expression, in any form, of a computer program. Since this is source code it fulfils this first criteria. However the underlying idea, to communicate a sentence to the user, is not protected. As said, the second criteria is that it should be the authors' own intellectual creation. Even if it is a simple computer program it is my own intellectual creation; I wanted the computer to write a sentence on the screen and this expression fulfils this purpose. However if I would apply the criteria from Painer it is not as certain that the originality criteria is fulfilled. First, it is difficult to

[109] Art. 2 WIPO Copyright Treaty.

[110] Case C-406/10 *SAS Institute Inc. v World Programming Ltd.* [2012], para. 39.

determine if the source code reflects my *personality*. The example only comprises three lines of effective code why it does not provide a lot of information to determine if it reflects my personality. Second, how can we determine if I have been able to express my creative abilities in the production of the work by making free and creative choices? I could for example prove this by developing the same function with another piece of source code, see *Source code 2* below.

```
#include <iostream>

int main()
{
  int number = 1;

  while (number == 1)
  {
  std::cout << "Hello thesis reader!";
  number++;
  }
}
```

Source code 2

This second example will also generate a software program that writes "Hello thesis reader!" on the screen, however as can be seen I have added lines of code. The added lines of code make this piece of software less memory efficient than *Source code 1* above. This might prove that I have made *free and creative choices* since I was able to choose between several alternative ways to express my idea already in *Source code 1*, however I would never use this second example since it is a less efficient and a less readable way to produce the same function. Since these are general values among the software developer community it is likely that two programmers would construct this function in an identical way independently from each other, i.e. in the simplest way which would be *Source code 1*.[111] This last example does also prove what is complicated with the criterion *personal touch*

[111] On the Internet there are challenges were instead of writing simple code, developers try to write the most complex code possible to solve a simple problem just for fun, however writing complex code just because you can is not commonly done when developing commercial software. For such a challenge see celtschk. Most complex "Hello world" program you can justify, *StackExchange*. 2012 (edited by daniero

in relation to computer programs. If we were able to find identical code snippets by other software developers, this would be an indication that the code lack personal touch.

As stated in section 2.2 above, one of the reasons to protect computer programs with copyright was to increase the incentives to invest in the development of quality software. The criteria for determining copyright protection like the double creation criteria, criteria on personal touch and free and creative choices, might suit to incentivise the creation of books and paintings but I find them problematic in relation to incentives in the creation of software since when developing code, the software developer is bound to certain expressions and certain structures. The incentive for a developer is often to develop code in the most efficient way possible, in terms of usage of computer memory or to promote readability, comprehensibility and maintainability. These interest are also recognised in the preparatory works were it is stated that "It is also evident that a computer program is an expression of human intellectual creativity, for which it requires a high degree of knowledge, capacity for logical thinking and intuition in order to *in the simplest* and *most effective manner* reach *the best result* [my translation and emphasis added]".[112] Such incentives makes the actual development of code less free and creative from a classic copyright perspective and it increases the chances of finding two pieces of source code that are identical even if they are developed independently from each other.[113]

### 2.4.3 The compilations of data

If the software contains a compilation of data this can be protected under several copyright categories. Depending on the category of copyright protection the normative experiences, i.e. consequences, will be different. First, each piece of data in the compilation can be protected under copyright as a literary work in accordance with art. 1 in the Swedish Copyright act (SFS 1960:729), if it is determined to be a result of

---

2014). Available at [http://codegolf.stackexchange.com/questions/4838/most-complex-hello-world-program-you-can-justify] Accessed 2015-07-17.

[112] Prop. 1988/89:85. *Om upphovsrätt och datorer,* pp. 10 – 11.

[113] This problem has been recognized in the US jurisdiction for example in the case *Computer Associates International, Inc. v. Altai, Inc., 982 F.2d 693 (2d Cir. 1992),* at para. 708 were the court states: "*Efficiency is an industry-wide goal. Since, as we have already noted, there may be only a limited number of efficient implementations for any given program task, it is quite possible that multiple programmers, working independently, will design the identical method employed in the allegedly infringed work".*

individual intellectual creation and has originality.[114] Secondly the compilation in itself can be a protected as a work under art. 5 in the Swedish Copyright act (SFS 1960:729) if it is deemed to be a *literary or artistic compilation* and is a result of individual intellectual creation and has originality. As a compilation of data in these categories it will be governed by the general rules of copyright, i.e. the data/compilation of data will be protected for seventy years after the author has died and it will not be seen as a computer program and thus the rights will not automatically be transferred to the employer under the circumstances mentioned in art. 40a in the Swedish Copyright act (SFS 1960:729).[115]

Thirdly, if the compilation of data does not fulfil the criterion for protection in accordance with art. 1 in the Swedish Copyright act (SFS 1960:729) it can be protected as a database under the sui generis protection in art. 49 in the Swedish Copyright act (SFS 1960:729), if the compilation of data is deemed to be a collection of a large quantity of data or if the compilation of data is a result of a substantial investment. The substantial investment should relate to the gathering and auditing of the data.[116] As a database under the sui generis protection the compilation of data will be protected for fifteen years after it has been finished and as with the first example, it will not be seen as a computer program and thus the rights will not automatically be transferred to the employer in accordance with art. 40a in the Swedish Copyright act (SFS 1960:729).

Fourthly, I argue that the compilation of data can be protected as a computer program in accordance with art. 1 in the Swedish Copyright act (SFS 1960:729).[117] This would be possible if the compilation of data itself is a set of source code commands. In such situation there would be demarcation difficulties between the two copyright categories. The database directive mentions a demarcation. According to the preamble in the database directive *the term 'database` should not be taken to extend to computer programs used in the making or operation of a database.*[118] Plogell argues that the logic in the preamble should also apply where the making or operation relate to a part of the database and that those types of software should be governed by the Directive 91/250/EEC, now Directive

---

[114] Olsson, 2015, p. 318 ff.

[115] Plogell, 1999, p. 13.

[116] Olsson, 2015, p. 318 ff.

[117] I have not found any case where this construction is tried by a court.

[118] Section 23 in the preamble of Directive 96/9/EC.

2009/24/EC, on the legal protection of computer programs.[119] I cannot conclude if he means that the compilation of data itself would be governed as a computer program or if it should be governed under the database directive. This is a problematic issue, even if a software program often appears to be one object, one file on the hard drive, from the user perspective, this is seldom the case. The compilation of data that is being used by the computer program might be located in a separate file from the computer program itself, for example in an excel file, which is a suitable solution if the user wants to extract the data. The compilation of data might also be located "within" the software program. If the category of protection for the compilation of data would be determined by the usage of the database this could cause a situation where, as long as the database is in used internally by the computer program it would be protected as a computer program under Directive 2009/24/EC but when the database is extracted (or copied) and no longer in use by the computer program it would be protected as a literary work or under the sui generis database protection. Since the categorization would be dependent on the usage of the database such solution would not be satisfactory.

### 2.4.4 The GUI

The graphical user interface is part of the visual aspects of the software program, it is what the user will see on the screen while using the software.[120] An example of a GUI is the iPhone OS with its icons that functions as button to enable the user to communicate with the iPhone. An issue regarding the protection of the GUI is whether the GUI is an expression of the computer program and thus protected in such category or if it should be protected by the general rules of copyright as an artistic work. In NJA 2000 s. 580, the Swedish Supreme Court made an indication that the GUI can be seen as an outflow of the computer program.[121] The base for the courts argument is a reference to Henry Olsson where he has states that "*there are good reasons to consider the graphical user interface as an expression of the underlying computer program and thus included in the protection of computer programs* [my translation]*"*.[122] This is a central issue since if the GUI would be protected

---

[119] Plogell, 1999, p. 11.

[120] Rachel Stigler. Ooey GUI: The Messy Protection of Graphical User Interfaces, *Northwestern Journal of Technology and Intellectual Property*. Vol. 12, No. 3, (2014).

[121] They stated that "*the images that appears on the screen should rather be seen as an outflow of the underlying computer program* [my translation]", see the Swedish Supreme Court in NJA 2000 s. 580 and Lindberg & Westman, p. 232, for further discussions.

[122] Henry Olsson. *Copyright : svensk och internationell upphovsrätt*. 6 Ed. Stockholm: Norstedts Juridik AB, 1998.

under the category of computer programs, the rights would automatically be transferred to the employer under the circumstances mentioned in art. 40a in the Swedish Copyright act (SFS 1960:729).

In contrast, in the case *Bezpečnostní softwarová asociace,* the CJEU tried whether a GUI would be protected as a computer program under the directive.[123] The court concluded that the GUI was not protected under the directive as an expression of a computer program. They concluded that the GUI is an interaction interface that enables communication between the user and the computer and that the GUI does not enable a reproduction of the computer program but only constitutes an element of the program that enables users to make use of the features of that program.[124]

However the court determined that the GUI could receive protection under the Infosoc Directive 2001/29/EC if it would be the author's own intellectual creation, and the court left the assessment whether a GUI would fulfil the criteria for protection for the national courts to decide.[125] The GUI may therefore be protected under the general rules of copyright if it fulfils the basic criteria of copyright protection under art. 1 in the Swedish Copyright act (SFS 1960:729). If the GUI would be protected under the general rules of copyright as an artistic work, the rights would not be transferred to the employer under the circumstances mentioned in art. 40a in the Swedish Copyright act (SFS 1960:729).

## 2.5 Transacting copyright

In the Swedish Copyright Act (SFS 1960:729), authorship is determined by who created the work. The general rule is that the author is also the right holder, see art. 1 in the Swedish Copyright act (SFS 1960:729), which states that "*Anyone who has created…has copyright in that work*". According to art. 27 in the Swedish Copyright act (SFS 1960:729) the exclusive rights can be transacted fully or partially (with the exception of the moral rights which can only be agreed by the author not the be enforced against the counterparty). If the exclusive rights are fully transacted to the buyer, the buyer becomes

[123] Since the dispute arose in 2001 the courts decision makes references to the old Council Directive 91/250/EEC of 14 May 1991 on the legal protection of computer programs, however these references should still be valid under the amendments that has led up to the current legislation, Directive 2009/24/EC.

[124] Case C-393/09 *Bezpečnostní softwarová asociace* [2010].

[125] Ibid., para. 46 - 47.

the new right holder and can choose to use the exclusive rights to exclude others from using the work in a manner that would infringe the exclusive rights. The parties can also choose to only partially transact the exclusive rights. For example, the right holder can agree to not enforce their copyright against the counterpart if the counterpart makes copies of the work; such limited transaction of the rights is often called a *license*.[126]

If an agreement regarding a transaction of copyright is vague, the agreement should be interpreted in favour of the author in accordance with the *principle of specification*.[127] This principle states that if a right is not explicitly mentioned to be transacted in the agreement it should be interpreted that the right is not transferred in the transaction but stays with the author.[128] Another important regulation is that it is specifically stated in art. 27 in the Swedish copyright act (SFS 1960:729), that a transaction of an expression of a copyright protected work does not include a transaction of the underlying exclusive rights, e.g. the buyer of a painting is not allowed to replicate and sell copies of that painting if not explicitly agreed upon. This regulation is somewhat problematic when applied to computer software, because a transaction of software often needs to include both an expression of the copyright protected work, i.e. the software program, and a right for the buyer to make copies of that same work, since copies are made each time the computer software is used in a computer.

The Swedish Copyright legislation only recognizes physical persons as creators of a work, however judicial entities can be right holders of copyright after a transaction of the right has occurred.[129] When an employee creates a work it is common that the employer (the judicial entity) becomes the right holder through transfer of the copyright.

### 2.5.1 Employers right

When an employee creates a computer program as a part of his or her tasks, the copyright in that computer program is entirely[130] and automatically transferred to the employer according to art. 40a in the Swedish Copyright act (SFS 1960:729). The article

---

[126] Olsson, 2015, p. 238.

[127] My translation of ”specifikationsprincipen”.

[128] See Olsson, 2015, p. 237.

[129] SOU 2010:24. *Avtalad upphovsrätt*, p. 67.

[130] Both the economic right and moral right are transferred. There has been a large debate that the moral right should not be transferred automatically, see Jan Rosén in SOU 2010:24. *Avtalad upphovsrätt*.

explicitly states that it is the copyright to a *computer program* that is transacted from the employee to the employer, and when reading the preparatory works for the article it is stated that this article should only govern copyright in *computer programs* and other categories of copyrighted works are excluded from this transfer.[131] This creates difficulties since a computer program is defined in the preparatory works as *a series of instructions to a computer,* this means that all other elements in a software program that are not a series of instructions to a computer, but still protected under other categories of copyright won´t be governed by the transfer of rights stated in art. 40a in the Swedish Copyright act (SFS 1960:729).[132] This is why it is important to differ between "software" and "computer program". Other protectable elements in a software program such as GUIs and databases would therefore need to be transferred by other means.

One solution would be to argue that these elements are transferred in accordance with the *"rule of thumb"*.[133] One of the descriptions of this principle is: "*The employer may, within its operational area and its normal activities, exploit those works that are created as a result of official duties against the employer. The employer's right refers uses for the purposes that can be anticipated when the work is created. To the extent that the work must be modified to fulfil the purpose of its creation, the modification is allowed"*.[134] However I don´t regard this as a satisfactory solution. The rule of thumb is not as certain as art. 40a in the Swedish Copyright act (SFS 1960:729) when it comes to determining transfer of rights between employee and employer since it is an uncodified principle and differences between the two are numerous. First, when the copyright to the source code will be fully transacted under art. 40a in the Swedish Copyright act (SFS 1960:729), it is uncertain whether the copyright to the GUI will be a full transfer (both economic and moral right), limited transfer (economic rights only), or a license (the developer keeps the ownership, while the employer has a right to use) under the rule of thumb. This uncertainty creates several sub difficulties. If the transfer is only a license, is the license an exclusive or non-exclusive license, i.e. is the employee

---

131 Prop. 1992/93:48. *Om ändringar i de immaterialrättsliga lagarna med anledning av EES-avtalet m.m.* p. 118 ff.

132 Prop. 1988/89:85. *Om upphovsrätt och datorer*, p. 7 ff. In the preparatory work they expressively differ between a series of instructions to a computer, i.e. a computer program, and a compilation of data, i.e. a work that may be protected under the protection of *catalogues*.

133 See SOU 2010:24. *Avtalad upphovsrätt* p. 144, to find a summary opinions regarding *the rule of thumb principle.*

134 See Bernitz et al. *Immaterialrätt och otillbörlig konkurrens*. 10 Ed. Stockholm: Jure, 2007, p. 94.

allowed to license the GUI to someone else? Secondly, since the copyright for the source code is fully transferred to the employer, the employer is allowed chose how to act with the source code, e.g. modify, sell, license or divest it etc. When it comes to elements like the GUI, transferred under the rule of thumb, the employer is only allowed to use it in their *operational area and its normal activities*, why the employer might not be able to sublicense the GUI if it is not in their normal activity to sublicense software. And under the rule of thumb the GUI can only be *modified to fulfil the purpose of its creation*, which means that the employer would not be allowed to reuse and modify the GUI to be used in other software programs if that was not the purpose when the employee created the GUI. Since a software program may contain several elements, some subject to the rule of thumb and some subject to art. 40a in the Swedish Copyright act (SFS 1960:729), this demarcation creates an uncertainty in employee to employer relationships as to which rights are transferred and to what extent are they transferred.

## 2.6 The infringement

This section concerns the scope of protection for a work, i.e. how similar to an original work can a new work be without infringing the copyright of the original work? According to art. 54 in the Swedish Copyright act (SFS 1960:729) a person infringes the copyright of a work if he or she uses the protected work, without permission, in a manner that is infringing in the exclusive rights in chapter. 1 and 2 in the Swedish Copyright act (SFS 1960:729), granted to the right holder, i.e. the right to reproduce, distribute, display or perform the protected work, or to make derivative works.[135]

### 2.6.1 The scope of protection

The exclusive right to reproduce the work granted to the right holder is generally wide. According to art. 2 in the Swedish Copyright act (SFS 1960:729), the right to reproduce the work includes any direct or indirect, temporary or permanent reproduction of the work, regardless in what form or by what method it takes place and whether in whole or in part. This means that even if someone only makes an indirect, temporary copy of a protected work, he or she will make an infringement. Thus it is possible to conclude that if someone copies a whole piece of source code, which fulfil the criteria for copyright protection, from a software program and paste the source code into his own software, without permission he commits an infringement in the copyright of the original work.

[135] Olsson, 2015, p. 256.

The same logic will apply if someone prints the same source code on a piece of paper.[136] In both situations a reproduction of the source code is made. Even if someone does not make a pure copy of a work, but instead modifies it, this new work might be deemed to be a derivate work and thus infringing the original work, see art. 4 in the Swedish Copyright act (SFS 1960:729). In such case, the permission by the right holder of the original work is necessary to be able to exploit the new derivative work. However if the modification is substantial enough, the new work can be considered to have been created in free association to the original work and thus will not be dependent on the original work, see art. 4 (2) in the Swedish Copyright act (SFS 1960:729).

To summarize we have three categories that a software developer might fall into when inspired by another work:

1. The new creation is a mere copy of another original work and thus requires the permission of the original right holder to be reproduced.
2. The new creation is a protected work in itself but it is dependent on another original work and thus requires the permission of the original right holder to be reproduced.
3. The new creation is a work independent from other works.

It can be difficult to assess in which of these categories a work inspired by another work should fall in, especially regarding a software program. Such assessment is dependent on the scope of protection for the specific element in question. In doctrine it is said that a very original work will have a broader scope of protection than a work with lower levels of originality and this also regards works of applied art.[137] This logic can thus be applied on industrial copyrights such as copyright protected software. For example Westman & Lindberg argues that software programs that are in certain standardised categories, for example text editors, have a narrow scope of protection, while other categories of software like games have a broader scope of protection.[138]

---

[136] Lindberg & Westman, p. 251.

[137] Levin, p. 188.

[138] Lindberg & Westman, p. 252. They must refer to the GUI, since even if most text editor look alike they might contain different source code structures.

This difficulty to assess the scope of protection is also intertwined with the difficulties arising from the idea/expression dichotomy regarding copyright protection for software.[139] For example, one can without difficulty differ between the idea of painting a screaming person by a fence and the expression of the painting "The Scream" made by Edvard Munch, however when it comes to software programs it is more difficult to differ between the idea of a certain function in the programming language C++ and the expression of the same function in source code. The problem can be illustrated by a comparative example:

If a teacher explains to her students how to draw a house using a two-point perspective (the idea), each of the students' own drawings (the expressions) would most likely be determined to be their own independent works since the idea of using a two-point perspective when painting a house is given by nature and is thus not copyright protected. However if a teacher explains an efficient structure or function in the computer language C++ (the idea), each of the students' application of said structure or function is prone to be so similar to the teacher's example that this can be deemed to be a verbatim copy or only a modification of the teacher's own example, i.e. a reproduction or a derivative work. In the case with the C++ source code it is difficult to determine the demarcation between the idea (not protectable) and the expression (protectable).

### 2.6.2 Subjective novelty and infringement

It is important to note that copyright does not protect *priority*, why the infringer must have been aware of the earlier work to be able to infringe in the copyright of such work.[140] This means that a situation might occur where two pieces of copyright protected source code is exactly alike, and this does not necessarily mean that a reproduction has been made.[141] Two pieces of similar source code might occur since there are only a limited number of ways to express a function in source code, due to the constraints while developing code, i.e. standardised language, efficiency and readability aims etc. which limits the freedom of the software developer compared to a writer of readable text who has a larger freedom due to less constraints. If both software developers have developed the code without being aware of the other developer's code, the criterion of subjective

[139] See Olsson, 2015, p. 50, and Levin, p. 179, for further references.

[140] Levin, p. 176.

[141] At least on a function or module level.

novelty is fulfilled why both can be granted copyright protection (if their works also fulfil the originality criterion) without infringing the other developer's work.[142]

In the reality, this is often a matter of evidence. Which party has the burden of proof for a certain fact and can they show such proof in court? The Swedish Supreme Court constructed a burden of proof principle for copyright infringement in case NJA 1994 s. 74 (Smultron). The court stipulated that "*The one who claims that there is an infringement of copyright in a work has the burden of proof that a reproduction or modification has occurred. A great resemblance to the original work, can in itself constitute a strong indicator that a reproduction or modification has been done. If this is the case the burden of proof is transferred to the defendant who has to prove that his work was created independently, without connection to the original work* [my translation]".[143] According to Levin, one has to do an overall assessment to determine the scope of protection in a particular situation.[144] The scope of protection should be wide enough to fulfil the aim of copyright, and still not hinder others fair use of expressions.

### 2.6.3 Intent

In accordance with art. 53 in the Swedish Copyright act (SFS 1960:729) one commit a copyright infringement if the infringing act is committed wilfully or by gross negligence. It can be argued to be gross negligent to copy a computer program that is highly similar or exactly alike to another computer program without investigating it further, however this has not been subject to a judgment in a Swedish court. According to art. 54 para. 1. in the Swedish Copyright act (SFS 1960:729), a person infringing in a copyright protected work has a strict liability to pay a reasonable remuneration to the right holder for the use, and additional damages for losses should be paid to the right holder by the infringer if the infringement was conducted with wilful or negligent intent, according to art 54. para 2. in the Swedish Copyright act (SFS 1960:729).

In NJA 1995 s. 164 the Swedish Supreme Court discussed the criteria for the assessment of *negligence* and the responsibility for a *vendor* to investigate whether ordered tunics committed copyright infringement when resold. The court specially discussed whether (1.) the defendant could have determined that the tunics was protected by copyright and

---

[142] Levin, pp. 80 - 82.

[143] See the Swedish Supreme Court in NJA 1994 s. 74.

[144] Levin, p. 188.

(2.) if the defendant should have had reasons to suspect that the tunics infringed in a copyright protected work. Regarding the first issue the court concluded that the tunic was designed in such a way that the defendant should have suspected that it was protected by copyright. When considering the second issue the majority of the court (three judges out of five) concluded that it was not negligent by the defendant to not have been aware of the publication of the original work because the publication were considered to be target mainly towards consumers. However the minority held that even if it cannot be required by the defendant that they should have been aware of the publication of the earlier work, they had an obligation to request information about the origin of the design. Since the defendant had failed to conduct such investigation they had acted negligently.

This case can give some notion on how to assess negligence of the buyer in software transactions. Regarding the first assessment, under an analogy to the case, the buyer would normally be able to assume that the software contains elements that are protected by copyright, even if he or she won´t be able to perceive other elements than the GUI. However regarding the second issue one can either choose to argue in line with the majority, that the buyer should not be expected to have knowledge about possible works which the elements might infringe in (other than the GUI), which would be natural since the buyer cannot determine what elements in the software are protected since he or she cannot perceive such elements, or one can argue in line with the minority that the buyer has an obligation to investigate the origin of the elements in the software to ensure that these are not infringing in other works. If the buyer has failed such investigation they would be negligent and liable to pay damages.

It should be noted that it could be argued that the case is not applicable on software transactions at all since there are several differences. In the NJA 1995 s.164 the court discusses the responsibility of a buyer as *vendor* not a buyer as *end-user*, and there are also a difference of transacted objects, for example the vendor could perceive the copyright protected expression, the jacket, while an end-user won´t normally be able to perceive the copyright protected expression, the source/object code, after compilation.

### 2.6.4 Discussion

As said above, a person won´t infringe another work if they are not aware of that work. However it is still difficult to determine when someone is *aware* of an earlier work.

According to Levin, it is enough that a person has made an *unconscious use* of an earlier work to be infringing in such work, and the infringer does not have to know that the earlier work is copyright protected.[145] This might be a reasonable policy for classic works of art and literature, but regarding an industrial copyright, such as computer programs, this can raise several issues. For example, since developing code has to be learnt by studying source code, how does one handle inspiration from study materials as prior works when it comes to copyright infringement? Is every source code that developer has seen during their education an earlier work that they risk to unconsciously infringe if they use what they have learnt or should these study materials be part of the *general common form* or the *scènes à faire* doctrine and thus will not be protected?

As said above, a situation that might arise is that a software developer is granted copyright protection to a certain source code because he or she is inexperienced and has not seen a lot of source code while another software developer who is more experienced and seen several pieces of source code is not allowed to use the same source code since it would constitute a copyright infringement in such original work. We can also construct a scenario where a software developer, who specializes in a certain unique source code related to random number generators, changes employer. The question is, since the copyright is transferred in the employment situation from the employee to the first employer regarding the source code under the circumstances stipulated in art. 40 in the Swedish Copyright act (SFS 1960:729), will the old employer be able to sue the employee if he or she uses similar source code for random number generators in the new employment? If source code would receive a broad protection from infringement this could the case regarding highly specialized software developers. If this would be the case the copyright legislation would have similar effect as a competition clause in an employment agreement, hindering the employee to use their knowledge in a competing company to the original employer. However while a competition clause is only allowed to hinder competition for a couple of years or else it would be invalidated, the copyright legislation can, at least in theory, hinder an employee from using their knowledge for as long as the source code has copyright protection, i.e. the rest of the employee's life. And while a competition clause is explicitly stated in an agreement, the copyright legislation

[145] Levin, p. 189.

would not be as explicit and would have to be interpreted by the employee. Because of this it is reasonable with a narrow protection of computer programs.

In NJA 2009 s. 159 (Mini Maglite), the Swedish Supreme Court concluded that a designed flashlight could receive copyright protection as a work of applied art. They concluded that the Mini Maglite lamp design rises above an only banal or trivial creation and does not only represent the result of a routine task. They also conclude that it seemed likely that another person in a similar situation would have given the flashlight a somewhat different design and that the overall impression is that the flashlight is more than a mere functional item and that it, by its design, have acquired its own identity. It is therefore protected by copyright, however with a *narrow scope of protection*.[146]

This argument can be applied on software programs as well. For example if a source code, GUI or other element is deemed to be protected by copyright, the scope of protection will be determined by the originality of that work. However, and this especially applies for the source code, since the development of software programs is hindered by several constraints, the scope of protection should be narrow and basically only hindering verbatim copying. Such solution would allow a protection for software under copyright and still permit the software developer to use their prior knowledge and even apply some source code reuse, as long as such reuse include that they modify the code to some extent.

### 2.6.5 Conclusion

A problem that I can observe is that while the criteria for protection, e.g. *the authors own intellectual creation,* the sub-criteria *the authors personality*, *free and creative choices* and *personal touch*, might be suitable for protecting original pieces of art, they are not as appropriate for protecting investments made in software development. It is reified in the judicial arena that a copy of a certain piece of source code, which is later pasted into a commercial software program, is a reproduction of such source code. If the source code is protected by copyright and the right holder has not given the permission for such copying, this will constitute an infringement. However it is not reified how the prior knowledge of a software developer will affect if they are aware of prior works or not and in what ways changing employer would hinder a software developer from using their

[146] Levin, p. 188.

own knowledge in the field of developing source code. It is likely that the elements in a computer software has a narrow scope of protection due to said constraints and that a wider scope of protection might only be granted if an element shows great signs of creativity where several different designs could be applied without any real hindrance of said constraints.

To summarise, the criteria to acquire copyright protection to an element in piece of software, is set to low threshold. It is enough that the element is the *author´s own intellectual creation*. Even if it is a simple creation it can acquire copyright protection, however the width of the protection will be narrow. Difficulties in assessments will arise since some elements of software can be argued to lack originality, why it does not receive copyright protection. The arguments that can be used to claim a lack of originality is that the element is either:

- Too simple for protection, which might be decided by the double creation criterion. However this criterion is not well suited for software since the specific program language and certain structures limits the alternative expressions for a developer to consider when developing the program why it is common that relatively advanced source codes might be somewhat similar.
- A part of the *general common form* and thus it will not be protected. This can be argued to be similar to the *scènes à faire* doctrine, where standard expressions do not acquire copyright protection.
- The only possible way of solving a problem/function, since the idea and the expression would merge, the element would not have originality why it would not be protected with copyright, according to the *merger doctrine*. It can be argued that the criteria that the work should express the author's *free and creative choices* relates to the merger doctrine since if there is only one available solution the author has not had the ability to do free and creative choices.

Since the width of the protection is determined on a case by case basis it is difficult to determine in advance if reused and modified source code should be seen as an own free expression or as a derivate work to the original source code. It is also uncertain whether the overall structure and order of the source code is protected in the Swedish

jurisdiction, and thus it is uncertain where the boundary between legitimate copying of elements such as source code and illegitimate copying of said source code is drawn.[147]

Before moving on to the next topic I will provide the reader with an example of how the end-user might commit an infringement in a protected software program.

### 2.6.6 An example of an infringement by the end user

A simple example of how an end-user might conduct copyright infringement is seen in the following example. A company is developing source code for internal uses, and a software developer is hired as a consultant to produce a function in that said source code. The consultant delivers the source code, which can be argued to fulfil the criteria to be protected as a work by copyright, but the exclusive rights are never transferred from the consultant to the company, why the consultant is still the right holder. The company uses the source code and implements it in their software for internal use. Later, when they are developing a new software product for external uses, they realize the potential in the source code developed by the consultant, why they reuse that code again in their new software. The company is now committing copyright infringement in the consultant exclusive rights, since they are producing unauthorised copies of a protected work. This software program is later licensed to a customer who downloads the software program from a homepage, i.e. the customer makes an unauthorised copy of a protected work, but when the downloading occurs the customer is still in good faith and is not committing the crime with intent.

But if the right holder, the consultant, realizes that his code has been reproduced without permission and sends a warning letter to the customer explaining that the customer has committed copyright infringement this changes. The customer is now in a position were they have information stating they are committing an infringement. In this situation the customer will have a difficulty to verify if this is true or not, since they have not been part of the development process. If they continue to use the software, they would make reproductions of the work when it is loaded into the RAM-memory, and because of the information in the warning letter they could be deemed to do it wilfully, or by gross negligence. Because of this, the end-user would be committing a copyright infringement.

---

[147] According to Lindberg & Westman, the overall structure would *probably* have protection in Sweden, see Lindberg & Westman, p. 230.

## 2.7 Modern market behaviours

After reading the last example in section 2.6, one might ask why the original right holder would send warning letters to the end-user rather than sue the software developer who made the initial reproduction of the protected work? The answer to this is that it can be more profitable for the right holder to claim licensing fees from a party further down in the value chain than risk litigation with a manufacturer. For example, if the right holder would sue the software developer for infringement, this might lead to one larger lawsuit where the right holder risks losing the litigation. In such situation it is not even certain that the software developer has the funds to pay damages if they are found to have infringed the original work. However if the right holder would send warning letters to end-users and claim licensing fees of an amount that is not large enough the provoke a litigation, the end-user might pay the licensing fee just to make the claim "disappear". If the right holder sends warning letters to enough end-users they can make a larger profit by choosing the second alternative compared to the first. From a cash-flow perspective it is also more viable to receive licensing fees closer in time than risk waiting several years to have a verdict in court before receiving revenue streams. This type of business model provokes a lot of bad will, but is generally done by non-practising entities (NPEs) or "trolls".

A "Patent troll" or non-practising entity is a now a common term especially within the technology intensive markets, and NPE-litigation has increased an average of 22% per year since 2004.[148] The definition of a patent troll differs but generally these can be defined as actors that do not generate value by production or innovation but rather by enforcing rights to receive revenue streams.[149] These behaviours have also moved into the field of copyright.

### 2.7.1 Illustration 3: *The case of SCO Group*

The story of Caldera International or *SCO Group* resembles a story of copyright trolls. Caldera International or *SCO Group* are two names of an American software company founded in 1994 that developed Linux and DOS-based operating system ("OS")

---

[148] PatentFreedom. Litigations over time, PatentFreedom. 2014. Available at: [https://www.patentfreedom.com/about-npes/litigations/] Accessed 2015-07-17.

[149] See Gene Quinn. In Search Of a Definition for the term "Patent Troll", *IPWatchdog*. 2010. Available at: [http://www.ipwatchdog.com/2010/07/18/definition-patent-troll/id=11700/] Accessed 2015-07-17.

products. In 2001 SCO Group purchased software rights to the operating system UNIX from Santa Cruz Operations, software IP-rights that originated from Unix System Laboratories (USL), a division at AT&T. USL had sold their all their rights, including copyrights, trademark rights and license agreements to Novell. Novell later sold some of these assets to Santa Cruz Operations who resold them to SCO Group.[150] Through this chain of transactions SCO Group claimed that they owned the rights to the operating system UNIX, including the copyright to the source code.

In addition to the ownership of UNIX, SCO Group claimed that the OS Linux contained pieces of source code from UNIX and thus infringed SCO Group´s intellectual property rights.[151] Because of this they sent cease- and desist letters to approximately 1.500 corporates claiming licensing fees based on the claim that the companies, when using Linux, infringed SCO Groups intellectual property rights in UNIX. This aggressive behaviour of SCO Group started several lawsuits, some of which are still on going.[152] A major question in these lawsuits was whether the original UNIX software was even protected by copyright at all. The costs incurred on the involved parties due to SCO Groups behaviour have been substantial.

### 2.7.2 Illustration 4: *The case of Righthaven LLC*

Another example is the company Righthaven LLC, a company that did not create, produce or distribute any goods or services. In contrast to SCO Group, who used claims of copyright protected software, Righthaven used claims of copyright protected articles. In 2010 the Las Vegas based company acquired several copyrights to articles in the Las Vegas Review-Journal and The Denver Post. They used these rights to sue a number of persons, who were publishing material on websites, for copyright infringement claiming

---

[150] Groklaw. Want to See One of the Letters to the Fortune 1500?, *Groklaw*. 2003. Available at: [http://www.groklaw.net/article.php?story=20031127100124265] Accessed 2015-07-17.

[151] SCOGroup. SCO Registers UNIX® Copyrights and Offers UNIX License, *SCO Group*. 2003. Available at: [http://web.archive.org/web/20100102232443/http://ir.sco.com/releasedetail.cfm?ReleaseID=114170] Accessed 2015-07-17.

[152]Lee Hutchinson. It's back: District court judge revives SCO v IBM, *arstechnica*. 2013. Available at: [http://arstechnica.com/tech-policy/2013/06/its-back-district-court-judge-revives-sco-v-ibm/] Accessed 2015-07-17.

$75.000 in damages from each one.[153] The defendants had allegedly used the articles without permission from the author, which was seen as an opportunity by Righthaven LLC to claim licensing fees. In the end Righthaven LLC was ultimately unsuccessful in their litigation, firstly due to judges who ruled that they had not been assigned the copyright of the articles (only the right to *sue* for infringement) why they didn´t have a standing to sue, and secondly even if they would have a standing to sue, the use was covered by the fair use exception.[154] In 2011 Righthaven LLC went into receivership due to unpaid legal fees.[155]

## 2.8 Conclusion

*How does uncertainty emerge in the intersection of copyright and development of software?*

As I have shown, the normative is wide space when arguing about whether a piece of software is protected by copyright. In the preparatory works it is stated that it is of importance that the protection of computer programs is comprehensible, since if the protection is not comprehensible the uncertainty may hinder innovation and impede importation of software.[156] As shown in this chapter, the protection of computer programs is far from comprehensible. The modern copyright system is able to handle verbatim copying of a whole software program, but the normative space and uncertainty is increased as I analyse copying of smaller excerpts of source code. I have shown in this chapter that the protection of software contains several uncertainties, either because important aspects of the software might not be protected or since it is uncertain how to interpretate the width of the protection in a infringement or because the right holder to

---

[153] Steve Green. Legal attack dog sicked on websites accused of violating R-J copyrights, *Las Vegas Sun*. 2010. Available at: [http://www.lasvegassun.com/news/2010/aug/04/unlikely-targets-emerging-war-media-content/] Accessed 2015-07-17.

[154] Nate Anderson. Copyright troll Righthaven finally, completely dead, *arstechnica*. 2013. Available at: [http://arstechnica.com/tech-policy/2013/05/copyright-troll-righthaven-finally-completely-dead/] Accessed 2015-07-17, and *Righthaven LLC v. Hoehn*, 716 F. 3d 1166 - Court of Appeals, (9th Cir. 2013).

[155] Steve Green. Receiver says Righthaven 'uncooperative' in surrendering copyrights, *VegasInc*. 2012. Available at: [http://vegasinc.com/business/2012/jan/04/receiver-says-righthaven-uncooperative-surrenderin/] Accessed 2015-07-17.

[156] Prop. 1988/89:85. *Om upphovsrätt och datorer*, p. 8.

the software is not certain due to uncertain transfer of rights. It is important to note that these uncertainties do rise in ordinary development of modern software. Due to that the normative space is wide, uncertainty emerges when the software developers reuses pieces of source code and pastes them into their own commercial software, software that is later transacted to end users. These uncertainties create risk as an obstacle when transacting the software, as seen in the different illustrations in this thesis. In the next chapter I will analyse different normative arguments of how this risk can be distributed between the licensor and the end-user in the absence of an agreement.

# 3 LAW – In the absence of an agreement

## 3.1 Introduction

As seen in chapter 2, it is uncertain to what extent an element in a software program is protected from unauthorised uses. It is also uncertain to what extent the copyrights to these elements are transferred in an employment situation in the absence of an agreement regulating the transfer of the rights. For a software vendor this is a problematic situation since this implies that some of their software might contain elements that are protected by copyright and also owned by third parties. While there are several laws in Sweden regulating the seller's responsibility in relation to the buyer in transactions regarding tangible goods, there is no law explicitly defining intellectual property defects, and there is no law explicitly governing the seller's responsibility for intellectual property defects in transacted software. In this chapter I will discuss the uncertainty during the transaction of the software from a software developer as seller to the end-user as buyer in the absence of an agreement regulating these matters.

The purpose of this chapter is to construct possible legal argumentations about the responsibility of the seller in software transactions.[157] This is not an attempt to find "the one answer" but instead one of several possible legal arguments about how to solve this situation in the absence of an explicit agreement regulating intellectual property defects. I will start with deconstructing what a transaction of software is, before going into the definition of intellectual property defects, especially relating to copyright, after which I will discuss the risk for the buyer in relation to intellectual property defects. I will then continue to present and analyse different analogies on how to responsibility of the intellectual property defect is allocated between the parties in the absence of an agreement regulating this matter.

[157] Linda Källström has written a master's thesis regarding the sellers responsibility for non-conformity in the copyright, in sales of software, however she explicitly delimits licensing arrangements, see Linda Källström. *Fel i rättighet till datorprogram - en utredning av upphovsmannens ansvar för immaterialrättsligt fel vid överlåtelse av en upphovsrätt* (Master's thesis). Lund: Department of law, Lund University, 2014. As I will present below, software transactions are often built on licensing agreements.

## 3.2 The nature of the software transactions

To be able to determine what rules and principles should govern the responsibilities for a software vendor and an end-user for intellectual property defects regarding copyright in software transactions I have to deconstruct the phenomena of *software transactions*. To be able to conclude the nature of such exchange it is important to differ between the right to make copies of a work (the license of the exclusive right) and the actual copy of a work (the software program per se). For example the Swedish Sale of Goods act (SFS 1990:931) is applicable on sales of property, excluding sales of real estate.[158] This includes the sale of an intellectual property right.[159] However it does not govern relationships between parties where the substantial part of the agreement is a service rather than a sale.[160] A common definition of a "sale" is an agreement were the owner transfers his ownership of an item, tangible or intangible, to an acquirer, in return for payment.[161] To determine the applicability of the Swedish Sale of Goods act (SFS 1990:931) the question is whether a *software transaction* constitutes a *sale* or a *limited right*, i.e. *a license*. First we have to consider the right to *use* a piece of software.

### 3.2.1 Transacting software – A sale of a good or a license to make copies of a work?

It is commonly said that software is *licensed rather than sold*.[162] The fundamental differences between a sale of software and a license to that software concerns ownership. If the buyer will receive ownership of the software than it is a sale and if the seller will maintain ownership it is a license.[163] Although that seems like a simple rule to distinguish between a sale and a license this is seldom the case. Software transfers can be constructed in several ways, which make them difficult to place in one category or the other.[164]

---

158 Art. 1 in the Swedish Sale of Goods act (SFS 1990:931).

159 Eric M. Runesson. Immaterialrättsliga fel vid köp. In *Festskrift till Gunnar Karnell.* Gorton, Lars (ed.) Stockholm: Carlsson Law Network, 1999, p. 626.

160 Art. 2. Para 2. in the Swedish Sale of Goods act (SFS 1990:931).

161 Case C-128/11 *UsedSoft GmbH v. Oracle International Corp.* [2012], para. 42.

162 See, Jan Leidö. *Realizing the single software market – cross-national validity of software license agreements*. Uppsala: Iustus förlag AB, 2014. p. 111, note. 554 for further references.

163 Ibid., p. 112.

164 Ibid., p. 114.

When a company acquires software it is common that they download a copy of the software from the software developer's homepage, which they install, and then they enter into a license agreement where they agree to pay a fee to receive the permission from the right holder to use the software. In this type of transaction it can be argued that the software vendor does not sell copies of computer programs, they only supply copies of those works on their homepage, free of charge, which the customer can download. However to *use* the works the customer needs to enter into a license agreement and pay the license fee.[165]

Naturally the licensor can only license what they are the right holder of. As a right holder of copyright to a computer program (or databases or other, by copyright, protected works) they have the exclusive right to exploit the work by making reproductions of the protected work and by making the work available to the public.[166] These are also the only rights that the right holder can license to others, for example this means that the licensor cannot license the right for someone to physically *read* the source code since this is not one of the exclusive rights in the copyright act. However in software license agreements it is common that the licensor licenses a right to *use* the software.[167] Copyright does not give the right holder an explicit right to *use* the work but only, as said, the right to make reproductions of the work and the right to make it available to the public. However when you start a software program, which is stored on a computer hard drive or server, the computer will make a copy of the computer program into its RAM.[168] Since the use of a software program creates copies of the underlying works, the user needs a license to make reproductions of the underlying works to use the software.[169]

---

165 The notion of software license agreements are also debated see Leidö, chapter 3.2 *Legal Categorization of Software License Agreements*.

166 Art. 2 in the Swedish Copyright act (SFS 1960:729).

167 See for example CISCO EULA under "License". Available at: [http://www.cisco.com/c/en/us/td/docs/general/warranty/English/EU1KEN_.html] Accessed 2015-07-17, or SAP Software License and Support Agreement clause 2.1.1. Available at: [http://global.sap.com/corporate-en/our-company/agreements/western-europe/index.epx]. Choose "SAP Software Agreement", "Sweden". Accessed 2015-07-17.

168 "Random Access Memory", for further details see Dave Coustan & Jeff Tyson. How RAM works, *howstuffworks*. 2015. Available at: [http://computer.howstuffworks.com/ram.htm] Accessed 2015-07-17.

169 Rosén, p. 268.

The substantial part of the agreement can be argued to be that the customer is allowed to *use* the software for a certain time; as such they pay the fee for the right to use the software and not the ownership of the copy of the work itself.[170] Because of this the transaction would not constitute a sale but a mere license of a right. However it can also be argued that if one considers the transaction as a whole, especially if the license is unlimited in time, this resembles a sale rather than a license. The intention of the transaction could be argued to be that the customer should be able to use the copy permanently in return for a fee that should correspond to the economic value of a copy of the work for the right holder, why this similar to a sale of the copy itself rather than a limited right, *a license*, to it.[171]

The conclusion is that there exists normative space to allow for both a solution were there is only a limited right and a solution were it is a sale. To determine which one of these a certain software transaction resembles we can specially focus on the duration of the license, the payment structure and limitations on modification.[172] Even if the transaction would only be deemed to be a limited right and not a sale, the Swedish Sale of Goods act can be analogously applicable on such transactions why it is still relevant.

## 3.3 Intellectual property defects, especially regarding copyright

As mentioned in the introduction to this chapter, there is no definition in the Swedish legislation regarding intellectual property defects in licensed intellectual property rights, or even sold goods. In the preparatory works for the Swedish Sale of Goods act it is mentioned that an intellectual property defect is a claim made by a third party based on an intellectual property right.[173] Another definition of intellectual property defects in sold

---

[170] It is often explicitly stated in the license agreement that the licensor retains ownership of all copies of the software. See for an example CISCO EULA under "General Limitations". Available at: [http://www.cisco.com/c/en/us/td/docs/general/warranty/English/EU1KEN_.html] Accessed 2015-07-17.

[171] This is the argument used by CJEU in UsedSoft v. Oracle were they decided that Oracles software licenses did include a transfer of ownership to the licensee and thus constituted a sale in the meaning of the exhaustion of rights principle, see Case C-128/11 *UsedSoft GmbH v. Oracle International Corp.* [2012], para. 44 – 46. It is important to note that since this argument is used in relation to *the exhaustion of rights* principle the judgement is only analogously applicable to the distinction of a sale in the Swedish Sale of Goods act (SFS 1990:931).

[172] For further discussions on this topic see Leidö, chapter 3.1 *Legal Categorization of Software Transfers.*

[173] Prop. 1988/89:76. *Om ny köplag* pp. 141 - 142.

goods can be found in art. 42 CISG, which states: "*The seller must deliver goods which are free from any right or claim of a third party based on industrial property or other intellectual property, of which at the time of the conclusion of the contract the seller knew or could not have been unaware, provided that the right or claim is based on industrial property or other intellectual property…".* Other definitions of intellectual property defects can also be found in literature, for example Runesson defines it as follows "*The fundamental condition for an intellectual property defect to occur is that a buyer through a resale or use of the subject matter infringe or is alleged to infringe in an intellectual property right of a third party or the seller* [my translation]".[174] What is important to note is that intellectual property defects is a special kind of defect in the sold object since it is difficult to determine *in advance* if the object is infringing and whether the intellectual property protection, underlying the third party claim, can be upheld in court.

It is relatively reified and thus uncontroversial among the legally schooled that an intellectual property defect is a claim towards a buyer of a good, made by a third party based on an intellectual property right. However it is not certain what kind of claim and to what extent the claim needs to be proven to constitute an intellectual property defect, e.g. would it be enough that the licensee can expect that a third party has the possibility to make a claim based on an intellectual property right towards the use of the transacted object or is it necessary that such claim is evident, for example by a warning letter sent by the third party, to be regarded as an intellectual property defect? It might even be argued that the claim should be assessed in a court to determine if it constitutes an intellectual property defect, since the underlying intellectual property right is dependent on such an assessment. For this thesis intellectual property defects are defined as when a third party makes a claim, based on an intellectual property right (copyright), to own the whole or parts of, the transferred work, which hinders the buyer from using the transferred work in the agreed manner.

An intellectual property defect in the relationship between the seller and the buyer typically arises in two situations when transacting software.[175] First, it occurs when the seller licenses a software program to buyer under license terms that are in breach of the license terms agreed upon between the original right holder (often the author) and the seller. For example this might happen if the GUI, in the absence of a contract, is licensed

[174] Runesson, p. 626.

[175] See Lindberg & Westman, p. 424.

under the *rule of thumb principle* to from an employee to the seller who licenses the GUI further to a buyer as end-user. As discussed in section 2.5.1 above, the rule of thumb principle does not necessarily include a right for the employer to sublicense the work further. Since an end-user will not acquire a better right to the licensed object than the seller has acquired from the original right holder, the end-user will commit copyright infringement when using the software.[176]

The second situation occurs if the software program infringes the rights of a third party. This occurs if the end-user enters into an agreement with someone who claims to be the right holder of the software but a third party is able to prove that the rights are actually owned by him or her. This situation can occur if a software developer has reused source code from Internet or if one of the company´s software developers have been working for one company and after changing employer he or she developed software that can be deemed to be a reproduction or derivate work of a work owned by the previous employer. The new copy is then transferred to an end-user who will make reproductions of the work when using the software and thus commits a copyright infringement in the right of the original right holder.

### 3.3.1 Intellectual property defects in transacted software: The risk for the buyer

To be able to determine what claims the buyer might be interested to make towards the seller if an intellectual property defect occur, I have to consider what the risks could be for the buyer if the software contains an intellectual property defect.

A risk for the buyer (or the defendant) is that a third party (or the claimant) makes a preliminary injunction claim to stop further infringement by the buyer, in accordance with art. 53 b para. 2 in the Swedish Copyright act (SFS 1960:729). Since the consequence of such claim is a preliminary action the result will have effect before the court has given their final judgement on the matter. The risk is also increased since the burden of proof for a preliminary injunction is set to "a probable" infringement in comparison to an established infringement "beyond reasonable doubt", which is the

[176] See section 2.6 about infringement and for information on acquisitions of intellectual property rights in good faith from an unlawful seller, see Jens Andreasson. *Intellektuella resurser som kreditsäkerhet: En förmögenhetsrättslig undersökning.* Göteborg: Chalmers tekniska högskola, Reproservice, 2010, pp. 293 – 294.

burden of proof to be convicted for the infringement in art. 53 in the Swedish Copyright act (SFS 1960:729).[177] [178] The risk with a preliminary injunction is that the licensee will suffer losses due to that the business will be interrupted since they will not be able to use the software program, this is especially the case if the software is a central part of the administration of the business.[179] It is less of a risk for the defendant to be convicted for the crime of copyright infringement since it must be proved beyond reasonable doubt that the defendant committed the crime with wilful misconduct or gross negligence intent, especially since the defendant needs to be the physical person committing the infringement. It would generally be difficult to *prove* that the buyer's employees knew or should have known about the rights of a third party, especially since
(1.) most often, neither the buyer nor their employees possess the source code for the licensed software and the claimed original work why they won´t be able to determine the similarity between the computer programs,
(2.) even if such employee would be able to determine that the code is similar or even exactly alike it does not necessarily mean that an infringement has been made since both computer programs can be protected as individual works as long as the works are created independently from each other and the criterion of subjective novelty is fulfilled,[180]
(3.) since copyright is an unregistered right and the court will determine the protection in retrospection, it is difficult for the buyer's employees to determine if and in what parts the licensed software and the claimed original work is protected by copyright.[181]

However the normal course of action is that the claimant first sends a warning letter to the alleged infringer stating that the they have committed an infringement in the

---

[177] "Probable" and "established beyond reasonable doubt" is my translations of "sannolikt" and "ställt bortom rimlig tvivel".

[178] The claimant's risk is that they can be held to pay remuneration for the damages caused to the defendant as a consequence of the preliminary injunction, if the claim of infringement isn't upheld by the court, in accordance with art 53 b para. 4 in the Swedish Copyright act (SFS 1960:729).

[179] An example of such software is WorkSite, which is used to administer and manage documents and is used by several law firms. If WorkSite would not be accessible for these law firms, the delivery of consultancy services to clients will most likely be severely delayed, resulting in losses of profit. More information about WorkSite can be found at: [http://www8.hp.com/us/en/software-solutions/document-management-software/] Accessed 2015-07-17.

[180] For intent, see section 2.6.3 above.

[181] I am not considering the copyright register that exists in some jurisdiction e.g. the US jurisdiction.

claimant's intellectual property rights when using the transferred object.[182] After the defendant has received such notice they now possess knowledge about the claimed infringement and it can be argued to be at least gross negligent to ignore such claim and continue to use the software without further investigation. This constitutes an obvious risk for the buyer since if they decide to start an investigation it is likely that they will have costs for interruption of business and consultancy for legal advice, and if they decide to continue to use the software without further investigation this might constitute copyright infringement with gross negligent intent. Another risk for the buyer is that they might have to pay a reasonable remuneration to the claimant for the use of the protected work. This is a strict liability, why it doesn't matter if they are not deemed to have committed the infringement by wilful misconduct or by gross negligence intent.[183] If the buyer is deemed to have committed the infringement *negligently* they might also have to pay additional remuneration for damages.[184]

In summary there are three types of risk for the buyer:

1. The risk of the physical person being convicted for the crime of copyright infringement including
    a. Imprisonment
    b. Fines
2. The risk of direct costs due to the infringement, including
    a. Reasonable remuneration and damages to be paid to the right holder
    b. Legal expenses
    c. License fees
3. The risk of indirect costs due to the infringement, including
    a. Costs due to loss of sales (business interruption, bad will etc.)

Of these risks it is most interesting to consider which of the above, except 1.a the buyer will be able to claim from the seller for the intellectual property defect.

---

[182] The warning letter often contains an opportunity to license the third party rights to avoid further litigation. The license is often more favourable cost-wise than to initiate an investigation and legal counter action, especially in the cases of the claimant being a non-practicing entity or as sometimes called a "copyright/patent troll".

[183] See art. 54 in the Swedish Copyright act (SFS 1960:729) and Olsson, 2015, p. 273 ff.

[184] See section 2.6.3 above, about the assessment of negligent behaviour.

## 3.4 The Swedish Sale of Goods Act

There are three types of defects handled in the Swedish principles for sale of goods, two of these are explicitly handled in the Swedish Sale of Goods act (SFS 1990:931). They are the *factual defect*, the *defect in rem* and the *disposition defect*.[185] For this thesis I will only consider the factual defect and the defect in rem. It is not certain how intellectual property defects relate to these. The Swedish Sale of Goods act (SFS 1990:931) does not contain an article explicitly regulating intellectual property defects. In the preparatory works for the act it is explicitly mentioned that intellectual property defects is excluded from the regulations on *defects in rem* due to third party ownership. It is further stated that third party claims based on intellectual property rights should be assed according to general rules.[186] I will start by an analogy to the general rule of non-conformity in art. 17 in the Swedish Sale of Goods act (SFS 1990:931) which regulates factual defects.

### 3.4.1 Factual defects - art. 17 in the Swedish Sale of Goods act

Art. 17 in the Swedish Sale of Goods act (SFS 1990:931) generally states that the good is to be considered to be defect if the good deviates from what the buyer can reasonably assume about the good. To be able to determine what the buyer can reasonably assume about the good one has to consider the nature of the object, the description of the object, the prior relationship between the parties and what properties a buyer typically assumes about similar objects.[187] In an analogy to art. 17 in the Swedish Sale of Goods act (SFS 1990:931) the responsibility for intellectual property defects would be determined by what the buyer could reasonably assume about the transferred software.

Generally when a company acquires software they enter the software seller's homepage. At the homepage there is typically information that describes the properties of the software program and the terms of the license agreement. There is often a link where the buyer can download a copy of the software program, which they later install. It is also common that they can try the software program for period of time before they have to pay license fees to continue to use it.[188] During this time the buyer will get a feel for the

---

[185] My translation of "faktiskt fel", "rättsligt fel" and "rådighetsfel".

[186] Prop. 1988/89:76. *Om ny köplag* pp. 141-142.

[187] Jan Ramberg & Johnny Herre. *Allmän köprätt*. 7 Ed. Stockholm: Norstedts Juridik AB, 2014, p. 85, and art. 17 para. 3 in the Swedish Sale of Goods act (SFS 1990:931).

[188] This is normally called a "trial period".

function of the software. In these situations it is normal that the buyer assumes that they can use the software in the manner described at the homepage and in the way that the software functioned on their computer. If the company would be aware of the third party right at the time when purchasing the license they would not be able assume that they could use the software without such claims why the software would not be considered defect. However if the buyer would license the software and it would later come to their knowledge that the software infringes third party copyrights, the software would be considered to be defect and the buyer would be able to make claims against the seller.[189]

### 3.4.2 Damages - art. 40 in the Swedish Sale of Goods act

In accordance with an analogy to art. 40 in the Swedish Sale of Goods act (SFS 1990:931) the buyer is entitled to compensation for damages for direct losses suffered due to that the software is defect, unless the seller shows that the defect is due to a *hindrance* beyond his *control* that he could not reasonably have *foreseen* prior to the transaction and which consequences he could not reasonably have *avoided or overcome*. These direct losses would mainly be those stated under 1.b and 2. a – c. under the summery of risks for the buyer above. The buyer is entitled to damages for indirect losses as well, mentioned under 3. if it can be proven that the defect is due to the seller´s negligent behaviour, or if the software would deviate from an agreed warranty.[190]

If it can be proved that the software is infringing in third party copyrights, either due to that one of the employees of the seller has copied code from the Internet without respecting the copyright within that work, or because the seller did not acquire all rights from consultants and employees to sublicense the work, it would be difficult for the seller to prove that the defect was outside his or her *control*, why he would be liable for direct damages towards the buyer. For the buyer to be entitled to compensation for indirect losses the burden is severely increased if the seller has not given an explicit warranty on this matter. The buyer would have to prove that the intellectual property defect has been caused by negligent behaviour by the seller. Generally if the employees of the seller has copied code from the Internet without respecting the copyright within

[189] Assuming that the company has given notice to the licensor in time within the time stated in the rule of limitation, see art. 32 in the Swedish Sale of Goods act (SFS 1990:931).

[190] Art. 40 in the Swedish Sale of Goods act (SFS 1990:931).

the work, it should be considered negligent behaviour since a legitimate behaviour would be to act cautiously while reusing the code of others to ensure to respect copyright and license obligations.[191] It should also constitute negligent behaviour if the seller has not acquired all rights from consultants and employees to sublicense the work. However it can be difficult for the buyer to successfully claim such acts towards the seller since the buyer most often won´t have the information necessary to prove this. The right to claim an infringement investigation in accordance with art. 56a and 56b para. 2. in the Swedish Copyright act (SFS 1960:729), to collect evidence of an infringement doesn´t apply to the buyer of the software since they are not the right holder or a legitimate licensee of the infringed copyright.

### 3.4.3 Defects in rem – art. 41 in the Swedish Sale of Goods act

An intellectual property defect can also be compared to the *defect in rem* regulated in art. 41 in the Swedish Sale of Goods act (SFS 1990:931). As defined in the Swedish Sale of Goods act (SFS 1990:931) a good contains a defect in rem if a third party has a property right in the good and the buyer has not accepted this condition. To license out a work that one does not possess the full copyright to can be argued to be comparable to selling a good that you do not have the ownership to, why an intellectual property defect resembles a defect in rem. In an analogy to art. 41 in the Swedish Sale of Goods act (SFS 1990:931), the seller would have a strict liability for intellectual property defects if the buyer didn´t know about, or had no reason to suspect, the intellectual property defect at the time of the conclusion of the contract.[192] In such analogy would construct a strict liability for the seller, including damages for both direct and indirect losses of the buyer, and the burden of proof for the buyer would be a to prove a *plausible* third party copyright to a work in the software.[193] If one would analogously apply *the defect in rem* construction on intellectual property defects this would create a severe burden on the seller, since as soon as a third party would make a claim towards the buyer that they own the copyright to a work in the software this would entitle the buyer to compensation from the seller for both direct and indirect losses of such claim. Since the claim is based

---

[191] Lindberg & Westman, p. 378.

[192] Lindberg & Westman mentions that the applicability of art. 41 in the Swedish Sale of Goods act (SFS 1990:931) is uncertain regarding intellectual property defects, however they mention that applying an analogy to art. 41 in the same act *might* be a solution in the absence of a contract. See Lindberg & Westman p. 425, note 9.

[193] See art. 67 in the Swedish Sale of Goods act (SFS 1990:931).

on an unregistered right it would be difficult to determine if the claim is genuine or not without a final judgement by a court.

An argument against applying art. 41 in the Swedish Sale of Goods act (SFS 1990:931) analogously is that the preparatory works explicitly state that intellectual property defects should not be regulated as a defect in rem.[194] An obvious difference between a defect in rem in a transacted good and an intellectual property defect in a the same good is that while the a physical good is rivalrous an intangible good is non-rivalrous. If a good is rivalrous this means that the consumption of the good hinders others from consuming the same good while if the good is non-rivalrous, the consumption of the good doesn´t hinder others to consume the same good.[195] This means that original owner to a physical good is hindered to use "their" good if the buyer would acquire the physical good, but the original right holder to a copyright is not hindered to use their copyright protected work even if the buyer acquires a right to use a copy of the work. [196] Due to this logic an intellectual property defect can be argued to be different from a defect in rem and an analogy would thus not be suitable, since the responsibility of the seller would be unreasonable burdensome in relation to the harm caused to the original right holder.

## 3.5 CISG

CISG regulates international sale of goods and has been ratified in Sweden.[197] Since CISG is only applicable to *the sale of goods*, the articles in CISG is only directly applicable on software transaction were the software can be argued to be a sale of a good, i.e. to repeat the argument above *if one considers the transaction (the copy of a software and a license to the right to use the protected work in such software) as a whole, especially if the license is unlimited in time, this resembles a sale rather than a license.*[198] However if the transaction is only deemed to be a license to the right to make copies of a copyright protected work, CISG would only be analogously applicable on intellectual property defects in such transaction.

---

[194] Prop. 1988/89:76. *Om ny köplag* pp. 141 – 142.

[195] Cooter & Ulen, p. 40.

[196] Andreasson discusses effect of non-rivalty when intellectual property rights are aqcuired in good faith from an unlawful seller, see Andreasson, pp. 294 – 296.

[197] Prop. 1988/89:76. *Om ny köplag* p. 24.

[198] Chapter 1. Art. 1 CISG. For CISG to be applicable, there is also a criterion that the parties place of business should be in different states. However I will not investigate the complexities of this criterion on software transactions further in this thesis.

### 3.5.1 Intellectual property defects – art. 42 in CISG

Intellectual property defects in the sold good are regulated in art. 42 CISG. In an analogy to art. 42 CISG the responsibility for intellectual property defects would be determined by two criteria. First it would have to be assessed whether the transacted software program infringed in a third party intellectual property right in a state where it would be resold or used, *if it was contemplated by the parties at the time of the conclusion of the contract that the goods would be resold or otherwise used in that State.* If the parties have not agreed on a state the seller liability is limited to the state where the buyer has his or her place of business in accordance with art. 10 CISG.

Secondly it would have to be determined whether the seller was in *bad faith* about the intellectual property defect at the time of the conclusion of the contract.[199] This criterion would be fulfilled if the seller could not have been *unaware* of the third party right or claim. The Secretariat Commentary on the draft of CISG states that the seller cannot have been unaware of the third party right if it is *based on a patent application or a grant which had been published in the country in question.* They further mention that third party rights might exist that are not published and that the seller is not liable to the buyer in such situations.[200]

If one would apply the same criteria as the secretariat it would probable be hard for the buyer to prove that the seller was in bad faith about the intellectual property defect, since it would based on a copyright claim, which is unregistered in Sweden and thus not published. In such case it can be argued that the seller is in bad faith only if the he or she has received a warning letter before the transaction.[201] Since the buyer would have the burden of proof for this fact and due to that companies seldom release warning letters to the public it would be hard for the buyer to successfully prove such claim in court.

---

[199] Bad faith is my translation of ”ond tro”, e.g. when the licensor is in bad faith, the licensor has knowledge about the intellectual property defect at the time of the conclusion of the contract.

[200] Guide to CISG - Text of Secretariat Commentary on article 40 of the 1978 Draft. No 6. Article 40, the draft counterpart of the current CISG article 42. Available at: [http://www.cisg.law.pace.edu/cisg/text/secomm/secomm-42.html] Accessed 2015-07-17. It can be argued whether the commentary should be used for interpreting CISG at all, since the commentary is not binding on the parties to the convention.

[201] Runesson, p. 641.

As stated in art. 42 (2) CISG, the seller´s liability does not extent to situations when the buyer know or could not have been unaware of the third party right at the time of the conclusion of the contract, why in the cases stated above, the seller would be able to rectify the defect by informing the buyer about the warning letter or that they suspect that ad-hoc code reuse which have infringed third party rights have occurred, and thus transfer this risk to the buyer who would need to consider if they want to enter into the agreement. If the seller does not inform the buyer, the buyer´s remedies for an intellectual property defect would primarily be claims of damages and cancellation of the contract, see art. 45, art. 49 and art 74 – 77 CISG. A claim of damages covers both direct and indirect losses, see art. 74 CISG. There are exceptions to the liability in art. 79 CISG, which resembles the regulation in art. 40 in the Swedish Sale of Goods act (SFS 1990:931), however these are not applicable to intellectual property defects since the seller needs to be in bad faith about the defect to be liable in accordance with art. 42 CISG.

## 3.6 Doctrine and discussion

As stated above it is not certain if a software transaction constitutes a sale or a limited right. Rosén argues that a sale of a copy of a computer program ought to include a license to make the copies necessary to use the computer program.[202] Even if Roséns argument is reasonable it must be noted that the express regulation in art. 27 in the Swedish Copyright act (SFS 1960:729) state that the copyright to a work is not included in the physical copy of the work. Runesson argues that it is uncertain how implied licenses are considered under Swedish law, and that it must be left to an interpretation of the agreement. He mentions that with copyright one has to consider the principle of specification, which promotes an interpretation in favour of the author.[203]

From the seller´s perspective it is preferable to argue and construct a transaction that constitutes only a non-exclusive license to the rights in the software that the seller owns, i.e. the seller commits to not enforce their own intellectual property rights to exclude the buyer when the buyer makes reproductions of the software. Thus the right to *use* the software as a whole, which might infringe any additional third party rights if any such exists in the software, is not included in the transaction. With this argumentation the

[202] Rosén, p 275.

[203] Runesson, p. 626.

buyer would bear the responsibility if the software infringes third party rights when used. However from the buyer´s perspective it is more suitable to argue and construct a transaction that constitutes a license to use the software as a whole, i.e. to access the function of the software. The buyer is most often interested in the function that the software performs and not in the underlying copyright in the source and object code, or the other works. They pay for a right to use an *expression of a function* and it would be meaningless for the buyer to pay the fee if they would not be able to access the function in the software. With this argumentation the seller would bear the responsibility if the software infringes third party rights that would limit the use.

Another obvious question is whether the Sales of Goods Act or CISG should be applied analogously, at all, to these agreements? There are many different opinions on this issue. Ramberg & Herre conclude that it is uncertain how the responsibility is allocated between the parties for intellectual property defects in the absence of an agreement. They argue that the Swedish Sale of Goods act (SFS 1990:931) should be applicable on intellectual property defects, and that such defects should be considered to be factual defects. They state that the seller, in accordance with the Swedish Sale of Goods act (SFS 1990:931), should bear an increased responsibility for intellectual property defects when compared with art. 42 CISG, however the modifications in art 42. CISG should still be analogously applicable to intellectual property defects.[204] Ramberg & Herre is also of the opinion that an intellectual property defect cannot be seen as a defect in rem with a strict liability for the seller, since such solution should have been present in the legislation. They further state that the responsibility for intellectual property defects should be determined on a case-by-case basis, where the court should specially consider the sellers knowledge about any third party claims based on intellectual property rights.[205]

---

[204]Jan Ramberg & Johnny Herre. *Köplagen – En kommentar*. 2 Ed. Stockholm: Norstedts Juridik AB, 2013, pp. 364 – 365. It should be noted that the authors discuss intellectual property defects in *sold goods*. Since goods can be argued to differ from licensed software it is uncertain whether they would apply the same logic on licensed software.

[205] Ramberg & Herre, Johnny, 2013, p. 189.

Hultmark argues that the Swedish Sale of Goods act (SFS 1990:931) can be analogously applicable on license agreements.[206] She argues for threating intellectual property defects under the Swedish Sale of Goods act (SFS 1990:931) as factual defects since from a systematic perspective it would not be satisfactory if intellectual property defects would have their own rules of limitation and own rules of actions separate from factual defects. She also argues that treating intellectual property defects in an own system would give rise to issues of demarcation between different defects, which can be avoided if threated under a single system.

Westman only state that it is uncertain how the responsibility is allocated between the parties for intellectual property defects in a licensed right in the absence of an agreement.[207] According to Plogell it might be difficult to determine whether a transaction of software is a limited right or a sale. He states that it is common to regard a transaction as a license when the buyer should return the object to the seller and it is normal to regard it as a sale when ownership of the medium for the copy has been transferred and when the buyer doesn't have any restrictions *in time* regarding the use of the software program and as a result one would have to analyse the clauses in the agreement to determine the nature of the agreement.[208] In contrast, Rosén argues that a software license agreement is a *limited right* for the buyer to make the copies necessary to run the software program.[209] He further argues that the Swedish Sale of Goods act is not applicable on a limited right to use a copyrighted work, i.e. a license, and that only slight analogies to general principles regarding sales of goods can be applied, however he does not specify which ones.[210]

From the international context the text of the Secretariat Commentary on the draft of CISG mentions that "*It appears to be the general rule in most, if not all, legal systems that the seller*

---

[206] Christina Hultmark. Köplagens tillämplighet på fel i patent, *Juridisk Tidskrift vid Stockholms Universitet. (1993-94), nr. 4,* p. 688 ff.

[207] Lindberg & Westman, p. 425.

[208] Plogell, Michael. Upphovsrätt till datorprogram ur ett EG-perspektiv. *Juridisk Tidskrift vid Stockholms Universitet. (1993-94), nr. 1,* p. 67. It is important to note that Plogell discusses the nature of a license in a section regarding the exhaustion of rights rather than the applicability of the Swedish Sale of Goods act (SFS 1990:931).

[209] Rosén, p. 274.

[210] Rosén, p. 110 ff.

*is obligated to deliver goods free from any right or claim of any third party based on industrial or intellectual property [industrial property or other intellectual property]. In the context of a domestic sale, this rule is appropriate. The producer of the goods should be ultimately responsible for any infringement of industrial or intellectual property rights [industrial property or other intellectual property rights] in the country within which he is both producing and selling. A rule that places the liability on the seller allows for this liability ultimately to be placed on the producer."*[211] It appears that according to the Secretariat it is uncontroversial to hold the producer liable for intellectual property defects in domestic sales, however they don´t mention the criteria for such liability for example if it should be based on negligence or strict liability.[212]

As seen above Hultmark argues that the Swedish Sale of Goods act (SFS 1990:931) is applicable on intellectual property defects, however she also argues that art. 42 CISG should *not* be analogously applicable on intellectual property defects on transactions within Sweden, referring to the conclusion made by the Nordic workgroup that art. 42 in CISG is too simplistic for an efficient regulation of this matter.[213]

Runesson constructs his own solution for intellectual property defects, which he builds on general principles in sale of goods law with an addition of a standard of precaution.[214] He states that it is uncontroversial that the seller should be responsible for an intellectual property defect when the seller has knowledge about or at least suspects a third party claim based on intellectual property right in the sold good.[215] In his solution the seller is responsible for intellectual property defects if he or she is negligent, i.e. the buyer can never reasonably assume that the good won´t ever infringe in any third party rights unless the seller has given such warranty, however the buyer should be able to assume that the seller has taken the necessary precaution to secure any third party rights in the good.

---

211 Text of Secretariat Commentary on article 40 of the 1978 Draft. No 4. of CISG.

212 The Secretariat does not share the same opinion regarding intellectual property defects in international trade, see Text of Secretariat Commentary on article 40 of the 1978 Draft. No 4. of CISG paragraph 4.

213 Hultmark, 1994, p. 693.

214 Runesson, pp. 651 - 656. It should be noted that the author discuss intellectual property defects in *sold goods*. Since goods can be argued to differ from licensed software it is uncertain whether they would apply the same logic on licensed software.

215 Ibid., p. 651 ff.

My own opinion is that the problem is that the legislation and most authors do not differ between the different categories of intellectual property rights. For example art. 42 in CISG doesn´t differ between infringements in copyrights or infringements in patents. I argue that it is difficult to build a suitable norm that would fit both the patent system, which builds on objective novelty, a registered right, with different publications and praxis in each jurisdiction, and the copyright system, which builds on un-registered rights in a fairly harmonized system over the world (compared to other IPRs) and a subjective novelty. By constructing one solution for the registered rights, i.e. patent rights, design rights, and registered trademarks and another solution for unregistered rights, i.e. copyrights and un-registered trademark rights (and possible un-registered Community design rights), the norms can be more appropriate for a balanced solution between seller and buyer regarding intellectual property defects.

To be able to determine the liability in regards to copyright protected software one should focus on the knowledge of the seller at the time of the conclusion of the transaction. I argue that it is unreasonable that the seller would be completely free from liability for intellectual property defects based on copyright claims in all other circumstances than when they had received a warning letter prior to the transaction. The infringement might for example be caused by the sellers own unjust verbatim copying of others code, i.e. unjust ad-hoc code reuse, an act that is likely to constitute a copyright infringement in most copyright jurisdictions since the copyright system is fairly harmonized due to the Bern Convention and thus the person conducting the infringing action, for example ad-hoc reuse of source code from the internet, should understand the risk of such behaviour.[216] In such circumstances the original right holder might not know of such copying until the transaction is concluded between the seller and buyer why they won´t be able to send the warning letter prior to such transaction. In these situations the possibility for the buyer to recover their losses from the seller will be dependent on the actions of a third party, i.e. that the third party sends a warning letter in time, which would be unsatisfactory since the foreseeability would decrease. It is therefore reasonable that the seller is liable if the software is infringing as a result of the sellers (or his employees) own copying of code. In such case it would not be reasonable if the seller could avoid liability towards the buyer by claiming that they didn´t know that

[216] At least in the jurisdictions where they protect computer programs as literary works.

they had copied copyright protected code from other sources since this would incentivize the software developing companies to be unaware of their conduct in the development stage.

However, it is also important to consider that the software developer should be able to have comfort in applying standard methods of software development without being limited by liability for copyright infringement. If software developers experience that they have complied with the moral and ethics of software development in society it is unsatisfactory if the law would say otherwise. This might lead to a situation were the law is not followed and thus the reification of software as a copyright protected and transferable property would be decreased. When the reification process is decreased it is possible that less developers and investors would feel confident that they would have exclusivity to the works that they have invested in, regardless whether that is time or money. With less confidence towards this exclusivity the investments in software would risk to decrease and thus the quality and development of software would also risk decreasing, going against the initial aim of protecting software against unjust copying.

## 3.7 Conclusion

> *When a seller transacts the software to a buyer, what responsibility do the seller have for the uncertainty found in question 1, especially regarding claims made by third parties based on copyright, in relation to the buyer if this is not regulated in an agreement?*

I can conclude that there is normative space between several alternatives regarding what a software transaction is, what constitutes an intellectual property defect and how the liability for such a defect should be handled. The alternatives would be to either apply the Swedish Sale of Good act (SFS 1990:931) and/or the CISG, directly, or analogously, or to disregard the applicability of the acts and chose another norm to regulate the liability, often built on general principles in sale of goods law.

By analysing the different arguments we can construct different normative claims. For example, if one would want to construct a solution were the seller carries the least responsibility as possible for intellectual property defects in an software transaction from the sources stated above, one should argue that art. 42 CISG and its commentary is

applicable (at least analogously) and since copyright is an unregistered and thus unpublished right, the seller cannot be held liable for such intellectual property defects since they cannot be aware of such claim. In contrast if one would want to construct a solution were the seller carries as much responsibility as possible for intellectual property defects in a software transaction from the sources stated above, one would argue that an intellectual property defect is a defect in rem in the Swedish Sale of Goods act (SFS 1990:931) why a strict liability for the seller applies.

In the next chapter I will analyse clauses that can be used in software end-user license agreements and show different options in how this risk can be distributed between the licensor and the end-user.

# 4 CONTRACT – Some examples of how the risk can be regulated in commercial agreements

## 4.1 Introduction

Lawyers commonly define a contract as ”An agreement creating obligations enforceable by law”.[217] This definition sets particular emphasis on the contract as a tool in a legal construction where it is used to enforce the parties to act through the threat of litigation. I believe that for most business actors it is not the ability to force the other party to act that is the driving intent behind a contract. Boyce makes another definition of a contract. He states that a contract is “A vehicle by which the risks inherent in the transaction are allocated as between buyer and seller”.[218] I find Boyce definition more suitable, especially for the subject of this thesis. As seen in the previous chapter the normative space regarding who should carry the risk of intellectual property defects in the absence of an agreement is generally wide. The applicable arguments vary from a strict liability for the seller for both direct and indirect losses of the buyer, to a lesser liability where the seller would only be liable for direct losses of the buyer if the seller had received a warning letter prior to the transaction, to no liability at all for the seller since copyrights are unregistered and such difficult for the seller to have knowledge about. Because of this normative space it is uncertain how much risk a party will carry when going into a software transaction without an agreement explicitly regulating the risks between the parties.

In this chapter the legal phenomenon used to distribute the liability is *freedom of contract*. From a legal constructivist approach the freedom of contract gives the parties a broad normative space where they can construct normative consequences in terms of obligations and liabilities. Because the normative space is wide in this creation it is not possible to regard every possible alternative there is to distribute liability for intellectual

---

[217] See Cornell University Law School. Contract, *Legal Information Institute*. 2015. Available at: [https://www.law.cornell.edu/wex/contract] Accessed 2015-07-17.

[218] Tim Boyce. *Commercial Risk Management : How to Identify, Mitigate and Avoid the Principal Risks in Any Project.* London: Thorogood Ltd, 1995, p. 9, see also Ulf Bernitz. *Standardavtalsrätt*. 8 ed. Stockholm: Norstedts Juridik AB, 2013, p. 21, who states that one of the purposes of the standard agreement is to allocate the risk between the parties. Even if he explicitly discusses standard agreements I would argue that this logic could be applied to software license agreements as well.

property defects by contract. In this section I will present different clauses inspired by agreements by commercial actors that affect the liability for intellectual property defects.

## 4.2 Regulating what the licensee can assume – warranties and warranty disclaimers[219]

As shown in the previous chapter about intellectual property defects, deciding whether an occurrence should be considered to be a defect or not is often determined by concluding what the buyer could reasonably assume about the transacted object. It is also apparent from the previous chapter that in the absence of an agreement it is often difficult to assess what a buyer can reasonably assume about the object when transacting software. Because of this it is often recommended that the parties agree on this matter in advance.[220] This can be achieved by using *warranty*- or *warranty disclaimer* clauses.

### 4.2.1 Warranties

Even if there are several definitions of a warranty, for example "*An assurance, promise, or guaranty by one party that a particular statement of fact is true and may be relied upon by the other party",*[221] or *"A warranty means that the seller in addition to what is apparent from the description of the item of purchase explicitly assumes responsibility for certain conditions [my translation]",*[222] there is no definition of a *warranty* in the European or Swedish laws. A warranty clause is often used to either (1.) indicate what should be considered to be a defect or (2.) regulate available remedies to a defect. In the former situation (1.) a warranty can be seen as regulating what the buyer should be able to reasonable assume about the transacted object.[223] A warranty clause is generally seen as a term with positive value for the recipient and such clauses can used to give the recipient additional rights in addition to the applicable law, for example by increasing the buyer's expectations of the good. [224] However sometimes they are also used to decrease the recipient's rights in comparison

---

[219] In Swedish the former is known as "garantier" and latter is known as "ansvarsfriskrivningar".

[220] Lindberg & Westman, p. 414.

[221] The Free Dictionary. Warranty, *The Free Dictionary*. 2015. Available at: [http://legal-dictionary.thefreedictionary.com/warranty] Accessed 2015-07-17.

[222] Christina Hultmark. *Kontraktsbrott vid köp av aktie*. Stockholm: Juristförlaget JF AB, 1992, p. 145.

[223] When using the terminology of the Swedish Sale of Goods act. See also Robert Sevenius. *Företagsförvärv*. Lund: Studentlitteratur AB, 2011, p. 304.

[224] Jan Ramberg & Christina Ramberg. *Allmän Avtalrätt*. 9 Ed. Stockholm: Norstedts Juridik AB, 2014, p. 226 ff. and Bernitz, p. 200.

with the legislation.[225] Supplying a warranty in a transaction will often generate normative consequences, for example if the seller provides the buyer with a warranty that is not fulfilled the seller has a strict liability towards the buyer for damages created by such defect in accordance with art. 40 in the Swedish Sale of Goods act. Warranties can also be used in license agreements, for instance if the seller warrants, through the agreement or their actions, that the buyer can assume that they should be able to use the licensed software freely without interference by third party rights, the presence of a claim made by a third party based on an intellectual property right in the software would be seen as a defect. An example of a warranty from the seller/licensor can be seen in *Example 1*.

> *1.1 The licensor warrants that it has the right to grant the licenses to the software in pursuant to this agreement.*
>
> *1.2 The warranty in 1.1 shall not apply if the software is not used in accordance with the documentation, or if the infringement is caused by a modification or add-on (other than a modification or add-on made by the licensor and which is provided through the licensors support).*

**Example 1.**

In clause 1.1 in *Example 1*, the licensor warrants that they have the right to license the software program, which indicates that the licensor owns all rights to the works in the software or that they have a license to such works that allows for sublicensing. Clause 1.2 is used to limit such warranty so that it does not apply if the licensee uses the software in other ways than agreed upon. This latter clause is useful in situations where the licensor has licensed in software and in such license agreement is bound to certain criteria that they would need to include in their own sublicense to the end user, e.g. commercial or non-commercial usage, it is also useful when the licensor wants to limit the warranty when the licensee has modified the software.

There are several effects of this warranty. The first is that the buyer should be able claim that the seller has breached the contract if the statement is not fulfilled, i.e. if the seller

[225] Ramberg & Ramberg, p. 227. Such use of warranties might be in violation of art 8 and 10 in the Swedish Marketing act (SFS 2008:486), see Bernitz, p. 121 and p. 200.

does not have the right to grant the license to the software. Since it is explicitly stated in the agreement it will be easier for the buyer to prove that this is a defect and a breach of contract. Secondly, since it is an explicit warranty the buyer is entitled to damages for both direct and indirect losses due to such defect.[226]

It is important to be specific when regulating warranties since vaguely written clauses increases uncertainty when the transaction does not enfold like intended.[227] If the clause is ambiguous it is difficult to determine what events is included in the warranty and which fall outside its regulation. For an example of an ambiguous clause regarding intellectual property defects, see *Example 2* below. This clause is vaguely written since it only states that the seller shall remedy or repair any defect in the goods resulting from faulty design, materials or workmanship, and in the next section it is stated that the seller disclaims all other liability for defects. Whether intellectual property defects is included in the sellers responsibility in this clause is uncertain.[228] It is uncertain since it is not clear whether an intellectual property defect is a *defect in the good resulting from a faulty design, material or workmanship* or if it should be seen as a defect falling outside of that scope. In the presence of an intellectual property defect the good is not *physically* defect, but the effect might be similar, i.e. that the buyer won´t be able to use the transacted object as intended. Another uncertainty with this clause is that even if intellectual property defects are included in the seller's responsibility, *repair or replacement* might not be a suitable remedy for the buyer. Even if the software is amended in such a manner that it no longer infringes in the third party copyright, the buyer's main concern is likely that they have had costs incurred due to that they would have to stop using their software system, or else risk facing a third party claim resulting in damages or additional license fees.

---

[226] This is subject to the application of art. 40(3) in the Swedish Sale of Goods act (SFS 1990:931). Even if the licensor is liable there is often a clause regulating a limitation on the liability, see limitation on liability below.

[227] Ramberg & Ramberg, p. 228.

[228] See Runesson, p. 625, for more information.

*1.1 The seller shall, by repair or replacement remedy any defects in the goods resulting from faulty design, material or workmanship.*

*1.2 Save as stipulated in clause 1.1 the seller shall have no liability for defects. This applies to any loss the defect may cause including but not limited to loss of production, loss of profit and any other consequential loss. This limitation of the seller liability shall, however, not apply if he has been guilty of gross negligence.*

**Example 2 – Inspired by clause 21 and 35 in the standard agreement NL 92 E**

### 4.2.2 Warranty Disclaimers

When the transaction does not enfold like the parties have intended, resulting in a negative outcome for one of the parties, it is common that injured party considers such occurrence a defect or a breach of contract to claim remedies. An agreed warranty disclaimer can be used as a tool to allocate the risk between the parties and limit such remedies.[229]

A warranty disclaimer can be defined as a clause where one of the parties limits their responsibility compared with their responsibility under the applicable law.[230] A warranty disclaimer can be used to (1.) indicate what should be considered to be a defect or (2.) regulate available remedies for a defect. As with a warranty clause, a warranty disclaimer clause can be seen as regulating what the buyer should be able to reasonable assume about the transacted object. The main difference between a warranty and a warranty disclaimer is that while a warranty gives the recipient an additional right in addition to the applicable law, a warranty disclaimer gives recipient fewer rights than compared to the applicable law.

---

[229] Ramberg & Ramberg, p. 229 ff., see also Thorsten Lundmark. *Friskrivningarsklausuler Giltighet och Räckvidd – Särskilt om friskrivning i kommersiella avtal om köp av lös egendom*. Uppsala: Iustus Förlag AB, 1996, p. 34 ff. and p. 77.

[230] Bernitz, p. 22, and also Lindberg & Westman, p. 500, and furthermore Leidö, p. 391.

Warranty disclaimers can regulate several topics in a transaction, however I will focus on such warranty disclaimers where the seller disclaims liability if the software infringes in third party intellectual property rights, i.e. a disclaimer of warranty for non-infringement. An example of such warranty disclaimer is seen in the end-user license agreement from Cisco, see. *Example 3*.

> "*Except as specified in this warranty section, all express or implied conditions, representations, and warranties including, without limitation, any implied warranty or condition of merchantability, fitness for a particular purpose, non-infringement, satisfactory quality, non-interference, accuracy of informational content, or arising from a course of dealing, law, usage, or trade practice, are hereby excluded to the extent allowed by applicable law and are expressly disclaimed by Cisco, its suppliers and licensors. To the extent that any of the same cannot be excluded, such implied condition, representation and/or warranty is limited in duration to the express warranty period referred to in the "limited warranty" section above. Because some states or jurisdictions do not allow limitations on how long an implied warranty lasts, the above limitation may not apply in such states. This warranty gives customer specific legal rights, and customer may also have other rights which vary from jurisdiction to jurisdiction. This disclaimer and exclusion shall apply even if the express warranty set forth above fails of its essential purpose.*"

**Example 3 Cisco EULA, Disclaimer of Warranty**

Since the focus of this thesis is intellectual property defects I have shortened the clause in Example 3 to only focus on the intellectual property defect, see *Example 4*, to make it more comprehensible.

> "*All express or implied conditions, representations, and warranties of non-infringement are hereby excluded to the extent allowed by applicable law and are expressly disclaimed by the Licensor.*"

**Example 4 – Shortened version of Cisco EULA, Disclaimer of Warranty**

The intended effect of this warranty disclaimer is that the buyer should not able to successfully argue that a claim from a third party based on an intellectual property right is a defect in the transacted software or a breach of contract. These types of clauses are common where the seller wants to limit their liability for intellectual property defects. It is common that software license agreements include warranty disclaimers, for example providing the software "as-is".[231] For an example of such a clause see *Example 5*. A piece of software provided on an "as-is" basis is often the most unfavourable disclaimer from the buyer's perspective.[232] However even if a warranty disclaimer has been entered into the agreement it can be uncertain whether the clause will have the desired effect since it can be argued that the seller in some occasions have an obligation to inform the buyer even if a disclaimer is used, see art. 19 (2) in the Swedish Sale of Goods act (SFS 1990:931) and Ramberg & Ramberg (2014) p. 230 - 231.[233] For instance even if a software has been provided with a disclaimer in the license agreement such disclaimer might be void if the seller had knowledge about the disclaimed defect and did not inform the buyer about such defect.

> "*No warranties: to the maximum extent permitted by law: the software, products and Skype websites are provided "as is" and used at your sole risk with no warranties whatsoever; Skype does not make any warranties, claims or representations and expressly disclaims all such warranties of any kind, whether express, implied or statutory, with respect to the software, products and/or Skype websites including, without limitation, warranties or conditions of quality, performance, non-infringement, merchantability, or fitness for use for a particular purpose. Skype further does not represent or warrant that the software, products and/or Skype websites will always be available, accessible, uninterrupted, timely, secure, accurate, complete and error-free or will operate without packet loss, nor does Skype warrant any connection to or transmission from the internet, or any quality of calls made through the software.*"

**Example 5 - Skype Terms of Use Clause 12.2**

---

[231] Leidö, p. 391.

[232] "As-is"can be translated to the Swedish expression "i befintligt skick". See also Ramberg & Ramberg, pp. 230 - 231.

[233] Ramberg & Ramberg, p. 231.

## 4.3 Regulating the amounts – liability disclaimers[234]

In addition to regulating what can be considered to be a defect and what remedies should be available if a defect is found, the parties can agree on an exclusion of certain liability, or a limit on the amount that a party can be liable for towards the other party. Such clauses are often called *liability disclaimers*.[235] In these types of clauses it is the liability for the economic costs or damages that a party have suffered due to other party's breach of contract that is in focus. This means that these clauses regulate a party´s obligation after a defect or breach of contract has been established. It is common that the parties limit the liability to a specific amount.[236] An example of a clause where the seller limits their liability to a fixed amount (a "cap") is seen in *Example 6*. In this example, Apple first limits their liability for all types of damages and if such limitation would be void they limit the liability to 50 U.S dollars.

> "*Limitation of Liability. To the extent not prohibited by applicable law, in no event shall Apple be liable for personal injury, or any incidental, special, indirect or consequential damages whatsoever, including, without limitation, damages for loss of profits, corruption or loss of data, failure to transmit or receive any data or information, business interruption or any other commercial damages or losses, arising out of or related to your use or inability to use the Apple software or services or any third party software or applications in conjunction with the Apple software or services, however caused, regardless of the theory of liability (contract, tort or otherwise) and even if Apple has been advised of the possibility of such damages. Some jurisdictions do not allow the exclusion or limitation of liability for personal injury, or of incidental or consequential damages, so this limitation may not apply to you. In no event shall Apple's total liability to you for all damages (other than as may be required by applicable law in cases involving personal injury) exceed the amount of fifty dollars ($50.00). The foregoing limitations will apply even if the above stated remedy fails of its essential purpose.*"

**Example 6 – Apple OS X Mavericks software license agreement clause 8**

---

[234] In Swedish known as "ansvarsbegränsningar", see Lundmark, p. 36.

[235] Leidö, p. 400.

[236] Ramberg & Ramberg, p. 232, and Bernitz, p. 171 – 175, and furthermore Lindberg & Westman, p. 500 ff.

The Swedish Sale of Goods act (SFS 1990:931) differs between direct and indirect losses and makes a general distinction between the two categories. In brief these rules state that a direct loss is directly related to the defect, e.g. the additional cost to buy a replacement for a malfunctioning product. Indirect losses are the losses due to that the buyer cannot use the transacted object, such as losses in production.[237] In commercial agreement it is common to deviate from this structure and choose an own distinction between reimbursable losses.[238]

### 4.3.1 Wilful misconduct or gross negligence

I will provide a short notion on limitation of liability for wilful misconduct or gross negligence. When limiting the liability an issue is whether the claimed damages are caused to the buyer by the seller's wilful misconduct or gross negligence (when the seller as software developer used *ad hoc code reuse* during the software development). If it would be deemed to be damages caused by the seller's wilful misconduct or gross negligence several jurisdictions, including Sweden, would declare any clause limiting such damages void.[239]

But the question is when ad hoc code reuse constitutes damages caused by wilful misconduct or by gross negligent nature?[240] One could argue that it would constitute damages caused by wilful misconduct to copy excerpts of someone else's source code, without permission, and implement it in their own software, since the act of copying is done wilfully. However not all code do automatically have copyright protection, as the code need to be the authors own intellectual creation and there are exemptions to the protection as seen above in chapter 2. Some source code can be argued to be standardised to a level were it is not the author's own intellectual creation of why it isn't protected by copyright, and therefore it is not an infringement when copied. If the developer copies the source code for a whole software they can almost be certain that the some parts of the copied source code are subject to copyright protection. The difficulty of the assessment is increased when regarding a situation where smaller excerpts of source code are copied. It is not certain that such copying constitutes a

[237] See art. 67 (2) in the Swedish Sale of Goods act (SFS 1990:931).

[238] Bernitz, p. 171.

[239] Ramberg & Ramberg, p. 233, and Leidö, p. 402 - 403, and Lindberg & Westman, p. 500 ff., and furthermore Lundmark, p. 133.

[240] See also section 2.6.3 above, regarding the assessment of negligence.

copyright infringement since it depends on what excerpts are copied. A difficult question is whether damages caused by such copying should be seen as caused by gross negligent behaviour, since such copying include the risk of copyright infringement. The problem is for the software developer is that they cannot themselves determine if the copied source code is subject to copyright protection, until a judge has given their judgement.

Although it would be interesting to conduct research to find where copying made by a software developer constitutes gross negligent nature or wilful misconduct, I will not go further into this subject.

## 4.4 Regulating whom will defend who

Besides regulating defects, remedies and liabilities the parties may also agree who shall defend whom if a third party makes a claim that an act constitutes an intellectual property infringement. This is often called that one party *indemnifies* the other party.[241] For this thesis these regulations will be called *indemnification clauses*, and regarding claims of intellectual property infringement: *intellectual property indemnification clauses*.

### 4.4.1 *Intellectual property indemnification*[242]

To indemnify can be defined as to *a process or arrangement under which a party will not suffer any loss.*[243] For the purpose of this thesis I focus on clauses where one party promises to give the other party monetary compensation if the latter gets sued for intellectual property infringement. An example of such clause where the licensor agrees to indemnify the licensee can be seen in *Example 7*.

[241] ContractStandards.com. Indemnification (License Agreement), *ContractStandards.com*. 2014. Available at: [http://www.contractstandards.com] Accessed 2015-06-16. Search for "indemnification".

[242] In Swedish agreements this is often identified under a clause named "intrångstalan".

[243] Wayne Courtney. The Nature of Contractual Indemnities, *Journal of Contract Law*, Vol. 27, No. 1, pp. 1-17, (2011).

*"SAP shall defend Licensee against claims brought against Licensee in the Territory by any third party alleging that Licensee's Use of the Software, in accordance with the terms and conditions of this Agreement, constitutes a direct infringement or misappropriation of such third party's patent claim(s), copyright or trade secret rights, and SAP will pay damages finally awarded against Licensee (or the amount of any settlement SAP enters into) with respect to such claims."*

**Example 7 – SAP Software License And Support Agreement - General Terms and Conditions clause 8.1 part 1.**

An indemnification clause often contain exceptions to limit the obligation to indemnify the other party, for an illustration see *Example 8,* which is the second part of Clause 8 in the SAP agreement.

*"This obligation of SAP shall not apply if the alleged infringement or misappropriation results from (i) Use of the Software in conjunction with any other software; (ii) Use of the Software with an apparatus other than a Designated Unit; (iii) failure e to promptly use an update provided by SAP if such infringement or misappropriation could have been avoided by use of the update; or (iv) any Use not permitted by this Agreement. This obligation of SAP also shall not apply if Licensee fails to timely notify SAP in writing of any such claim; however Licensee's failure to provide or delay in providing such notice shall not relieve SAP of its obligations under this Section except to the extent SAP is prejudiced by Licensee's failure to provide or delay in providing such notice. SAP is permitted to control fully the defense and any settlement of any such claim as long as such settlement shall not include a financial obligation on or admission of liability by Licensee. In the event Licensee declines SAP's proffered defense, or otherwise fails to give full control of the defense to SAP's designated counsel, then Licensee waives SAP's obligations under this Section 8.1. Licensee shall reasonably cooperate in the defense of such claim and may appear, at its own expense, through counsel reasonably acceptable to SAP. SAP expressly reserves the right to cease such defense of any claim(s) in the event the Software is no longer alleged to infringe or misappropriate, or is held not to infringe or misappropriate, the third party's rights. SAP may settle or mitigate damages from any claim or potential claim by substituting alternative substantially equivalent non-infringing programs and supporting documentation for the Software. Licensee shall not undertake any action in response to any infringement or misappropriation,*

**Example 8 – SAP Software License And Support Agreement - General Terms and Conditions clause 8.1 part 2**

The parties can also agree that the licensee should indemnify the licensor. This might be suitable when the software is based on the servers of the licensor and when the licensee can use the software to upload files to that server that might infringe in third party intellectual property rights. For an example of a clause where the licensee indemnifies the licensor see the terms for using Skype in *Example 9*.

> *"If any third party brings a claim against Skype in connection with, or arising out of (i) your breach of these Terms; (ii) your breach of any applicable law of regulation; (iii) your infringement or violation of the rights of any third parties (including intellectual property rights); (iv) your User Submissions or (v) your complaint in relation to any user submission, you will indemnify and hold Skype harmless from and against all damages, liability, loss, costs and expenses (including reasonable legal fees and costs) related to such claim."*

**Example 9 – Skype Terms of Use Clause 12.7**

## 4.5 Conclusion

> *How can the parties in a transaction handle the uncertainty found under question 1 and 2, with contractual tools, especially in end-user license agreements?*

As seen in this chapter there exists several means to regulate the liability for intellectual property defects between the parties by using the freedom of contract, however if the clauses are written in an ambiguous manner they might even increase the uncertainty for the parties. For this reason it is important that both parties consider how to regulate the liability while bargaining. To conclude I have shown three options used to construct a distribution of liability between the parties. The first option was to use warranties and warranty disclaimers to regulate what the buyer could reasonable assume about the transacted object. The second option was to regulate a limitation on the seller's liability for losses of the buyer, which included both categories of liability and a fixed maximum amount of liability. The third option was to regulate who should defend whom in the event of litigation.

According to Coase theorem successful bargaining, for example a distribution of liability between the parties through the clauses stated above, would provide for the most efficient resource allocation, given zero transaction costs. If these criteria is fulfilled the

uncertainty emerging due to copyright protection of software would be allocated in the most efficient manner by the parties even without the law explicitly regulating these matters. Regarding the ability to transfer risk by disclaimers in a perfect market with rational and informed parties, Lundmark states that "*In practice this would most likely as a rule mean that the risk is transferred to the party who can best assess the risk and its consequences or who can best disseminate or by other means, for example by means of insurance absorb, the economic implications of the risk* [my translation]".[244] In the next chapter I will analyse whether an efficient allocation of resources is a probable scenario given the nature of copyright protected software and the ability of the parties to reach a successful bargaining.

[244] Lundmark, p. 47 - 48.

# 5 EFFICIENCY - Given my findings in the previous chapters, are there risks for an inefficient allocation of resources?

As seen in chapter 2 – 4 there is a wide normative space in several issues relating to the copyright protection of the interests behind software development and the transactions and infringements in such copyright. These wide normative spaces leads to uncertainty, and as seen in the previous chapter, the parties to a software transaction have the opportunity to allocate this uncertainty between themselves in license agreements. In theory the risk should reside with the party that can best cope or absorb the risk.[245] In this chapter I will use economic theory to determine whom of the parties that should carry the risk of intellectual property defects. I will do this by determining what distribution of liability in a software transaction would create the most efficient allocation of resources and compare such distribution with my findings in the previous chapters. To define an effective allocation of resources for intellectual property defects I will analyse whether the parties have a unilateral or bilateral ability to take precaution against the risk and I will also analyse whether the parties are informed or uninformed about the likelihood of an intellectual property defect and the cost of such defect. This chapter does also determine whether a rule of *no liability*, *strict liability* or *negligence based liability*, would create the most efficient allocation of resources regarding intellectual property defects in software transactions.

## 5.1 Unilateral vs bilateral precaution

This measure concerns whom of the parties, or if both, have the ability to limit the likelihood of an intellectual property defect. When regarding whom of the parties that can take precaution we can divide this into different stages relating to the infringing act, for example: precautions during the development of software and precaution during the use of the software. I will also touch upon efficient allocation of resources during the defence against a claim of intellectual property infringement.

### 5.1.1 Precaution during the development of the software

The precaution during the development of the software relates to who of the parties have the ability to ensure that no copyrighted works are infringed during the

[245] See Lundmark, pp. 47 - 48 and Boyce, p. 6.

development of the software. When regarding this matter we must differ between customized software made from specifications delivered by the buyer and software that is developed solely by the software developer and then licensed to a buyer, the former called *customized software* and the latter *commercial-off-the-shelf software* (COTS software). The buyer of a COTS software program is seldom involved in the development of the software and as such has little or no information about the specific process in the development stage. Since the buyer is not involved in the development stage it is difficult for the buyer to control this stage. In contrast to developing COTS software, the buyer is usually better informed when they have sent specifications to the software developer and the software developer follows these instructions to develop the software.

However even in the situations where the buyer has provided some instructions, the control over whether a piece of software will infringe a third party copyright lies mainly with the seller, as software developer. It is the software developer who can control the creation of the software, including what policies the employees are applying in regards to code reuse and license compliance and it is the software developer who is in control to ensure that all rights are transferred from the employees and hired consultants to the software developer to enable sublicensing. The seller, as the software developer, can choose to educate the employees in code reuse procedures that would lessen the risk for infringements in third party copyright and the software developer could for example choose to use a "clean room" where the development of code is conducted in a clinical setting, restricted from external sources of third party source code.

### 5.1.2 Precaution during the use of the software

As said in section 2.6, the buyer will make an infringement each time they are using the software since a copy will be made in the computers memory. The buyer will have the ultimate control regarding their use of the software program and will be able to stop using the software program if they receive a warning letter. It can also be argued that it is easier for the buyer to investigate the copyright system and the criteria for protection and infringement of copyright in their own jurisdiction. However the buyer seldom has the possibility to inspect the original source code since the software is received after it has been compiled as an executable program, and even if they would be able to decompile the software they would not see the original source code but only a close resemblance to such code. If the buyer would have access to the source code they would also need to have access to all other pieces of copyright protected source code that the software

developer has seen or used, to be able to determine if the source code in the software program have been copied. This is not realistic. Even if the buyer would have access to the all other copyright protected source code they often lack the necessary knowledge to inspect and compare such source code against the source code in the software to determine if an infringement have occurred.

### 5.1.3 Efficient allocation of resources when defending against a claim

The party that is able to most efficiently defend against an infringement claim, is generally the seller since they have developed the software, they know the industry and they have the information to defend against such claim, e.g. the information if they have used clean rooms and what license arrangements has taken place. The buyer most often lacks the information necessary to efficiently investigate and defend against a copyright infringement claim. They lack this information since when the software developer has compiled the program into an executable file, the buyer will not be able to see the original source code and they won´t have access to the license agreements that regulate the transfer of copyrights during the development stage.

## 5.2 Informed vs. uninformed buyers

As discussed throughout this thesis, the buyer is most often uninformed about the risk of third party intellectual property claims in a particular software program, at least when these claims relate to copyright, why we can assume that they can´t properly consider this risk when choosing between competing software programs. As said by Cooter & Ulen, imperfectly informed customers doesn´t necessarily choose the most efficient product.[246] In this section I will do an analogy to the economic theory of consumer product injuries, adjusting the theory to software transactions and instead of an *injury* use the concept of *intellectual property defects*.

When using economic theory for the subject of this thesis we can argue that the buyer doesn´t necessarily choose to buy the most efficient software program, see Table 2. In this example the buyer chooses between purchasing Software 1 or Software 2. Software 1 is produced with less precaution in comparison to Software 2, meaning that the development of Software 1 might be achieved by applying ad-hoc code reuse, non-compliance with OSS licenses and without an active policy for transfer of rights from

[246] Cooter & Ulen, p. 227.

employees and contractors to the seller as licensor. In comparison the software developer behind Software 2, has implemented an active precaution policy for securing that they have the right to license the software. This policy might include education for employees in how to use third party source code like OSS, to apply the use of clean rooms and a implement a system to secure compliance with licenses and transfer of rights from employees and contractors to the licensor.

| **Cost of Software Program** | **A** | **B** | **C** | **D** | **E** |
|---|---|---|---|---|---|
| **Choice of licensee** | **Licensors average cost of production per unit** | **Probability of a intellectual property defect (a third party claim of infringement in IPRs)** | **Average loss if a third party claim occurs** | **Expected loss** | **Full cost per unit** |
| Software 1 | 20 Euro | 1/10 000 | 200 000 Euro | 20 Euro | 40 Euro |
| Software 2 | 25 Euro | 1/20 000 | 100 000 Euro | 5 Euro | 30 Euro |

Table. 2[247]

Column A represents the average cost of production per unit, i.e. per licensed software program.[248] Since the software developer behind Software 2 has invested in precaution their average cost of production per unit is greater than the average cost of production per unit for the software developer behind Software 1. Due to this investment, I have assumed that Software 2 will have an average cost of production per unit that is 125% of the average cost of production of Software 1.[249] Column B represents the probability of a intellectual property defect, i.e. that a third party will make a claim towards the buyer,

[247] All figures in this section are assumptions made by myself to prove an example.

[248] Normally it will cost more to develop the software program than it will cost to produce copies and transact these copies, why the average cost of production for each unit is the cost of development divided by number of licenses. For this model I have assumed that both Software A and Software B has the same amount of licenses, why the only difference is in development costs.

[249] 125% is my assumption for this example.

claiming that the buyer infringes the third party's intellectual property rights when using the software. Since the software developer behind Software 1 has not taken precaution I have assumed that it is twice as probable that an intellectual property defect will occur when using Software 1 compared to if the buyer would choose to use Software 2. Column C presents the average losses if a intellectual property defect occurs. For this example I have assumed that the expected loss would be twice as great for Software 1 compared to Software 2, this is based on the assumption that Software 1 due to less precaution during the development contains an increased amount of segments that potentially infringes third party rights compared to Software 2. Column D represents the expected loss and is calculated by multiplying the probability of an intellectual property defect (column B) with the average loss if an intellectual property defect occurs (column C). The full cost per unit (column E) is given by adding the seller's cost of production per unit (column A) with the expected loss (column D). By comparing the two software solutions we can observe that the full cost per unit is lower for Software 2 than for Software 1 why an efficient use of resources would be if the buyer chooses Software 2.

I assume perfect competition and rational market actors, why the market price of a unit would equal the seller's production cost plus the cost of the seller's liability.[250] Therefore under a rule of no liability for the seller, or where the seller excludes their liability for intellectual property defects through warranty disclaimers and/or limitation of liability clauses in the license agreement, the price of each unit will only be equal to the seller's production cost, i.e. 20 Euros for Software 1 and 25 Euros for Software 2. If the buyer would be perfectly informed they would know that they must carry the costs in the event of an intellectual property defect, and they would also know the probability and the cost of such defect. The rational buyer would therefore choose Software 2 since the total costs would be lower than if they would choose Software 1. The buyer would choose the *most efficient product*.

However as discussed above, the buyer is seldom informed of the software developers' development processes when choosing between software programs to license, especially for COTS-software. Because of this the buyer will not be able to determine whether the probability of an intellectual property defect is high or low for a certain type of software

[250] Cooter & Ulen, chapter 2 and p. 226.

compared to another. It is also difficult for the buyer to determine the average cost if an intellectual property defect occur, why they will not be able to correctly calculate the full cost per unit for a particular software. Because of this the buyer is *imperfectly informed*.[251] If the seller has limited their liability for intellectual property defects, which resembles a rule of no liability for the seller, there is a risk that the imperfectly informed buyer would overestimate, underestimate or disregard the probability or cost for an intellectual property defect associated with either Software 1 or Software 2, why they might choose the *less efficient product* Software 1, if they falsely perceive Software 1 as cheaper than Software 2.[252]

In comparison, if a rule of strict liability for the seller were introduced, either by contract or legislation, the price of each unit would equal the seller´s production cost plus the cost of the seller's liability, i.e. 40 Euros for Software 1 and 30 Euros for Software 2. In such circumstances, even the imperfectly informed buyer would choose Software 2 even if they would overestimate, underestimate or disregard the probability or cost for an intellectual property defect associated with Software 1 or Software 2. The conclusion is that the allocation of resources is more efficient under a rule of strict liability for the seller, rather than under a rule of no liability for the seller, when the buyer is imperfectly informed about intellectual property defects in software transactions.

## 5.3 Compared to the allocation of resources with and without agreement

In this section, I will compare the distribution of liability between the parties that I have found to be the most efficient allocation of resources, in section 5.1 and 5.2 above, with the possible distributions of liability found in the previous chapters, chapter 3 and chapter 4. I will start with a short analysis on whether the resources would be efficiently allocated in the absence of an agreement regulating liability for intellectual property defects before analysing a situation where such agreement exists.

### 5.3.1 Allocation of resources in the absence of an agreement

As I have shown in the chapter on risk distribution without an agreement, there are several possible normative claims regarding how to distribute the liability for intellectual

[251] Cooter & Ulen, p. 226.

[252] Ibid.

property defects in the absence of an agreement regulating this matter. Depending on which one of the solutions that would govern the transaction this will cause different allocations of resources. Some of these solutions will position the liability of intellectual property defects with the seller, which in accordance to this chapter would be an efficient allocation of resources, however several of the alternative normative claims would also restrict the liability of the sellers, which in contrast risks an inefficient allocation of resources.

Since it is uncertain to what extent the seller is liable or not for an intellectual property defect in a software transaction, I can assume that several sellers would underestimate or disregard the probability of them being liable for such a defect. Because of this, the price of each their software programs would only be equal to the seller's production cost of that software, not regarding the cost of their possible liability. Software programs offered from such sellers would have a lower market price compared with software programs offered from sellers that would account for their possible liability as well as their production costs, all other things equal. As shown above, the buyer is generally imperfectly informed, why they risk choosing the *less efficient product.*

### 5.3.2 Allocation of resources when using an agreement to regulate the liability

As I have shown in chapter 4, the parties can distribute the liability of intellectual property defects between themselves or towards one of the parties. According to Coase theorem, this would be a satisfactory solution given zero transaction costs since the parties would allocate the resources efficiently through successful bargaining. However if there exists transactions costs these might hinder successful bargaining and an efficient allocation of resources.

#### 5.3.2.1 Transaction costs in software transactions

According to Coase theorem, rational parties will allocate legal entitlements in the most efficient way, given zero transaction costs.[253] This means that it would not matter how the legal entitlements are distributed between the parties from start since they will redistribute the resources through successful bargaining,[254] i.e. agreement. As said in the

[253] Cooter & Ulen, p. 85 and p. 291.

[254] Ibid., p. 84.

section on economic theory, transaction costs can be divided into three subcategories: search and information costs, bargaining and decision costs, policing and enforcement costs.[255]

The main hindrance for successful bargaining, in transactions of software, as seen above in section 5.2, is that the buyer is not informed about whether the software infringes in third party copyrights or not, why the buyer will not be able to make an informed decision about total cost of product, and thus which of the software programs that would be the most efficient product. This means that even if the parties are able to reach an agreement and transact the software under agreed terms, including warranties and disclaimers to distribute the liability, there is a risk that this is not the most efficient allocation of resources due to uninformed buyers. The imperfectly informed buyer could still overestimate, underestimate or disregard the probability or cost for an intellectual property defect associated with the offered product when the seller has disclaimed liability for intellectual property defects, why the buyer might choose the *less efficient product*.

To be able to have an informed buyer who could estimate the total cost of product, both parties would have to understand what is *transacted*. This information is linked to the first category of transaction costs, *search and information costs*, which include costs for finding information about the transacted product, including the quality of the product. This relates to what I have shown in chapter 2 about how uncertainty emerges in the protection of software by copyright. If it would be less difficult to determine to what extent elements in software are owned, protected, transacted and infringed, it would be less difficult for both parties to understand and communicate what would be transacted in a software transaction, why such change would lessen the transaction costs. With less transactions costs it would be less difficult for the parties to reach a successful bargaining and an efficient allocation of resources.

#### 5.3.2.2 The rational behind distribution of liability

As seen above, transaction costs might hinder successful bargaining. From a legal constructivist approach it is also interesting to analyse the logic of the actors in the market regarding their behaviours in the distribution of liability.

---

[255] Dahlman, pp. 141 - 162.

According to Jimmy Ahlberg,[256] companies are usually risk averse and will try to avoid taking on additional liabilities if it at all can be helped, the ones that are willing to carry risk are typically two types of companies. The first type is those types of companies whose business model is to provide indemnities, for example the Linux distributer Red Hat. From these actors you can download the software for free but you will pay for the indemnity clause and for the software support. These business models are similar to an insurance provider business model. The second type are smaller companies that are inferior to a larger counterpart. The logic for these actors to commit to an indemnity clause towards the larger actor is that the deal is seen as so important for the smaller actor that they regard the indemnity as an acceptable risk. If they would have to indemnify the larger actor, after an infringement claim, it is likely that they would go bankrupt why they would not be able to pay the indemnity anyway.[257]

Runesson mentions that it is common that the responsibility for intellectual property defects depends on who writes the agreement. If the seller writes the agreement he or she often disclaims all responsibility. If the buyer writes the agreement the seller takes all responsibility.[258] Olsson, however, states that it is common that the licensor warrants that they have the right to license the software and that the licensed work will not infringe in third party rights.[259]

It seems that even in the event that an agreement is used to regulate the liability for intellectual property defects, it is common that the both parties' tries to avoid such liability. While searching for different contract clauses for chapter 4, I did not find any agreement where the licensor made a warranty that they had the right to license the software, a type of clause commonly used according to Olsson.[260] However SAP did provide an indemnification clause, which would make SAP, as the seller, liable for

---

[256] Jimmy Ahlberg works as Defense IPR strategist at Ericsson AB.

[257] Jimmy Ahlberg. Interview 2015-06-10. The interview was conducted in Swedish why I have translated Mr Ahlberg's answers. Mr Ahlberg approved of the text after amendment 2015-07-04.

[258] Runesson, p. 625.

[259] Olsson, 2015, p. 238.

[260] Olsson, 2015, p. 238.

intellectual property defects under the circumstances stated in *Example 7* and *Example 8* in section 4.4 above.

## 5.4 Conclusion

> *Is it so that the uncertainty associated with our current copyright and contractual system risks an inefficient allocation of resources, and although I conclude that it is possible to manage such uncertainty (as demonstrated in the previous question) with the contractual tools, the ability of the parties to be informed and take precaution still risks an efficient allocation of resources?*

As seen in this chapter the most efficient way to allocate the resources in regards to ensuring that software programs does not infringe third party copyrights is by holding the seller liable for intellectual property defects, mainly because they have the control over the development process and as such can take the necessary precaution during the development process to lower the risk of intellectual property defects. As seen it is also difficult for the buyer to take any precaution against an intellectual property defect during the development and/or use of the software. Since the buyer is not able to take precaution it is reasonable to make the liability strict for the seller rather than based on negligence. The seller should also carry the liability since the buyer is generally imperfectly informed about the likelihood of an intellectual property defect and the costs for such a defect, why they can´t make an informed decision about the total cost of product, which creates a risk of inefficient allocation of resources.

In this chapter I have also shown that there is a risk of an inefficient allocation of resources regardless of whether the parties has chosen to regulate the liability of intellectual property defects in an agreement or not. In the absence of an agreement regulating this matter the buyer/licensee risk end up carrying the risk for intellectual property defects to a varying extent since there is no reified legal construction for intellectual property defects in the Swedish jurisdiction. Because of this uncertainty some sellers would disregard their possible liability when calculating the price of the product, why they would offer the software at a price that would not reflect the total cost of product. Because of this the imperfectly informed buyer would risk choosing the less efficient product, which would lead to an efficient allocation of resources.

When the parties chose to use an agreement to distribute the liability, uncertainties relating to ownership and third party rights, as shown in chapter 2, combined with transaction costs due to search and information costs, hinder successful bargaining. The parties might reach an agreement and transact software under terms that limit the liability for the seller and the uninformed buyer will not be able to assess the total cost of the software why there is a risk that the buyer choses a less efficient product. In both situations there is a risk for an inefficient allocation of resources. In the next chapter I will provide two possible legislative solutions that would affect the allocation of resources towards an increased efficiency.

# 6 SOLUTION – Given that I found that the allocation of resources risk being inefficient, what could be a solution to achieve a more efficient allocation of resources?

## 6.1 Introduction

As seen in the previous chapter, there is a risk for inefficient allocation of resources if the risk of intellectual property defects resides with the buyer/licensee. Because of this I will present a proposal of a legislation that would aim to allocate the risk for intellectual property defects with the seller/licensor. The purpose of this legislation is to stipulate a guideline for the allocation of liability between licensor and licensee in software transactions. In chapter 2 on protection of software by copyright I also provided an argument that it is uncertain how the rights for elements that are not classified as computer programs under the Swedish Copyright act (SFS 1960:729) are transferred to the employer in the absence of an agreement. To lower this uncertainty I argue to broaden the transfer of rights in relation to computer programs and ultimately software by amending the current art. 40a in the Swedish Copyright act (SFS 1960:729).

The proposed articles relate to each other since the first proposed article will increase the liability for the seller for intellectual property defects, and to weigh against the increased liability, the second proposed article would automatically transfer the copyrights to the employer, which would decrease the likelihood of intellectual property defects. The purpose of suggesting these two following articles is therefore that they will balance each other to some extent.

## 6.2 One new article and a change in an existing article

I suggest that one new article is introduced in the Swedish Copyright act (SFS 1960:729) whose purpose is to allocate the resources efficiently, see *Example 10*. As I have concluded above, the allocation of resources is more likely to be efficient under a rule of strict liability for the seller of software programs even if this places an uncertainty on the seller. In this new article the seller/licensor should be liable for intellectual property defects, unless they have informed the buyer/licensee of any third party claims in the computer program. I also suggest that a change is made in art. 40 a in the Swedish Copyright act (SFS 1960:729) to increase the transfer of copyrights to the employer. The purpose of this change is to decrease transaction costs during software development and

later transactions of such software, which would facilitate successful bargaining, see *Example 11* below. Both of these articles will be optional, meaning that the parties will be able avoid the regulations by agreement.

### 6.2.1 Allocation of resources through a new article

In the first paragraph I choose to construct a strict liability for the seller/licensor for intellectual property defects in licensed software, see *Example 10*. This construction is inspired by art. 41 in the Swedish Sale of Goods act (SFS 1990:931) on *legal defects*. Several authors are against a strict liability for intellectual property defects, often based on the argument that it is difficult to foresee whether a transacted object would infringe in intellectual property rights in another jurisdiction. However regarding copyright protected software the licensor as software developer can be fairly certain that their software will not infringe in third party copyrights as long as one avoid verbatim copying of others source code and ensure license compliance. Because of this I have chosen to limit the scope of this clause to copyright and especially to elements protected by copyright as *computer programs*.

> **X §** Om tredje man gör anspråk på upphovsrätt till ett verk i ett licenserat datorprogram (immaterialrättsligt fel baserat på upphovsrätt) och det inte följer av avtalet att licenstagaren skall licensera datorprogrammet med den begränsning som tredje mans rätt medför, gäller bestämmelserna om reklamation i 32 § första stycket köplagen och 33 § köplagen, om avhjälpande och omleverans i 34--36 §§ köplagen, om prisavdrag och hävning i 37--39 §§ köplagen, om skadestånd i 40 § köplagen samt om licenstagarens rätt att hålla inne betalningen i 42 § köplagen.
>
> Om inget annat har avtalats så har licenstagaren endast rätt till ersättning för den skada han lider genom ett immaterialrättsligt fel baserat på upphovsrätt som förelåg vid köpet, om han varken kände till eller borde ha känt till felet.
>
> Påföljder av immaterialrättsligt fel baserat på upphovsrätt får även göras gällande, om tredje man påstår att han har en sådan rätt som avses i första stycket och det finns sannolika skäl för påståendet.

**Example 10 – A new article in the Swedish Copyright Act**

As said above it is important to differ between a situation where the seller/licensor has done the sole contribution of the development and a situation where the buyer/licensee has been part of the development as well, e.g. when the buyer provides instructions on the development. Because of this I have included a second paragraph in the article were the seller/licensor is not liable if the buyer/licensee has been aware about the intellectual property defect. In such situation I assume that the buyer/licensee will be able to make a better assessment of the total cost of the product, and in such situations it is not reasonable with a strict liability for the seller. This second paragraph would also incentivize the seller to inform the buyer about eventual intellectual property defects prior to the transaction. If the seller informs the buyer, the buyer should be able to make an informed decision on whether they would like to buy the software and carry the risk why a strict liability for the seller is not as necessary for an efficient allocation of resources.

For the regulation to have a real impact it is important that a buyer can make the seller liable without having to verify the intellectual property defect in court. The process to verify an intellectual property claim in court often takes several years and would be costly for a buyer. In addition it might not be a reasonable decision for the buyer to defend against a claim from a third party since it might be more expensive for the buyer to defend against the claim than to pay the damage for an infringement and a license to a third party. It is unreasonable that the buyer would be unable to hold the seller/licensor liable in such situation only due to the fact that the third party claim is not verified in court. The third paragraph is hence introduced to handle these situations. In accordance with the third paragraph it would be enough that a third party make a claim that they have a right in the licensed software, e.g. by sending a warning letter or similar, for the buyer to be able to hold the seller/licensor liable for the such claim. I believe that this is an important measure to be able to coup with possible non-practising entities with litigious behaviours.

### 6.2.2 Decreased transaction costs through a change in art. 40a in the Swedish Copyright act

The second legislative solution that I propose relates to an increase in copyrights transferred to the employer in commercial software development projects. The solution involves a modification in art. 40a in the Swedish Copyright act (SFS 1960:729), see *Example 11*. As said in section 2.5.1 above, the current art. 40a in the Swedish Copyright

act (SFS 1960:729) only transfers the copyright for elements claimed as *computer programs* from the employee to the employer, leaving the copyright to other elements in the software with the employee. This causes uncertainty regarding the rights to those elements between employee and employer, especially regarding the right to sublicense and modify those elements since art. 28 in the Swedish Copyright act (SFS 1960:729) hinder such actions in the absence of an agreement. This uncertainty affects the likelihood of intellectual property defects in transactions between the seller and the buyer. The purpose of the suggested article is therefore to transfer the copyrights regarding other elements in the software program to the employer as well,[261] i.e. the elements that are protected under copyright but not as a *computer programs,* for example the GUI or the databases. By transferring the copyrights regarding the other elements to the employer, difficulties with art. 28 in the Swedish Copyright act (SFS 1960:729) can be avoided. The desired effect of this new article is that uncertainties regarding the ownership of copyright, the right to modify and the right to sublicense the works in commercial software development would decrease. Since it would be less uncertain who hold these rights it would lead to decreased transaction costs in software transactions as both seller and buyer can presume that the ownership of the copyright have been transferred to the employer automatically, why there would be less need for ownership clarifications and agreements allowing for sublicenses and modifications. Since search and information costs would decrease this would lead to an increase in successful bargaining and thus an increase in efficient allocation of resources.

> **40 a §** Upphovsrätten till ett datorprogram, *eller ett annat verk eller närstående rättighet, som har skapats för att ingå tillsammans med ett datorprogram*, som skapas av en arbetstagare som ett led i hans arbetsuppgifter eller efter instruktioner av arbetsgivaren, övergår till arbetsgivaren, såvida inte något annat har avtalats.

**Example 11 – A change in article 40a§ in the Swedish Copyright Act, the amendment is shown in Italic style.**

[261] I use the word ”transfer” to indicate that there is a full transfer of the copyright, i.e. both the economic and moral right should reside with the employer.

The suggested article would be aligned with the underlying interests of copyright protected software. In the section regarding the history of copyright protected software, I mentioned that the rational for protecting software was to protect investments made in software development and to increase the quality of software programs. By transferring the copyright from the employee to the employer, the copyrights will be owned by the entity where the monetary investment into quality software generally is made, i.e. the company. This is also mentioned in SOU 2010:24, "*Moreover, computer programs are usually developed for commercial purposes that involves large investments. It is essential for the employer's ability to exploit their computer program that it can be modified and tailored to meet such requirements that may arise in the marketing and utilization of the program. This might be difficult if the moral rights remains with the author. It is therefore reasonable to have the moral rights transferred to the employer, and it is also reasonable for employees in a business, where the development of computer program is a natural part of their job, to accept that his or her programs are subject to modifications* [my translation]".[262] These arguments apply to the other elements in a software program as well, and not just the source or object code, why I argue that it is natural that the copyright to these elements should be transferred in full to the employer.

To limit the scope of this new article I have included in the suggested article that such elements are only transferred as long as they have been developed to be included with a *computer program*. This means that the copyright to a database is not transferred automatically if the database is not created with the purpose to be included with a computer program, even if the database is *later* included with a computer program. Thus the intention when creating a work determines whether the copyright to the work will be transferred to the employer in accordance with the new article. When an employee in a software development project develops an element, which should operate with a computer program, it should generally be considered that the intention is that the element should be included with the computer program. My suggested legislative construction with its limitation, already exist in the Finnish jurisdiction where both the copyright to the computer program and the copyright to other works, that are in direct connection with the computer program, are transferred in full to the employer, see art. 40b in the Finnish Copyright act.

---

262 SOU 2010:24. *Avtalad upphovsrätt*, pp. 152 – 153.

## 6.3 Consequences

It is likely that the two new articles will give rise to some consequences that has not been touched upon. I will briefly touch upon some consequences that I have estimated could arise.

### 6.3.1 Consequences of a new article

I will start with the consequences of the new article that would allocate the liability for intellectual property defects with the seller/licensor (*Example 10*). An expected consequence is that the price for software programs for the end-user would increase. This will happen because of two reasons; the first reason is that it will be more common than it is today for the seller to internalize the full cost of the software program, i.e. both the production cost and the cost for liability. The seller/licensor would therefore have to allow for costs of liability claims when transacting the software to end users why the price for the software program will increase. The second reason for an increased price for software programs is that the seller will need to take precaution while developing the software. This would typically mean that the seller would have to invest in educating their employees about copyright in software, create policies and use clean rooms during software development, to ensure compliance with licenses and to avoid copyright infringements while reusing source code and other elements. Such precaution will lead to increased production cost for the seller due to increased development times, and because of this the price of the software program for the end user would also increase.

The costs in the first and second reason will most likely counteract each other, i.e. if the seller invests in precaution during the development of software it would be less likely that the software program infringes in any third party copyrights why the seller could expect less expenses for liability claims. As said, both costs would be transferred to the buyer through an increased price on software programs why it would be the buyer who would carry the ultimate costs for precaution and increased liability for the seller. However the cost would be spread across a community of buyers rather against a single one why this solution would spread the risk.[263] Although, the buyers would be able to compare the full cost of the software programs and choose the most cost-efficient software among competing products, which would lead to an efficient allocation of resources.

---

[263] Calabresi pp. 500 - 501.

### 6.3.2 Consequences of changing art. 40a in the Swedish Copyright act

A consequence of extending the rights transferred in art. 40a in the Swedish Copyright act (SFS 1960:729) to include the copyright of other elements outside the category of *computer programs*, is that it would decrease the uncertainty whether the employer holds all necessary rights to sublicense and modify a copyright protected element in a piece of software, especially in the absence of an agreement. This would lead to less transaction cost since the employer would not have to ensure through contracts that the copyright to such elements are licensed to the employer. In addition the employer would also be able to avoid the hindrance in art. 28 in the Swedish Copyright act (SFS 1960:729) to sublicense and modify the work when transacting the software. This amendment to the article would thus make the costs to ensure that all copyrights are transferred decrease. Since costs of production would decrease it is possible that this legislation could decrease the price of software towards the buyers, even if only to a small extent. This might balance the above-suggested article regarding liability for the seller/licensor, by decreasing the costs of precaution.

## 6.4 Discussion and counterarguments

It should be noted that my recommendation to increase the copyrights transferred to the employer in art. 40a in the Swedish Copyright act (SFS 1960:729) is differing from Roséns recommendations in SOU 2010:24. I will briefly discuss his arguments. Rosén argues for a decrease of copyrights transferred regarding computer programs and that a change in the current art. 40a in the Swedish Copyright act (SFS 1960:729) should be made, where the moral rights should reside with the employee rather than being transferred to the employer. The rational for Rosén to strengthen the rights of the author is that the authors, he argues, has a difficult time to claim their rights and that the attitude towards copyright in society has increased this hardship. As a result we should return to the fundamental interests behind copyright and especially the moral right.[264] I find it interesting that he doesn´t mention the other fundamental interest behind copyright, i.e. the interest to protect the economic investments in intellectual creation. As mentioned above, the interest to protect the economic investments in intellectual creation is specifically mentioned as the logic for the copyright protection of software in the preparatory works.[265] In contrast the moral right for the author is only discussed to a

[264] SOU 2010:24. *Avtalad upphovsrätt*, p. 173.

[265] See SOU 1985:51. *Upphovsrätt och datorteknik*, p. 44 ff.

lesser extent.[266] It is also interesting that Rosén argues that the author would have less of a difficulty defending their rights if they would retain the moral rights. I would argue that it is commonly difficult for an author to claim their moral right in software programs since the counterpart is often a large company with great financial means, and a sole author seldom has the economic strength to prove and litigate their rights against such an actor. My belief is that strengthening the moral rights of the author in regards to computer programs would not help the authors to claim their rights but only decrease the respect for the copyright system as a suitable mean to protect economic investments in software development.

Rosén further states that the rights that are necessary for the employer in software development and software transactions, such as the right to modify and sublicense the work, are already governed by custom, implicit consent or express agreements why art. 28 in the Swedish Copyright act (SFS 1960:729) is not a hinder for the employer.[267] He briefly discusses the economic consequences of his suggestion, stating that his suggested amendment would only affect companies and individuals to a small extent when reviewing the employment agreements.[268]

My belief is that if Rosen's proposal would be transformed into legislation it would increase the uncertainty for the parties concerned, as the difficulty to determine which rights are licensed and/or transferred to the employer would increase. Even if the copyrights can be agreed upon, with Roséns suggestion there will be uncertainties since there will be a normative space regarding to what extent the employer will be allowed to modify, sublicense, or change the work and the employer would also have the burden of proof that the rights have been licensed from the employee which increases this uncertainty. These uncertainties would create increased transaction costs, relating to search and information costs for the parties, which would hinder successful bargaining and decrease efficient allocation of resources. I also believe that Roséns suggestion would increase the possibility of intellectual property defects since the copyrights to different elements in a piece of software will reside with several different authors. An obvious issue, that I cannot answer, is whether the authors will decide to use the

---

[266] Ibid., pp. 81 - 82.

[267] SOU 2010:24. *Avtalad upphovsrätt*, p. 176.

[268] Ibid., p. 298.

normative space to claim their rights against their employer and/or the end-user of the software. A somewhat pessimistic vision is that the current market trend, with patent and copyright trolls emerging, has increased the option for these authors to decide to partner with such NPE and enforce their rights to gain additional revenue streams. As I have touched upon, such behaviours is often more profitable if one choses to enforce the rights against end-users rather than the employer or licensor why under Rosen's suggestion we might see an increase in litigation towards end-users.

## 6.5 Conclusion

> *If I under question 4. will find that there is a risk of an inefficient allocation of resources, what could be a suitable legislative measure to increase the efficient allocation of resources?*

In this chapter I have provided two suggested articles that would increase the efficient allocation of resources by altering the liability of market actors and by transferring ownership of copyrights to increased number of works from the employee to the employer in software development projects. A probable consequence if both of these changes would be introduced is that software programs would have a slight increase in market price. The new article that would allocate the liability with the licensor would increase the market price, while the amended art. 40a in the Swedish Copyright act (SFS 1960:729) would slightly decrease the market price. As seen in the discussion, there are opinions that are against my suggested amendment of art. 40a in the Swedish Copyright act (SFS 1960:729), since they argue for a decrease of copyrights transferred from the employee to the employer. I believe that the suggested amendment in SOU 2010:24 would increase the uncertainty relating to ownership of copyright and because of this lead to increased transaction costs and a decrease in successful bargaining. In the long-term perspective the suggested change might lead to a decrease in reliance on copyright as an efficient mean to protect the interests behind software development.

# 7 Concluding remarks

The flexibility of the copyright system has ensured that it has made it trough the technological advancements of society during the last century. The flexibility is seen in the criteria for assessment such as the originality criterion and the idea/expression dichotomy, where the courts have a large freedom to decide case-by-case if copyright should reside in a work or not. However because of this flexibility, the legal construction of copyright protected software has not yet become reified to a large extent in society and there is a wide normative space in several of the issues regarding the protection, transaction and infringement of copyright protected software. This is in part because software is a phenomenon much different from the "classic" copyright subjects, such as music or paintings, why analogies to case law and the preparatory works are not always suitable. Thus when copyright is applied to software programs there are several alternative normative claims about what the legal construction of software programs, i.e. there is a wide normative space.

A wide normative space is not fully undesirable since it means that the courts will be able to be flexible and make decisions on case-by-case basis, which might provide for fair verdicts. However as shown in this thesis the wide normative space regarding the legal phenomena of the protecting, transacting and infringing copyright protected software creates an uncertainty for the actors in the market since it is difficult to assess the risk of third party claims based on ownership of copyright. Such third party claim can, under certain circumstances, be defined as an intellectual property defect in the transacted good. However intellectual property defects are only reified to some extent and it is not certain how the liability for such defects is distributed between the parties in the absence of an agreement. Even when the parties use an agreement to distribute the risk, it is likely that they will not be able to make informed decisions about the risk. These uncertainties cause transaction costs that hinder successful bargaining and an efficient allocation of resources.

If copyright protected software does not become reified in society to an extent where the actors in society feel that they can rely on the protection given it is possible that we will see trends where the interests behind software development are protected by other legal phenomena, such as by patents or as trade secrets. Such normative claims might provide the actors with a less uncertain protection, however this should be subject for further

research. We might also see technical solutions to protect the interests behind software development, for example cloud-based solutions were the software is not downloaded or copied, but accessed through an Internet based interface or advanced DRM techniques.

# 8 Further research

During my work with this thesis I have found subjects where it would be interesting to conduct further research. One such subject is the intersection between patents, software and copyleft licenses. In my thesis I have provided the reader with illustrations where open source software has allegedly been copied into the software of a commercial party and as such third parties has claimed the right to receive and use the source code in accordance with the open source license, often the GPL license. However it would be interesting to research whether if such third party would be able to claim a right to license a patent, from the commercial party without license fees, that reads on the source code in accordance with the copyleft licenses. If it would be the case the importance to ensure license compliance for software providers with valuable patent portfolios should not be underestimated.

Since this research is centred on copyright protection of software programs it would be interesting to broaden the research and conduct a comparative study of several different means of protecting the interests behind software development, to find the best alternatives that would also provide for an efficient allocation of resources. It would also be interesting to conduct a comparative study of the protection of computer programs between different jurisdictions and draw conclusions from such study.

# 9 Bibliography

## 9.1 Litterature


- Andreasson, Jens. *Intellektuella resurser som kreditsäkerhet: En förmögenhetsrättslig undersökning*. Göteborg: Chalmers tekniska högskola, Reproservice, 2010.
- Berger, Peter and Luckmann, Thomas. *The social construction of reality - A Treatise in the Sociology of Knowledge*. London: Penguin Books, 1991.
- Bernitz, Ulf. *Standardavtalsrätt*. 8 ed. Stockholm: Norstedts Juridik AB, 2013.
- Bernitz, Ulf., Karnell, Gunnar., Pehrson, Lars., and Sandgren, Claes. *Immaterialrätt och otillbörlig konkurrens*. 10 Ed. Stockholm: Jure, 2007.
- Boyce, Tim. *Commercial Risk Management : How to Identify, Mitigate and Avoid the Principal Risks in Any Project*. London: Thorogood Ltd, 1995.
- Brinkmann, Svend. & Kvale, Steinar. *Interviews – Learning the craft of qualitative research interviewing*. 3rd Ed. Thousand Oaks, California: SAGE Publications, Inc., 2015.
- Bryde Andersen, Mads. *IT-Retten*. Publisher: Author, 2001.
- Cooter, Robert. & Ulen, Thomas. *Law and Economics*. 6th Ed. Boston: Pearson Education, 2012.
- Glavå, Mats. & Petrusson, Ulf. Illusionen om rätten! – juristprofessionen och ansvaret för rättskonstruktionerna. In *Erkjennelse og engasjement : minnesseminar for David Roland Doublet [1954-2000],* Askeland, B & Bernt, J.F. (eds.). Bergen: Fagbokforlaget, 2002.
- Hultmark, Christina. *Kontraktsbrott vid köp av aktie*. Stockholm: Juristförlaget JF AB, 1992.
- Karnell, Gunnar W. G. European Originality: A Copyright Chimera. In *Intellectual Property and Information Law, Essays in Honour of Herman Cohen Jehoram,* Jan J.C. Kabel, Gerard J.H.M. Mom (eds.). Kluwer Law International, 1998.
- Lantz, Annika. *Intervjumetodik*. 2d Ed. Pozkal: Studentlitteratur, 2007.
- Leidö, Jan. *Realizing the single software market – cross-national validity of software license agreements*. Uppsala: Iustus förlag AB, 2014.
- Levin, Marianne. *Lärobok i immaterialrätt – Upphovsrätt, patenträtt, mönsterrätt, kännetecknensrätt – i Sverige, EU och internationellt*. 10 Ed. Stockholm: Norstedts Juridik AB, 2011.

- Lindberg, Agne. & Westman, Daniel. *Praktisk IT-rätt*. 3rd Ed. Stockholm: Norstedts Juridik AB, 2004.
- Lundmark, Thorsten. *Friskrivningarsklausuler Giltighet och Räckvidd – Särskilt om friskrivning i kommersiella avtal om köp av lös egendom*. Uppsala: Iustus Förlag AB, 1996.
- Olsson, Henry. *Copyright : svensk och internationell upphovsrätt*. 9 Ed. Stockholm: Norstedts Juridik AB, 2015.
- Olsson, Henry. *Copyright : svensk och internationell upphovsrätt*. 6 Ed. Stockholm: Norstedts Juridik AB, 1998.
- Petrusson, Ulf. *Intellectual Property & Entreprenuership – Creating Wealth in an Intellectual Value Chain*. Gothenburg: Center for Intellectual Property Studies (CIP), 2004.
- Ramberg, Jan., & Ramberg, Christina. *Allmän Avtalrätt*. 9 Ed. Stockholm: Norstedts Juridik AB, 2014.
- Ramberg, Jan., & Herre, Johnny. *Allmän köprätt*. 7 Ed. Stockholm: Norstedts Juridik AB, 2014.
- Ramberg, Jan., & Herre, Johnny. *Köplagen – En kommentar*. 2 Ed. Stockholm: Norstedts Juridik AB, 2013.
- Rosén, Jan. *Upphovsrättens avtal : regler för upphovsmäns, artisters, fonogram-, film- och databasproducenters, radio- och TV-bolags samt fotografers avtal*. 3rd Ed. Stockholm: Norstedts Juridik AB, 2006.
- Runesson, Eric M. Immaterialrättsliga fel vid köp. In *Festskrift till Gunnar Karnell*. Gorton, Lars (ed.) Stockholm: Carlsson Law Network, 1999.
- Searle, John. *The construction of social reality*. New York: Simon and Schuster, 1995.
- Sevenius, Robert. *Företagsförvärv*. Lund: Studentlitteratur AB, 2011.
- Sim, Susan Elliott. & Gallardo-Valencia, Rosario E. Introduction: Remixing Snippets and Reusing Components. In *Finding Source Code on the Web for Remix and Reuse*, Sim, Susan Elliott. & Gallardo-Valencia, Rosario E (eds.). Pages 1 – 14, New York: Springer-Verlag, 2013.
- Sim, Susan Elliott. & Stenberg, Erik B. Intellectual Property Law in Source Code Reuse and Remix. In *Finding Source Code on the Web for Remix and Reuse*, Sim, Susan Elliott. & Gallardo-Valencia, Rosario E (eds.). Pages 311 – 322, New York: Springer-Verlag, 2013.

- Sunstein, Cass R. & Nussbaum, Martha C. *Animal rights: current debates and new directions.* USA: Oxford University Press, 2005.

## 9.2 Articles

- Agresti, William W. Software Reuse: Developers' Experiences and Perceptions, *Journal of Software Engineering and Applications, 2011, 1, 48-58,* (2011).
- Bing, Jon. Vurdering av opphavsrettslig krenkelse av datamaskinprogrammer et praktisk perspektiv, *Nordisk immateriellt rättsskydd*, NIR 2/1999, (1999).
- Calabresi, Guido. Some Thoughts on Risk Distributions and the Law of Torts, *The Yale Law Journal*, Vol. 70, No. 4, (1961).
- Courtney, Wayne. The Nature of Contractual Indemnities, *Journal of Contract Law*, Vol. 27, No. 1, pp. 1-17, (2011).
- Dahlman, Carl J. The Problem of Externality, *Journal of Law and Economics,* Vol. 22, No. 1, The University of Chicago Press for The Booth School of Business of the University of Chicago and The University of Chicago Law School (1979).
- Hultmark, Christina. Köplagens tillämplighet på fel i patent, *Juridisk Tidskrift vid Stockholms Universitet. (1993-94), nr. 4.*
- Matson, Per. Förhandling om upphovsrätten till komponentutvecklade applikationer, *Juridisk Tidskrift vid Stockholms universitet. Årg. 10 (1998-99), nr. 3, p. 788-793*, (1999).
- McIlroy, Douglas. Mass produced software components. In *Software Engineering Report on a conference sponsored by the NATO Science Committee Garmisch, Germany, 7th to 11th October 1968*. Naur, Peter. & Randell, Brian. (eds.) NATO Scientific Affairs Division, 1969.
- Plogell, Michael. Upphovsmannen, *Nordisk immateriellt rättsskydd*, NIR 1/99, (1999).
- Plogell, Michael. Upphovsrätt till datorprogram ur ett EG-perspektiv. *Juridisk Tidskrift vid Stockholms Universitet. (1993-94), nr. 1.*
- Sim, Susan Elliott., Umarji, Medha., Ratanotayanon, Sukanya., and Lopes, Cristina V. How Well Do Search Engines Support Code Retrieval on the Web?. *ACM Transactions on Software Engineering and Methodology (TOSEM),* Vol. 21, No. 1, Article 4 (2011), 25 pages. DOI=10.1145/2063239.2063243 [http://doi.acm.org/10.1145/2063239.2063243]

- Sojer, Manuel. & Henkel, Joachim. License Risks from Ad-Hoc Reuse of Code from the Internet. *Communications of the ACM*, Vol. 54, No. 12, pp. 74-81, (2011). Available at [https://mediatum.ub.tum.de/doc/1083689/1083689.pdf] Accessed 2015-07-17.
- Stigler, Rachel. Ooey GUI: The Messy Protection of Graphical User Interfaces, Northwestern Journal of Technology and Intellectual Property. Vol. 12, No. 3, (2014). Available at: [http://scholarlycommons.law.northwestern.edu/njtip/vol12/iss3/3/] Accessed 2015-07-17.


## 9.3 Other written sources


- Naur, Peter. & Randell, Brian. *Software Engineering Report on a conference sponsored by the NATO Science Committee Garmisch, Germany, 7th to 11th October 1968*. NATO Scientific Affairs Division, 1969. Available at: [http://www.scrummanager.net/files/nato1968e.pdf], Accessed 2015-07-17.
- WIPO. (1978). *Model Provisions on the Protection of Computer Software, WIPO publication ; no. 814*.


## 9.4 Preparatory works


- SOU 2010:24. *Avtalad upphovsrätt*.
- SOU 1985:51. *Upphovsrätt och datorteknik*.
- SOU 1956:25. *Upphovsmannarätt till litterära och konstnärliga verk*.

- Prop. 1992/93:48. *Om ändringar i de immaterialrättsliga lagarna med anledning av EES-avtalet m.m.*
- Prop. 1988/89:85. *Om upphovsrätt och datorer*.
- Prop. 1988/89:76. *Om ny köplag*


## 9.5 Case law

### 9.5.1 EU jurisdiction


- Case C-128/11 *UsedSoft GmbH v. Oracle International Corp.* [2012].
- Case C-604/10 *Football Dataco Ltd and Others v Yahoo! UK Ltd and Others* [2012].
- Case C-406/10 *SAS Institute Inc. v World Programming Ltd.* [2012].
- Case C-145/10 *Eva-Maria Painer v Standard VerlagsGmbH and Others* [2011].

- Case C-393/09 *Bezpečnostní softwarová asociace* [2010].
- Case C-5/08 *Infopaq International A/S v Danske Dagblades Forening* [2009].


### 9.5.2 Swedish jurisdiction


- NJA 2000 s. 580
- NJA 1996 s. 79
- NJA 1995 s. 256
- NJA 1994 s. 74
- NJA 1990 s. 499
- T 2714-07


### 9.5.3 U.S. jurisdiction


- *Righthaven LLC v. Hoehn*, 716 F. 3d 1166 - Court of Appeals, (9th Cir. 2013).
- *Lexmark International Inc. v. Static Control Components, Inc.*, 387 F.3d 522, 535 (6th Cir. 2004).
- *Gates Rubber Co. v. Bando Chem. Indus., Ltd.*, 9 F.3d 823 (10th Cir. 1993).
- *Computer Associates International, Inc. v. Altai, Inc.*, 982 F.2d 693 (2d Cir. 1992).
- *Cain v. Universal Pictures, Co.*, 47 F. Supp. 1013 (S.D. Cal. 1942).


## 9.6 Websites


- Anderson, Nate. Copyright troll Righthaven finally, completely dead, arstechnica. 2013. Available at: [http://arstechnica.com/tech-policy/2013/05/copyright-troll-righthaven-finally-completely-dead/] Accessed 2015-07-17.
- ContractStandards.com. Indemnification (License Agreement), ContractStandards.com. 2014. Available at: [http://www.contractstandards.com] Accessed 2015-06-16. Search for "indemnification".
- Cornell University Law School. Contract, Legal Information Institute. 2015. Available at: [https://www.law.cornell.edu/wex/contract] Accessed 2015-07-17.
- Free Software Foundation. GNU General Public License version 3. GNU Operating System. 2007. Available at: [https://gnu.org/licenses/gpl.html] Accessed 2015-07-17.
- Free Software Foundation. GNU General Public License version 2. GNU Operating System. 1991. Available at: [www.gnu.org/licenses/old-licenses/gpl-2.0.en.html] Accessed 2015-07-17

- Green, Steve. Legal attack dog sicked on websites accused of violating R-J copyrights, Las Vegas Sun. 2010. Available at: [http://www.lasvegassun.com/news/2010/aug/04/unlikely-targets-emerging-war-media-content/] Accessed 2015-07-17.
- Green, Steve. Receiver says Righthaven 'uncooperative' in surrendering copyrights, VegasInc. 2012. Available at: [http://vegasinc.com/business/2012/jan/04/receiver-says-righthaven-uncooperative-surrenderin/] Accessed 2015-07-17.
- Groklaw. Want to See One of the Letters to the Fortune 1500?, Groklaw. 2003. Available at: [http://www.groklaw.net/article.php?story=20031127100124265] Accessed 2015-07-17.
- Hutchinson, Lee. It's back: District court judge revives SCO v IBM, arstechnica. 2013. Available at: [http://arstechnica.com/tech-policy/2013/06/its-back-district-court-judge-revives-sco-v-ibm/] Accessed 2015-07-17.
- Kang, Peter Y. XimpleWare, Versata Settle Insurance Software IP Dispute, LAW360. 2015. Available at: [http://www.law360.com/articles/620898/ximpleware-versata-settle-insurance-software-ip-dispute] Accessed 2015-07-17.
- Lombard Hill Group. Software Reuse 101: What Is Software Reuse?, Lombard Hill Group. Available at: [http://lombardhill.com/articles/software-reuse-101-what-is-software-reuse/] Accessed 2015-07-17.
- Nullbyte. Open curtains in swish payments service, Nullbyte. 2014. Available at: [http://blog.nullbyte.eu/open-curtains-in-swish-payments-service/] Accessed 2015-07-17.
- PatentFreedom. Litigations over time, PatentFreedom. 2014. Available at: [https://www.patentfreedom.com/about-npes/litigations/] Accessed 2015-07-17.
- Perrin, Chad. Why clean code is more important than efficient code, TechRepublic. 2011. Available at: [http://www.techrepublic.com/blog/software-engineer/why-clean-code-is-more-important-than-efficient-code/] Accessed 2015-07-17.
- Quinn, Gene. In Search Of a Definition for the term "Patent Troll", IPWatchdog. 2010. Available at:

[http://www.ipwatchdog.com/2010/07/18/definition-patent-troll/id=11700/] Accessed 2015-07-17.

- Radcliffe, Mark. GPLv2 goes to court: More decisions from the Versata tarpit, opensource.com. 2014. Available at: [http://opensource.com/law/14/12/gplv2-court-decisions-versata] Accessed 2015-07-17.
- Rouse, Margaret. decompile. WhatIs.com. 2005. Available at: [http://whatis.techtarget.com/definition/decompile] Accessed 2015-07-17.
- Ryberg, Jonas. Säkerhetsgurun: Swish har stulit min kod, IDG.se. 2014. Available at: [http://www.idg.se/2.1085/1.593454/sakerhetsgurun-swish-har-stulit-min-kod] Accessed 2015-07-17.
- Sanders, Rick. Copyright Protection of APIs after Oracle v. Google: Poppin a Whelan, IP breakdown. 2012. Available at: [http://ipbreakdown.com/blog/copyright-protection-of-apis-after-oracle-v-google-poppin-a-whelan/] Accessed 2015-07-17.
- SCOGroup. SCO Registers UNIX® Copyrights and Offers UNIX License, SCO Group. 2003. Available at: [http://web.archive.org/web/20100102232443/http://ir.sco.com/releasedetail.cfm?ReleaseID=114170] Accessed 2015-07-17.
- Williamsson, Aaron. Lawsuit threatens to break new ground on the GPL and software licensing issues, opensource.com. 2014. Available at: [http://opensource.com/law/14/7/lawsuit-threatens-break-new-ground-gpl-and-software-licensing-issues] Accessed 2015-07-17.

## 9.7 Interview

- Ahlberg, Jimmy. Interview 2015-06-10.

## 9.8 Contracts

- Apple Inc. Software License Agreement for OS X Mavericks, *apple.com*. 2015. Available at: [http://images.apple.com/legal/sla/docs/OSX109.pdf] Accessed 2015-07-18
- Cisco Systems Inc. End-user license agreement, *cisco.com*. 2013. Available at: [http://www.cisco.com/c/en/us/td/docs/general/warranty/English/EU1KEN_.pdf] Accessed 2015-07-18.

- Hovedorganisationen Dansk Industri, Danmark,. Metalliteollisuuden Keskusliitto Metallindustrins Centralförbund r.y., Finland,. Teknologibedriftenes Landsforening, Norge, samt Sveriges Verkstadsindustrier, Sverige. *NL 92 E General Conditions*. 1992.
- SAP. Software License and Support Agreement General Terms and Conditions ("GTC"), *sap.com*. 2014. Available at: [http://global.sap.com/corporate-en/our-company/agreements/western-europe/index.epx] Accessed 2015-07-18. Choose "SAP Software Agreements" and then "Sweden".
- Skype and Microsoft. Skype Terms of Use. *Skype.com*, 2015. Available at: [http://www.skype.com/en/legal/tou/] Accessed 2015-07-18.

Jag, Johan Corell, registrerades på kursen första gången VT15. Jag har ej omregistrerats och ej heller deltagit i något tidigare examinationstillfälle.