Legal Issues Regarding Software Use and Reuse within the European Union Legislation

Giancarlo Succi[1,3], Gianpiero Succi[2] and Marco Ronchetti[1]

[1] DISA, Università di Trento, Italy
[2] Law School, University of Genova, Italy
[3] Department of Computer Science, Virginia Tech, USA.

This paper addresses the problem of legal issues in the use and reuse of a software artifact with reference to the European Union regulations. Up to now software have been protected by means of the author law, however they are very different from other artifacts subject to the author law. This problem is getting more urgent as reuse is becoming a widely used software development methodology. Guidelines for implementing an "almost safe" contract for using and reusing software artifact are presented here. Questions regarding common misconceptions of the rights of the purchasers are also described.

Keywords: software legal issues, software use and reuse, European union legislation, author law, patent, trademark.

1. Introduction

Software reuse is becoming more popular, however not only is there a considerable lack in the current legislation on it, but the whole juridical discipline on software, is far from being well assessed. This paper addresses the problem of legal issues connected the use and reuse of software artifacts. The reference legislation is that of the European Union as it is implemented and applied in Italy; nevertheless the problems analysed there are rather general and can be applied not only to other European Union countries but also to a larger community of the industrialized world. US, Canadian and Japanese legislation are also inspired by the same general principles, on which the European legislation is founded, because most of the countries have joined the same world-wide conventions, starting from the Bern Convention of 1886. Furthermore a EU perspective can be extremely interesting because it is already a synthesis of the legislation of 15 member states and is the reference to the future laws of the other European and non European countries aiming either to join EU, such as Slovenia, Croatia, Poland, . . . , or to establish with it preferential trade agreements, such as Israel, Turkey, and many other countries from Middle East and Mediterranean Africa regions.

For the time being, up to the writers' knowledge, there is not a single country in the world that has a law regulating all the activities related to software development and usage, taking into account the specificity of the software. The current approach is to reason by analogy, applying to software the discipline of Author Law.

Section 2 examines this current approach describing the reasons that have led to it. Section 3 describes the rights of the authors of software artifacts, also with reference to current industrial practices. Section 4 goes in to the core of the problems connected with the usage of a software artifact. Implications and limitations of the current approach are evidenced and discussed in section 5, to make the reader aware of the uniqueness of the software production within the framework of the Author Law and to evidence potential black holes in the definition of any agreement on software usage. Section 6 draws some conclusions and identifies guide-

lines for further research.

2. Background: The Rights of the Authors

In our discussion the term "object" refers to the entity that is referenced by a regulation or law, regardless of its own nature; thus we refer to the object of the author law, the object of the patent and the object of the trademark. Mostly there are three instruments for protecting such rights, depending on the kind of the object to be protected and the kind of protection sought. They are:

- patent
- trademark
- author law

Furthermore, the authors right following from an artifact are recognized throughout the world, hence results the need to have treaties established between countries to integrate the applications of these instruments across the boundaries of the countries.

First this section introduces the reader to current EU regulations on patent, trademark and author law, then it sketches the history of the author law and of the related international agreements.

2.1. Patent

The first instrument used to protect the rights of a human craft's author is that of *patents*. Patents are currently protected in Italy by RD1127/39 6, stating that patents can be issued for new inventions.

A patent can be issued for a new invention, that is, it requires an inventive activity and an *industrial application*. Consequently of that, patents cannot be issued for:

- *discoveries, scientific theories and mathematical methods*, since they can be regarded as discovery of something already existing rather than new inventions,
- *plans, principles and means for intellectual activities, for games or business related activities*, that can be mostly kept hidden
- *presentations of information.*

Patents regarding something having a well defined physical structure created for a specific industrial use.

Furthermore, a patent is a government *concession* issued for a limited amount of time. In the European Union the duration of a patent is 20 years and *cannot* be extended.

Patent is not a *right*: such approach comes from the need of th a society to take care not only of the inventor but also of the whole community which can benefit from it. This is why to obtain a patent one must describe not only its *novelty* but also its *industriability and its production process*. Description of the novelty is useful to avoid people re-inventing the wheel again and again. The industriability prevents protection of meaningless creations. The production process has to be described in a *repeatable* way, so that, at the end of the concession, all the community can have access to it.

2.2. Trademark

A trademark is a sign stamped on an object so that some of its properties can be distinguished. The sign can be composed by an icon and/or a logo. Typical trademarks are:

- *production marks*, certifying the producer,
- *trademarks* properly defined,
- *service marks*, identifying services provided by enterprises to make usage of a given good, examples of service marks or transportation marks (i.e., DHL, UPS, . . .)

For being registered and so protected, a trademark must be true and new, that is, it must refer unambiguously to a property of an object (viz., not be misleading such as calling a company merchandising canned fish "fresh fish Inc.") and it must not be similar to a previously existing one. Once registered, the owner of the mark has the right to the exclusive use of trademark.

These marks are inadequate to protect software since the law protects the trademark itself and not the object with the trademark.

A different category of marks is that of "collective marks". A collective mark is a certificate granted to a product or a process by an independent organization interested in the "quality" of the item. Examples of collective marks

are the Italian DOC and Parmigiano Reggiano. DOC refers to the production process of wine; the acronym stands for Denominazione di Origine Controllata, Controlled Source Name, that refering the fact that the producer of that kind of wine is well known and controlled, and uses a standard production process, without modern short-cuts, such as adding sugar (that is also forbidden in Italy). Parmigiano Reggiano identifies a cheese produced according to some very ancient procedures. The difference with the American definition is striking: in Europe, only a cheese that has been produced in a specific zone of Italy, following rigorous specifications, can be defined as parmesan cheese.

The family of the ISO 9000 certificates is a well known example of collective marks; again, rather than the product itself, they are used to certify the production process as a whole.

2.3. Author Law

Objects of authors' rights are creative products of the human intellect, within the framework of literature, music, opera, drama, visual arts, cinema, photographs, architecture and (recently) computer programs. To protect the authors' rights throughout the world, the so called "author law" has been established. Authors' rights have been protected since the ancient Greeks. On all his works Focidides used to write the statement "$\kappa\alpha\iota\ \tau o\delta\varepsilon\ \Phi o\kappa\iota\delta\iota\delta\varepsilon o\nu$", "also this work belongs to Focidides". In the entire history of literature breaking authors' rights has been regarded as a humiliating unfairness, i.d. the case of Lawrence the Magnifier, Lord of Tuscany, who was accused of not writing his poems himself. Regardless of any legal formality the authors' rights are linked with the creation of the work, however such work has been expressed and hold, even when it has not been publicly revealed. To protect the authors' rights throughout the world the so called "author law" has been established.

The central idea is therefore that of creation, which is very different from that of invention, typical of the patents (see above). Therefore, the essence of any work protected by the author law is by its own nature very different from that of patents. First of all, patents refer to something having a well defined physical structure, while author rights are associated to more "abstract" entities. Thus, while there is only one application of the object of a patent, and that is to put it into work, there are multiple means of using an object with authors' rights: it can be reproduced, transcribed, executed, diffused, commercialized, translated and elaborated.

A patent is a concession from government. On the contrary, the author right is an "absolute" right comparable to that of property; this is why the author right lasts forever, is inherited by the relatives till 70 years after the death of the author, and, after that, it does not cease but becomes property of the author's country.

The object of a patent is something physical, while the object of the author law has a more abstract nature. Therefore, patents refre to "inventions", author law deals with "creations".

It is useful to understand the origin of the words in question. Invention comes from the latin "invenire", to find, while creation comes from the latin "creare", to create, and the latin jurisdiction defines "creare est ex nihilo facere", to create is to make from nothing. Therefore patents refer to something that is built on the basis of already existing means and such that if the inventor had not invented it someone else could have discovered it later, while author rights are appropriate to something that has been generated from nothing, so that if the creator had not created it, hardly anyone else could have done it. The following example can help clarify this concept: how many people are reinventing the wheel, but no other musicians but Beethoven could have done the ninth symphony. For the same reason patents are related to object with an industrial application, while no such constraint is imposed on the objects protected by author rights. Furthermore, as previously described, there is usually only one way of sing the object of a patent,but there are multiple means for making use of the author law's object.

Last but not least, there are different economic advantages for patent related objects on one side and those covered by author rights on th eother. While the former lasts 20 years after the concession and cannot be prolonged, the latter lasts 70 years after the death of the author; the reason for that is the fact that in the case of patents the community considers two facts:

(a) someone else could have invented it,

(b) the advantages for the whole community.

History provides evidences of (a); for example the telephone — no one knows for sure whether it has been invented by Bell or by Meucci. As said, (b) is so important that the production process must be described in a repeatable way.

It has been decided that computer programs should be protected by the author law.

2.4. A Brief History

As previously explained, author rights have their root in the history. The first international convention about author rights took place in Bern in 1886. After it, several other were organized (Paris 1896, Berlin 1908, Bern 1914, Rome 1928, Brussels 1948, Stockholm 1967, Paris 1971). At the end of 1994/95 countries from all over the world joined the Bern Convention: almost all European countries among them.

A *Universal Convention*
/ was signed in Geneva in 1952 and revised in Paris in 1971. This convention integrates the countries that have signed the Bern Convention with those having legislations incompatible with it. The list of countries that join the Universal Convention includes United States, Argentina, Chile, Mexico, African and Asian countries and some of the new countries of former Soviet Union. More specifically, within the framework of Author Law, the Universal Convention regulates the relationships:

- between the countries that have joined the Bern Convention and the member countries of Universal Convention only,
- among the countries that have joined the Universal Convention only.

EU authors can protect their work only in the countries member of the Universal Convention, by means of the so called "reserve mention" expressed in these work a symbol © followed by their name and the year of the first publication.

The need of a specific protection for software authors has been identified since the early '70–ies. In 1980 the US promulgated the first law in this field, the so called "Computer Software Amendment Act" (Public Law 96–517, 12–12–1980). Australia followed the US example in 1984, and in the following years some EU countries also settled their internal regulations (France, Germany and UK in 1985, Spain in 1987). On 14th may 1991 the EEC promulgated its directive number 250 about the protection of computer programs, which is the reference point of this paper; the directive required that by the end of 1992 all the member states should have made their internal legislation conform to it.

The European Convention on Patents held in Munich in 1975 stated that computer programs cannot be protected with patents, since they are not something physical that can be assimilated to a new invention but are rather the result of a creative activity of the human mind; thus the preference was to protect software authors with the discipline of the author law. This was not an easy decision since software has a clear industrial application, unlike other kind of artifacts protected by the author law. Apart from the projects made by architects, that also have an industrial implication, even though much more limited than that of computer programs, a software has a clear industrial application. However, similarities in the multiple potential usages and other aspects of items protected by the author law have led to this decision.

The industriability poses a further problem. Unlike the programs, most of the objects protected by the author law, though beautiful, are not physically necessary for the community. In this sense, protection supplied to softwares seems to be too high. Moreover the fact that after a few years any program becomes obsolete, makes this problem void; yet this situation may change in the future. By the way, it is worth mentioning that the author law protects the expression of the idea, not the idea itself; therefore the idea of an algorithm cannot be protected, only the specific C code implementing the algorithm.

3. Taxonomy of the Rights of the Software Author

The rights of an author of a software artifact can be divided as follows:

1. economic rights
2. moral rights

3.1. Economic Rights

The author of a software artifact has the unique right of using the artifact in any shape or by any means, and of reproducing the artifact in any way he/she can, of divulgating the usage of the artifact by any possible media (e.g., floppy disk, CD/ROM, net, ...) and of commercializing it, of translating it into different "expressing paradigm", i.e., if the artifact is a piece of BASIC code, he/she can translate it into Fortran, if it is a LaTeXdocument, he/she can translate it into Microsoft Word, and collect it into a library together with other artifacts belonging to her/him.

All these unique rights are independent one of another and the unique use of one of them does not compromise in any way the use of another.

These rights are called *unique*, since they are up to the author as the original owner of the artifact and of its pertinence.

In one word, the author can do with his/her software artifact whatever he/she wants, from making it public, or marketing it to withdrawing it from the market. Therefore she/he can also alienate partly or wholly from these rights.

Notwithstanding the above mentioned rights of the author, unless a different agreement has been set up, the employer has all the economic rights (in the sense previously mentioned) of software artifact developed by his/her employees.

3.2. Moral Rights

Regardless of the economic rights that can be alienated, the author retains for ever all the moral rights of the software artifact. Moral rights allow the author:

(a) to have the acknowledgment of the paternity of the artifact

(b) to forbid any modification of the artifact that may damage the artifact itself, causing a prejudice toward his/her honour and reputation.

4. Reusing a Software Artifact

A software artifact is defined in terms of the EEC directive 91/250 as any kind of computer program, including those embedded into hardware, taking into account not only the code listing, but all the preparatory and complementary documents needed to develop the code, provided that there is the clear connection between them and the developed code. This definition is reflected in the Italian law DLg518/92 14.

Moreover, when we say "we purchase a software", in reality we mean two separate things:

(a) the purchase of a hardware support in which the software is stored, say floppies or CDs

(b) the purchase of the license to use the software

Thus, while we can do whatever we want with the hardware, since such hardware belongs to us, with the software we have just a limited license, and we must stick to it. However, the conditions of such license are those established by the law, as we shall explain in the rest of this section. Stricter conditions must be explicitly agreed before the purchase, as the conditions found in the package are, in our opinion, not valid.

It is important to notice two aspects of the law. The first is, that it cannot be identified in terms of qualitative or quantitative criteria what a program actually is. The law protects any kind of program, regardless of its size or quality. The second one is that only the *expression* of a program is protected, and not the *ideas*, or the principles upon which the program is built.

A very recent example of the latter fact is the decision of the Supreme Court of February 7, 1996, to consider Borland not guilty in the Lotus vs. Borland case. Lotus cited Borland in 1991 because Borland introduced in its spreadsheets a feature very similar to the one that Lotus had already introduced in its "Lotus 1–2–3", the so called "command hierarchy". At the first trial Lotus won, however the Appeal Court rejected such decision in March 1995, since, as they said, the command hierarchy could not be copyrighted. The Supreme Court took the same approach as the Appeal Court, the decision was so relevant that the president of Borland, Gary Wetsel, called it as a victory of the software producers and users of the whole world.

4.1. Taxonomy of Software Artifact Usage

A software artifact can be reused in different ways depending on the kind of the artifact and the need of the user. For the sake of simplicity, we differentiate three kinds of artifact:

1. machine code

2. source code

3. documentation

By **machine code** we mean any code, that can be run directly on the machine hardware, without the need of compilation or of being interpreted by some software tool, such as a virtual machine. By **source code** we mean any portion of a program that can be run *latu sensu* on a computer in a human readable language, such as a FORTRAN function, a C++ class, a PROLOG clause, a hypercard stack and so on. By **documentation** we mean any further software artifact that is produced to help in the program development, maintenance, usage or any other computer related activity other than coding; user manuals, OMT diagrams, requirement documents are all examples of documentation. The boarders around the first class are rather sharp. The ones between the other two are more fatuous: for instance, a Z specification can be considered both as a source code and as documentation.

4.2. Using Machine Code

The legal issues concerning the usage of machine code are quite well defined both in the USA law [12] and in the European Directive [13]. They regulate the rights of using a software product intended as an executable code packaged with its environment, of reproducing it and also of analysing it, of reverse engineering and reengineering under certain conditions, mainly that the user can do almost whatever she/he wants to make the program interoperable with his/her environment.

It is not always clear when the activities of reverse engineering and modifying a software product are legal: for instance, the European legislation permits to reverse engineering and modify a program to make it interoperable with the working environment of the purchaser; this seems to mean that the purchaser can legally build his/her own APIs for the program; but then, if the product is bought in a version for work in the environment XXX with the application YYY, can the purchaser reverse engineer it to make it working in the environment WWW with the application ZZZ? And can he/she do it even if XXX is also sold to work in WWW with the application ZZZ, maybe at a higher price? The EU directive seems to answer 'yes' to both this questions.

Then, there is the problem of copying and duplicating software. Here the position of the EU directive is rather clear, despite some software factories seem to forget it. Each purchaser has the right of producing a back-up copy of software he/she bought regardless of the conditions signed or printed on the licensed one. Furthermore, the EU directive states that a software can be freely used to accomplish the goals for which it has been purchased. Thus it is not clear whether it is forbidden for students or universities to duplicate a software bought for educational purposes. The Italian "Avvocatura Generale dello Stato" (the Office of the State Lawyers) gave a recommendation to software producers not to be too strict in the controls operated toward university and students.

Furthermore, the EU directive makes a difference between the two cases, when the copy of software is made for money and when not. Both cases are forbidden, but the penalty is much different. In Italy, for the former crime the punishment is imprisonment from 3 months to 3 years and a fine from 500.000 Italian Liras to 6.000.000 Italian Liras (from 350 to 3900 US$), while for the latter there is just a fine from 20.000 Italian Liras to 800.000 Italian Liras (from 13 to 500 US$). Now the point is, saving money is "for money"? That is, when a programmer duplicates a software for his/her personal interests, say for studying it, is he/she liable according to the harder regulation? There is no proper interpretation of this point, however the writer tend to believe that the answer is "no".

It is still unclear, whether downloaders of the socalled shareware software are required to pay the "shareware fee"; analogy can be established with public musical entertainments: when one stops listening to street musicians he can feel obliged to give them some money, but he/she is not forced by the law to do so. Another useful analogy can be with the habit of tipping in

the US: in a restaurant customers always leave a tip between 10% and 20% of the total price. On one side this is just their kindness, on the other, if they do not do so, they may face rude reactions!

4.3. Using Source Code and Documentation

Very different is the situation concerning the usage of the source code. There is a general lack in the legislation regarding the reuse of software source code and documentation. The european directive does not address such issue at all, leaving the problem completely open.

Common cases bring forward many serious questions; for instance, if we buy a routine, then

1. how 'far' can we use it?

2. how 'far' can we adapt it to our needs?

3. can we sell an executable made with such routine?

4. can we sell a source code containing the routine?

5. how long can we use it, adapt it, sell as executable and sell as source?

6. after the expiration of the license, what can we do?

7. who is liable if the routine contains an error?

8. who is liable if the routine is used in an improper way?

The only reasoning that appears sound is the analogy with other fields stipulated by the author law. So, for instance, for questions 2 and 3 we may refer to musical arrangements.

5. Consequences of the Current Legislation

Our impression is that the current European software legislation is aimed to protect small corporations and free lance programmers: the ones in greatest most danger nowadays. The fact that reverse engineering is permitted to perform system integration and the big differences in punishments between the two kinds of violations of copyrights lead towards limiting the risks of creating monopolies. Suppose that such activities were forbidden; then once a firm has widespread its operating system and/or its integrated environment and kept secret the details of their implementations, no one else could compete with it, since none else could build a software that could use the already existing modules.

Besides, in the European Union there is no regulation about the liability of a software engineer: this is again a protection of those small companies that do not have access to expensive top-level lawyers. In one opinion this is not a miss, since ISO9000 and CMM rules and certifications are better incentives to produce high quality and defect-free software than the threat of a severe punishment.

6. Conclusion and Further Research

The new EU directive addresses the protection of a software and its inter-conenction with using and re-using software components. Currently, it is protected by the promisions of the author law. These promisions seem adequate for protecting executable programs. However some problems remain open, such as to what extent moral rights of the authors can limit the usage of programs, or how long the rights of the authors should last. Furthermore, the boundary between the idea of a program (which cannot be protected) and its expression, is not strictly defined. Anyway, it seems that the general approach of the directive is trying to limit the creation of monopolies.

The situation is much harder dealing with source code: there are still several problems to be addressed properly, and in the current framework some of them are not considered at all. Consider, for instance, the question of the liability of the prime developer and that of the reuser.

Therefore, the resulting 'scenario' is that of an evolving situation and we may expect new regulations coming out soon.

References

[1] Berne Convention for the Protection of Literary and Artistic Work, *Final Agreement*, September 9,1886, Berne.

[2] Paris Convention for the Protection of Literary and Artistic Work, *Completion of the Berne Convention for the Protection of Literary and Artistic Work*, May 4, 1896, Paris.

[3] Berlin Convention for the Protection of Literary and Artistic Work, *Revision of the Berne Convention for the Protection of Literary and Artistic Work*, November 13, 1908, Berlin.

[4] Berne Convention for the Protection of Literary and Artistic Work, *Completion of the Berne Convention for the Protection of Literary and Artistic Work*, March 20, 1914, Berne.

[5] Rome Convention for the Protection of Literary and Artistic Work, *Revision of the Berne Convention for the Protection of Literary and Artistic Work*, June 2, 1928, Rome.

[6] Regio Decreto 1127/39, it Brevetti per le invenzioni industriali, 1939, (King Decree 1127 of 1939 on *Patents for industrial inventions*, identified in the paper as RD1127/39).

[7] Legge 633/41, *Protezione sul Diritto d'Autore*, 1941. (Law 633 of 1941 on the *Protection of the Author Rights*, identified in the paper as L633/41).

[8] Regio Decreto 929/42, Marchi Registrati, 1942. (King Decree 929 of 1942 on *Trademarks*, identified in the paper as RD929/42).

[9] Brussels Convention for the Protection of Literary and Artistic Work, *Revision of the Berne Convention for the Protection of Literary and Artistic Work*, June 26, 1948, Brussels.

[10] Stockholm Convention for the Protection of Literary and Artistic Work, *Revision of the Berne Convention for the Protection of Literary and Artistic Work*, July 14, 1967, Stockholm.

[11] Paris Convention for the Protection of Literary and Artistic Work, *Revision of the Berne Convention for the Protection of Literary and Artistic Work*, June 24, 1971, Paris.

[12] Public Law 96–517, 12/12/1980, *Computer Software Amendment Act*, 1980.

[13] EEC Directive 91/250, *Tutela giuridica dei programmi per elaboratore*, 1991.

[14] Decreto Legislativo 518/92, 1992.

[15] L. CHIMIENTI, *La tutela giuridica dei programmi per elaboratore nella legge sul diritto d'autore*, Giuffrè editore, Milano, Italia, 1994.

[16] E. GIANNANTONIO, *Manuale di Diritto dell'Informatica*, CEDAM, Padova, Italia, 1993.

[17] A. STRACUZZI, *La tutela del software*, Notizie Assoft 2:1–4, May 1995.

Received: February, 1996
Accepted: October, 1996

Contact address:
Giancarlo Succi
DISA — Università di Trento
via Inama 5, I–38100 Trento
Italy
phone: +39 (0) 461 882307
fax: +39 (0) 461 882124
e-mail: Giancarlo.Succi@cs.unitn.it
e-mail: Giancarlo.Succi@acm.org

GIANCARLO SUCCI (PhD, 1993) is visiting assistant professor at Virginia Tech, Computer Science on leave from the Department of Computer and Management Sciences of the Università di Trento.

GIANPIERO SUCCI is graduate student at the Law School, Università di Genova.

MARCO RONCHETTI is assistant professor at the Università di Trento.