Software: Protection, Licensing and Rewarding Researchers in Computer Science – an Overview of Challenges in the European Innovation Ecosystem

Urška Fric, Špela Stres[1], Robert Blatnik[2]
Faculty of Information Studies in Novo mesto Slovenia
[1,2]Jožef Stefan Institute
E-mail: urska.fric@fis.unm.si, spela.stres@ijs.si, robert.blatnik@ijs.si

Keywords: software, protection, licensing, researchers, computer science, rewarding, challenges

Received: November 2, 2022

With the transition of innovation to the digital sphere, software has become an essential part of contemporary inventions and creations. It is also an essential part of intellectual property. The state of software protection in the European innovation ecosystem still needs to be considered fully regulated. However, researchers in computer science also face several challenges when exploiting intellectual property rights in the software. The field, therefore, offers opportunities for researching, which should steer research to (1) present the critical points, and (2) update some of the legal frameworks to address more clearly the field that deals with the issue of remuneration for researchers in computer science. The overview paper discusses software, focusing mainly on the challenges that researchers in computer science face in protecting and licensing software in the European innovation ecosystem. The following paper presents the state of the researchers' remuneration under protection and licensing.

Povzetek: Pregledni članek obravnava področje programske opreme. Prvotno se posveča izzivom, s katerimi se raziskovalci na področju računalništva soočajo pri zaščiti in licenciranju programske opreme, nadalje pa predstavlja ključne izzive nagrajevanja raziskovalcev iz naslova zaščite in licenciranja.

1 Introduction

Patent applications for computer-based inventions are amongst those with the highest growth rate across all planet categories arriving at the European Patent Office (EPO). A thorough examination process awaits all new applications in this field. The crucial aim is to distinguish genuine technological innovations – which contribute to the overall level of progress of technology – from straightforward and inventiveness of computer-implemented inventions. [1, 2]

Over the last decade, there has been equally intense debate over how software should be the subject of patent protection instead of a program's copyright protection. Different answers have been reached in the US, Europe, and the rest of the world. Many companies in the software industry feel that the perceived difficulty of defining the scope of software patents still needs to be determined. A plaintiff can sue under such patents, relying on their ambiguity and one of the significant legal fees involved in obtaining clarity on the scope to force a settlement.
However, others feel equally strongly that the software industry needs patents. [2, 3]

Software that does not demonstrate a technical contribution can only be protected by copyright, which does not protect ideas. The appearance of a command line or graphical interface can be protected as a registered design. In contrast, a patent for computer or mobile applications can be granted if a technical contribution is demonstrated. Under EPO rules, the software must relate to the hardware in case of such. [3]

At least part of the reason why there is still no appropriate legal base is that such inventions are particular and proving their technical contribution and industrial applicability can be challenging. [3]

To successfully market software, the Public Research Organization (PRO) system needs to regulate the motivation and reward of researchers in computer science from successful marketing. Currently, the European innovation ecosystem is nothing to motivate researchers in computer science.

The current situation calls for the study identification of the most critical points to update some of the legal bases, to address this field more clearly and to address the issue of rewarding researchers in computer science – which is therefore addressed in this paper with a focus on software.

The paper is overview oriented and presents software, (1) focusing mainly on the challenges researchers in computer science face in protecting and licensing software in the European innovation ecosystem, and (2) the state of researchers' remuneration under protection and licensing.

While the starting points of the conceptual framework are based on secondary data derived from the current and updated legal frameworks of the European innovation ecosystem, the presentation of the current state-of-the-art of rewarding researchers in computer science is based on primary data derived from the knowledge of operating in PROs.

2 Software in theory and practice

2.1 Software and related terms

Before addressing the matter of this paper, let us clarify the key concepts related to the software and, therefore, necessary for the understanding of this paper:

1. *Software*: a group of computer programs that, together with the hardware in a computer, form a whole.
2. *Computer Program*: an algorithm written in one of the programming languages that can be executed on a computer.
3. *Algorithm*: a sequence of defined rules and commands that allow a problem to be solved in a finite number of steps.
4. *Source Code*: code in a form suitable for translation into an appropriate programming language.
5. *Machine Code*: text or operating code in the form of executable files.
6. *Computer-Implemented Invention (CII)*: an invention implemented using the computer, computer network or other programmable devices, with one or more features that are either partially or fully implemented using one or more computer programs [4].
7. *Technical Contribution*: a contribution to the state-of-the-art in a field of technology which is new and not apparent to a person skilled in the art; it is not assessed in the light of the difference between state-of-the-art and the patent application, which must contain technical features, whether non-technical features accompany them [4].
8. *Command Line Interface (CLI):* connects the user to a computer program or operating system, where users interact with the system or application through a CLI by entering text.
9. *Graphical User Interface (GUI):* computer environment that allows a user to interact with the computer through visual elements [5].

2.2 Legal framework in european innovation ecosystem

Computer programs are defined both in *Directive 2009/24/EC* of the European Parliament and of the Council of 23 April 2009 on the legal protection of computer programs [6], and in the European Patent Convention. The latter stipulates in *Article 52(2) (c)* that programs for computers are not regarded as inventions [7]. In this Article, the European Patent Convention excludes computer programs from patentability. It is important to stress the distinction between "*software patents*", which are excluded according to the Article mentioned above, and the so-called "*computer-implemented inventions*", which are accepted at the EPO [8].

Software that does not demonstrate a technical contribution can only be protected by copyright, which does not protect ideas. The appearance of a *command line interface* or a *graphical user interface* can be protected as a registered design. In contrast, a computer or mobile application patent can be granted if a *technical contribution* is demonstrated. Under EPO rules, the software must relate to the hardware in case of such. [9]

Although the European Patent Convention excludes *computer programs* from patentability to the extent that a patent application relates to a computer program *as such*, this is interpreted to mean that any invention that makes a non-obvious *technical contribution* or *solves* a *technical problem* in a non-obvious way is patentable, even if the *technical problem* can be solved by running a computer program. [10]

The problem of strictly classifying software as a literary work appears when one considers computer programs have other elements that are usually not protected by copyright – software is not just a literary expression; lines of code have a function that does not depend on their grammatical construction. Problems with the protection of additional elements of computer programs have created a perceived need for software patentability.

Today, the world's three most prominent patent offices, the European, US and Japanese, allow the patenting of specific software. However, there are differences between the criteria used in accepting applications in these offices. All new and non-obvious software that produces valuable concrete and tangible results are eligible for patent protection in the US. In contrast, in Europe, the invention's technical contribution must be defined as described above, which is the same in Slovenia.

These discussions led to the now widely accepted principle that computer programs should be protected by copyright, while apparatus using software or software-related inventions should be protected by patent.

Protecting and obtaining intellectual property rights (IPR) in fast-growing areas such as artificial intelligence is challenging. Artificial intelligence enables entirely new approaches to creating intellectual property (IP). Questions arise regarding the eligibility of patent protections, authorship and rights ownership of a newly developed technical solution or creation autonomously created, enabled, or co-created by a program. The subject of intense debate and accelerated activity at the EPO is how these issues will be resolved in a way that does not stifle innovation potential [11].

3 Software and exploitation of intellectual property rights

3.1 Ways of licensing software

IP is an essential tool for protecting the value created by the software. As a rule, almost all software is protected, including the smallest libraries and subroutines. IPRs are divided into economic and moral rights. [12]

Economic rights give the holder the right to exploit the work, prevent others from using it without consent, and aim for financial gain. The license can grant the right to use; if it is exclusive, it allows the holder to exclude others from using the IP in question. If it is transferable, it will enable the holder to grant third parties the rights to use it. A *license* is a permission granted by the *licensee* to use an

identified asset under certain conditions. In doing so, the *licensor* may determine at its discretion the extent of the exclusive IPRs granted in respect of the support (and, conversely, the rights it reserves for itself).

Moral rights include the right to authorship, the right to publish the work anonymously or under a pseudonym, and the right to the integrity of the work. In most countries (including all EU countries), copyright protection lasts throughout the author's lifetime and for 70 years after his death.

As far as IP is concerned, the software can be protected by several IPRs on the borderline between pure creations of the mind and technical inventions. However, even more complexity arises from the intangible nature of software, the variety of uses and the different means of creating value from software.

Therefore, the means of creating value from software can vary considerably depending on the exploitation scheme chosen and the associated ecosystem to which the use of the software in question is directed.

Nevertheless, licensing is essential in creating value by managing the IP associated with software development. Business models are formalised in a contract, usually in the form of licensing agreements, which impose specific rules of use on third parties who intend to exploit the software. Figure 1 shows typical software licensing models.

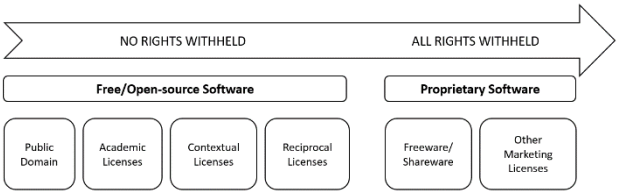

Figure 1: Classification of typical software licenses

Free and open-source software rights include use, inspection and modification, and distribution of modified and unmodified copies. It typically allows use for any purpose without restriction; in the case of reviewing and changing code, it requires that the modified code be made available again under the same conditions; and it also provides for the possibility of distributing modified and unmodified copies. When free and open-source software is modified, derivative works are created, and when various software components are assembled, composite parts of the underlying components are made. *Component A* and *Component B* are formed, and *Component A* is also modified. *Component C* is created as both a derivative work of *Component A* and a composite work of *Component B*. Different economic rights may arise from open-source and free software. Free software derives from licenses granted by the *Free Software Foundation*, while open-source software is defined by the *Open-Source Initiative*, which has a more business-oriented approach. We consider the following:

1. Academic licenses
   a. Extremely open, "*permissive licenses*".
   b. Licensees can perform, modify, and distribute derivative works without restrictions.
   c. Licenses for derivative works may lead to new licensing terms, including proprietary ones.
   d. Generally accepted in academia.
2. Contextual licenses
   a. Licensees may use, modify, and distribute derivative works, provided that the derivative or composite works are distributed under the same license.
   b. "*Copyleft license*": copyleft grants the right to freely distribute and modify IP, with the requirement to preserve the same rights in derivative works created from that property [13]. The main advantage of such licenses is to ensure joint investment, as no derivative or significant works can be licensed under another license. They allow the original licensor to be granted the same rights in the derivatives as those initially acquired by the original code licensees. [13]
3. Reciprocal licenses
   a. Very complex.
   b. Licenses of major works using an unmodified version of the original component under a contextual license are not limited by the original license.
   c. A derivative product containing a modified component must be released under the same license.

Many different contractual relationships contend with actual sets of rules derived from proprietary licenses, all of which typically require a financial contribution from the user. We consider the following:

1. *Freeware*, where the software is available free of charge, but any code modification is prohibited.
2. *Shareware*, where the user is free to use the software for a limited period or with limited functionality but to gain access to the full unrestricted version, an additional license must be obtained.

All proprietary licenses prohibit software modification, impose strict use conditions, and usually do not allow access to the source code. We consider the following typical models:

1. *End-user licensing*, where the license is allowed to be used by a specific user, but sharing with other users is not allowed, but the same user on different devices can use the license.
2. *Node licensing*, where the license can be used by multiple users but on the same device rather than at the same time; site licensing, where the software may be used by multiple users on various devices in a specific area or company but the number of users may be limited.
3. *Network licensing (floating licensing)*, where the same software may be used by multiple users simultaneously, but a central server authorises access to the application.

3.2 Management of intellectual property rights for software

Managing IP in software requires the strategic and complementary use of different types of IP. Exploitation and licensing strategies must be carefully considered, considering all associated costs and market opportunities. Two fundamental issues should be addressed in the assessment and planning process:

1. *Why was the software created:* was it intended to generate income through licensing to end-users or was it developed as part of a scientific project without an exploitation strategy in mind*?* Even if we focus only on the technical challenges of R&D, we should pay attention to the long-term benefits of protecting IP, not only from a revenue perspective but also because in research work, we may want to reuse the developed software in future applications.
2. *How was the software developed*: what are our components and what have we obtained from elsewhere, and if so, under which licenses*?* Designing based on foreign parts can bring legal challenges, as the individual licenses of different third-party software may not be compatible.

Derivative works based on licensed academic software components may be re-licensed under the same license type or upgraded to contextual or reciprocal (mutually compatible) licenses. If necessary, contextual licensing code can be re-licensed by reusing the same license, upgrading the license to a newer version that remains in the same contextual field, or switching to reciprocal licenses. We cannot embed free and open-source software in proprietary software. However, combining copyleft software without copyright and some contextual rights is possible.

However, suppose the software is protected exclusively by copyright. In that case, it is possible to circumvent all prior rights easily but effortlessly if we have access to the source code: we must implement the same idea in another source code. A new implementation of the code is the only legal way if we want to convert academic or reciprocal software code into proprietary code and sell and license it.

3.3 Software marketing and challenges of rewarding researchers in computer science: typical examples

The applied and commercially attractive results of R&D, such as computer programs, (graphical) user interfaces, databases, and other software, represent a potential value for customers or users in a commercial activity (sale of rights) and, in some instances, a more comprehensive use value for society, for which the rights holders decide to make the rights available free of charge under certain conditions.

The following are some typical examples of software development and commercial exploitation and the difficulties in providing an incentive environment and reward mechanisms for creators or authors.

1. *The authors develop the software, publish it online and make it free to use under certain conditions*. These cases may be conditional on a funding contract for the R&D project that resulted in the software, or it may be a decision by the wider research team (not necessarily the authors) that the software is for the broader good of society and a reference that, alongside the wider use of the software, brings specific other results that are important to the research team (e.g. raising the impact factor through published papers, completed projects, number of citations, raising the international reputation of the R&D team and the PRO, which in turn makes it more successful in obtaining new funding). It can also be a business model to acquire a critical mass of users or developers, allowing later market exploitation (open-source marketing models). There are no legally required ways of rewarding authors for such successful results (demonstrably used for the benefit of society but not valued through the direct income of the PRO). Rewarding is possible through other mechanisms, e.g., internal rules at the group level, which provide rewards or incentives to collaborators for successful work. In cases where project funding agreements do not already limit this, the decision to grant rights (e.g., free use) is left to the group or project leaders, and authors' involvement in such decision-making processes is only sometimes guaranteed. The absence of decision-making mechanisms that weigh the pros and cons of the chosen model for the allocation of software usage rights on expert grounds, such as assessments of technological and market potential, reviews of the feasibility and options of different open source models for software protection or licensing, estimates of the academic and broader societal benefit potential, leads to a less stimulating environment for the commercial exploitation of R&D results in the field of software creation.
2. *Authors develop software based on third-party software, restricting them from commercially exploiting or even publishing freely the newly designed software based on prior rights.* These cases arise, for example, when a newcomer to a long-running project takes over work on software which, during development and contribution by the newcomer, is found to have commercial value (perhaps because of the newcomer's innovation or contribution). Practical realisation of the market value through the sale of the rights to such software is not possible because the design of the software or the project itself needs to consider the limitations of the third-party rights of the authors of the selected software or the potential business opportunity.
3. *Authors develop software and, due to the limitations of the reward possibilities in the PRO, prefer to use it in their own company or a third company with private business links to the authors.* These cases arise for several reasons:
   a. There are no established mechanisms for rewarding authors of commercially exciting software, which, to be successful on the market, also requires personal involvement in the development of a specific, customer-tailored application, its sale, the establishment of marketing, user support and all the business model-related factors for the successful sale or realisation of the company on the market.
   b. A problem related to the previous point is the need for mechanisms for fast and efficient software registration as trade secrets (e.g.,

technical improvements) at the PRO. This would allow the registration of an intangible asset and the related determination of its value and impact on the PRO's business result or open the possibility of a transparent transfer of the rights to use the asset to a company (spin-off or external, unrelated to the PRO, through a license agreement) or through a capital injection in a newly created company (a spin-out, in the case of legal possibilities).

4. *The authors develop the knowledge and software used in a software development project commissioned by the customer (R&D or commercial contract with industry).* Under the contract, the buyer acquires all rights to the newly developed software. However, the rights to the previously developed software and know-how that made the deal for the PRO or the innovative potential of the newly designed software for the buyer possible in the first place are not defined in the contract. They are indirectly valued through the value of the contract, which is cost-accounted for in terms of person-hours worked on this commercial R&D project. There is no legally required remuneration to authors for exploiting the innovation resulting from the innovative potential of the previously developed software. The problem manifests itself on several levels:
   a. *Researchers at the PROs primarily seek funding through projects that cover person-hours and material costs.* The value of the IP that enables the sale of TR hours is generally ignored or forgotten, as the need and responsibility of R&D teams to secure sufficient funding or projects to cover the hours is at the forefront. Obtaining R&D contracts directly from the industry is highly scarce or requires significantly more commitment from researchers than getting publicly available resources through calls for tenders. On the other hand, TR projects represent a particular reference. R&D groups are willing to give up rights to previously generated IPs.
   b. *Often, there is even a free assignment of all rights of newly created inventions in favour of companies and the clients of the R&D services* (e.g. PRO is not even listed as the applicant of a patent application based on software code authored by the researchers of the PRO).
   c. *Under the pressure of securing a deal with the industry, with all the conditions of negotiating the best price for the solution that is still acceptable to the company, the authors are forced to compromise or set calculations that only cover the person-hours* for the development of the ordered solution, which in the analysis does not cover the rewards for the authors of the previously developed software, which, due to its innovative potential, made it possible to obtain the deal in the first place. The rewards are only made possible through the source, i.e., the business, which also covers the compensation financially.
5. *Undergraduate students work with the research team on a student-placement basis.* They may be students whose graduate advisors are employed by a PRO or a research group developing software. There are no copyright agreements with the students to regulate the transfer of copyright and other rights (e.g., in the case of a working invention of other authors, employees of the research group members in which the students are involved). There is an interplay between the roles of the advisor of the first PRO, the project implementer, within which an individual graduation problem is being solved or is related to the project, who trains the student through a graduation thesis, in which a professor employed by another PRO participates as an advisor. Thus, on the one hand, the student thesis is the basis for the diploma thesis, over which certain rights belong to the university. On the other hand, the project's outcome belongs to the PRO, which has certain obligations towards the project funder, wishes and commercialising the work results. The problem arises for several reasons:
   a. *Project managers hire students for routine programming tasks under time pressure*. Collaboration, involving the student in the projects, and training the students through advising lead to more complex tasks that result in original works and inventions.
   b. *The results of the student work are usually based on already created original work or tacit knowledge*, which the advisors make available to the student for use to develop new versions of the software, or this may lead to registered inventions at the PRO.
   c. *Due to the unregulated mechanisms in the field of software IP, the tacit pitfalls of unregulated IP rights concerning student work are not known to the employees of the PROs*, which, in the case of late regulation of rights over IP created with students, usually in the past, requires an additional investment of time and participation in the subsequent regulation procedures, typically between the PRO, the project promoter, and the PRO, the pedagogical program promoter. This may inhibit or even prevent the commercial exploitation of IP for which a market interest has been identified, e.g. industry demand requiring rapid response, contracting and delivery of the solution.

4 Conclusion

The PRO system needs to motivate and reward researchers for marketing the software. Software is legally the property of the employer, who also has all economic rights over it, without having to take possession of it, as is required by law for patentable inventions. As a result, although the software is the property of the employer as soon as it is created, the inventor has no interest in participating in its marketing, as he is not additionally remunerated for any successful sale or licensing due to the

exemption of the Inventions under the *Employment-Related Inventions Act* (in Slovenia).

Nevertheless, it should be remembered that any invention that makes a non-obvious *technical contribution* or *solves a technical problem* in a non-obvious way may be patentable, even if that technical problem can be solved by running a computer program. Consequently, a program code whose technical effect (even if in a non-obvious way) constitutes a technical improvement which is patentable by its very nature. At the same time, the trade secret segment is important since disclosure of program code without a proper proprietary license, or even without any license, may result in commercial damage. By combining the technical effect of the software code with the trade secret effect, it is possible to register the invention also for the software code case and, consequently, to reward the researcher accordingly.

We, therefore, propose, also in the case of software code, regular reflection among researchers within the PROs should be facilitated concerning new, commercially viable software code, a check should be introduced concerning any technical contribution and, consequently, an appropriate registration of the invention based on the software code should be made. TTOs have a crucial role in this respect. Their expertise can contribute to the proper assessment and registration of service inventions and the broader popularisation of the possibility of commercialising software (also protected and registered in this way). At the same time, the proposed method allows researchers working in the field of software code development to be rewarded for their work on an equal footing with those working in the fields of new materials, medical devices, or biotechnology.

Acknowledgement

The operation is partially co-financed by the European Union from the European Regional Development Fund and the Ministry of Education, Science and Sport of the Republic of Slovenia. The operation is implemented under the Operational Program for the Implementation of European Cohesion Policy for 2014–2020, priority axis 1, strengthening research, technological development, and innovation.

References

[1] Closa, D., Gardiner, A., Giemsa, F. and Machek, J. (2011). *Patent Law for Computer Scientists. Steps to Protect Computer Scientists*. Springer-Verlag, Berlin, Heilderberg, Dordrecht, London, New York.

[2] Fric, U., Stres, Š. and Blatnik, R. (2021). Software Protection and Licensing Challenges in Europe: An Overview. *14th International Technology Transfer Conference.* http://library.ijs.si/Stacks/Proceedings/InformationSociety/2021/IS2021_Volume_E.pdf

[3] Johnson, S. (2015). *Guide to Intellectual Property. What it is, how to protect it, and how to exploit it.* The Economist in Association with Profile Books Ltd. And PublicAffairs, New York.

[4] Zakon o industrijski lastnini (Uradni list RS, št. 51/06 – uradno prečiščeno besedilo, 100/13 in 23/20).

[5] Stigler, R. (2014). Ooey GUI: The Messy Protection of Graphical User Interfaces. *Northwestern Journal of Technology and Intellectual Property*, 12, 3, 215–252.

[6] EUR-Lex. (2009). Access to European Union Law. *Directive 2009/24/EC of the European Parliament and of the Council of 23 April 2009 on the legal protection of computer programs.* https://eur-lex.europa.eu/legal-content/EN/TXT/?qid=1598852616560&uri=CELEX:32009L0024

[7] European Patent Office. (2007). *European Patent Convention (EPC 1973)*. https://www.epo.org/law-practice/legal-texts/html/epc/1973/e/ar52.html

[8] European IP Helpdesk. (2020). *Copyright or Patent – how to protect my Software?* https://www.iprhelpdesk.eu/news/copyright-or-patent-how-protect-my-software

[9] Fric, U. and Tomić Starc, N. (2021). Computer-Implemented Inventions and Computer Programs – Status Quo in Slovenia and EU. *Informatica*, 45, 5, 667–673, DOI: https://doi.org/10.31449/inf.v45i5.3468

[10] Neuhäusler, P. and Frietsch, R. (2019). *Computer-Implemented Inventions in Europe. In Springer Handbook of Science and Technology Indicators*, W. Glänzel, H. F. Moed, U. Scmoch, and M. Thelwall, Eds. Springer Nature Switzerland AG, Switzerland, Cham, 1007–1022.

[11] European Patent Office. (2020). The Role of Patents in an AI Driven World. *Digital Conference*. 17–18 December 2020. https://www.epo.org/news-events/events/conferences/ai2020.html

[12] European IPR Helpdesk. (2014). *Fact Sheet IPR Management in Software Development.* https://iprhelpdesk.eu/sites/default/files/newsdocuments/Fact-Sheet-IPR-Management-in-Software-Development.pdf

[13] Cunningham, R. (2007). The Road of Computer Code Featuring the Political Economy of Copyleft and Legal Analysis of the General Public License. In Handbook of Research on Open Access Software: Technological, Economic, and Social Perspectives, K. St. Amant and B. Still, Eds. IGI-Global, 348–362, DOI: https://doi.org/10.4018/978-1-59140-999-1