Artificial intelligence as a new challenge for software law

Dominika Galajdová[a]*

[a]*Institute of Law and Technology, Masaryk University, Czech Republic*

* dominika.galajdova@mail.muni.cz

The author is a PhD student with main interest on software law and impact of artificial intelligence. This work was funded by grant of Masaryk University No. PrF/09/2018 and its contribution towards the author´s thesis “The future of software law in the age of Artificial Intelligence”.

Artificial intelligence as a new challenge for software law

Artificial intelligence (here-after “AI”) has attracted a great deal of attention in the last year. AI is not a new concept or discovery as itself, however, recent progress in this field has been remarkable. Especially, advances in machine learning technology have been noticed by experts as well as by non-professionals. The impact of AI is discussed not just in the fields of technology and science but also in the field of law. The question of regulation of AI by law has been raised in regards to many legal issues, i.e. legal personhood, civil liability, intellectual property, etc. Obviously, the usage of AI varies in different fields with its specific implication for existing legal framework.

AI and its nature resemble the well-known field of software engineering and software as itself. Furthermore, AI technology requires the implementation of software to be functional. Due to this, elements of AI can be copyrighted or patented as software based on its nature. The question is whether these two technologies are the same, or similar enough, so that potential legal regulation of AI can be based on existing software law

Moreover, AI has been used for the purpose of software design, which makes the relationship between software and AI even closer. In the case of software design by AI, the consideration whether there might be an “artificial” software developer is even more interesting. The questions of allocation of authorship and copyright protection of software developed by AI are one of many legal implications of AI on software law.

This article aims to provide an analysis of AI and software based on the state of art of these technologies and open up discussion on the implications of software developed by AI. Therefore, this article is divided into 3 parts with the first part considering existing copyright protection of software with an emphasis on European legislation. The second part provides a description of the current state of art in the field of AI, in particular on AI software development. The final part analyses the clash of copyright and AI with an emphasis on the protection of software developed by AI.

Keywords: artificial intelligence; software development; copyright protection; preparatory design material

1. Software as a subject-matter of copyright protection

Today, software is an "umbrella" term, which encompasses a wide variety of programmes and information sources that controls hardware.[1] Software can be defined as "computer programs, procedures and possibly associated documentation and data pertaining to the operation of computer system" according to the IEEE Standards.[2] This definition involves more than just the final product of software development. However, notably there is no statutory definition of software in existing law, so it can include future developments of unknown technologies.[3]

Software is a broad term, which is heterogeneous and composed of several different elements that vary depending on the type of software and its functionality. In most cases, these elements consist of source code, object code, data flows, algorithms, programming language and general-user interface (here-after GUI). Because of the multi-faceted nature of software, the development of software protection went through different stages of assessing the limits of protection for these aforementioned elements (e.g. Samuelson 1991; Stigler 2014) (See case Oracle, Am. Inc. v. Google Inc., 2014; case *BSA v. Ministerstvo kultury ČR*, C-393/09, 2010; case *SAS Institute Inc. v. World Programming Ltd*, C-406/10, 2012). The nature of software is defined by the very fact that software consists of a set of instructions which are executable by a computer. This

---

[1] The Oxford Dictionary of Computer Science (2016, 510) defines software as a generic term for those components of a computer system which are intangible rather than physical. However, the author is aware that the ambiguous definition of software can lead to conflicting interpretations.

[2] IEEE Std 610.12-1990 Standard Glossary of Software Engineering Terminology. Noteworthy is that this IEEE Standard does not have a legal binding effect.

[3] Definition of computer program in the EU law, deliberately not included because it would be outdated soon (see: Walter and von Lewinski 2010, 100).

means that software will behave in the manner in which it was programmed to by its developer. In such a way; software, in the sense of the final product of software development, behaves in a predictable and predetermined manner. Irrespective of its computational powers and its functionality, software is in some sense a brainless machine, which executes instructions given to it by its operator; therefore, the role of the software developer is undeniable.

The origin of the protection of software was established in the U.S. in the 1970's and 1980's.[4] The National Commission on New Technological Uses of Copyrighted Works (here-after the "CONTU") published in 1978 their Final Report with recommendations on a copyright amendment to Congress. The CONTU report suggested that copyright would be, above all other solutions, the most suitable regime for the protection of software (Mark A. Lemley 2011). Beside CONTU efforts, the World Intellectual Property Organization (hereafter "WIPO") also took a stand on software protection in their Model Provisions[5] and concluded that copyright might be one solution for the protection of software (Shalgi 1986). However, such a solution was subject to discussion and critical evaluations (e.g. Karjala 1987; Samuelson 1988; Goldstein 1990; Ginsburg 1994; Stallman 1994). The fact that software has a utility aspect and it is used for its functionality was the main source of doubts whether copyright protection is the best solution for software (Samuelson 2016). An interesting analogy regarding the functionality of software has been linked to recipes (Longdin and

---

[4] Short overview of origin of software protection in US in Nimtz, R.O. 1979. Development of the Law of the Computer Software Protection. J. Pat. Off. Soc'y 61:43.

[5] Model Provisions For National Law On The Protection Of Computer Software, Agcp/Ngo/Iv/8, World Intellectual Property Organization, Advisory Group Of Non-Governmental Experts On The Protection Of Computer Programs, July 25, 1977

Lim 2013); recipes, like software, consist of a set of instructions and eventually lead to the creation of a product (food or a computer program). The utility nature of recipes is unquestionable and can be viewed as highly valuable. Recipes, like software, are protected as a literary work by copyright, which draw limits on the protection only of the expression of these concepts. It is obvious that functionality is a concept, which goes beyond its expression.

Copyright protection is generally based on a principle called the idea/expression dichotomy. This is a well-established principle which defines the scope of copyright protection (Samuels 1988). Generally, the idea/expression dichotomy divides work, as a fundamental subject-matter of copyright, on aspects which can or cannot attract copyright protection. Copyright protection is exclusively provided just for an author's expression of the work and will protect an author's copyright against illicit copying by others (Buccafusco 2016). Such an expression can be based on an idea; however, copyright does not protect the underlying ideas (Cohen, 1990). The idea/expression principle limits the scope of copyright protection and secures a free flow of ideas, information and abstract concepts. In the case of software, it should be noted that its expression usually takes the form of source code[6] and object code[7]; therefore, the

[6] Source code is the typed instructions that programmers write in a high-level programming language before the program is compiled or interpreted into machine-readable instructions that the computer can execute. A software developer uses a particular programming language for the purpose of creating source code. Programming languages can be divided into high- and low-level languages. High-level languages provide a level of abstraction above the assembly language and enable a programmer to create programs more quickly and easily than assembly language.

[7] The object code is the machine-readable instructions in binary code (consisting of binary digits, e.g. ones and zeros) created by a compiler or interpreted from the source code.

prevailing approach to the protection of software by copyright law was the protection of the source code as a form of expression of a particular software (Bently 2016, 243).

Copyright protection of software is well established within the majority of national legal regimes as well as by international agreements.[8] Any future amendments will not possibly lead to a rebuttal of copyright protection of software at the international level. This would require an adoption of a new legal protection of software since copyright protection is the most common way of software protection in most states (Donát et al. 2012). Moreover, most states are bounded by the international agreements to guarantee minimal standards of copyright protection of software (see below). Nevertheless, states have the option to establish complementary protection out of scope of international regulation while preserving copyright protection.

1.1 European copyright protection of a computer programs

EU legislation has been established by the Directive on the legal protection of computer programs[9] which was later amended (here-after "Directive"[10]). Article 1 of the Directive deals with protected subject-matter and in connection with Recital 7 provides that a protected computer program is a program in any form, typically source code or object code; however, it also adheres to the idea/expression dichotomy. This is evidenced in how the Directive deals with algorithms, which depends on understanding

[8] Article 10 of the Trade-related Aspects Of Intellectual Property Rights, 1994.

[9] Council Directive on the legal protection of computer programs [1991] OJ 2 122/0042.

[10] Directive 2009/24/EC of the European Parliament and of the Council on the legal protection of computer programs [2009] OJ L 111/16.

of the term “algorithm” (Walter and von Lewinski 2010, 103). Whenever, the term algorithm is understood as an abstract idea it does not possess copyright protection.[11]

Another consideration is how the Directive deals with preparatory design materials (here-after “PDM”). The Directive states that PDM are protected as a computer program under the provisions of the Directive. Since such PDM are reflected in the computer program, the special provisions provided for computer programs must also be extended to such materials. Therefore, the PDM benefit from same scope of protection, i.e. set of exclusive rights obtained by the right-holder, as other forms of computer programs protected by copyright, e.g. source code (Walter and von Lewinski 2010, 100). The exclusive set of rights and their exemptions are influenced by the technical nature of a computer program and aim to secure operability and interoperability of a computer program (de Cock Buning 2007). Moreover, the PDM would be expressed in the form of literal, graphical or scientific work (e.g. flow charts or description of computer program sequences) (Bently 2016, 244). However, the character of PDM is standardly different from other forms of expression of a computer program, since the PDM includes all preparatory stages of a program developed with regard to its final structure and features, such as data flow diagrams, descriptions of the program sequences, etc. The scope of protection of PDM in accordance with provisions of the Directive is subject to the same limits as other elements of a computer program, e.g. functionality, programming language, GUI (see below). The application of Article 4, 5 and 6 of the Directive on PDM will depend on the wording of these provisions. While article 4 deals with restricted acts in regards to *computer program* [emphasised

---

[11] Recital 11 of the Directive: “In accordance with this principle of copyright, to the extent that logic, algorithms and programming languages comprise ideas and principles, those ideas and principles are not protected under this Directive.”

added], article 6 specifically refers to the *code* [emphasised added]. Thus, the application of article 6 is excluded in case of PDM. Also, the term *back-up copy* [emphasised added] in paragraph 2 of article 5 of the Directive is related to the computer program in the form of its code.

Article 2 defines the conditions which should be met to provide a computer program protection and says that the computer program is protected if it is its author's own intellectual creation. The requirement of an author´s own intellectual creation has not set other criteria to be applied to determine copyrightable subject matter. Recital 8 states that a computer program enjoys copyright protection regardless of their aesthetic or functional merits and irrespective of whether a minimum level of creativity is involved. The EU legislation is based on a low threshold of originality in the sense that a computer program shall not be just a copy of another program (Walter and von Lewinski 2010, 95). The fundamental principle is that the author shall be the person who is responsible for the manner in which a computer program is expressed not a person who proposes the ideas and principles that underlie it (Bently 2016, 248). The final version of a software product depends on the process of software development, which is one step of software engineering[12]. However, the exact parts of software development depend on the chosen software paradigm and methodology (Ralph 2015). Nonetheless, we can divide software development into software design, which does not involve coding, and the actual coding. The early stages of software development were

[12] Software engineering can be defined as a set of activities which deals with designing, creating, testing and maintaining software products (see generally: Darnell and Margolis 1996, 9). The software development as itself falls into several parts, i.e. software requirements, software design, software construction, software testing, and software maintenance (Bourque 1999).

surrounded by the debate whether the creation of software is more of an art or a science (Lessig 2006). The main argument to support software development as a field of art is obviously the fact that one software product can be implemented in numerous ways and by numerous methods. It is unquestionable that software development generally gives a software developer plenty of opportunity for creative freedom (see for example studies of Fagan 2004 or Gu and Tong 2004) (see case *Eva-Maria Painer*, C-145/10, EU:C:2011:798, 2011, paragraphs 86-94). However, the genesis of software engineering reflects the logic and methodical approach towards software development.

It is apparent that the Directive initially intended to vest copyright in a human author (Bently 2016, 248). The Directive explicitly allows authorship of a legal person which is different to general copyright protection in the Directive on the harmonisation of certain aspects of copyright and related rights in the information society[13] (Rosati 2017); however, the Directive leave particular application of authorship of legal person on legislation of Member States. The Directive also has a special provision for joint authorship works and employment work.

The scope of copyright protection of a computer program is given by a set of exclusive rights of right holder in article 4 of Directive. The fundamental rights of the right-holder are the right for reproduction, right for alterations, and, finally, the right for distribution of a computer program. The right for reproduction has wide application even for the act of mere utilization of a computer program that is subject to authorisation by the right-holder (Walter and von Lewinski 2010, 130). The reproduction of a computer program is an essential technical feature of running a

[13] Directive 2001/29/EC of the European Parliament and of the Council on the harmonisation of certain aspects of copyright and related rights in the information society, [2001] OJ L 167

computer program that such exclusive right covers its standard usage. The text of the Directive also does not cover loading of an embedded computer program as a restricted act (Walter and von Lewinski 2010, 131). This is especially relevant regarding the emerging sector of software as a service where software is delivered from a central host server without executing a copy of the accessed software (Brereton and Budgen 2000).

The acts of alterations cover adaptation, translation or alteration and are dealt with paragraph 1b) of article 4 of the Directive. Adaptations are understood in the sense of a new and original creation, and so are protected by copyright. On the other hand, translations are deemed to be a particular form of adaptation or transformation in the literary area; in regards to software, translation means transformation from one programming language to another while adaption is further development of the software from one development stage to the subsequent. However, the translation of source code to the object code is not an adaptation of software, because of lack of its originality; however, such act requires a permission of right-holder in compliance with the Directive. The Directive does not expressly define the aforementioned terms and does not properly distinguish between paragraph 1a) and 1b) of Article 4 when such state does not have further negative implications for legal practice (Walter and von Lewinski 2010, 131-2).

Lastly, the Directive in Article 4 paragraph 1c) deals with the right of distribution to the public. This right is independent and complementary to the right of reproduction. The distribution rights secure the right holder interest also against distribution of “pirated copies” which state is less difficult to prove within proceeding (Bently 2016, 254). It is apparent that the major source of software protection is secured by copyright law, however, design of relevant legislation highly reflects nature of software and aim to secure the interoperability of software (de Cock Buning 2007).

As this system reflects traditional copyright principles, at the same time it establishes *sui generis* regime for the regulation of software from traditional copyright (see the Directive on the harmonisation of certain aspects of copyright and related rights in the information society). Previous section briefly describes existing European copyright protection of computer programme with its emphasis on principal statutory requirements for copyright protection as standard of originality, authorship of computer programme as well as set of exclusive rights. The European copyright protection of computer program is well-established legislation and as itself serve to its desired purpose. However, the progress in the area of software development might challenge this established legal framework, so the software development will no longer meet statutory requirement for gaining copyright protection. Such scenario will be described in following parts.

2. AI implementation in software development

The technical part of software development has transformed considerably from its earlier stage at the beginning of the rise of the software industry (Darnell and Margolis 1996, 2). In its earliest days, software development would require an inordinate amount of time and effort of a software developer with programming in assembly language and defining various elements of the software. While today, there are multiple tools and programming languages available to software developers. These tools are commonly known as Computer Aided Software Engineering tools (here-after "CASE tools").[14] The development of CASE tools challenged the newly established

[14] The broader term is Computer-Aided Design which can be defined as an application of computer technology/software to the design of product, or the design itself. The Computer-Aided Design is used in architecture, engineering, etc. See generally, A

legal protection of software regarding the subject-matter and scope of copyright protection (e.g. Arsenault 1994). CASE tools provide better technical means for software design; however, the manual participation of a software developer on the final output is decreased. For example, most common CASE Tools are a source code editor, build automation tools and a debugger and might be implemented in an integrated development environment (IDE), which provides comprehensive facilities for a software developer. Simply said an IDE is like a word processor for software development where the user will write a textual expression of a desired program while the system will execute the machine-readable code and point out possible errors in the program. Moreover, it is apparent that there is no longer a need for a software developer to manually write source code in one of the aforementioned programming languages because usage of CASE tools even allows the generating of source code by itself (e.g Budinsky et al. 1996). However, the human programmer remains integral to the development process. Nevertheless, utilising AI to aid the development of computer programs has been a long-held desire (see: Manna and Waldinger 1974). The next section will provide a brief description of AI as a technology and will be followed by demonstration of AI software development at the current state.

2.1 AI as technological progress

While AI is a well-established and commonly used term within a wide variety of

---

Dictionary of Computer Science (2016, 106). For the purpose of software design, there are specialised tools (CASE tools) for the creation of software. The IEEE Std 610.12-1990 Standard Glossary of Software Engineering Terminology defines CASE as a use of computer to aid in software engineering process which may include and application of software tools to software design requirement tracing, code production, testing, document generation.

media and communication channels; a search for an exact definition of AI, can prove confusing. There is no general definition of AI within either the research field or a statutory definition enacted by laws. Furthermore, an AI may be implemented in many ways and has many forms, which does not help in an attempt to adopt such a definition.

John McCarthy, who first coined the term artificial intelligence, presented the general definition of AI as the "science and engineering of making intelligent machines, especially intelligent computer programs".[15] The definition was provided specifically for non-professionals and thus has not been widely accepted by researchers in the field of AI. Another early pioneer in AI technology, Marvin Minsky, defined AI as "the science of making machines do those things that would be considered intelligent if they would done by people" (Minsky 1967). Both of these definitions provide a very general concept and involve general terminology, which can lead to ambiguous interpretations of what constitutes AI.

Another definition of AI is provided by Stuart Russell and Peter Norvig who approached the definition of AI from different perspectives, i.e. thought process, computational science, and reasoning. Russell and Norvig (2009) provided four approaches for how to define AI: the rational agent approach, the "law of thought" approach, the cognitive modelling approach, and the Turing test approach. The four approaches provide the characteristics that define an AI. So, the aforementioned

---

[15] McCarthy presented his definition at a conference on the campus of Dartmouth College in the summer of 1956 indicating the beginning of AI research, and the attendees, including John McCarthy, became the leaders of AI research for many decades. The record of the FAQ answered by John McCarthy is available on http://jmc.stanford.edu/articles/whatisai/whatisai.pdf

approaches can be used for the purpose of development of AI, which would reflect its definition.

It is therefore apparent that the community working in the AI field does not rely on exact terminology, which makes it harder to provide clear unambiguous explanations even of its basics. However, AI can be understood as a field of research not just as a technology, which may implement an AI. Since the methods of implementation of AI technologies can vary distinctively such an attempt to try and define based on the technology can exclude some other concepts which belong to the field of AI. However, for the sake of further text, AI will be understood as set of components (e.g. software, dataset, etc.), which allow a machine to execute operations that can be deemed as intelligent.

There is obviously a strong link between software, as it was described in previous sections, and AI. We can conclude that both technologies are implemented in a computational machine, are universal and heterogeneous. There is no statutory definition in the case of software neither of AI. Also, both technologies include an algorithm and data. Generally speaking, AI can be perceived as software from a technical point of view; however, AI enhances software capabilities beyond the original inception of software and with further development may possibly transform software into complex more human-like machines which may lose its role as a pure tool in human control. Generally, the difference between software and AI is the fact that software is most commonly algorithmic whereas AI is typically heuristic. At the moment, there is an obvious distinction between AI and software in their functionality and nature.

2.2 AI approach to software development

Outstanding progress in the area of software development by AI has been placed

in program-like neural networks (Graves, Wayne and Danihelka 2014). However, such systems cannot be truly viewed as creative as they do not generate human readable source code. Therefore, recently interest has been redirected to inductive programming (here-after "IP"), whereby inductive reasoning methods are used for programming, algorithm design, and software development (Kitzelmann 2009). An example of this technique can be seen in the automated string process in recent versions of Microsoft Excel (Singh and Gulwani 2012). The underlying algorithm is able to automatically generate a vast amount of outputs based on just a limited number of example outputs. A more complex example is the STOKE super-optimization system developed at Stanford University (e.g. Schkufza, Sharma and Aiken 2013). STOKE uses stochastic local search to identify input programs which can be made more efficient based on pre-existing assembly programs.

The next step has been to combine IP technology with a neural network, which was demonstrated by the Cambridge University and Microsoft developed Deep Coder (Balog et al. 2016). Given just a list of desired inputs and outputs, Deep Coder can piece together lines of pre-existing code. Deep Coder is based on a Learning Inductive Program Synthesis (LIPS) approach and includes these components: 1) Domain Specific Language (DSL) specifications, 2) a data-generation procedure, 3) a machine learning model, and 4) a search procedure. The main idea behind Deep Coder is to use a neural network to guide the search for a program consistent with a set of input-output examples without directly predicting the entire source code. For that purpose, the neural network with three hidden layers is trained on a specified dataset to identify functions of the DSL and provide encoding of each individual input-output. Besides other things, it has been demonstrated that Deep Coder is able to generalize beyond programs of the same length that it was trained on. There are obvious limitations of DeepCoder and its

results, e.g. the complexity of synthesized programs; however, it shows possible future direction for the usage of AI for program synthesis.

Another example of AI software development focuses on learning representation of software through graphs which represent both syntactic and semantic structure of code. Then these graphs are used for deep learning methods to learn to reason over program structure. The main assumption is that this method should lead to a decrease in the amounts of training data, model capacity and training regime and so improve the present state-of-the-art. This method also allows detection of bugs in mature open-source software projects. Outcomes of this work should be valuable for code completion and more advanced bug finding and can be considered as a first step towards the core challenge of learning the semantic of source code (Allamanis, Brockschmidt and Khademi 2017).

The last example of AI software development is pix2code, which is a deep learning method to generate code from a screenshot of a GUI. pix2code is based on Convolutional and Recurrent Neural Networks allowing the generation of computer tokens from a single GUI screenshot as input (the model learns from the pixel values of the input image alone). This method can be used for generating computer GUI code for various platforms (i.e. iOS and Android native mobile interfaces, and multi-platform web-based HTML/CSS interfaces) without the need for any change or specific changes to the model. In fact, pix2code can be used as such to support different target languages simply by being trained on a different dataset. This method can minimize the time that a software developer would spend on coding of particular GUI (Beltramelli 2018).

These examples are by no means an exhaustive list of examples of AI software development; however, what they demonstrate is that although AI is not yet able to create new programs solely by itself the role of a human programmer can be minimised

in particular development phases to merely providing a list of desired inputs and outputs. While software development is composed of different phases with its level of abstraction in regards to it required usage of specific methods, present state-of-art does not allow software development purely based on AI. Such a transfer of software developments towards AI may require more sophisticated technology nearly general intelligence.[16] The general intelligence would probably allow a combination of different methods and approaches necessary during software development as well as a high level of abstraction, generalization, incremental learning and even creativity. At this stage, unsupervised learning can be deemed as the next step towards more autonomous software development (Murphy 2012). If such movement can rebut the lack of creativity and role of human developer, is a good question.

Since copyright protection is strongly linked to the expression of software in the form of source and object code and vests the rights in the human (see above), such technological development may lead to legal uncertainty whether the software developed in the described way can be protected by copyright. Therefore, the protection of non-literal and abstract elements of software may become increasingly important, since the participation of a human can be solely connected to such non-literal elements represented by the provided inputs and outputs or more abstract concepts of software. The presented disparities might prove to be one of the main sources of the future struggle of tackling the synergy of copyright law and AI technology.

---

[16] The recent research on general intelligence and its outcomes were discussed on conference Human-Level AI 2018, 22-25th August, Prague; for more information see https://www.hlai-conf.org/.

3. Clash of copyright and AI

The law has been trying to cover AI and its different aspects. It is evidently apparent that AI and its usage can have significant impact on human lives and society as a whole.[17] The law has faced several challenges by the AI, i.e. civil liability, legal personality and even copyright. Thus far, the legal framework has not enacted any direct regulation of AI. Most suggestions related to AI are based on present legislation. Despite this fact, AI is a subject-matter of particular legal provisions since the elements of AI are protected by copyright as well as patent law[18,19].

The main problem of AI and copyright is the status of AI as an author and protection of AI generated outputs (Lambert 2017). While attention is mainly focussed on AI generated outputs in textual, visual or audio form as the traditional subjects of copyright, AI developed software is not explicitly discussed in literature (Butler 1981). Nonetheless, the same grounds and remarks in the former area can be applied in case of AI generated software. The question of how AI may influence intellectual property has been raised multiple times by the WIPO (Gaudamuz 2017). The WIPO Worldwide Symposium on the Intellectual Aspects of Artificial Intelligence[20] was held in 1991 with

---

[17] Recently, the attention of the public is caught by fatal accident of self-driving car, whether question of liability is discussed. See for example: Levin (2018).

[18] For example: U.S. Patent No. 5,695,666 (filed Oct. 13, 1994), US 8,126,832 B2 (filed Mar. 6, 2008), EP1277201B1 (granted on Oct. 10, 2010), EP0681284B1 (granted on Jul. 3, 2007), etc.

[19] European Patent Office released Guidelines for Examination of Artificial Intelligence and Machine learning technology and its patentability on its website (https://www.epo.org/law-practice/legal-texts/html/guidelines2018/e/g_ii_3_3_1.htm).

[20] See WIPO Worldwide Symposium On The Intellectual Property Aspects Of Artificial Intelligence, Stanford University, Stanford (California), United States of America, March 25 to 27 1991, World Intellectual Property Organization, (hereafter WIPO Symposium 1991) available at ftp://ftp.wipo.int/pub/library/ebooks/wipopublications/wipo_pub_698e.pdf.

the participation of experts from the field of computer science, law and business. The WIPO Symposium 1991 provided space for discussion of different aspects of AI and also suggested some possible approaches toward this new technology. During the WIPO Symposium 1991, an appealing idea was presented, that AI regulation should be considered from the point of view of software protection (Davis, 1991).

The next section provides an assessment of clash of copyright legislation with AI at the international and national level. The outcome illustrates the very friction areas between traditional concepts of copyright and new phenomena of AI. This section is followed by current limits of copyright protection of software and its possible application in case of AI development. The last section demonstrates potential future consideration of status of AI outcomes and their protection.

3.1 International and European copyright perspective on AI

The current legal framework of copyright developed through two different concepts, copyright as the right to copy (Goldstein 2013), and author´s moral rights (Sterling 1998; Goldstein 2013). The Berne Convention[21], as well as other international treaties, helped to bridge the two traditions with minimum standards, which dictate substantively similar rules for countries in both systems. The role of the Berne Convention is undoubtedly crucial in the course of creating a universal procedural framework based on the principle of national treatment. The fundamental principles and concept of copyright law has been harmonised at a minimum level (e.g. authorship) via international instrument or left outside (e.g. creativity). One issue which was identified

[21] Berne Convention for the Protection of Literary and Artistic Works, as amended on September 28, 1979, (hereinafter the "Berne Convention") available at http://www.wipo.int/wipolex/en/treaties/text.jsp?file_id=283698.

based on collision between AI and existing copyright law is the question of authorship. It is noteworthy that the international treaties deal only with the terms of author and leaves the specific wording to their Member States (Ginsburg 2002). The necessary space for national rules, which vary on the question of the person who possess copyright, remains (Rosati 2017). The Berne Convention conjunct the term author in Articles 3 and 7 with his nationality and death, moreover, it declares also moral right to the author, so it is apparent that the Berne Convention recognises as an author the natural person. Noteworthy, the Article 1 of TRIPS agreement expressly states that the nationals of other Member States shall be understood as those natural or legal persons that would meet the criteria for eligibility for protection. At the European level, in the case of computer program, the Directive expressly defines an author as a natural or legal person in article 2 (see above). In conclusion, copyright law always requires at least a legal person in common and does not allow to vest copyright in a non-human entity without any legal status (Denicola 2016).

The provision of US Copyright Code might seem more favourable towards the possible adoption of a new concept of author, however, the US Copyright Office has expressly demonstrated[22] that copyright could be granted only to the works of a natural person — "*the U.S. Copyright Office will register an original work of authorship, provided that the work was created by a human being….the Office will not register works produced by a machine or mere mechanical process that operates randomly or automatically without any creative input or intervention from a human author*". This opinion is also supported by analogy with animals and divine person (Denicola 2016).

[22] U.S. Copyright Office, Compendium Of U.S. Copyright Office Practices §§ 306, 313.2 (3d ed. 2017)., available at https://www.copyright.gov/comp3/

The insistence on the requirement of human authorship provides the main obstacle over what copyright rules possess for the usage of AI technologies.

The second crucial issue of AI and copyright is creativity as one of the requirements for copyright protection (Bridy 2016, Schönberger 2018). The question whether an AI can be truly creative is the topic of hot debate. Ginsburg and Budiardjo (2019) argue that current AI technology cannot be deemed truly creative and further observe possible allocation of authorship of humans and generative machines (i.e. ordinary tools, partially-generative machines, fully-generative machines), contrarily Bridy (2012), and Dorotheou (2015), have argued that in terms of computational creativity, machines can pass the standard of creativity set up in *Feist Publications Inc. v. Rural Tel. Service Co.,1991*. Boden distinguishes machine creativity and defines it as the ability to come up with ideas or artefacts that are new, surprising and valuable, thus machines can be deemed creative (Boden 2004). Assessment of machine creativity has a significant impact on the protection of AI outputs. Regarding to machine creativity, we can also distinguish between situation where AI is merely used as a tool, AI and human cooperation and AI acting autonomously.

Lastly, the question of allocation of copyright amongst persons who participated on an outcome is yet another issue of AI and copyright law. Davies (2011) presented several options on the fundamental question who should have copyright to the outcome. The present debate covers the author of AI, owner, author of dataset, user or AI itself (Bridy 2012; Dorotheou 2015). The last suggestion, AI as an author, may lead to some surprising conclusions whereby an AI would be treated as a subject-matter of rights and simultaneously would have the capability to possess rights (Krausová 2007). The implication of this concept was discussed in Galajdová (2018). There is no prevailing solution over the issue of allocation of copyright; however, this question is subsequent

to the question of authorship and criteria of creativity of AI which should be solved first.

In the European context, there is discussion over the issues mentioned in previous paragraphs which are partially addressed at the legislative level by a Report of The Committee of Legal Affairs (hereafter "CLA")[23].The CLA in relation to robots states in its recommendations (point 59 f. of Motion for European Parliament Resolution) that the Commission should consider an option of creation of specific status of electronic person in connection with usage of autonomous robots with its possible implication on civil liability.[24] It is not sure if the status of electronic person will be placed on an equal footing with a natural or legal person or whether it status will be distinguished from these traditional concepts. In such a case, AI might not be deemed as an author with this special status in copyright perspective. In the Explanatory Statement of Report of CLA, the need for reconsideration of standard of originality in connection with computer or robots generated outputs is highlighted. While, the effort at the EU level is welcomed there is not universal agreement regarding the regulation of AI and how to best proceed. The European Economic and Social Committee stated that "is opposed to any form of legal status for robots or AI (systems), as this entails an

---

[23] Report with recommendations to the Commission on Civil Law Rules on Robotics, Committee on Legal Affairs, Rapporteur Mady Delvaux, published on 27th January 2017, (here-after the "Report of CLA") available at http://www.europarl.europa.eu/sides/getDoc.do?pubRef=-//EP//TEXT+REPORT+A8-2017-0005+0+DOC+XML+V0//EN

[24] Report of CLA, Motion for European Parliament Resolution, Liability, AB: "whereas the more autonomous robots are, the less they can be considered to be simple tools in the hands of other actors (such as the manufacturer, the operator, the owner, the user, etc.)"

unacceptable risk of moral hazard" in its Opinion INT/806.[25] The fact that there is significant attention on the issue raised by AI at the EU level implies that policy makers are aware of limitations of current legal framework and its ability to accommodate AI. However, it is clear that there are significant opposition and hurdles that must be faced before a conclusive policy towards regulation of AI and its potential creation can be formulated. Nonetheless, software protection, in part thanks to its fragmented nature, can provide a temporary solution in its shift to protection of PDM as it will be described below. However, the lack of analysis of the application of the Directive in relation to PDM raises a question about complexity and strength of such protection.

3.2 Case of software protection and its limits – AI perspective

While the European legislation is well-established in the Directive, the Court of Justice of EU (hereafter "CJEU") has tested the limits of copyright protection of software covered by the text of the Directive. The CJEU has ruled over scope of protection of various elements of software and in some cases refused to broaden copyright protection. The CJEU based its decisions on various ratios as well as on existing state-of-art of software and its development. The question is whether such restrictive approach cannot provide to be obstacle with future progress in the area of software development.

In case C-393/09, *Bezpečnostní softwarová asociace - Svaz softwarové ochrany v. Ministerstvo kultury*, the CJEU brought a decision on the protection of other elements

---

[25] European Economic and Social Committee, Artificial intelligence – The consequences of artificial intelligence on the (digital) single market, production, consumption, employment and society, INT/806, 31/05/2017.

of software by EU law.[26] The CJEU concluded that the GUI is not a form of expression of a computer program, so it cannot be protected by copyright as a computer program under existing EU law. Moreover, the CJEU stated that the GUI does not enable the reproduction of software, but merely constitutes one of its elements by means of which users make use of the features of that program (Griffiths 2011). The CJEU interpreted what forms of a program should be understood in the terms of the Directive, where the most important is that the form of expression of a program enables the program to perform the task for which it was developed. However, the CJEU admitted that a GUI can be protected by copyright as a work if that interface is its author's own intellectual creation but there is limited chance of such result (Griffiths 2011; Šavelka 2011). Consideration of this case with the application pix2code may provide an interesting outcome. If there is no copyright protection of the GUI as software, there is no restriction on the usage of such an application to generate GUI code. However, there is a principal question whether AI as itself can even breach someone´s copyright (Grimmelmann 2016). Furthermore, the case C-393/09 provides a clearance on the question of form of expression of computer programme and which should be created by human author or legal person in regards to be protected by European copyright.

The functionality is another essential element of every software and adds to software its utility nature. Generally, the functionality determines what a particular software does and for what purpose it can be used. This very aspect highlights the different nature of software from other subject-matters protectable by copyright.[27] A book, film or song does not include a utility element and its copyright protection flows

---

[26] Council Directive on the legal protection of computer programs [1991] OJ 2 122/0042.

[27] General advocate opinion in case C-393/09, p. 59.

from the utilitarian approach (protection of investment) (Goldstein 2013, 48) or natural theory (personal rights of author) (Sterling 1998, 16). Protection of the functionality of software is closely linked with the non-literal infringement of software. The crucial feature of software is that its functionality can be accomplished in numerous ways. This fact leads to situations where different implementations by different source code can lead to the same functionality of two or more programs (Donát et al 2012). Since copyright protects the source code, the risk of collision of two programs with similar functionality but different source codes exist.

Relevant case to the protection of functionality was ruled by the CJEU in C-406/10, *SAS Institute Inc. v. World Programming Ltd* where a number of preliminary questions were raised by the High Court of Justice of England and Wales. The CJEU followed its previous ruling in the case C-399/09 and denied protection of functionality of computer program as well as protection of programming language and the format of data files used in a computer program within the scope of existing EU law (see C-406/10, *SAS Institute Inc. v. World Programming Ltd*, paras. 39-46*)*. The court relied on the General Advocate opinion in reasoning why functionality should not be copyrightable, because it would have a negative effect on technological progress and industry development (see C-406/10, *SAS Institute Inc. v. World Programming Ltd*, para. 40). In the next lines, the CJEU declined protection of a programming language and data files as they are elements by means of which users exploit certain functions of a particular program (see C-406/10, *SAS Institute Inc. v. World Programming Ltd*, para. 42). The court has not found the form of expression of computer program in any of these elements, so the CJEU has not established copyright protection. The decision was commented on by the public and critically assessed for its restriction on protection of

software as well as for its reasoning (e.g. Šavelka 2011; Samuelson, Vinje and Cornish 2012; Vezzoso 2012).

This ruling is important to the question of extending copyright protection of software within the EU framework. The CJEU points out the rationale behind declining protection of functionality as a protection of technological progress and industry development. Such rationale is not alien to copyright justification as it should not serve as a stiff instrument which hampers creativity and information flow. The CJEU also ruled over protection of abstract concepts when it applied the idea/expression dichotomy principle and declined protection of abstract ideas. The functionality is protected via its expression in final product, however, the concept behind its expression has not copyright protection. In relation to AI, the question of abstract concept is important since it was demonstrated that AI has not yet achieved capability of abstraction and higher level of generalization. It is hard to presume when the AI might achieve such capability and if it is even possible at the current state. Logically, the space for human development of software stays out of the scope of AI. This shift leads as to reassessment whether the abstract concept as functionality should be protected if they are created by humans and if they meet statutory requirements. While, historically, such a shift might have hampered the whole field of software development, the current situation might require such a change. The software development process can currently be divided into parts which include human creativity and AI creativity. The AI can successfully generate the specific expression of final product which would be based on abstract and general ideas of human creator, such as essential functionality of the final product. In reflect of case C-406/10, there will not be no protection of such abstract elements as well as current legal framework will not allow protection of AI output. Such

approach seems to be *a contrario* to justification of protection of technological progress and industry development which was made by the CJEU in case C-406/10.

Considering the shift to more abstract elements, another area that requires further clarification regards the scope of protection provided to PDM. Recently, the CJEU obtained a request for a preliminary question by the Swedish Court of Appeal in the case C-313/18, Dacom Limited v IPM Informed Portfolio Management AB, lodged on 9 May 2018. Amongst others, the Swedish court asked the following questions:

- what criteria determine whether material constitutes such PDM according to the Directive,
- whether documents which set out the requirements as to the functions which are to be performed by a program and the result which the program must achieve (e.g. detailed descriptions of investment principles, risk models for asset management) can constitute PDM,
- if material must be so complete and detailed that in practice it requires no independent choices on the part of the person who actually writes the code of a computer program, in order to constitute PDM,
- if the exclusive right to PDM means that the program in which the PDM subsequently results is to be regarded as its adaptation and therefore a dependent work for the purpose of copyright, or that the PDM and software are to be regarded as different forms of expression of the same work, or that they are two independent works.

The ruling over the protection of PDM is important in the question of interpretation of PDM in the sense of the Directive. Thus far the PDM has not been subject to legal dispute and so has not been critiqued in the legal literature. However,

given the rise of AI involvement in software development the protection of more abstract form of expression of software such as PDMs may become more important. Therefore, clarity regarding what constitutes PDM and what forms of PDM are protected is necessary. The CJEU in the aforementioned dispute has the chance to provide such clarity and set up regime of protection of PDM. The crucial element of the ruling will be interpretation of PDM in terms of the Directive and its compliance with the reality of software development. Such interpretation of the form of PDM should closely reflect existing practice in the technological and legal field. If such harmony will not be reached, the newly established manner of copyright protection might become immediately obsolete and useless. There are no doubts that the CJEU is bound by its previous ruling on the Directive as well as by the text of the Directive itself. In such a scenario, the CJEU shall not protect elements of software, e.g. programming language or GUI, which are excluded from protection even though they are expressed in the form of PDM.

There is a recent decision delivered by the Dutch Supreme Court on copyright protection of PDM which is restricted to the scope of protection. The Dutch Supreme Court concluded that if the PDM needs further translation into a software by means of creative steps in programming, it cannot qualify as protected PDM (Cohen 2018). The decision limits protection of PDM to the stage in which it is protected or not. However, such limitations might have been problematic especially because the evaluation of PDM is done *ex post* while at the moment of securing investment in the software development the proper means of protection might later be proved to be insufficient.

Moreover, the question of the relationship between PDM and the final product is crucial in the way that if the final program is independent work it will have to always establish its own protection and meet the requirements of that criteria. Especially, in the

case of AI development, it would mean that the final product in form of its code is not protected if it is autonomously developed by AI, while the PDM developed by a software developer will be protected. In contrast, if the final product is just derivative work, we might assume that it can be protected by copyright of author of PDM.

There is apparent tension between the protection of software by copyright and the nature of software. The copyright protection has been applied by the CJEU in a narrower way. There is an option of protection by patent, trade secret or contract provisions as a supplementary means. However, the nature of software also led to discussion of adoption of sui generis protection, which can provide legal certainty for software protection (e.g. Phillips 1991; Ginsburg 1994; Toeniskoetter 2005). There was also provided analogy with enactment of sui generis protection of semi-conductor chips (Samuelson 1986). Thus far, *sui generis* protection has not moved beyond discussions in the academic sphere and possible adoption does not appear on the horizon. However, the AI and its capabilities might be appropriate incentive to adopt such *sui generis* regime and reassess current European copyright protection of software, especially, since AI highlights the gaps in the current legal framework. Moreover, future development and progress in this area can demonstrate that copyright protection cannot be stretched to secure interest of right-holders.

3.3 The protection of software developed by AI

There are possible other considerations of status and protection of AI outcomes on the current legal framework. Such considerations can be made particularly in case of cooperation of human and AI. The cooperation of human and AI and its implication for copyright protection will depend on the participation of a human in the process of generation of output. While some applications are mere tools in the hands of a human, some applications operate autonomously with only minimised human interaction of

mere switching on the application (Ginsburg and Budiardjo 2019). In the latter case, the human role and his activity can be separated into creative and non-creative activity, which results in the idea or expression. While creative activity which results in the expression of an idea will be automatically protected by copyright with no doubts, the residue will depend on assessment and potential copyright protection is jeopardised. Butler (1981) concludes that, if the computer owner or the owner of the underlying AI software rights claims he is directly involved in the authorship of the resulting expressions, he must be able to prove his input in the creation of the finished work was substantial. Thus, in the normal programming environment, the developer can exercise a large degree of originality and creative choices in the formulation of an algorithm and in the expression of idea underlying software. However, applications like DeepCoder require only a problem statement, the computer then develops a solution. The element of human input into the program-creating process is minimal.

Wu (1997) suggested a test for derivative works in case that AI also participated in output. A similar test according to the assessment of copyright protection of output of human and AI cooperation can be suggested also in case of software. In case of software development, the test is little modified.[28] Firstly, the outcome will be separated from the elements, which are not protected by law, such as GUI or programming language. Such separation of unprotected elements has been used in USA case law in non-literal infringement in the case Computer Associates International, Inc. v. Altai, Inc., 982 F.2d 693 (2d Cir. 1992). The second step will involve an analytical assessment to determine which part(s) of the outcome has risen from a human developer activity. Then the outcome is split into parts which are linked to a human and which are linked to

[28] For the purpose of this test, the outcome is understand as a final software product as whole.

an AI. After such determination, the assessment whether the parts associated with a human are fixed in the expression of the outcome and it is not only linked to the idea underlying such expression. If linked to the expression of the outcome, the last step would be to assess whether it fulfils the requirement of originality. If so, part of the outcome in the question is protected by copyright, which flows from the human. The question of copyright protection of the outcome as a whole would probably depend on the extent and importance of the human participation on the final outcome. The copyright protection of a human author can be extended if there is a nexus between outcome and human creativity (Bridy 2012). This test can provide a tool to analyse the status and nature of an outcome.

The decision whether the threshold of extent of the human participation on the final outcome will be low or high would depend on court decision. Logically, a high threshold does not make sense by virtue that this would reflect the concept of AI as a mere tool. A lower threshold might provide an adequate solution for most cases of AI and human cooperation. This test will not be applicable in cases where it is apparent that AI generated its outcomes almost autonomously and independently, where the human (user) input consisted of a simple order to generate code it cannot be deemed as participation on final product (Wu, 1997). In such scenarios, there is still possible participation of person who developed the AI which can be examined by the same test. There is still opportunity to modify the aforementioned test based on the nature of the outcome when in the case of a database the criteria can be oriented in investment in the input (Ramalho 2017). Also, in case of national application of this test, the threshold might be modified in regards to subject-matter of the test, e.g. uniqueness criteria for originality (Zibner 2017). Importantly, the described test provides a solution for

copyright protection of AI outputs which is vested in a human author. This assessment excludes the concept of AI as an author.

Conclusion

In the case that AI will continuously be implemented for software development, the established scope of the copyright protection will not be enough (Goldberg and Carson 1991, Schrönberger 2018). Although, AI cannot fully replace a human software developer yet, we can already recognize the possible side effects of AI on software protection. The fact that AI can generate code, which is one of the essential elements for protection, might shift the whole idea of copyright of software. The key moment is when a human will no longer be creator of the code, so the code will not be the human developers own creation. The code will still be the form of expression of software; however, it will not originate in the human activity but be created by a machine. In such a case, the overall question of authorship of code arises.

Will software in the form of its source code still be protected by copyright if AI generated it? Is it still a human creation? Compared to CASE tools, mentioned earlier, AI demonstrates skills to generate code with minimum human inputs, while CASE tools still present application software where the role of the developer is indisputable.[29] This shift in software development may challenge copyright protection in the near future since it was proved that the current state of art in AI research continues to improve in the field of code generation and future developments may provide even more complex code production. With ongoing research and its outcomes, it might be important to detect the link between a human author and the produced source code. In this scenario,

[29] See part 2.

the protection of other forms of software will become important, although current practice shows limited desire to protect these other forms. However, these forms will probably be tightly bounded with human development in the long-term period. As it was discussed, fully AI software development is a future desire which goes beyond the horizon of existing technology. Regarding to that the pending preliminary ruling before the CJEU might become a breakthrough in terms of future copyright protection of software. The CJEU interpretation of PDM might demarcate the very scope of remaining copyright protection of software, if the AI software development upgrades to the level when it will generate longer and more complex codes. While the CJEU already declined copyright protection of various elements of software such as GUI, programming language, format of data files and mostly its functionality, the protection of leftover elements is gaining importance. Although, the CJEU has not showed any hesitation to protect software in the form of its source code or object code, such a protection might become obsolete in the case that these are generated by AI and are no longer a human creation. The only remaining element of software to be protected under the existing copyright framework will be PDMs and other abstract elements such as algorithms. Whilst the protection of abstract elements of software might be difficult to obtain as well as it may possess adverse effect on the software industry in the form of increasing risk of monopoly over fundamental concepts, the protection of PDMs shall be a pragmatic solution. As it was mentioned, there is a high probability that in upcoming future the software development with its more abstract concept and preparatory works will be still carried by human developers.

There is ongoing discussion of consequences of AI on copyright and authorship of AI in light of traditional copyrighted works (e.g. Guadamuz 2017). There are suggestions towards AI generated creations with the desire to track back to a human

author, which could be either AI developer or AI user (e.g. Denicola 2016). The refusal of protection of AI generated outputs has been proposed as well (e.g. Grimmelmann 2015). Lastly, the AI authorship has been discussed as a possibility (e.g. Hristov 2016). Parallel evaluation can be made in relation to AI software development. However, if the fully AI generation of code will be reached, tracking a human author of code and proof of his creation might be impossible (Liu 2018). In conclusion, upcoming challenge for copyright protection of software appears on the horizon while the certain collapse of software copyright protection has not been foreseen yet. Nevertheless, a shift in software protection can be perceived in the near future.

Acknowledgements, avoiding identifying any of the authors prior to peer review

This article benefited from numerous discussions with doc. Dr. Radim Polčák and Dr. Matěj Myška as well as two anonymous reviews. Additionally, Dr. Stephen Collett is thanked for proofreading and English corrections.

References:

Allamanis, M., M. Brockschmidt, M. and M. Khademi. 2017. Learning to represent programs with graphs. arXiv preprint arXiv:1711.00740.

Arsenault, J.G. 1994. Software without Source Code: Can Software Produced by a Computer Aided Software Engineering Tool Be Protected. Alb. LJ Sci. & Tech. 5:131-162.

Balog, M., Gaunt, A., Brockschmidt, M., Nowozin, S., Tarlow, D. 2017. Deep Coder: Learning to Write Programs. ICLR. Toulon. France

Baroni, M., A. Joulin, A. Jabri, G. Kruszewski, A. Lazaridou, K. Simonic, and T. Mikolov. 2017. CommAI: Evaluating the first steps towards a useful general AI. arXiv preprint arXiv:1701.08954.

Beltramelli, T. 2018. pix2code: Generating code from a graphical user interface screenshot. In Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems.

Bently, Yin-Harn. 2016. Concise European Copyright Law. Directive 2009/24/EC – on the legal protection of computer programs (Computer Programs Directive). Netherlands: Wolters Kluwer International.

Boden, M. 2004. The Creative Mind: Myths And Mechanisms. London: Routledge.

Bourque, P., R. Dupuis, A. Abran, J.W. Moore, and L. Tripp. 1999. The guide to the software engineering body of knowledge. IEEE software 16:35-44.

Brereton, P., Budgen, D. 2000. Component-based systems: A classification of issues. Computer 33: 54-62.

Bridy, A. 2012. Coding Creativity: Copyright and the Artificially Intelligent Author. Stan. Tech. L. Rev. 5.

Bridy, A. 2016. The Evolution of Authorship: Work Made by Code. Colum. JL & Arts 39:395.

Buccafusco, C. 2016. A Theory of Copyright Authorship. Va. L. Rev., 102:1229.

Budinsky, F.J., M.A. Finnie, J.M. Vlissides, and P.S. Yu. 1996. Automatic code generation from design patterns. IBM systems Journal 35:151-171.

Butler, T.L. 1981. Can a Computer be an Author -Copyright Aspects of Artificial Intelligence. Comm/Ent L.S. 4:707.

Cohen, A.B. 1990. Copyright Law and the Myth of Objectivity: The Idea-Expression Dichotomy and the Inevitability of Artistic Value Judgements. Ind. L.J. 66:175-232.

Cohen, J.T. 2018. Dutch Supreme Court limits copyright protection for preparatory material of software. De Brauw, Blackstone, Westbroek, February 15,

Darnell, P. A., Margolis, P. E. C.1996. A Software Engineering Approach. Third Edition. New York: Springer-Verlag.

Davies, C.R. 2011. An evolutionary step in intellectual property rights - Artificial intelligence and intellectual property. Computer Law & Security Review 27:601-619.

Davis, R. 1991. Intellectual Property and Software: the assumptions are broken. WIPO Symposium 1991.

de Cock Buning, M. 2007. The history of copyright protection of computer software: the emancipation of a work of technology toward a work of authorship. In The History of Information Security (pp. 121-140).

Dacom Limited v IPM Informed Portfolio Management AB, case C-313/18, points 1.1-1.3., request for a preliminary ruling from the Svea hovrätt (Sweden) lodged on 9 May 2018.

Denicola, R.C. 2016. Ex Machina: Copyright Protection for Computer Generated Works. Rutgers UL Rev. 69:251.

Donát, J., M. Maisner, R. Polčák, J. Šavelka, M. Myška and T. Kyselovská. 2012. Software Protection - A Comparative Perspective. München: Medien und Recht.

Dorotheou, E. 2015. Reap the benefits and avoid the legal uncertainty: who owns the creations of artificial intelligence? Computer and Telecommunications Law Review 21:85-93.

Fagan, M.H. 2004. The influence of creative style and climate on software development team creativity: an exploratory study. Journal of Computer Information Systems 44:73-80.

Feist Publications Inc. v. Rural Tel. Service Co., 499, U.S. 330. 1991

Galajdová, D. 2018. Deadlock in Protection of Software Developed by AI. Jusletter IT 22. February 2018.

Gaudamuz, A. 2017. Artificial Intelligence and Copyright, WIPO Magazine, October. http://www.wipo.int/wipo_magazine/en/2017/05/article_0003.html.

Ginsburg, J.C. 1994. Four Reasons and a Paradox: The Manifest Superiority of Copyright over Sui Generis Protection of Computer Software. Columbia law review 94:2559-2572.

Ginsburg, J.C. 2002. The concept of authorship in comparative copyright law. DePaul L. Rev., 52:1063.

Ginsburg, J.C. and Budiardjo, L.A. 2019. Authors and Machines. Berkeley Technology Law Journal 34.

Goldberg, M.D. and D.O. Carson. 1991. Copyright Protection for Artificial Intelligence Systems. J. Copyright Soc'y USA, 39:57.

Goldstein, P. 1990. Copyright in the New Information Age. Cath. UL Rev. 40:829.

Goldstein. H. 2013. International copyright: principles, law, and practice. 3rd ed. New York: Oxford University Press.

Graves, A., Wayne, G., Danihelka, I. 2014. Neural Turing Machines. CoRR, abs. vol. 2014

Griffiths, J. 2011. Infopaq, BSA and the 'Europeanisation' of United Kingdom Copyright Law. Media & Arts Law Review 16:6.

Grimmelmann, J. 2015. There's No Such Thing as a Computer-Authored Work-And It's a Good Thing, Too. Colum. JL & Arts, 39:403.

Grimmelmann, J. 2016. Copyright for Literate Robots. Iowa Law Review 101: 657

Groover, M. 2014. Fundamentals of Modern Manufacturing: Materials, Processes, and Systems. Hoboken, New Jersey: John Wiley and Sons inc.

Gu M., Tong X. (2004) Towards Hypotheses on Creativity in Software Development. In: Bomarius F., Iida H. (eds) Product Focused Software Process Improvement. PROFES 2004. Lecture Notes in Computer Science 3009. Heidelberg-Berlin: Springer.

Hristov, K. 2016. Artificial Intelligence and the Copyright Dilemma. IDEA 57:431.

Humphreys, Lloyd G. 1979. The construct of general intelligence. Intelligence 3:105-20.

Judgement of 22 December 2010, *BSA v. Ministerstvo kultury ČR*, C 393/09, EU:C:2010:816.

Judgement of 1 December 2011, *Eva-Maria Painer*, C-145/10, EU:C:2011:798, paragraphs 86-94.

Judgement of 2 May 2012, *SAS Institute Inc. v. World Programming Ltd*, C-406/10, EU:C:2012:259.

Karjala, D.S. 1987. Copyright, computer software, and the new protectionism. Jurimetrics 28:33-96.

Kitzelmann, E. 2009. Inductive Programming: A Survey of Program Synthesis Techniques. In: Schmid, U; Kitzelmann, E.; Plasmeijer, R. (eds) Approaches and Applications of Inductive Programming. AAIP 2009. Lecture Notes in Computer Science 5812. Heidelberg, Berlin: Springer.

Krausová, A. Legal Regulation of Artificial Beings. Masaryk U. J.L. & Tech. 1:187.

Lambert, P. 2017. Computer-generated works and copyright: selfies, traps, robots, AI and machine learning. E.I.P.R. 39:12-20.

LeCun Y., Y. Bengio, G. Hinton. 2015 Deep Learning. Nature 521: 436-44.

Lemley, M.A. 1995. Convergence in the law of software copyright. High Tech. LJ 10:1-34.

Lemley, M.A., Menell, P. S., Merges, R. P., Samuelson, P., Carver, B. W. 2011. Software and Internet Law. New York: Wolter Kluwer Law & Business.

Lessig, L. 2006. Code: And Other Laws of Cyberspace, Version 2.0. New York: Basic Books.

Levin, S. 2018. Uber crash shows 'catastrophic failure' of self-driving technology, experts say. The Guardian, March 22.

Longdin, L., Lim, P.H. 2013. Copyright in High Technology and Haute Cuisine: Distinguishing Inspiration and Infringement. European Intellectual Property Review 35:74-81.

Liu, D. 2018. Forget the monkey copyright nonsense for goodness sake, dude! E.I.P.R. 40(1): 61-65.

Maier, G.J., 1987. Software protection-integrating patent, copyright and trade secret law. J. Pat. & Trademark Off. Soc'y 69:151.

Malovic, N. 2018. Swedish Court requests CJEU to clarify meaning of 'preparatory design material' and 'employee' within Software Directive. IPKitten. August 10. http://ipkitten.blogspot.com/2018/08/swedish-court-requests-cjeu-to-clarify.html

Manna, Z., Waldinger, R. 1975. Knowledge and Reasoning in Program Synthesis. Artificial Intelligence. 6: 175-208

Min, S., B. Lee and S. Yoon. 2017. Deep learning in bioinformatics. Briefings in bioinformatics 18:851-869.

Minsky, M. 1967. Computation: Finite and Infinite Machines. Engelwood Cliffs: Prentice-Hall

Murphy, Kevin P. 2012. Machine Learning: A Probabilistic Perspective. Cambridge, MA: The MIT Press.

Noble, D. 2017. Forces of production: A social history of industrial automation. Oxford, U.K.: Routledge.

Oracle, Am. Inc. v. Google Inc., 750 F.3d 1339 (Fed. Cir. 2014), cert. denied, 135 S. Ct. 2887 (2015).

Parasuraman, R., and V. Riley. 1997. Humans and automation: Use, misuse, disuse, abuse. Human factors, 39:230-53.

Phillips, J.C. 1991. Sui Generis Intellectual Property Protection for Computer Software. Geo. Wash. L. Rev. 60:997.

Ralph, P. 2015. The sensemaking-coevolution-implementation theory of software design. Science of Computer Programming 101:21-41.

Ramalho, A. 2017. Will Robots Rule the (Artistic) World? A Proposed Model for the Legal Status of Creations by Artificial Intelligence Systems (June 13, 2017). Available at

SSRN: https://ssrn.com/abstract=2987757 or http://dx.doi.org/10.2139/ssrn.2987757.

Rosati, E. 2017. The Monkey Selfie case and the concept of authorship: an EU perspective. Journal of Intellectual Property Law & Practice, 12:973-977.

Russell, S., Norvig, P. 2009. Artificial Intelligence: A Modern Approach (3rd Edition). Harlow: Pearson.

Samuels, E. 1988. The idea-expression dichotomy in copyright law. Tenn. L. Rev. 56:321.

Samuelson, P. 1986. Creating a New Kind of Intellectual Property: Applying the Lessons of the Chip Law to Computer Programs. Minn. L. Rev. 70:471.

Samuelson, P. 1988. Modifying Copyrighted Software: Adjusting Copyright Doctrine to Accommodate a Technology. Jurimetrics 28:179-221.

Samuelson, P. 1991. Some New Kinds of Authorship Made Possible by Computers and Some Intellectual Property Questions They Raise. U. Pitt. L. Rev. 53: 685-704.

Samuelson, P. 2012. A Fresh Look at Tests for Nonliteral Copyright Infringement. Nw. UL Rev. 107:1821.

Samuelson, P. 2016. Functionality and Expression in Computer Programs: Refining the Tests for Software Copyright Infringement. Berkeley Tech. L.J. 31:1215-1300.

Samuelson, P., T.C. Vinje and W.R. Cornish. 2012. Does Copyright Protection Under the EU Software Directive Extend to Computer Program Behaviour, Languages and Interfaces? European Intellectual Property Review. February 2012.

Šavelka, J. 2011. Exploring the Boundaries of Copyright Protection for Software: An Analysis of the CJEU-Case C-393/09 on the Copyrightability of the Graphic User Interface. Medien und Recht–International Edition 8:11-16.

Schalkoff, R.J. 1997. Artificial neural networks (Vol. 1). New York: McGraw-Hill.

Schkufza, E., Sharma, R., Aiken, A. 2018.Stochastic Program Optimization. Communications of the Ach. 59,2: 114-122

Schönberger, D. 2018. Deep Copyright: Up - And Downstream Questions Related to Artificial Intelligence (AI) and Machine Learning (ML) Droit d'auteur 4.0 / Copyright 4.0, DE WERRA Jacques (ed.), Geneva / Zurich (Schulthess Editions Romandes) 145-173.

Shaeffer, J., 2016. Software as Text. Santa Clara Computer & High Tech. L.J. 33:324.

Shalgi, M. 1986. Copyright in Software and Data. Israel Law Review 21:15-22.

Silver, D., J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton and Y. Chen. 2017. Mastering the game of Go without human knowledge. Nature, 550:354.

Singh, R., Gulwani, S. 2012. Synthesizing number transformations from input-output examples. In International Conference on Computer Aided Verification Springer, Berlin, Heidelberg. 634-651.

Stallman, R. 1994. Why software should not have owners. http://www.ru.j-npcs.org/usoft/WWW/www_gnu.org/philosophy/why-free.html.

Sterling, A. 1998. World copyright law: protection of authors' works, performances, phonograms, films, video, broadcasts, and published editions in national, international, and regional law. London: Sweet & Maxwell.

Stigler, R. 2014. Ooey GUI: The Messy Protection of Graphical User Interfaces. Nw. J. Tech. & Intell. Prop. 12:215-252.

Toeniskoetter, S.B. 2005. Protection of software intellectual property in Europe: an alternative sui generis approach. Intell. Prop. L. Bull. 10:65.

Turing, A.M. 1950. Computing Machinery and Intelligence. Mind 49:433-60.

Vezzoso, S. 2012. Copyright, Interfaces, and a Possible Atlantic Divide. J. Intell. Prop. Info. Tech. & Elec. Com. L. 3:153.

Vinje, T.C. 1992. The beginning of a new era in us Software protection law: Computer Associates v. Altai. Law, Computers & Artificial Intelligence 1:241-245.

von Lewinski, S. 2008. International copyright law and policy. Oxford: Oxford University Press.

Walter, M.M., and S. von Lewinski. 2010. European copyright law: a commentary. New York: Oxford University Press.

Wu, A.J. 1997. From Video Games to Artificial Intelligence: Assigning Copyright Ownership to Works Generated by Increasingly Sophisticated Computer Programs. AIPLA Q. J. 25:131-173

Zibner, J. 2017. Originalita v pojetí práva Evropské unie. Revue pro právo a technologie. 8: 15, p. 217-260.