Open Source Software and Some Licensing Implications to Consider

Iryna Lishchuk
Institut für Rechtsinformatik
Leibniz Universität Hannover
Hannover, Germany
e-mail: lishchuk@iri.uni-hannover.de

***Abstract* — As more and more areas of science make use of open source software, legal research seeks to reconcile various open source licenses (OSS) (which may be used in a single research project) and explores solutions to allow exploitation of software outcomes in a license-compliant way. In this paper, we consider some licensing implications of open source licenses along with solutions on how to distribute software developments in a license compatible way. The steps undertaken in course of defining a license and checking license compatibility are demonstrated by a case study.**

Keywords - open source software; free software; open source licensing; copyleft.

I. INTRODUCTION

As previously discussed in the paper "Licensing Implications of the Use of Open Source Software in Research Projects", presented at INFOCOMP 2016 [1], the use of open source software in IT-projects may produce licensing implications. Such implications may in turn interfere with the plans of the developer on the potential exploitation of newly developed software. However, as we found out and describe below, some potentially risky legal issues can be avoided a priori by applying the basic knowledge of license terms and managing the use of dependencies in a legally and technically skillful way. We describe in simple terms the basic ideas and principles of free and open source software (FOSS) and suggest some guidelines, which should help a developer to make such uses of OSS, which would go in line with the exploitation plans of the developer and the license terms.

Some key areas of computing, such as Apple/Linux/GNU, Google/Android/Linux, rely on open source software. There are numerous platforms and players in the market of OSS, which offer their tools "open source", but dictate their own rules for using their developments. Well-known examples are the Apache Software Foundation (ASF) and the Apache http server; the Mozilla Foundation, whose browser Firefox makes strong competition to Google Chrome and Microsoft Internet Explorer; the Free Software Foundation with its benchmarking GNU project. The bringing of such innovative products to the market enriches the software development community and helps solving various technical problems. On the other hand, binding the use of such products within the rules of the platforms may also cause legal challenges for the developers, who try to combine products of several platforms in one project.

Many research projects use the potential of OSS and contribute to the open source movement as well. One example is the EU FP7 CHIC project in the area of health informatics (full title "Computational Horizons In Cancer (CHIC): Developing Meta- and Hyper-Multiscale Models and Repositories for In Silico Oncology" [2]). CHIC is engaged in "*the development of clinical trial driven tools, services and infrastructures that will support the creation of multiscale cancer hypermodels (integrative models)"* [2]. In the course of this, it makes use of OSS. For example, the hypermodelling framework VPH-HF relies on an open source domain-independent workflow management system Taverna [3], while an open source finite element solver, FEBio, is used in biomechanical and diffusion modeling [4].

CHIC also explores the possibility of releasing the project outcomes "open source" as well. This is part of a wider trend in all areas of scientific research, in which OSS is becoming increasingly popular. However, while the use of OSS may benefit the conduct of the project and promote its outcomes, it may at times limit the exploitation options.

In this paper, we look into the licensing implications associated with the use of OSS and open sourcing the project outcomes. Also, we seek to suggest solutions on how licensing implications (and incompatibility risks) may best be managed. The rest of this paper is organized as follows. Section II describes the notion of FOSS and elaborates on the license requirements for software distribution. Section III addresses peculiarities of the set of GNU General Public Licenses (GPL) and points up some specific aspects stemming from the use of GPL software. In Section IV, we consider some instruments for solving license incompatibility issues. The article concludes by way of a case study in Section V, showing how the use of OSS may impact on future licensing of software outcomes.

II. FREE AND OPEN SOURCE SOFTWARE

Open source software is not simply a popular term, but it has its own definition and criteria, which we describe below.

A. Open Source Software

According to the Open Source Initiative (OSI), "*Open source doesn't just mean access to the source code. The distribution terms of open-source software must comply with the following criteria…"* [5]. These requirements normally dictate distribution of a program: either in source form (a script written in one or another programming language, such as C++, Java, Python, etc.) or as a compiled executable, i.e., object code ("*a binary code, simply a concatenation of "0"'s and "1"'s*." [6]).

The basic requirements of OSS are as follows:

1. *Free Redistribution*. The license may not restrict distributing a program as part of an aggregate software distribution and/or may not require license fees.
2. *Source Code.* The license must allow distribution of the program both in source code and in compiled form. By distribution in object code, the source code should also be accessible at a charge not exceeding the cost of copying (download from Internet at no charge).
3. *Derived Works.* The license must allow modifications and creation of derivative works and distribution of such works under the same license terms.
4. *Integrity of The Author's Source Code.* The license may require derivative works and modification to be distinguishable from original, such as by a version number or by name.
5. *No Discrimination Against Persons or Groups.*
6. *No Discrimination Against Fields of Endeavor.*
7. *Distribution of License.* The license terms apply to all subsequent users without the need to conclude individual license agreements.
8. *License Must Not Be Specific to a Product.* The license may not be dependent on any software distribution.
9. *License Must Not Restrict Other Software.* The license must not place restrictions on other programs distributed with the open source program (e.g., on the same medium).
10. *License Must Be Technology-Neutral.* The license may not be pre-defined for a specific technology [5].

There are currently more than 70 open source licenses, which can be categorized according to the license terms.

B. Free Software

One category is free software, which also has its own criteria. As defined by the Free Software Foundation (FSF), a program is free software, if the user (referred to as "you") has the four essential freedoms:

1. *"The freedom to run the program as you wish, for any purpose (freedom 0).*
2. *The freedom to study how the program works, and change it so it does your computing as you wish (freedom 1). Access to the source code is a precondition for this.*
3. *The freedom to redistribute copies so you can help your neighbor (freedom 2).*
4. *The freedom to distribute copies of your modified versions to others (freedom 3). By doing this you can give the whole community a chance to benefit from your changes. Access to the source code is a precondition for this."* [7].

The GPL, in its different versions, is a true carrier of these freedoms and GPL software (when distributed in a GPL compliant way) is normally free. The licenses, which qualify as free software licenses are defined by the FSF [8].

C. Free Software and Copyleft

The mission of free software is to provide users with these essential freedoms. This mission is achieved in a way that not only the original author, who licenses his program under a free license first, but also the subsequent developers, who make modifications to such free program, are bound to release their modified versions in the same "free" way.

Maintaining and passing on these freedoms for subsequent software distributions are usually achieved by the so called copyleft. "*Copyleft is a general method for making a program (or other work) free, and requiring all modified and extended versions of the program to be free as well."* [9]. A copyleft license usually requires that modified versions be distributed under the same terms. This distinguishes copyleft from non-copyleft licenses: copyleft licenses pass identical license terms on to derivative works, while non-copyleft licenses govern the distribution of the original code only.

D. Licensing Implications on Software Distribution

From the whole spectrum of FOSS licenses, mostly the free licenses with copyleft may produce licensing implications on software exploitation. The other free licenses without copyleft are, in contrast, rather flexible, providing for a wider variety of exploitation options, subject to rather simple terms: acknowledgement of the original developer and replication of a license notice and disclaimer of warranties.

Such more relaxed non-copyleft licenses usually allow the code to be run, modified, distributed as standalone and/or as part of another software distribution, either in source form and/or as a binary executable, under condition that the license terms for distribution of the original code are met. Among the popular non-copyleft licenses are: the Apache License [10], the MIT License [11], the BSD 3-Clause License [12], to name but a few. "*Code, created under these licenses, or derived from such code, may "go "closed" and developments can be made under that proprietary license, which are lost to the open source community.*" [13].

The conditions for distributing the original code under these non-copyleft licenses are rather simple. The basic rationale is to keep the originally licensed code under the original license (irrespective whether it is distributed as standalone or as part of software package) and to inform subsequent users that the code is used and the use of that code is governed by its license. The basic principle, which, generally, not only these, but all open source licenses follow, is that the use of the original code and its authors should be acknowledged. For instance, the MIT license requires that "*copyright notice and this permission notice shall be included in all copies or substantial portions of the Software*" [11]. The easiest way to fulfill this license requirement is to keep all copyright and license notices found in the original code intact. By this, the copyright notice, the program license with disclaimer stay replicated (maintained) throughout the whole re-distribution chain.

Failure to do so may, on the one hand, compromise the ability of the developer to enforce his own copyright in parts of the code, which he wrote himself, and, on the other hand, put him at risk of becoming an object of cease and desist action or a lawsuit [13].

E. Copyleft Licenses

At the same time though, the free licenses with copyleft, in promoting the four essential freedoms to the users, may take away the developer´s freedom to decide on licensing of his own software, by pre-determining a license choice for him. While supporters of free software speak about copyleft as protecting the rights, some developers, affected by the copyleft against their will, tend to refer "*to the risk of "viral" license terms that reach out to infect their own, separately developed software and of improper market leverage and misuse of copyright to control the works of other people*." [14].

The GPL Version 2 (GPL v2) [15] and Version 3 (GPL v3) [16] are examples of free licenses with strong copyleft. GPL copyleft looks as follows. GPL v2, in Section 1, allows the user "*to copy and distribute verbatim copies of the Program's source code… in any medium"* under the terms of GPL, requiring replication of the copyright and license notice with disclaimer and supply of the license text. In Section 2, the GPL license allows modifying the program, "*thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above*", i.e., under GPL itself. In doing so, it implies that a developer may distribute his own developments, only if he licenses under GPL. In some cases, this binding rule may place the developer in a dilemma: either to license under GPL or not to license at all.

A more positive aspect of GPL is that at times it may be rather flexible. In particular, not all modes of using a GPL program create a modified version and not all models of software distribution are necessarily affected by GPL.

III. GPL AND GPL COPYLEFT

Among the decisive factors whether software is affected by GPL copyleft are: the mode, in which software uses a GPL program, the version and wording of the applicable GPL license, and the method of how software will be distributed.

A. Mode of Use

The mode of use essentially determines whether a development qualifies as "*a work based on a GPL program*" or not. If because of using a GPL program, software qualifies as a derivative work, i.e., a *"work based on the Program",* then according to the terms of GPL it shall go under GPL [15]. Otherwise, if a program is not a modified version of GPL, then there is no binding reason for it to go under GPL.

In this regard, not all uses of a GPL program will automatically produce a derivative work. For example, developing a software using the Linux operating system, or creating a piece of software designed to run on Java or Linux (licensed under GPL v2 [17]) does not affect licensing of this software (unless it is intended to be included into the Linux distribution as a Linux kernel module). Also, calculating algorithms by means of a GPL licensed R (a free software environment for statistical computing and graphics [18]) in the course of developing a software model does not affect licensing of a model, since the model is not running against the GPL code.

Even so, a distinctive feature of GPL is that, in contrast to the majority of other open source licenses, which do not regard linking as creating a modified version (e.g., Mozilla Public License [19], Apache License [10]), the GPL license considers linking, both static and dynamic, as making a derivative work. Following the FSF interpretation criteria, "*Linking a GPL covered work statically or dynamically with other modules is making a combined work based on the GPL covered work. Thus, the terms and conditions of the GNU General Public License cover the whole combination*" [20]. This is interpretation of GPL license by the FSF and this position is arguable. When testing whether linking programs produces a GPL-derivative, the technical aspects of modification, dependency, interaction, distribution medium and location (allocation) must be taken into account [21].

The controversy Android v Linux [22] illustrates how Google avoided licensing of Android under GPL because the mode, in which it used Linux stayed beyond the scope of Linux GPL license. This case concerned the Android operating system, which relies on the GPL licensed Linux kernel and which was ultimately licensed under the Apache License. Android is an operating system, primarily used by mobile phones. It was developed by Google and consists of the Linux kernel, some non-free libraries, a Java platform and some applications. Despite the fact that Android uses the Linux kernel, licensed under GPL v2, Android itself was licensed under Apache License 2.0. "*To combine Linux with code under the Apache 2.0 license would be a copyright infringement, since GPL version 2 and Apache 2.0 are incompatible"* [22]. However, the fact that the Linux kernel remains a separate program within Android, with its source code under GPL v2, and the Android programs communicate with the kernel via system calls clarified the licensing issue. Software communicating with Linux via system calls is expressly removed from the scope of derivative works, affected by GPL copyleft. A note, added to the GPL license terms of Linux by Linus Torvalds, makes this explicit:

*"NOTE! This copyright does *not* cover user programs that use kernel services by normal system calls - this is merely considered normal use of the kernel, and does *not* fall under the heading of "derived work". Also note that the GPL below is copyrighted by the Free Software Foundation, but the instance of code that it refers to (the linux kernel) is copyrighted by me and others who actually wrote it."* [17].

Examples of normal system calls are: *fork(), exec(), wait(), open(), socket(),* etc. [22]. Such system calls operate within the kernel space and interact with the user programs in the user space [23]. Taking into consideration these technical details, "*Google has complied with the*

requirements of the GNU General Public License for Linux, but the Apache license on the rest of Android does not require source release." [22]. In fact, the source code for Android was ultimately released. However, in the view of the FSF, even the use of Linux kernel and release of the Android source code do not make Android free software. As commented by Richard Stallman [22], Android comes up with some non-free libraries, proprietary Google applications, proprietary firmware and drivers. Android deprives the users of the freedom to modify apps, install and run their own modified software and leaves the users with no choice except to accept versions approved by Google. What is most interesting, that the Android code, which has been made available, is insufficient to run the device. All in all, in opinion of Richard Stallman, these "faults" undermine the philosophy of free software [22].

B. GPL Weak Copyleft and Linking Exceptions

Another factor that determines whether a development is subject to GPL copyleft is the form of GPL license used.

Some GPL licenses have so-called weak copyleft. Examples are the GNU Library or "Lesser" General Public License, Version 2.1 (LGPL-2.1) [24] and Version 3.0 (LGPL-3.0) [25].

By the use of these licenses, a program or an application, which merely links to a LGPL program or library (without modifying it), does not necessarily have to be licensed under LGPL. As LGPL-2.1 explains, "*A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License*." [24]. LGPL allows combining external programs with a LGPL licensed library and distributing combined works under the terms at the choice of the developer. What LGPL requires is that the LGPL licensed library stay under LGPL and license of the combined work allow "*modification of the work for the customer's own use and reverse engineering for debugging such modifications*" [24].

Some practical consequences of how a switch from LGPL to GPL in one software product may affect exploitation and usability of another software product are demonstrated by the dispute that arose between MySQL and PHP [21].

PHP is a popular general-purpose scripting language that is especially suited to web development [26]. PHP was developed by the Zend company and licensed under the PHP license, which is not compatible with GPL [27]. PHP is widely used and distributed with MySQL in web applications, such as in the LAMP system (standing for: Linux, Apache, MySQL and PHP), which is used for building dynamic web sites and web applications [28]. MySQL is the world's most popular open source database, originally developed by MySQL AB, then acquired by Sun Microsystems in 2008, and finally by Oracle in 2010 [29].

In 2004, MySQL AB decided to switch the MySQL libraries from LGPL to GPL v2. That is when the controversy arose. The PHP developers responded by disabling an extension in PHP 5 to MySQL. If PHP was thus unable to operate with MySQL, the consequences for the open source community, which widely relied on PHP for building web applications with MySQL, would be serious [21]. To resolve the conflict, MySQL AB came up with a FOSS license exception (initially called the FLOSS License Exception). The FOSS license exception allowed developers of FOSS applications to include MySQL Client Libraries (also referred to as "MySQL Drivers" or "MySQL Connectors") within their FOSS applications and distribute such applications together with GPL licensed MySQL Drivers under the terms of a FOSS license, even if such other FOSS license were incompatible with the GPL [30].

A similar exception may be found in GPL license text of the programming language Java. Java is licensed under GPL v2 with ClassPath Exception [31]. ClassPath is a classic GPL linking exception based on permission of the copyright holder. The goal was to allow free software implementations of the standard class library for the programming language Java [21]. It consists of the following statement attached to the Java GPL license text: "*As a special exception, the copyright holders of this library give you permission to link this library with independent modules to produce an executable, regardless of the license terms of these independent modules, and to copy and distribute the resulting executable under terms of your choice, provided that you also meet, for each linked independent module, the terms and conditions of the license of that module. An independent module is a module which is not derived from or based on this library."* [31].

As we explore further in Section IV, a developer may be motivated to add such linking exceptions to solve GPL-incompatibility issues, which can arise if a GPL program is supposed to run against GPL incompatible programs or libraries. Such linking exception may also allow certain uses of GPL software in software developments, which are not necessarily licensed in a GPL compatible way.

C. Mode of Distribution

Thirdly, the mode of distribution, namely: whether a component is distributed packaged with a GPL dependency or without it, may matter for the application of GPL.

According to the first criterion of OSS, which says that a license must permit distribution of a program either as standalone or as part of "*an aggregate software distribution containing programs from several different sources*" [5], the GPL license allows distributing GPL software "*as a component of an aggregate software"*. As interpreted by the FSF, "*mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License"* [33]. Such an "aggregate" may be composed of a

number of separate programs, placed and distributed together on the same medium, e.g., USB. [33].

The core legal issue here is of differentiating an "aggregate" from other "modified versions" based on GPL software. "*Where's the line between two separate programs, and one program with two parts? This is a legal question, which ultimately judges will decide."* [33]. In the view of the FSF, the deciding factor is the mechanism of communication (exec, pipes, rpc, function calls within a shared address space, etc.) and the semantics of the communication (what kinds of information are exchanged). So, including the modules into one executable file or running modules *"linked together in a shared address space"* would most likely mean "*combining them into one program*". By contrast, when *"pipes, sockets and command-line arguments"* are used for communication, "*the modules normally are separate programs"* [33].

These observations bring us to the following conclusions. Distributing an independent program together with a GPL program on one medium, so that the programs do not communicate with each other, does not spread the GPL of one program to the other programs. Equally, distributing a program, which has a GPL dependency, separately and instructing the user to download that GPL dependency for himself would release a program from the requirement to go under GPL. However, distributing a program packaged with a GPL dependency would require licensing the whole software package under GPL, unless exceptions apply.

D. Commercial Distribution

In contrast to the open source licenses, which allow the code to go "closed" (as proprietary software "*lost to the open source community*" [13]), GPL is aimed to preserve software developments open for the development community. For this reason, GPL does not allow "burying" GPL code in proprietary software products. Against this principle, licensing GPL software in a proprietary way and charging royalties is not admissible.

Alternative exploitation options for GPL components, though, remain. One of these may be to charge fees for distribution of copies, running from the network server as "Software as a Service" or providing a warranty for a fee. For instance, when a GPL program is distributed from the site, fees for distributing copies can be charged. However, *"the fee to download source may not be greater than the fee to download the binary*" [34].

Offering warranty protection and additional liabilities would be another exploitation option. In this regard, GPL allows providing warranties, but requires that such provision must be evidenced in writing, i.e., by signing an agreement. A negative aspect here is that by providing warranties a developer accepts additional liability for the bugs, caused by his predecessors, and assumes "*the cost of all necessary servicing, repair and correction*" [16] for the whole program, including modules provided by other developers. Nonetheless, the business model of servicing GPL software has proven to be quite successful, as the Ubuntu [35] and other similar projects, which distribute and provide services for Linux/GNU software, demonstrate.

At the same time, the open source requirement and royalty free licensing of GPL software are not very convenient for some business models. In this regard, businesses, which are not comfortable with GPL (or, to be more exact, with licensing their software developments under GPL), may on occasion be tempted to test the boundaries of what uses of GPL software are still controlled under the GPL license [36]. This has given rise to a number of lawsuits, involving allegations of improper circumvention of GPL license requirements, one of which we consider in more detail below.

E. GPL and Copyright Relevant Actions

The case in question is Oracle America, Inc. v. Google Inc., C 10-03561 WHA [37]. The case dealt with a question in how far Google´s use of Java´s API violated Oracle´ copyright in Java.

Java is a powerful object oriented programming language, developed by Sun Microsystems, first released in 1996, and acquired by Oracle in 2010. Java is a popular programming language and makes an integral part of many contemporary software. Between 2006 and 2007 Java migrated to GPL v2 and continued under GPL v2, when it was acquired by Oracle in 2010. Java was designed to run on different operating systems and makes use of Java virtual machine for that. "*Programs written in Java are compiled into machine language, but it is a machine language for a computer that doesn't really exist. This so-called "virtual" computer is known as the Java virtual machine"* [38].

Java created a number of pre-written programs, called "methods", which invoke different functions, such as retrieving the cosine of an angle. These methods are grouped into "classes" and organised into "packages". Software developers can access and make use of those classes through the Java APIs [37]. In 2008 Java APIs had 166 "packages", split into more than six hundred "classes", all divided into six thousand "methods".

A very popular Java project is the Open JDK project [39]. Open JDK was released under GPL v2 license with the ClassPath exception. However, the package, which was involved in the dispute, was Java ME phone platform development (known as PhoneMe [40]). The package PhoneMe) did not contain the ClassPath exception. Google built its Android platform for the smartphones using the Java language. The GPL v2 license was inconvenient for Android's business model. So, apparently, Google used the syntax of the relevant Java APIs and the Java virtual machine techniques, but with its own virtual machine called the Dalvik [41] and with its own implementations of class libraries [21]. According to Oracle, Google "*utilized the same 37 sets of functionalities in the new Android system callable by the same names as used in Java*" [37].

By doing that, Google wrote its own implementations of the methods and classes, which it needed. The only one

substantial element, which Google copied from Java into Android was the names and headers of 37 API packages in question. Such copying of the headers amounted to replication of the structure, sequence and organization of Java APIs. Oracle claimed copyright infringement, and Google defended with fair use, arguing that Java is an open solution (which Oracle did not dispute) and there was no literal copying of the Java code.

In fact, 9 lines of Java code were copied verbatim into Android, but those 9 lines related to a Java function of 3179 lines called Range Check [37]. The judge assessed such copying as accidental and not substantial enough to qualify for copyright violation.

As regards the structure of Java APIs, the district court qualified the headers and method names in Java APIs as non-copyrightable, referring to the interpretation criteria of the US Copyright Office: "*Even if a name, title, or short phrase is novel or distinctive or lends itself to a play on words, it cannot be protected by copyright*." [42].

In terms of the copying of the declarations and duplicating the command structure of Java APIs, the court found that the command structure of Java APIs amounts to a method of operation – a material not subject to copyright in the US [42]. In Java programming, the specific declarations in the Java APIs designate a method. A method can be implemented in different ways, but is invoked by that specific declaration only. The command format, used to call the methods in Java, reads:

"java.package.Class.method()."

Here, a formula "*a = java.package.Class.method()"* sets the field "a", which is equal to the return of the method called. For example, the following call would call the method from Java:

"int a = java.lang.Math.max (2, 3)"

This command line would instruct the computer to fetch "*the max method under the Math class in the java.lang package, input "2" and "3" as arguments, and then return a "3," which would then be set as the value of "a."* [37].

As interpreted by the district court judge, in Java, each symbol in a command structure is more than a simple name - each symbol carries a task to invoke a pre-assigned function.

Considering that for using Java class methods software developers need to replicate the Java declarations, the judge qualified the command structure of Java APIs as a method of operation – a functional element essential for interoperability, not subject to the US Copyright Act. This position was based on the merger doctrine and non-copyrightability of structures dictated by efficiency: *"... When there is only one way to express an idea or function, then everyone is free to do so and no one can monopolize that expression.*" [37].

However, on appeal, the Federal Circuit Court reversed that ruling [43]. The appellate court found the declaring code and the structure, sequence and organization of packages in Java APIs were entitled to be protected by copyright.

The appellate court supported its decision by the argument that Java programmers were not limited in the way they could arrange the 37 Java API packages at issue and had a choice to organize these API packages in other ways. For instance, instead of using the command format "*java.package.Class.method()"*: language – package – class – method, the same method could be called by the format: method – class – package – language. By making a decision to arrange the declarations in Java in this way and by having also other choices, the programmers were not prevented by the factor of efficiency, which would preclude copyright. Rather, the programmers had a scope to exercise their creation, which they, in view of the court, exercised, indeed. This creation, realized in sequencing the Java APIs, amounted to a copyrightable expression. Against these considerations, the court concluded that, "*the structure, sequence, and organization of the 37 Java API packages at issue are entitled to copyright protection."* [43].

Google argued fair use and petitioned the US Supreme Court to hear the case. The US Supreme Court, referring to the opinion of the US Solicitor General, denied the petition. In the result, a new district court trial began. On 26 of May 2016 the district court jury found that Google´s Android did not infringe Oracle copyrights, because Google´s re-implementation of 37 Java APIs in question amounted to and was protected by fair use. According to a Google spokesperson, *"Today's verdict that Android makes fair use of Java APIs represents a win for the Android ecosystem, for the Java programming community, and for software developers who rely on open and free programming languages to build innovative consumer products."* [44].

This lawsuit, although not concerning the GPL license directly, sheds some light on very important questions of software copyright: free use of Java APIs, copyrightability of interfaces and an attempt "*to control APIs with copyright law*" and counter-balance between copyrights and "fair use" [44]. As established in this case, the APIs, although elements responsible for interoperability, can be protected by copyrights (at least in the opinion of one court of appeals); the APIs, although protected by copyright, may be reused in other software systems, if such re-use is covered by fair use of open and free programming languages, like Java.

Another conclusion, which may be drawn from this litigation, is that copying structure, sequence and organization of someone else's GPL program or APIs, and in the process making a GPL program and a newly developed program compatible with each other, may be not the best solution to avoid GPL copyleft. Such copying may, under some circumstances and unless exempted by "fair use" doctrine, infringe third party copyright and lead to litigation and associated financial costs, which might be spared if compliance with GPL had been observed.

Also, as may be observed, although the programming languages, which comprise ideas and principles, may not be subject of copyright, at least not in the EU [45], Java is an object oriented programming language, which tested this assumption under the US law and has passed the copyrightability test [21].

IV. MANAGING LICENSE INCOMPATIBILITY

In this section, we consider some examples and practices of managing license incompatibility issues.

A. Exceptions and Permissions

There are about 70 open source licenses and some of them are incompatible with each other in some respect [46]. The FSF made an attempt to analyze open source licenses on compatibility with GPL and published the list of GPL-compatible and GPL-incompatible licenses on the FSF website [8]. Also, compatibility checks and the lists of compatible and non-compatible licenses have been identified by the Apache Software Foundation [47], the Mozilla Foundation [48], etc.

The FSF developments are powerful software and are very popular with the software development community. By that, the specifics of GPL license often causes license incompatibility issues. The reason for this is the position of FSF to consider linking as creating a derivative work: "*Linking a GPL covered work statically or dynamically with other modules is making a combined work based on the GPL covered work. Thus, the terms and conditions of the GNU General Public License cover the whole combination"* [20]. In contrast, in terms of Apache License, Version 2.0, "*Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof*" [10]. Also, Mozilla Public License, Version 2.0 (MPL 2.0), which has a weak copyleft, allows "*programs using MPL-licensed code to be statically linked to and distributed as part of a larger proprietary piece of software, which would not generally be possible under the terms of stronger copyleft licenses.*" [48].

However, what approach should a developer adopt, who intends to release his program under GPL, but uses GPL-incompatible dependencies, modules or libraries linking to his code? In this situation, the FSF recommends the developers to provide a permission to do so. The appropriate examples are: systems call exception added by Linus Torvalds to the GPL license terms for Linux [17] or GNU ClassPath exception, aimed at allowing free software implementations of the standard class libraries for Java [31].

For GPL v3, the FSF advises adding the linking permission by making use of Section 7 GPL v3 "Additional permissions". Section 7 GPL v3 allows adding terms that supplement the terms of GPL license by making exceptions from one or more of its conditions [16]. For adding a linking permission to the GPL v3 license text, the FSF advises developers to insert the following text after the GPL license notice:

"*Additional permission under GNU GPL version 3 section 7. If you modify this Program, or any covered work, by linking or combining it with [name of library] (or a modified version of that library), containing parts covered by the terms of [name of library's license], the licensors of this Program grant you additional permission to convey the resulting work. {Corresponding Source for a non-source form of such a combination shall include the source code for the parts of [name of library] used as well as that of the covered work.}"* [32]. If a developer does not want everybody to distribute source for the GPL-incompatible libraries, he should remove the text in brackets or otherwise remove the brackets.

In GPL v2, a developer may add his own exception to the license terms. The FSF recommends the following notice for that:

"*In addition, as a special exception, the copyright holders of [name of your program] give you permission to combine [name of your program] with free software programs or libraries that are released under the GNU LGPL and with code included in the standard release of [name of library] under the [name of library's license] (or modified versions of such code, with unchanged license). You may copy and distribute such a system following the terms of the GNU GPL for [name of your program] and the licenses of the other code concerned{, provided that you include the source code of that other code when and as the GNU GPL requires distribution of source code}."* [32].

By this, the FSF notes that people who make modified versions of a program, licensed with a linking exception, are not obliged to grant this special exception for their modified versions. GPL v2 allows licensing a modified version without this exception. However, when such exception is added to the GPL license text, it allows the release of a modified version, which carries forward this exception [32].

However, only an original developer, who creates a program from scratch and owns copyrights in it, may add such permission. This would be the case when a developer does programming as a hobby or in his spare time. At the same time, when a developer writes a program in the employment relation, then, according to the work-for-hire doctrine, a developer is the author and owns moral rights in the program (such as a right to be named as the author), however, economic or exploitation rights in the program (such as to distribute or license) pass to the employer [45]. This principle may, however, be derogated from by the contract. On the other hand, when a developer writes a program as a freelance, then, unless the contract foresees otherwise, software copyright would pass to the developer. In case of doubt, it is advisable to check the contractual basis or consult a lawyer.

It may also be said that although such a linking exception may be added and would be valid for a program, which a programmer creates by himself, it would not apply to the parts of other GPL-covered programs. If a developer intends to use parts of other GPL licensed programs in his code, a developer cannot authorize this exception for them and needs to get the approval of the copyright holders of those programs [32].

B. License Upgrade

License upgrade may be considered and suggested as

another option for dealing with license incompatibility. It may be considered, if such upgrade is provided for by the license. This may be explained by the fact that in the process of open source movement, some licenses, issued in initial versions, underwent changes, were adapted and became more flexible and compatible with the other open source licenses.

Examples of license upgrades, which provided for a better license compatibility, include: upgrade of MPL 1.1 to MPL 2.0, Apache 1.1 to Apache 2.0, GPL v2 to GPL v3, BSD original to BSD 3-Clause, etc.

Thus, for instance, whereas the original Mozilla Public License was incompatible with GPL, MPL 2.0 provides an indirect compatibility with GNU GPL version 2.0, the GNU LGPL version 2.1, the GNU AGPL version 3, and all later versions of those licenses. Section 3.3 MPL 2.0 gives a permission to combine software, covered by these GPL licenses, with MPL software and distribute a combined work under a GPL license, but requires to leave the MPL code under MPL [8]. In any case, it is advisable to check the MPL license notices, before making a GPL-MPL-combined work. This is also important, given that developers, who release their software under MPL, may opt out of the GPL-compatibility by listing GPL licenses in Exhibit B "Incompatible With Secondary Licenses", declaring in this way that MPL code is not compatible with the GPL or AGPL. Although software originally released under earlier versions of MPL may be brought to compatibility with GPL by upgrade or dual licensing under MPL 2.0, the software, which is only available under the previous MPL versions, will remain GPL-incompatible. Also, whereas the original BSD license because of its advertising clause was recognized as GPL-incompatible, a modified BSD 3-Clause License complies with GPL [8].

Although GNU GPL accepts BSD 3-Clause License as a lax permissive license, the FSF rather supports Apache v2. Apache v2 has been recognized by the FSF as free software license and compatible with GPL v3. Therefore, Apache v2 programs may be included into GPL v3 projects. However, this compatibility works in one direction only: Apache v2→GPL v3 and does not work vice versa [50]. Thus, software under GNU GPL licenses, including: GPL, LGPL, GPL with exceptions may not be used in Apache products. In opinion of the Apache software foundation, "*the licenses are incompatible in one direction only, and it is a result of ASF's licensing philosophy and the GPL v3 authors' interpretation of copyright law"* [50].

V. CASE STUDY

In this paper, we have considered some licensing implications, which may arise by the use of open source software. We conclude by way of a case study, showing how the use of OSS may affect licensing of a project component.

In this example, let us consider licensing of a repository for computational models. The repository links, by calling the object code, to the database architecture MySQL, licensed under GPL v2 [51], and a web application Django, licensed under BSD 3-Clause License [52].

We may identify the future (downstream) licensing options for the repository in the following way. GPL v2 considers, "*linking a GPL covered work statically or dynamically with other modules making a combined work based on the GPL covered work. Thus, GNU GPL will cover the whole combination"* [20]. In terms of GPL, a repository, which links to GPL MySQL, qualifies as a work based on a GPL program.

Assuming the repository is distributed packaged with MySQL, then, in order to be compliant with GPL license, the repository must go under GPL as well. BSD 3-Clause License is a lax software license, compatible with GPL [8]. GPL permits BSD programs in GPL software. Hence, no incompatibility issues with the BSD licensed Django arise. Section 9 GPL v2, applicable to MySQL, allows a work to be licensed under GPL v2 or any later version. This means, a repository, as a work based on GPL v2 MySQL, may go under GPL v3. Hence, GPL v3 has been identified as a license for this repository. The license requirements for distribution are considered next.

A repository may be distributed in source code and/or in object code. Distribution in object code must be supported by either: (a) source code; (b) an offer to provide source code (valid for 3 years); (c) an offer to access source code free of charge; or (d) by peer-to-peer transmission – information where to obtain the source code. If the repository is provided as "Software as a service", so that the users can interact with it via a network without having a possibility to download the code, release of the source code is not required.

In distributing this repository under GPL v3, the developer must include into each source file, or (in case of distribution in an object code) attach to each copy: a copyright notice, a GPL v3 license notice with the disclaimer of warranty and include the GPL v3 license text. If the repository has interactive user interfaces, each must display a copyright and license notice, disclaimer of warranty and instructions on how to view the license.

Django and MySQL, as incorporated into software distribution, remain under BSD and GPL v2, respectively. Here the BSD and GPL v2 license terms for distribution must be observed. This means, all copyright and license notices in the Django and MySQL code files must be reserved. For Django, a copyright notice, the license notice and disclaimer shall be retained in the source files or reproduced, if Django is re-distributed in object code [12]. Distribution of MySQL should be accompanied by a copyright notice, license notices and disclaimer of warranty; recipients should receive a copy of the GPL v2 license. For MySQL, distributed in object code, the source code should be accessible, either directly, or through instructions on how to get it.

At the same time, as we described above, MySQL GPL

v2 will spread its copyleft effect upon the repository only, if the repository is distributed packaged with GPL-covered MySQL. On the other hand, if the repository is distributed separately from MySQL with clear instructions to the user to download and install MySQL on the user´s machine separately, licensing of the repository will not be affected and the repository may go under its own license. A user, who runs GPL covered MySQL when using the repository, will not be affected by GPL either, because GPL v2 does not consider running a GPL program as producing a license relevant action. According to GPL v2, "*Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does."* [15].

As this case study suggests, licensing software under copyleft licenses, such as GPL, may be a preferred option for keeping the project components open for the software development community. By contrast, the use of dependencies under copyleft licenses will not be suitable for business models, pursuing commercial purposes. If commercial distribution is intended, use of dependencies under lax permissive licenses, such as BSD 3-Clause License, Apache v2 or MIT License would suit these interests better.

VI. CONCLUSIONS

In this paper, we considered the spectrum of FOSS licenses, identified essential criteria of different categories of open source licenses, such as free software and copyleft, and tested different uses of software against license terms.

The three categories of licenses were distinguished:

a) Non-copyleft licenses, examples: Apache and BSD. The use of non-copyleft licenses, in principle, does not cause serious licensing implications, except that the license terms for the distribution of the original code must be observed. The best mode to come to terms with this is to keep all license notices in the original code files intact. The modification and distribution of such software as part of other software and under different license terms is generally allowed, as long as the original code stays under its license.

b) Licenses with weak copyleft, examples: LGPL and MPL. These licenses require that modifications should go under the same license, but programs, which merely link to the code with weak copyleft are released from this obligation. Therefore, linking an application to a program with weak copyleft does not bring an application under the same license terms and, in general, should not limit the licensing options for an application. Distribution of the original code is governed by the original license.

c) Copyleft licenses, example: GPL. GPL requires that modified versions should go under the same license terms and also spreads this requirement to the programs, which merely link to a GPL-program. When testing whether linking programs produces a modified version of GPL-software, the technical aspects of modification, dependency, interaction, distribution medium and location (allocation) must be taken into account. The distribution of programs, developed with the use of or from GPL-software should normally follow the GPL license terms and pass on the same rights and obligations to subsequent licensees. Commercial uses of GPL software are restricted.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Union Seventh Framework Programme FP7/2007-2013 under grant agreement No 600841. Particular credit is given to Luis Enriquez A. for his insightful Master Thesis "Dynamic Linked Libraries: Paradigms of the GPL license in contemporary software", which was of great help in doing the research.

REFERENCES

[1] I. Lishchuk, "Licensing Implications of the Use of Open Source Software in Research Projects," in Proc. INFOCOMP 2016, The Sixth International Conference on Advanced Communications and Computation, Valencia, Spain, 22-26 May, 2016, ISBN: 978-1-61208-478-7, pp. 18-23.

[2] CHIC, Project, <http://chic-vph.eu/project/> [retrieved: 23 November 2016].

[3] D. Tartarini et al., "The VPH Hypermodelling Framework for Cancer Multiscale Models in the Clinical Practice", In G. Stamatakos and D. Dionysiou (Eds): Proc. 2014 6th Int. Adv. Res. Workshop on In Silico Oncology and Cancer Investigation – The CHIC Project Workshop (IARWISOCI), Athens, Greece, Nov.3-4, 2014 (www.6thiarwisoci.iccs.ntua.gr), pp.61-64. (open-access version), ISBN: 978-618-80348-1-5.

[4] F. Rikhtegar, E. Kolokotroni, G. Stamatakos, and P. Büchler, "A Model of Tumor Growth Coupling a Cellular Biomodel with Biomechanical Simulations", In G. Stamatakos and D. Dionysiou (Eds): Proc. 2014 6th Int. Adv. Res. Workshop on In Silico Oncology and Cancer Investigation – The CHIC Project Workshop (IARWISOCI), Athens, Greece, Nov.3-4, 2014 (www.6thiarwisoci.iccs.ntua.gr), pp.43-46. (open-access version), ISBN: 978-618-80348-1-5.

[5] Open Source Initiative, Open Source Definition, <http://opensource.org/osd> [retrieved: 23 November 2016].

[6] Whelan Associates Inc. v. Jaslow Dental Laboratory, Inc., et al, U.S. Court of Appeals, Third Circuit, November 4, 1986, 797 F.2d 1222, 230 USPQ 481.

[7] GNU Operating System, The Free Software Definition, <http://www.gnu.org/philosophy/free-sw.en.html> [retrieved: 23 November 2016].

[8] GNU Operating System, Various Licenses and Comments about Them, <http://www.gnu.org/licenses/license-list.en.html> [retrieved: 23 November 2016].

[9] GNU Operating System, What is Copyleft?, <http://www.gnu.org/licenses/copyleft.en.html> [retrieved: 23 November 2016].

[10] OSI, Licenses by Name, Apache License, Version 2.0, <http://opensource.org/licenses/Apache-2.0> [retrieved: 6 April, 2016].

[11] OSI, Licenses by Name, The MIT License (MIT), <http://opensource.org/licenses/MIT> [retrieved: 23 November 2016].

[12] OSI, Licenses by Name, The BSD 3-Clause License, <http://opensource.org/licenses/BSD-3-Clause> [retrieved: 23 November 2016].

[13] A. St. Laurent, "Understanding Open Source and Free Software Licensing," O´Reilly, 1 Edition, 2004.

[14] R. Nimmer, "Legal Issues in Open Source and Free Software Distribution," adapted from Chapter 11 in Raymond T. Nimmer, The Law of Computer Technology, 1997, 2005 Supp.

[15] GNU General Public License, Version 2 (GPL-2.0), <http://opensource.org/licenses/GPL-2.0> [retrieved: 23 November 2016].

[16] GNU General Public License, Version 3 (GPL-3.0), <http://opensource.org/licenses/GPL-3.0> [retrieved: 23 November 2016].

[17] The Linux Kernel Archives, <https://www.kernel.org/pub/linux/kernel/COPYING> [retrieved: 23 November 2016].

[18] The R Project for Statistical Computing, R Licenses, <https://www.r-project.org/Licenses/> [retrieved: 23 November 2016].

[19] Mozilla, MPL 2.0 FAQ, <https://www.mozilla.org/en-US/MPL/2.0/FAQ/> [retrieved: 26 November 2016].

[20] GNU Operating System, Frequently Asked Questions about the GNU Licenses, <http://www.gnu.org/licenses/gpl-faq#GPLStaticVsDynamic> [retrieved: 26 November 2016].

[21] L. Enriquez, "Dynamic Linked Libraries: Paradigms of the GPL license in contemporary software," EULISP Master Thesis, 2013.

[22] R. Stallman, "Android and Users' Freedom," first published in The Guardian, <http://www.gnu.org/philosophy/android-and-users-freedom.en.html> [retrieved: 26 November 2016].

[23] H. Kroat, "Linux kernel in a nutshell," O'Reilly, United States, 2007.

[24] Open Source Initiative, Licenses by Name, The GNU Lesser General Public License, version 2.1 (LGPL-2.1), <http://opensource.org/licenses/LGPL-2.1> [retrieved: 26 November 2016].

[25] Open Source Initiative, Licenses by Name, The GNU Lesser General Public License, version 3.0 (LGPL-3.0), <http://opensource.org/licenses/LGPL-3.0> [retrieved: 26 November 2016].

[26] The PHP Group, <http://php.net/> [retrieved: 26 November 2016].

[27] OSI, Licenses by Name, The PHP License 3.0 (PHP-3.0), <https://opensource.org/licenses/PHP-3.0> [retrieved: 26 November 2016].

[28] Building a LAMP Server, <http://www.lamphowto.com/> [retrieved: 26 November 2016].

[29] Oracle, Products and Services, MySQL, Overview, <http://www.oracle.com/us/products/mysql/overview/index.html> [retrieved: 26 November 2016].

[30] MySQL, FOSS License Exception, <https://www.mysql.de/about/legal/licensing/foss-exception//> [retrieved: 26 November 2016].

[31] GNU Operating System, GNU Classpath, <http://www.gnu.org/software/classpath/license.html> [retrieved: 26 November 2016].

[32] GNU Operating System, Frequently Asked Questions about the GNU Licenses, <http://www.gnu.org/licenses/gpl-faq#GPLIncompatibleLibs> [retrieved: 26 November 2016].

[33] GNU Operating System, Frequently Asked Questions about the GNU Licenses, <http://www.gnu.org/licenses/gpl-faq#MereAggregation> [retrieved: 26 November 2016].

[34] GNU Operating System, Frequently Asked Questions about the GNU Licenses, <https://www.gnu.org/licenses/gpl-faq.html#DoesTheGPLAllowDownloadFee> [retrieved: 26 November 2016].

[35] Ubuntu, <http://www.ubuntu.com/> [retrieved: 26 November 2016].

[36] Software Freedom Conservancy, Conservancy Announces Funding for GPL Compliance Lawsuit, VMware sued in Hamburg, Germany court for failure to comply with the GPL on Linux <https://sfconservancy.org/news/2015/mar/05/vmware-lawsuit/> [retrieved: 26 November 2016].

[37] U.S. District Court for the Northern District of California, Ruling of 31 May 2012, Case C 10-03561 WHA, Oracle America, Inc., v. Google Inc.

[38] E. David, "Introduction to Programming using Java," Hobart and William Smith Colleges, 1996.

[39] Java.net, JDK Project, <https://jdk7.java.net/> [retrieved: 26 November 2016].

[40] Java ME phone platform development, <http://java.net/projects/phoneme> [retrieved: 26 November 2016].

[41] B. Cheng and B. Buzbee, "A JIT Compiler for Android's Dalvik VM," May 2010, pp.5-14, <www.android-app-developer.co.uk> [retrieved: 26 November 2016]

[42] U.S. Copyright Office, Circular 34; "Copyright Protection 'Not Available for Names, Titles or Short Phrases", rev. January 2012.

[43] U.S. Court of Appeals for the Federal Circuit, Ruling of 09 May 2014, Oracle America, Inc., v. Google Inc., Appeals from the United States District Court for the Northern District of California in No. 10-CV-3561.

[44] J. Mullin, "Google beats Oracle—Android makes "fair use" of Java APIs," Ars Technica, 26 May 2016, <http://arstechnica.com/tech-policy/2016/05/google-wins-trial-against-oracle-as-jury-finds-android-is-fair-use/> [retrieved: 26 November 2016].

[45] Directive 2009/24/EC of the European Parliament and of the Council of 23 April 2009 on the legal protection of computer programs, Official Journal of the European Union (OJEU), L 111/16 – 111/22, 5 May 2009.

[46] D.Rowland, U.Kohl, A.Charlesworth, "Information Technology Law", 4th edition, Routledge, Taylor&Francis Group, 2012, p. 412 et seq.

[47] The Apache Software Foundation, ASF Legal Previously Asked Questions, <http://www.apache.org/legal/resolved.html> [retrieved: 26 November 2016].

[48] Mozilla, MPL 2.0 FAQ, <https://www.mozilla.org/en-US/MPL/2.0/FAQ/> [retrieved: 26 November 2016].

[49] GNU Operating System, Various Licenses and Comments about Them, BSD original license

[50] The Apache Software Foundation, GPL-Compatibility, <http://www.apache.org/licenses/GPL-compatibility.html> [retrieved: 26 November 2016].

[51] MySQL, MySQL Workbench, <http://www.mysql.com/products/workbench/> [retrieved: 26 November 2016].

[52] Django, Documentation, <https://docs.djangoproject.com/en/1.9/> [retrieved: 26 November 2016].