Less may be more: copyleft, -right and the case law on APIs on both sides of the Atlantic

Walter H. van Holst

Senior IT-legal consultant at Mitopics, The Netherlands (with thanks to the whole of the FTF-legal mailinglist for contributing information and cases that were essential for this article)

DOI: 10.5033/ifosslr.v5i1.72

Abstract

Like any relatively young area of law, copyright on software is surrounded by some legal uncertainty. Even more so in the context of copyleft open source licenses, since these licenses in some respects aim for goals that are the opposite of 'regular' software copyright law. This article provides an analysis of the reciprocal effect of the GPL-family of copyleft software licenses (the GPL, LGPL and the AGPL) from a mostly copyright perspective as well as an analysis of the extent to which the SAS/WPL case affects this family of copyleft software licenses. In this article the extent to which the GPL and AGPL reciprocity clauses have a wider effect than those of the LGPL is questioned, while both the SAS/WPL jurisprudence and the Oracle vs Google case seem to affirm the LGPL's “dynamic linking” criterium. The net result is that the GPL may not be able to be more copyleft than the LGPL.

Keywords

Law; information technology; Free and Open Source Software; case law; copyleft, copyright; reciprocity effect; exhaustion; derivation; compilation

Introduction

A recurring issue surrounding copyleft licenses is the question at which point the reciprocal effect (which has been called the "viral effect" of these licenses by some) ceases to exist. There has hardly been an issue of IFOSSLR that did not touch this particular issue. Since the most important family of copyleft licenses is the GPL family of licenses and the GPL (both version 2 and 3) states that the rightsholder's authority solely derives from copyright law, the boundaries of copyright on software are paramount in order to be able to answer this question. To some extent, the boundaries of software copyright have been addressed in recent case law, both in the European Union (SAS Institute vs WPL Ltd) and in the United States (Oracle vs Google). The subject of this article is the interplay between the aforementioned copyleft provisions in the GPL-family of free software licenses and these fairly recent developments in jurisprudence. The conclusion is that the net difference between the LGPL and the GPL may be a lot less than intended by their drafters.

In this article the issue at hand, the scope of the reciprocal effect of the GPL-family of licenses, is addressed through an analysis of the applicable rules as supplied by said family and copyright law on software with a focus on reciprocity in case of inclusion (and no other adaptation) of (L)GPL software in other software. Although the prism through which this is looked at is primarily the EU Software Directive (and more precisely the Dutch transposition of it into law as well as wider Dutch copyright jurisprudence), other jurisdictions, notably the US, will be taken into account by an analysis of the case law mentioned above.

Legal framework as provided by the GPL family

Roles of the GPL family of licenses

It is important to understand the GPL-family as dual-purpose licenses. They provide both an end-user license and a distribution license. The end-user license is relatively simple, the core of it is included in art. 2 GPL v3, which among other things says "This License explicitly affirms your unlimited permission to run the unmodified Program". The distribution license is where the pitfalls lie, but again from a relatively uncomplicated basis in section 4 GPL v3:

> *"You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program."*

And also a delineation of its scope in section 5 GPL v3 (see also GPL v2, section 2, final paragraph):

> *"A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate*

> *does not cause this License to apply to the other parts of the aggregate."*

The complexity starts here, because the GPL speaks about not being "by their nature extensions of the covered work". In other words: as long as no derivation takes place. And then it becomes relevant what defines derivation, does the GPL family of licenses take precedence here or does copyright law?

Bare licenses based on copyright law

Section 0 of GPL v2 is rather explicit about its tie-in with copyright:

> *"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. "*

Somewhat more implicit, but with seemingly the same meaning is Section 0 of GPL v3; several core concepts are defined within that core concept by using copyright (or related rights such as semiconductor masks) as the explicit reference for their scope.

This is no surprise since the Free Software Foundation (FSF) has always claimed that the GPL-family of licenses is a set of so-called bare licenses,[1] meaning that they should be interpreted solely through the prism of copyright law and not contract law.[2] This distinction is mostly relevant in common law jurisdictions since in most civil law jurisdictions the exonerations of liability in the GPL-family of licenses are most likely to be treated as negative obligations of the licensee, which automatically makes the license a bilateral contract. The upside of civil law jurisdictions is that generally speaking the licensor will not be deprived from enforcement options based on copyright infringement by the mere fact that there is a contractual relationship with the licensor. Basically, copyright infringement overrides the safeguards that a liable party which is in breach of contract could otherwise rely on.

The net result of all this is that in order to find the scope of what constitutes a derivative work under the GPL-family of licenses we will have to focus on software copyright and as far as that does not provide answers, to copyright law in general. Neither the EU Software Directive nor art. 117 of the US Copyright Act of 1976 (USC) contain specific provisions about derivation of software. Also literature on this subject is relatively scarce, with the notable exception of Pamela Samuelson's impressive analysis of derivation under US copyright law.[3]

So we have to turn to 'classical' copyright on the subject of what constitutes a derivative work under copyright law. Article 2 sub 3 of the Berne Convention defines derivative works as:

> *"Translations, adaptations, arrangements of music and other alterations of a literary*

[1] Moglen, E. (2001), Enforcing the GNU GPL, http://www.gnu.org/philosophy/enforcing-gpl.en.html
For a similar analysis of how the GPL works see also Stoltz, Mitchell L. (2005), The Penguin Paradox: How the scope of derivative works affects the effectiveness of the GNU GPL, in Boston University Law Review, Vol 85, nr. 5, December 2005, pp. 1440-1477.

[2] See for example: Henley, Mark (2009) 'Jacobsen v Katzer and Kamind Associates – an English legal perspective', IFOSS L. Rev., 1(1), pp 41 – 44, and Rosen, Lawrence (2009) 'Bad facts make good law: the Jacobsen case and Open Source', IFOSS L. Rev., 1(1), pp 27 – 32.

[3] Samuelson, P. (2012), The Quest for a Sound Conception of Copyright's Derivative Work Right (August 29, 2012). Georgetown Law Journal, Forthcoming; UC Berkeley Public Law Research Paper No. 2138479. Available at SSRN: http://ssrn.com/abstract=2138479 or http://dx.doi.org/10.2139/ssrn.2138479

> *or artistic work shall be protected as original works without prejudice to the copyright in the original work."*

Copyright laws in the various signatory countries of the Berne Convention tend to be variations of that theme, examples are art. 101 USC, art. 13 of the Dutch Auteurswet (Aw), art. 23 of the German Urhebegesetzbuch (UHG) and art. 21 of the UK Copyright, Designs and Patents Act 1988. The operative term in all these legislative terms is 'adaptation' in the sense of alterations made to the work.

This is not wholly reflected in section 5 of GPL v3 which starts with:

> *"You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4..."*

and then continues with a series of conditions, among them the famous reciprocity clause:

> *"c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it."*

To clarify this further, at the very bottom of the GPL v3 the following note can be found:

> *"The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>. "*

Section 2 of the GPL v2 and another note at the end of the GPL v2 contain very similar language.

It is therefore not unreasonable to conclude that the GPL family of licenses generally assumes that the notion of derivative work extends beyond actual transformations or adaptations to include the use of API calls to libraries. This is underlined further by the way the LGPL (both v2 and v3) treat this, although not entirely consistently. In the LGPL this is treated as a "Combined work" per the definitions of section 0 LGPL v3 and to which notably the rules of section 4 LGPL v3 apply. These state the requirement of a 'suitable linking mechanism', which is a fascinating read on its own and will be discussed shortly. The reasoning seems to be based on the idea that a creation of dependencies on L(GPL) software through library calls is use of the software beyond the permitted use and distribution of Sections 2 and 4 of the GPL, which gets us to the extent such library calls are indeed covered by copyright as protected acts.

Analysis and application to libraries

Linking mechanisms

Before getting into detail on the copyright aspects of library calls, a minimal explanation of library calls and linking mechanisms is in order. A lot of software, especially application software, relies on function libraries that are typically employed by calling the Application Programming Interface (API) of those libraries. To use a real-world analogy, it is not unlike using Lego s as an underlying foundation for a top layer that is typically written by the author(s) of the software. There are typically two ways of providing the foundations. The first one is so-called static linking, which works like incorporating the building blocks permanently in the application. The other way is to give the operating system a bill of materials stating the building blocks needed (including version information) and to have them assembled at run-time. This is called dynamic linking. As a result multiple programmes can share common building blocks which save both storage and memory space. Again, this is a very minimal explanation, for a truly thorough overview of the amount of copying and remixing of software that goes on during the normal usage of a contemporary computer that is accessible to a lawyer, I refer to Determann.[4]

It can be argued however that dynamic linking is less of an adaptation than for example a collection of poems, or the incorporation of graphical materials, musical scores or photographs in a text, which usually are not considered as adaptations (which in certain jurisdictions would be treated as collective works). The actual act of setting up the necessary references to successfully make library 'calls' is done at run-time by the operating system, when loading the calling programme into memory, not when distributing the programme that is dependent on the libraries.

It should also be noted that similar mechanisms are employed in the case of contemporary multi-platform language frameworks such as Java, Dalvik and .NET which all use virtual machines as target platforms, but in practice often rely on Just-In-Time (JIT) compilers that ultimately function very similarly to that of traditional compilers with the difference being that they are invoked on the fly during startup of a programme written in a higher or intermediate level language. Even more dynamic are programmes written in so-called dynamic languages such as Python, PHP, JavaScript (ECMA script) and Ruby, but ultimately the lower level mechanisms are not dissimilar to those described above.

Transformation and derivation in case law

Neither with static nor with dynamic linking there is much, if any, transformation of the work at a technical level, although practically speaking it will be very difficult to separate the building blocks from a statically linked executable. When looking at the rare cases about derivative works these tend to concentrate on the edges of exhaustion of copyright (also known as the first sale doctrine). They also appear to be toss-ups between being qualified as derivative works (and therefore infringing) or as mere exhaustions of existing copies. An example in the US is **Lee v. A.R.T. Company**, 125 F.3d 580 (7th Cir. 1997).[5] One Deck the Walls store sold note cards and small lithographs created by Lee to A.R.T. Company, which mounted the works on ceramic tiles (covering the art with transparent epoxy resin in the process) and resold the tiles. Lee was of the opinion that this constituted an adaptation and therefore a derivative work, while A.R.T. Company claimed that this was a case of copyright exhaustion. The 7th Circuit Court of Appeals upheld the District Court's view that this was copyright exhaustion whereas in a similar case, **Mirage Editions, Inc. v. Albuquerque A.R.T. Company**, 856 F.2d 1341 (9th Cir. 1988)[6] the 9th Circuit Court of Appeals came to an opposite decision. In this case the 9th Circuit wrote “We conclude, though, that appellant has certainly recast or transformed the individual images by incorporating them into its tile-preparing process.”, thereby referring to art. 101 USC which describes recasting

[4] Determann, L. (2006), Dangerous Liaisons – Software Combinations As Derivative Works? Distribution, Installation And Execution Of Linked Programs Under Copyright Law, Commercial Licenses And The GPL, 21 Berkeley Technology Law Journal 1421 (2006)

[5] A commentary on this decision can be found at http://www.law.cornell.edu/copyright/cases/125_F3d_580.htm

[6] The decision can be found at https://bulk.resource.org/courts.gov/c/F2/856/856.F2d.1341.87-6465.html

or transforming as one of the ways a work can be derived.

Closer to the copyright in software is a series of video game console related cases: **Sega Enterprises v. Accolade**, 977 F.2d 1510 (9th Cir. 1992),[7] **Sony Computer Entertainment, Inc. v. Connectix Corp.**, 203 F.3d 596 (9th Cir. 2000)[8] and **Lewis Galoob Toys, Inc. v. Nintendo of America**, Inc. 964 F.2d 965 (9th Cir. 1992).[9] Surprisingly enough, in none of the three cases it was claimed that interfaces are copyright protected material. This despite the fact that in all three cases use or even reimplementation of APIs lied at the heart of the matter. Stolz rightfully notes in his analysis that "the cases strongly imply that any parts of a program that must necessarily be copied in order to create a compatible module are not protected by copyright, denying copyright holders one of their key tools for controlling unauthorized linking".[10] Very striking is that in Sega the Ninth Circuit Court held that under the circumstances of the case the functional need to use some of Sega's code to use the functionality of its cartridge interface was grounds for a fair use defence, despite the literal copying and distribution of code from Sega involved. In Connectix the full reimplementation of Sony's PlayStation game console in software was not even the heart of the dispute; the core arguments were about to what extent intermediate versions of Connectix's product had been a derivative work of Sony's software as embedded in the PlayStation. And here the Ninth Circuit ruled that since the end result was free of Sony's code, there was only indirect derivation. From a pure derivative works perspective this jurisprudence is mostly tangentially relevant, but does not explicitly answer the question. Stolz also describes related cases[11] in which derivation was judged to exist, but he clearly thinks these cases do no longer provide much precedent after Connectix.

For case law from this side of the Atlantic we stay in the realm of repurposing popular art, since there is no jurisprudence equivalent to the aforementioned game console cases in the United States. For example in **Rien Poortvliet**[12] the Dutch High Court ruled that cutting up a calendar with authorised reproductions of the artist Rien Poortvliet and selling the pieces after having glued them to cardboard constituted an infringing derivation. One of the reasons the High Court found these infringing was that the author's partial transfer of copyright only had calendars within its scope and never was intended to encompass other markets than calendars. With the interesting consequence that a contractual limitation was deemed relevant for third parties that had no way of knowing about that contractual limitation. Equally similar to the US jurisprudence were the recent **Pictoright/Allposters**[13] cases in the Netherlands in which for reasons very similar to Mirage vs Alberquerque it was decided that the sale of art posters transferred on canvas surface constituted sale of derivative works, not exhaustion, and therefore infringement. Another case of creative reuse of existing artwork was the German **Flachmembranlautsprecher** case[14] in which electrostatic loudspeakers had been fitted with art posters on their surface. The Upper State Court of Hamburg followed a reasoning that the artwork still performed a very similar function on the electrostatic loudspeakers, namely wall decoration, as on the original medium (the posters) and that it therefore was not being used outside the economic scope for which it had been licensed to by the poster publisher.

Applying the foregoing jurisprudence, the inclusion of libraries in other code through the

[7] Retrieved from: https://bulk.resource.org/courts.gov/c/F2/977/977.F2d.1510.92-15655.html

[8] Retrieved from: https://bulk.resource.org/courts.gov/c/F3/203/203.F3d.596.99-15852.html

[9] Retrieved from: https://bulk.resource.org/courts.gov/c/F2/964/964.F2d.965.91-16205.html

[10] Stolz, p. 1458.

[11] Worlds Of Wonder v. Veritel Learning Systems, 658 F.Supp. 351 (1986) and Micro Star v. FormGen Inc. 154 F.3d 1107 (9th Cir. 1998), which narrowed Galoob down considerably.

[12] HR 19 januari 1979, NJ 1979, 412 m.nt. LWH; AMR 1979, p. 50 m.nt. JHS; AA 1980, p. 311 m.nt. Cohen Jehoram.

[13] Rb Roermond, 22 september 2010, Pictoright v Art & Allposters, overturned by Hof Den Bosch, 3 januari 2012, HO 200.079.664, LJN: BV0773 which can be found at http://www.rechtspraak.nl/ljn.asp?ljn=BV0773

[14] OLG Hamburg - Urteil vom 10.10.2001 (5 U 86/01) - DRsp Nr. 2003/6820

mechanism of static linking as described above may be qualified as derivative works on either side of the Atlantic, even though one could argue that the linked code has not been adapted otherwise. It should also be added that different jurisdictions already have differing outcomes when it comes to relatively simple cases such as repurposed art publications, so that this already is a very grey area in copyright law.

The analogy with “recasting” as was made in the Mirage decision becomes difficult to hold onto when applied to dynamic linking. By its very nature dynamic linking only takes place at runtime, so the “recasting” only takes place at the end-user's machine, not during distribution. The distribution itself may be accompanied with the dependent application, but not necessarily so. Extending the prism of “recasting” to dynamic linking of libraries would make a lot of the dependencies of applications on operating systems a reason to assume that such applications would be a derivative of the operating system. For example a great deal of non-kernel API calls tend to employ dynamic linking mechanisms. Typical examples are graphical user interface (GUI) elements and other standard components of contemporary operating systems. A stronger argument may be the economic reasoning taken by European courts in the cases quoted above because they do focus on the market as intended by the author, but this still assumes that the API itself is subject to copyright.

This was in essence one of the questions raised in both the **SAS/WPL**[15] (in the EU) and **Oracle/Google**[16] (in the USA) cases. The dust has not settled on either case yet and in the case of Oracle/Google an appeal has been filed, so especially regarding the situation in the USA this analysis is somewhat preliminary.

In SAS/WPL one of the main questions was whether a reimplementation of a programming language in a new piece of software would be an infringement of the copyright of the original piece. This is relevant in the context of library calls because the keywords and syntax of a programming language in themselves do constitute a (high level) API, but as Vezzoso[17] rightly points out, this decision does not expressly concern APIs. The European Court of Justice (ECJ) built further on its earlier **Bezpečnostní softwarová asociace**[18] decision and ruled that this matter falls outside the scope of the Software Directive (91/250 EC), but in such a way that it does not explicitly place APIs outside the scope of general copyright:

> *“Consequently, the answer to Questions 1 to 5 is that Article 1(2) of Directive 91/250 must be interpreted as meaning that neither the functionality of a computer program nor the programming language and the format of data files used in a computer program in order to exploit certain of its functions constitute a form of expression of that program and,* ***as such****, are not protected by copyright in computer programs for the purposes of that directive.” (emphasis mine)*

The strange reminiscent of the European Patent Convention, use of 'as such' implies that under certain (however unspecified) circumstances functionality or a programming language (which are a species of API) may be protected by copyright in computer programs for the purposes of Software Directive (91/250 EC). It also does not exclude the possibility that an API may be covered by general copyright law at all, but given the technical nature of APIs they by and large

[15] SAS Institute Inc. vs World Programming Ltd, ECJ May 2nd, 2012, C-406/10, retrieved from http://curia.europa.eu/juris/document/document.jsf?text=&docid=122362&pageIndex=0&doclang=EN&mode=req&dir=&occ=first&part=1&cid=972439

[16] US District Court for the Northern District of California, No. C 10-03561 WHA

[17] Vezzoso, S. (2012), Copyright, Interfaces, and a Possible Atlantic Divide, in: JIPITEC no 3, p. 153, para. 1.

[18] Bezpečnostní softwarová asociace, ECJ December 22nd, 2010, C-393/09, retrieved from http://curia.europa.eu/juris/document/document.jsf?text=&docid=83458&pageIndex=0&doclang=EN&mode=lst&dir=&occ=first&part=1&cid=713053

are unlikely to fall within the scope of general copyright law. This is crucial for the question where the reciprocal nature of the GPL ends since the GPL, as earlier mentioned, relies solely on copyright law.

For the purpose of the question where the reciprocal nature of the GPL ends when it comes to (dynamic) linking of libraries, the ECJ's implicit caveat about general copyright law is of lesser importance than for interoperability matters. Because even if an API falls inside the scope of copyright, it generally is no longer constrained by the intended use of articles 4 and 5 of the Software Directive (91/250 EC), unless the unspecified circumstances of the 'as is' are in play. This also means that the exceptions of classical copyright can be invoked and may overrule the GPL as far as the API of a (L)GPL-covered piece of software is concerned. It even opens the door for potential corner-case scenarios in which minor cases of static linking (so the inclusion of parts of a GPL-covered library into a calling program) may possibly fall outside the scope of the reciprocity clauses of the GPL-family of licenses. It also puts the AGPL's reciprocity clause which extends distribution to the provisioning of online services into a new light as far as its applicability to APIs for web-services is concerned.

Although admittedly a lower court, so not necessarily setting a precedent for the whole of the US yet, the US District Court of Northern California went a significant step further than the ECJ in Oracle vs Google when confronted with the question whether an API is covered by copyright. The court answered it with a rather resounding no:

> *“This order holds that, under the Copyright Act, no matter how creative or imaginative a Java method specification may be, the entire world is entitled to use the same method specification (inputs, outputs, parameters) so long as the line-by-line implementations are different.”*

The conclusion of all of this is that if the Java API falls outside the scope of copyright protection and if we can extend this reasoning to any library API, the particular use of a library API without the full inclusion of the library cannot possibly constitute the type of adaptation that is covered by art. 117 USC or equivalent laws in European jurisdictions. In the European context an API may possibly fall within the scope of copyright protection, although likely a very limited protection due to the highly technical constraints within which APIs typically are designed. Arguments based on the intended use of the GPL-covered library cannot hold up either because a) in the case of dynamically linked libraries that use is by the end-user, not the publisher of the library-dependent programme, and b) they are self-contradicting with both sections 2 and 5 GPL v3.

It should be noted that this analysis does not deviate from Stolz's earlier analysis of the GPL v2 based on earlier case law that was more implicit on the question of derivation in software.

Conclusion

In order to establish at which point the reciprocal nature of the GPL in case of inclusion (and no other adaptation) of (L)GPL software in other software should take place, I have assessed both the GPL and the LGPL in their role as distribution licenses. In order to establish the precedence of the GPL-family of licenses over copyright, I have established that the latter takes precedent since they are designed as bare licenses. This means that they cannot redefine what constitutes a derivative work and can only cover that what is governed by copyright law as far as the question when the GPL should be applied to computer programmes that are dependent on (L)GPL libraries is concerned. As a consequence, the question to what extent inclusion of a covered library constitutes the creation of a derivative work beyond “mere aggregation” becomes relevant. When analysing the typical mechanisms for inclusion, both static and dynamic linking, it must be concluded that the closest analogies to dynamic linking in jurisprudence are in a grey zone. Furthermore, these analogies are of limited use since the mechanisms of dynamic linking are common practice in most contemporary computer systems and are generally understood not to constitute derivation. When taking the most recent jurisprudence on software APIs into account, one can argue that the LGPL is not really the Lesser GPL, but that the GPL is based on a by now outdated understanding of software copyright and effectively becomes equal to the LGPL. In light of the fact the open source communities tend to think of the currently most popular sets of licenses as a continuum from permissive (Apache 2.0) to very far copyleft (AGPL 3.0), the conclusion that this continuum does not stretch much further in the copyleft spectrum than LGPL is not a happy one. It means that there is a serious disconnect between the expectations developers may have from their chosen license in the GPL family and the legal reality. The intent of these developers is not necessarily reflected in the effects of their chosen licenses, which is rather unfortunate.

About the author

Walter van Holst *is a senior legal consultant with Mitopics (http://www.mitopics.nl)*