UIC John Marshall Journal of Information Technology & Privacy Law

Volume 25
Issue 3 *Journal of Computer & Information Law - Summer 2008*

Article 2

Summer 2008

Deadly Combinations: A Framework for Analyzing the GPL's Viral Effect, 25 J. Marshall J. Computer & Info. L. 487 (2008)

Ron Phillips

DEADLY COMBINATIONS: A FRAMEWORK FOR ANALYZING THE GPL'S VIRAL EFFECT

RON PHILLIPS†

I. INTRODUCTION

Free and Open Source Software ("FOSS") is a popular movement that at first blush seems to be a win-win for everyone. Community source[1] projects, such as these, offer an attractive alternative to buying licenses for proprietary software. The software is free to use and can be downloaded from the Internet. Second – and in stark contrast to most proprietary software – the source code for the software is freely available. This availability allows licensees to modify and adapt the software for a particular need. For the unwary software development organization, however, the infectious nature of community source licenses can make "free" software a costly option.

This paper will argue that adaptations combining community source licensed software with an organization's own intellectual property can trigger viral terms of the community source licenses in unexpected ways. Those terms require public disclosure of the software's source code, which in turn can erode or completely destroy the commercial value of intellectual property. This paper will propose a model framework for analyzing software combinations to determine whether the viral terms are triggered, and illustrate that analysis against various technical combinations of community-sourced and proprietary software.

This paper begins with a brief primer on community source and its licensing models. It then introduces the GNU General Public License as a prototypical licensing model. Next, there is a discussion about the con-

† Vice President and General Counsel, Serlio Software Development Corporation; J.D. Marquette University Law School; B.S. University of Washington. Special thanks to my colleagues at Serlio for their insights, and to Lynn, Tony, Danielle and Amanda for their patience and support.

1. The advocates of FOSS projects draw strong distinctions between the meanings of "free software" and "open source" software. This paper uses the term "community source" to refer to FOSS projects as a term that is neutral with respect to the motivations of the individuals and organizations behind these projects.

cept of "copyleft" and how it applies when software developers combine community source software with their own new works. Additionally, two "bright line" examples of combinations of software illustrate the legal boundaries of software combinations and the legal uncertainty of the combinations within that spectrum. Finally, this paper introduces a framework for technical and legal analysis of software combinations that is designed to mitigate some of the uncertainty stemming from ambiguity in the GNU General Public License. This discussion reviews different technical approaches of combining community source and other software and suggests likely legal conclusions.[2]

II. BACKGROUND

A. Technical Essentials

A modest amount of technical background on the internal workings of software is necessary to understand some of the inherent subtleties of software and copyright law. Without understanding, it is not always clear whether and how a particular piece of software may infringe on copyrights in other software and who the ultimate infringer could be.

The process of building software involves writing instructions in a structured form that can be readily understood by a software developer and then transforming those instructions into a sequence of numbers. Computer processors are only capable of understanding data and instructions as series of numbers. However, humans are much more efficient when dealing with higher level concepts. Programming languages provide a textual grammar that the software developer can use to express instructions for the computer ("source code"), which are translated or compiled into the numeric values, the computer can understand, called object code. This is accomplished using a special compiler program. Before the computer can execute those instructions, the object code must be linked with a separate linker program. The linker program combines the object code into a file, packaging all of the instructions along with additional system code in a format that the computer can load, parse and run. See Appendix A, Figure 1.

Computer programs – with the exception of the most trivial ones are commonly comprised of thousands of lines of source code stored in many separate files. The process of building an executable program involves compiling the source code of many files, into corresponding object code files. The linker then puts all the parts together. See Appendix A, Figure 2.

2. Please note: it is explicitly not the author's intention to judge community source projects as inherently bad. The author has extensive experience and deep appreciation for community source projects. It is the ignorance of the ramifications of software licenses that is inherently bad.

Parts of software programs that are useful enough to be reused, are often compiled into a library. For example, rather than reinventing algorithms to draw charts, software developers might get an existing library that already contains those routines and use it in their programs. A static library is a special kind of compiled file that contains the reusable routines stored as object code. The static library file can be read by the linker program, which then copies the parts of the library that are needed along with the developer's object files. It then packages all of those parts into the executable file. See Appendix A, Figure 3.

A dynamically linked or "shared" library serves the same purpose as a static library, but the contents of the library are not copied into the program's executable file by the linker. Instead, the linker generates instructions in the executable file, directing it to load the library into memory when the program runs. See Appendix A, Figure 4. Note that the executable file does not include the object code for the routines from the dynamically linked library. In order for the program to run, the dynamically linked library must be installed on that computer.

B. Community Source – A Brief Primer

Free Software and Open Source software, which is referred to collectively as "community source" in this article, is software that can be characterized as (a) having been developed by a loosely-grouped team of volunteer software developers, (b) that is distributed without charge, and (c) makes the source code freely available.[3] The economics of a volunteer workforce are plain. As a product of volunteer contributions,[4] the software can be distributed freely without the need to recoup research and development costs. Further, community source advocates claim that an open development process makes the source code available for anyone to study, review, and critique results in software that is substantially higher quality than that of closed, proprietary software.[5]

The availability of the source code, however, has more than academic interest. Most commercial software is sold as a "black box" and the source code carefully guarded as a trade secret. In contrast, the source code for community source software is freely available. In effect,

3. *See generally* Marcus Mahler, *Open Source Software: The Success of an Alternative Intellectual Property Incentive Paradigm*, 10 Fordham Intell. Prop. Media & Ent. L.J. 619, 638-40 (2000).

4. *See* Karim Lakhani & Robert G. Wolf, Perspectives on Free and Open Source Software 3, 9 (Joseph Feller ed. 2005). The reader should not infer, however, that the contributors are never compensated. Several commercial software development companies pay employees to develop software contributed to community source projects, including Microsoft, IBM, and Sun. Many other companies unwittingly pay employees for contributions to community source projects developed during working hours.

5. Open Source Initiative, http://opensource.org (last visited Oct. 23, 2007).

the blueprints for building and changing the software are free for anyone. Having source code available to modify for a particular business or market need can substantially decrease production costs and accelerate the time to market. Industry leaders, including Hewlett Packard, IBM, and Nokia have implemented modified versions of community source in a broad range of products including routers, "smartphones," and PDA's.[6]

Depending on the license for community source software, commercial use may come with strings attached. There are countless licenses used by community source projects, but three license models predominate: BSD, GNU General Public License, and Open Source Initiative.[7] The BSD license is an "attribution only" license that places few limitations on derivative works.[8] The license anticipates and encourages commercial exploitation of the software[9] and requires little more than acknowledgement in the copyright for the derived work.[10] The GNU and Open Source Initiative licenses are drafted with the expectation of extending their terms to other software that is combined with software already under that license.[11] The GNU General Public License will be the main focus of this article, as it is more frequently used than others licenses.[12]

The GNU General Public License ("GPL") currently covers about sixty to seventy percent of community source software. This includes the Linux operating system and a vast body of software that runs on Linux.[13] Drafted by the Free Software Foundation ("FSF") and now in its third revision, the GPL is perhaps the most exacting community source license. Like the BSD model, the GPL favors free and open distribution of software with its source code.[14] The founders of the Free Software Foundation, however, assert that software source code should be freely available as a matter of right.[15] The GPL permits copying,

6. Richard Koman, *First GPL Lawsuit Settling Out of Court*, SCI-TECH TODAY, Sept. 24, 2007, http://www.sci-tech-today.com/story.xhtml?story_id=11200DQNRSSG (last visited Aug. 1, 2008).

7. Mahler, *supra* note 4.

8. Greg R. Vetter, *Infectious Open Source Software: Spreading Incentives or Promoting Resistance*, 36 RUTGERS L.J. 53, 74 (2004).

9. Bruce Montague, *Why You Should Use a BSD Style License for Your Open Source Project,* 6 (2006), ftp://ftp.freebsd.org/pub/FreeBSD/doc/en_US.ISO8859-1/articles/bsdl-gpl/article.pdf.zip (last visited Aug. 1, 2008).

10. FreeBSD Copyright, http://www.freebsd.org/copyright/freebsd-license.html (last visited Sept. 23, 2007).

11. Vetter, *supra* note 9, at 65.

12. Liz Laffan, Vision Mobile, GPLv2 vs GPLv3, http://linuxdevices.com/files/misc/GPLv2_vs_GPLv3.pdf (last visited Sept. 23, 2007).

13. *Id.*

14. GNU General Public License, http://www.gnu.org/licenses/old-licenses/gpl-1.0.txt (last visited Aug. 1, 2008).

15. *Id.*

changing, and redistributing the licensed software. However, the GPL in turn imposes significant restrictions and requirements for distribution, such as imposing limits on what an author can charge for the software, requiring publication of the source code for the modified software, and prohibiting use of GPL software for certain applications.[16] The FSF aggressively investigates reported violations of the GPL and funds its Free Software Licensing and Compliance Lab to enforce the terms of the GPL.[17]

The GPL drafters assert it is a license, not a contract, and base its enforceability on copyright law.[18] The license is essentially a promise not to sue for conduct that would, absent the license, constitute a violation of the rights of the copyright owner.[19] This is most particularly, the right to reproduce and the right to make derivative works.[20] The GPL grants a license to distribute and create derivative works if, and only if, the licensee complies with the specific terms in the license.[21] Without such compliance, there is no license for use, and the copyright owner can sue for any use that infringes on that copyright.[22] Compliance with the licensing terms for derivative works seems to be one that generates particular controversy.

The GPL includes a "copyleft" term that requires derivative works to be licensed under the same terms as the original GPL work.[23] The copyleft clause is the quid pro quo for the privilege of being granted license to the software. It permits the licensee to make new works based on the GPL software and to distribute those new works (even copies of the original software itself), but compels the licensee to either distribute those combined works under the same terms of the original software's license agreement or not distribute at all.[24] When the copyleft terms are triggered, the new combination has to be distributed with the source code freely available.[25]

16. *Id,; see also* Laffan, *supra* note 13.

17. FSF Free Software Licensing and Compliance Lab, http://www.fsf.org/licensing (last visited Aug. 1, 2008).

18. GNU General Public License Version 2, http://www.gnu.org/licenses/old-licenses/gpl-2.0.txt (last visited Aug. 1, 2008).

19. *Id.*

20. Vetter, *supra* note 9, at 69.

21. GNU General Public License, *supra* note 19.

22. *Id.*

23. *Id.*

24. *Id.*

25. *Id.*

III. DERIVATIVE WORKS - PROCEED AT YOUR OWN RISK

The text of the GPL is not very helpful in sorting out what combinations will trigger copyleft and require the entire combination to be licensed under the terms of the GPL.[26] An organization that wants to capitalize on intellectual property that adds value to GPL-licensed software faces substantial risks.[27] Unless that organization accepts the risk of a lawsuit or being required to open their source code to the public, they need to understand what they are doing both technically and legally.[28] Software developers who want to create and sell proprietary software for use with GPL-licensed software may find their software infected by the copyleft terms of the GPL. The terms and effect of copyleft are illustrated by Version 2 of the GPL, under which most community source software is licensed.[29] GPL Version 2 grants a conditional license to create derivative works:

> You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work . . . provided that you also . . . cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.[30]

The plain language of this clause permits the licensee to copy the work in its entirety, change it, or create new software based on the GPL licensed work. However, if that work is distributed to a third party, the licensee must license that work as a whole under the same GPL terms. This includes terms that require releasing source code for changes and prevent charging for the software. Reading the clause literally, it seems to infect *any* proprietary software combined with copylefted software, such as putting GPL software on the same disk with proprietary software. In contrast, the GPL provides that, "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."[31] Here, the license specifically stipulates that mere aggregation will not impose the terms of the GPL on unrelated proprietary software that is, for exam-

26. *See generally* Lori E. Lesser, *Open Source Software 2006: Critical Issues in Today's Corporate Environment*, 885 PRACTICING L. INST. 9 (2006).

27. *Id.*

28. *Id.*

29. Few Takers for Latest Version of GPL, VNUNET.com, http://www.vnunet.com/itweek/analysis/2200759/few-takers-latest-version-gpl (last visited Aug. 1, 2008).

30. GNU General Public License, *supra* note 19.

31. *Id.*

ple, distributed on the same DVD as GPL-licensed software.[32] On the other hand, it is axiomatic that changing the GPL program's source code creates a derivative work.[33] Distributing that derivative work makes it subject to the terms of the GPL.[34] These two scenarios are the only bright line rules for copyleft in the GPL.[35] Between the end-points of mere aggregation and direct source modification lays a broad spectrum of possible combinations that the terms of the GPL may or may not reach.[36]

There is little certainty for software developers whose combinations of GPL and proprietary software fall within that spectrum because it is not easy to determine when combinations impose the copyleft terms of the GPL on the proprietary software.[37] The terms of the GPL provide a safe harbor clause for wholly independent works that are not distributed with the GPL licensed software:

> If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.[38]

The safe harbor clause turns on whether (a) the other software is an independent and separate work, and (b) that independent and separate work is not distributed as part of a whole based on the GPL-licensed software. The terms "independent and separate work" and "whole" are not defined in the GPL, but it seems that a "whole" means a combination where the parts are somehow more closely related than a "mere aggregation." The FSF comments:

> What constitutes combining two parts into one program? This is a legal question, which ultimately judges will decide. We believe that a proper criterion depends both on the mechanism of communication (exec, pipes, rpc [sic], function calls within a shared address space, etc.) and the semantics of the communication (what kinds of information are

32. *Id.* This provision is more important than it would seem at first blush because merely installing software onto a hard drive creates this sort of aggregate combination.

33. *See, e.g.*, Computer Assoc. Int'l v. Quest Software, Inc., 333 F. Supp. 2d 688, 699 (N.D. Ill. 2004).

34. GNU General Public License, *supra* note 19.

35. *Id.*

36. Vetter, *supra* note 9.

37. Lesser, *supra* note 27, at 24.

38. GNU General Public License, *supra* note 19.

interchanged).[39]

Whether the FSF could convince a court to enforce copyleft on these kinds of combinations remains to be seen.[40] The FSF's license enforcement group has charged many organizations with violating the GPL, but every case in the United States has been quietly settled outside of court.[41] There is literally no legal precedent in the United States concerning enforcement of the GPL at the time of this writing.[42] Without legal precedent establishing which specific technical software combinations impose copyleft, practitioners must predict their legal standing by determining whether the proprietary software within a combination, infringes on the distribution rights of the GPL software licensor. They also must consider whether the proprietary software constitutes a derivative work.[43]

IV. DISTRIBUTION AND DERIVATION – VIRAL CATALYSTS

The reach of the copyleft terms of the GPL is fundamentally limited to the extent to which a combination of GPL-licensed software and another work infringes on the GPL licensor's rights to distribution and to derivative works.[44] Copyright law specifically identifies distribution and derivative works in the bundle of rights afforded to the owner of a copyright.[45] Without that backing authority of 17 U.S.C. Section 106, the FSF would have no enforceable rights under which it could limit the ability of third parties to combine GPL-licensed software with other software.[46] If the combination neither infringes on the distribution right nor is a derivative work, there is no legal basis by which the GPL terms could be imposed on the non-GPL software.[47] The two scenarios de-

39. GNU Operating System, *Frequently Asked Questions About the GNU*, http://www.gnu.org/licenses/gpl-faq.html (last visited Aug. 1, 2008).

40. Brian W. Carver, *Share and Share Alike: Understanding and Enforcing Open Source and Free Software Licenses*, 20 BERKELEY TECH. L.J. 443, 468 (2005).

41. *See First U.S. GPL Lawsuit Filed*, LINUX WATCH, (Sept. 20, 2007, http://www.linuxwatch.com/news/NS3973290690.html (last visited Aug. 1, 2008); *see also* Monsoon Multimedia, Press Release, Monsoon Multimedia to Comply with GNU General Public License (Sept. 21, 2007), *available at* http://www.myhava.com/press_releases_monsoon_open_source.html.

42. *Id.*

43. Lesser, *supra* note 27, at 25.

44. Lothar Determann, *Dangerous Liaisons -Software Combinations as Derivative Works?* 21 BERKELEY TECH. L.J. 1421, 1465 (2006).

45. 17 U.S.C. § 106 (2006).

46. 42 U.S.C. § 106; *see* Mathias Strasser, *A New Paradigm in Intellectual Property Law? The Case Against Open Sources*, 2001 STAN. TECH. L. REV. 4, 32 (2001). Since the FSF asserts that the GPL is not a contract, I do not consider any contractual rights that might exist here.

47. *See* GNU General Public License, *supra* note 19 (stating license of right under copyright law are the only means by which GPL software can be used).

scribed above, as endpoints of a range of possible combinations, illustrate this point. Where the combination is a mere aggregation, the unrelated proprietary software does not infringe on the GPL licensor's right to derivative works: the added software makes no actual or conceptual use of the GPL-licensed work.[48] Nonetheless, distributing the GPL software without a license to do so would be actionable.[49] Where the combination is created by modifying and recompiling the GPL source, the program is a derivative work because it is one in which the GPL program has been "recast, transformed or adapted"[50] and could infringe on the licensor's rights.

The copyleft terms of the GPL only come into effect when and if the work derived from the GPL-licensed software is distributed. This stands in contrast to United States' copyright law, which does not limit a copyright holder's rights to distributed derivative works.[51] A derivative work made without permission infringes, whether published or not.[52] The drafters of the GPL, however, intended that distribution must be a condition precedent to enforcing copyleft:[53]

The GPL does not require you to release your modified version. You are free to make modifications and use them privately, without ever releasing them. This applies to organizations (including companies), too; an organization can make a modified version and use it internally without ever releasing it outside the organization. But if you release the modified version to the public in some way, the GPL requires you to make the modified source code available to the program's users, under the GPL.[54]

The terms of the GPL grant a virtually unlimited license for non-distributed derivative works, and the copyleft terms under the GPL do not spring into force without distribution.[55]

V. ANALYZING COMBINATIONS OF GPL, PROPRIETARY SOFTWARE

A framework for analyzing various software combinations may help to determine the reach of copyleft. The text of the GPL circumscribes the outer bounds of what is allowed and what is prohibited, but provides little guidance for combinations within the spectrum of possible combinations.[56] Close analysis of potential combinations to expose the means by

48. *Id.*
49. *Id.*
50. 17 U.S.C. § 101 (2006).
51. 17 U.S.C. § 106(2) (2006).
52. 17 U.S.C. § 104(a) (2006).
53. GNU General Public License, *supra* note 19.
54. Free Software Foundation, *supra* note 41.
55. GNU General Public License, *supra* note 19.
56. Vetter, *supra* note 9, at 92.

which the combination is formed provides a factual basis for application of copyright principles. A candidate framework for such analysis is introduced here, and is demonstrated with analysis of commonly-occurring technical combinations of software. The framework helps to identify scenarios that are likely to impose copyleft by categorizing patterns of software combinations.

A. Framework for Analysis

Determining if a particular combination of software will trigger copyleft terms requires a technical analysis of how those works are combined. It also involves a legal analysis of that combination against the terms of the GPL and copyright law. The specific nature of the combination is tested against the GPL to determine if that combination is explicitly permitted or prohibited. If not, then the combination is analyzed to determine whether it constitutes a derivative work under copyright law (in turn providing substance for enforcing the terms of the GPL).[57] See Appendix A, Figure 5.

The technical analysis is required to determine precisely how the GPL-licensed software and the proprietary software are related and interact. For example, copyleft terms are not implicated where the relationship between the GPL and proprietary software is simply having been distributed together on the same disk.[58] Where there is a more intimate nexus between the two programs, however, more investigation is required to determine if the combination creates a copy of all or part of the GPL program permanently onto a disk, creates an in-memory copy, or whether the two programs interoperate in some manner.[59] This investigation establishes a factual basis upon which to develop a legal analysis.[60]

The purpose of the legal analysis is to determine whether the proprietary software constitutes a derivative work by incorporating protected expression in some definite, concrete form.[61] That incorporation may be through literal or non-literal copying.[62] Based on an understanding of the precise nature of the relationship between the GPL-licensed and proprietary software, the analysis uncovers whether copying has occurred, and whether that copying involved protected expression.

The framework is comprised of three primary steps. The first step, safe harbor analysis, determines if the combination falls completely

57. Determann, *supra* note 45, at 1465.
58. GNU General Public License, *supra* note 19.
59. Free Software Foundation, *supra* note 41.
60. *See, e.g.*, Vetter, *supra* note 9, at 112.
61. Micro Star v. FormGen, Inc., 154 F.3d 1107, 1111 (9th Cir. 1998).
62. Determann, *supra* note 45, at 1433.

outside the reach of the GPL or is an explicitly permitted combination. The next step, suspect combinations, identifies likely-infringing scenarios where the developer of the combination has created a derivative work. The final step, zone of uncertainty, provides guidance for technical and legal analysis of combinations where the question of whether copying has occurred may not be readily evident.

B. Application of Analysis – Safe Harbor Analysis

The safe harbor analysis determines whether a software combination falls outside the reach of the GPL or is an explicitly permitted use under the GPL that does not impose copyleft. First, if proprietary software is in no way related to GPL-licensed software, it is axiomatic that the GPL cannot be imposed upon it. Second, even if the software is somehow combined with GPL software but is not distributed, the GPL copyleft terms are not imposed.[63]

The GPL provides a safe harbor clause permitting distribution of a mere aggregation of proprietary and GPL licensed software without triggering copyleft terms.[64] The GPL attempts to explain mere aggregation as one where the proprietary software is "reasonably considered [an] independent and separate [work]."[65] This language suggests that where the proprietary software and the GPL-licensed software are merely packaged together on the same distribution medium, the copyleft terms are not implicated.[66] If, however, the software is bundled somehow as a whole, the terms of the GPL infect the proprietary software.[67]

Consider this example: Client wants to distribute his proprietary tax software with OpenAccolade, an Enterprise Resource Planning ("ERP") package licensed under the GPL. He plans to distribute them both on a single DVD. The two programs are otherwise unrelated and do not interoperate. Here, the client's packaging likely falls under the safe harbor permitting distribution of combinations that are mere aggregations, since there is no closer nexus.[68]

C. Application of Analysis – Suspect Software Combinations

The next step, the suspect combinations analysis, looks for specific practices in software development that are likely to result in derivative works because the practice involves copying of protected expression. If

63. *See* GNU General Public License, *supra* note 19.
64. *Id.*
65. *Id.*
66. David McGowan, *Legal Implications of Open-Source Software*, 2001 U. Ill. L. Rev. 241, 255 (2001).
67. *Id.*
68. *Id.*

the developer has somehow brought protected expression from the GPL software into a new work, the result is a derivative work.[69] Though copying may not be obvious or intentional, the resulting work will be infected and subject to the copyleft terms[70]

The most obvious case occurs when the developer copies GPL source code into his own source. See Appendix A, Figure 6. Here, the developer engages in literal copying of protected expression. Source code is protected under copyright as a literary work[71] and literal copying infringes on the copyright holder's right to derivative works.[72]

Linking object code compiled from a legitimately obtained GPL-licensed source file into the proprietary work is similarly infringing. Object code is given the same copyright protection as source code and is usually considered an analog to source code.[73] The act of compiling the GPL-licensed source creates a non-literal copy by transforming it into object code. See Appendix A, Figure 7. The act of copying alone is not enough to trigger copyleft, because no derivative work has been distributed.[74] Linking the object code into an executable file, however, creates an additional copy of the object code into the executable file. If that executable file is distributed, the copyleft terms spring into effect against the whole program.[75] Where the program is built by linking routines from a GPL-licensed library, the result is the same. Linking the program makes an in-memory copy of copyrighted object code from the library and copies it in the executable file. The resulting executable file incorporates the copyrighted code and the copyleft terms are imposed on the resulting program.[76]

Non-literal copying can occur when a developer translates GPL software from one programming language to another.[77] The GPL explic-

69. 17 U.S.C. § 101 (2006).

70. Lesser, *supra* note 27, at 24.

71. Computer Associates v. Altai, 982 F.2d 693, 702 (2d Cir. 1992).

72. *See, e.g.*, Computer Associates v. Quest Software, 333 F. Supp. 2d 688, 699 (N.D. Ill. 2004).

73. Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240, 1249 (3d Cir. 1983).

74. GNU General Public License, *supra* note 19 (requiring distribution of the original source code, the clause in the GPL would be triggered if the developer merely distributed the object code file generated from the GPL).

75. *Id.*

76. *Id.*; *but see* GNU Lesser General Public License (2007), *available at* http://www.gnu.org/licenses/lgpl.html (last visited Dec. 1, 2007). Libraries are sometimes licensed under the Lesser General Public License, which permits linking and distribution of literal copies of libraries without imposing copyleft. Use of the LGPL is discouraged by the FSF. *Id.*

77. 17 U.S.C. § 101 (2006) (defining derivative works to include translations of the original work).

itly describes translation as a modification that triggers copyleft.[78] Even though translation of software may not result in any literal copying of the original source code or object code, the original software is protected beyond those literal elements.[79] See Appendix A, Figure 8. The structure and form of the software can likewise be protected by copyright and a translation can be infringing.[80]

Here is an example. Client publishes "Money In Your Pocket", a personal finance application that runs on smart phones. He has learned that one of his programmers has taken a small sub-program from OpenAccolade that calculates compound interest, made minor modifications, and used it in the MIYP software. Here, if the client has distributed the MIYP software, he has infringed on the OpenAccolade copyright by copying part of the protected work and adapting it and may have triggered the copyleft clause.[81]

D. Application of Analysis – The Zone of Uncertainty

The final step of the analysis is the zone of uncertainty, which includes software combinations where the GPL code is not used within the proprietary code or vice versa. However, the two are nonetheless related in origin or by some manner of interoperation. Conclusions made in the zone of uncertainty are somewhat speculative; determining whether a particular combination involves copying of protected expression can be a highly fact-intensive inquiry, and without any court decisions to establish legal precedent for particular fact patterns, predictions involve a fair amount of uncertainty.

Where GPL and proprietary software interoperate in some way, the author of the proprietary software may be able to avoid the effect of copyleft by avoiding distribution of his software with the GPL software.[82] The GPL appears to impose copyleft on all software that is part of a distributed whole even if the proprietary software is only tangentially related to the GPL software.[83] Even proprietary software that is an independent and separate work is subject to the copyleft terms if it is distributed as part of a whole based on the GPL software.[84] The GPL

78. GNU General Public License, *supra* note 19.
79. Whelan Assoc., Inc. v. Jaslow Dental Lab., Inc., 797 F.2d 1222, 1248 (3d Cir. 1986).
80. *Id.*
81. While the facts indicate a prima facie case of infringement, the client may raise a defense that the work is not protected expression, or that this is fair use under 17 U.S.C. §107. That discussion is beyond the limited scope of this paper.
82. GNU General Public License, *supra* note 19. (distinguishing works published with GPL software as part of a whole subject to copyleft, from the same works published apart from GPL software).
83. *Id.*
84. *Id.*

does not define "whole," but it appears to broadly include any work bundled with GPL software except mere aggregations.[85] Proprietary software that somehow interoperates with GPL software is likely to trigger copyleft terms if the two are distributed together. The author of the related work may avoid copyleft by distributing the GPL software independently of the related work. This approach allows distribution of the GPL-licensed software under the terms of the GPL and avoids the problem of distributing the combination as a whole.

Proprietary software that interoperates with, but is distributed independently of, GPL software may sidestep the rubric of the GPL "whole," but risks imposition of copyleft if the licensor proves that the proprietary software is a derivative work that was distributed. The FSF asserts that the GPL may impose copyleft terms against software that interoperates in some manner with GPL software, depending on how it interoperates.[86] Convincing a court that copyleft terms were triggered, however, requires showing both derivation and distribution.

Proprietary software that incorporates GPL code as a dynamically linked library probably creates a legally derivative work, but likely will not trigger the GPL's copyleft terms if the library is not distributed with the proprietary software.[87] The proprietary program by itself does not contain the object code from the GPL software and can easily be distributed without the dynamically linked library. The end user can get the library independently and install it on his or her computer, all the while complying with the GPL. When the end user runs the program, however, the program causes the operating system to copy the object code from the dynamically linked library into memory and the program communicates with the library through shared memory space. See Appendix A, Figure 9. Copying the object code from disk into memory can violate the reproduction right of the copyright holder,[88] and the publisher of the proprietary software might be liable for contributory infringement for inducing infringement by the end user.[89] Further, the in-memory combination of the GPL library and the proprietary program that caused it to be loaded creates an in-memory derivative work.[90] The GPL, however, gives liberal allowance for end users to execute GPL software without constraints and given that an in-memory derivative work is not distrib-

85. Vetter, *supra* note 9, at 92 (discussing whether an alleged infringer might successfully raise a defense that the expansive reach of the GPL constitutes copyright misuse remains to be decided by the courts).

86. Frequently Asked Questions about the GNU Licenses, Free Software Foundation, http://www.gnu.org/licenses/gpl-faq.html (last visited Aug. 1, 2008).

87. Determann, *supra* note 45, at 1460.

88. MAI Sys. Corp. v. Peak Computer, 991 F.2d 511, 518 (9th Cir. 1993).

89. *See generally* Metro-Goldwyn-Mayer Studios Inc. v. Grokster, 545 U.S. 913 (2005).

90. Determann, *supra* note 45, at 1460.

uted, it would not be subject to copyleft.[91] Without any direct infringement by the end user, there is no basis for contributory infringement by the publisher of the software referencing that library.[92]

An argument that the proprietary program infringes on the GPL software because it takes information about the GPL software's interfaces will not succeed.[93] For example, the parameters and data structures used to communicate with a library might be argued to be protected expression.[94] Interfaces and data structures, however, are often considered as elements dictated by external factors or as *scenes a faire*.[95] The interfaces may be held to be functional elements, and unprotected by copyright.[96] Finally, interfaces and data structures may be so closely tied to the expression of interaction that the doctrine of merger may prevent successful claims of infringement.[97]

Most other means of interoperation between proprietary and GPL software likely will not constitute a derivative work that would impose the copyleft terms. Where the GPL and proprietary programs interoperate through some means other than dynamic linking – for example, as independent processes running on the same or separate computers – the inquiry follows the same examination of the interface between the programs and the nature of data structures used to communicate.[98] Programs that communicate through some form of inter-process communication ("IPC") such as sockets and named pipes must conform to the interface and data standards imposed by those IPC mechanisms. Since those external factors shape the contours of the interface and data structures, those elements are probably not protected works.[99]

Interoperability that is based on reading and writing files stored on a disk probably does not constitute a derivative work, provided that the file formats were legitimately reverse engineered or in the public domain. If the proprietary program has incorporated protected works from the GPL program in order to read and write those files, then the work is probably derivative under the suspect combinations analysis. However, if the proprietary program's ability to read and write those programs was the result of legitimate reverse engineering, that is likely to be fair use

91. GNU General Public License, *supra* note 19 ("The act of running the Program is not restricted.").

92. Determann, *supra* note 45, at 1486.

93. Mitchell Stoltz, Comment, *The Penguin Paradox: How The Scope of Derivative Works in Copyright Affects the Effectiveness of the GNU GPL*, 85 B.U.L. REV. 1439, 1451 (2005).

94. *Id.*

95. *See, e.g.*, Computer Associates Int'l. v. Altai Inc., 982 F.2d 693, 709 (2d Cir. 1992).

96. *Id.* at 714.

97. *See id.* at 708.

98. Determann, *supra* note 45, at 1449.

99. *Computer Associates Int'l.*, 982 F.2d at 709.

under copyright law and seems unlikely to impose the copyleft terms on the proprietary program.[100]

The final inquiry in the zone of uncertainty concerns proprietary software that neither incorporates GPL software nor makes use of it, but is in some other way based on GPL software. For example, Pidgin is GPL-licensed instant messaging software that works with multiple instant messaging accounts and providers at the same time. Proprietary software subsequently developed that supports multiple instant messaging accounts in the same way may infringe on the GPL software. Here, the inquiry is deeply fact-intensive and focuses on whether substantial similarities exist between the proprietary software and the GPL software.[101] Were such a case to come to trial, the court would likely apply an abstraction/filtration/comparison analysis to separate ideas from expression, isolate protected expressions, and compare those with the alleged infringing work.[102]

Here is an example. OpenAccolade provides a means of adding new functionality through "snap-ins" – software extension modules that allow licensed users of OpenAccolade to customize specific business transactions within that software without rewriting the OpenAccolade source itself. Writing the snap-in modules, however, is difficult even for experienced programmers. A client's program generates most of the code needed to create a snap-in module, making it much easier to build one. The program does not incorporate any code from OpenAccolade and the client wants to keep the source as a trade secret in order to sell it for a profit. In this instance, incorporating the snap-in module with OpenAccolade probably does create a derivative work and the snap-in itself is likely subject to copyleft. But here, it is the end user, not the client, who is creating the combination. The end user of GPL licensed software is free to make use of the software and derivative works for his own use, with few limitations. Provided that the end user is prohibited from distributing a snap-in without the code generated by the client's software, there is no direct infringement by the end user or the client. Therefore there is no contributory infringement for a derivative work. While the client's software must incorporate some knowledge of the interfaces exposed by OpenAccolade in order to generate the snap-in modules, those interfaces are likely beyond copyright protection.[103] Finally, unless OpenAccolade provides a similar means of creating the snap-in modules, the client's software probably bears no substantial similarities to OpenAccolade and is not an infringing work.

100. *See generally* Sega Enterprises Ltd. v. Accolade, Inc., 977 F.2d 1510 (9th Cir. 1992).

101. *See generally Computer Associates Int'l.*, 982 F.2d 693.

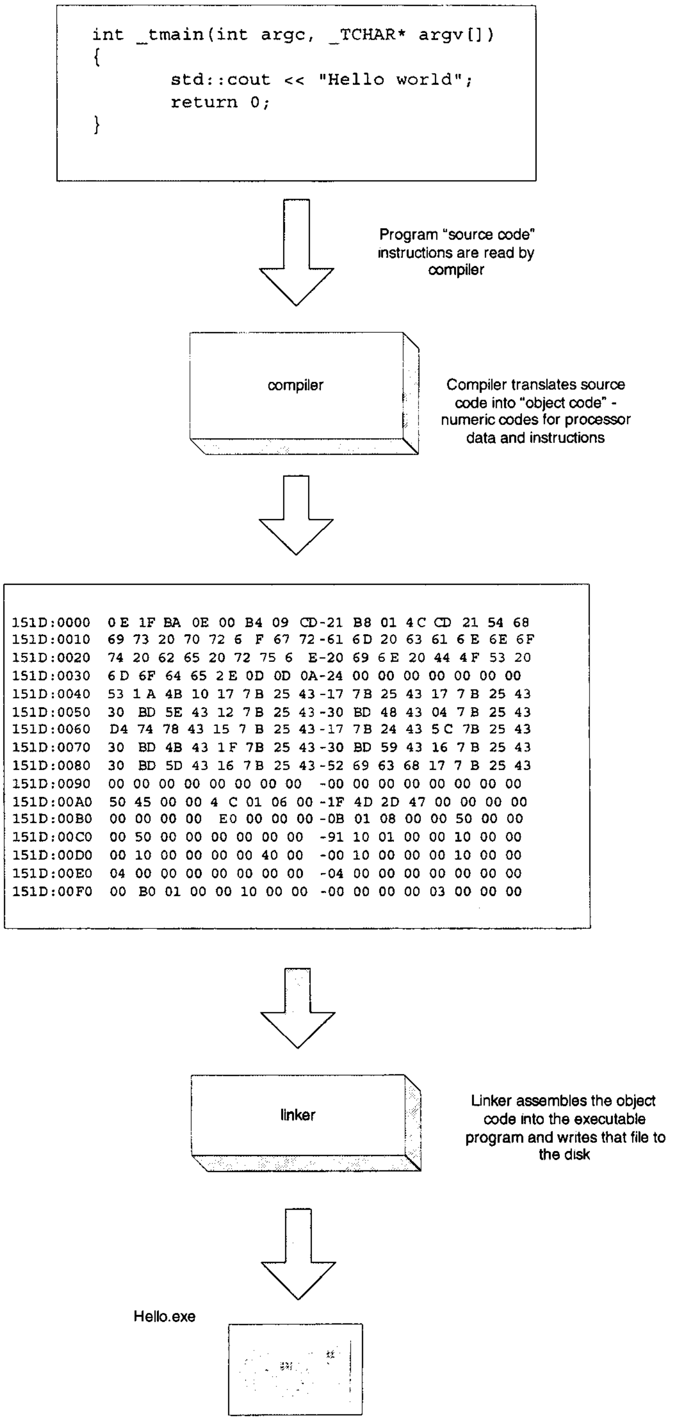
102. *Id.* at 705-06.

103. *Computer Associates Int'l.*, 982 F.2d at 709.

VI. CONCLUSION

Combinations of GPL software with proprietary software can often result in copyright infringement and the viral terms of that license will infect the proprietary source code under many circumstances. This paper presented a framework for both technical and legal analysis to uncover both obvious and obscure forms of copying and to provide a basis for determining whether particular technical combinations will impose the viral copyleft terms on a whole work. That framework is intended as an aid in assessing risk of infringement and mitigating ambiguity inherent in the GPL copyleft terms.

APPENDIX A



```
int _tmain(int argc, _TCHAR* argv[])
{
        std::cout << "Hello world";
        return 0;
}
```

Program "source code" instructions are read by compiler
compiler
Compiler translates source code into "object code" - numeric codes for processor data and instructions

```
151D:0000  0E 1F BA 0E 00 B4 09 CD-21 B8 01 4C CD 21 54 68
151D:0010  69 73 20 70 72 6 F 67 72-61 6D 20 63 61 6E 6E 6F
151D:0020  74 20 62 65 20 72 75 6 E-20 69 6E 20 44 4F 53 20
151D:0030  6D 6F 64 65 2E 0D 0D 0A-24 00 00 00 00 00 00 00
151D:0040  53 1A 4B 10 17 7B 25 43-17 7B 25 43 17 7B 25 43
151D:0050  30 BD 5E 43 12 7B 25 43-30 BD 48 43 04 7B 25 43
151D:0060  D4 74 78 43 15 7 B 25 43-17 7B 24 43 5C 7B 25 43
151D:0070  30 BD 4B 43 1F 7B 25 43-30 BD 59 43 16 7B 25 43
151D:0080  30 BD 5D 43 16 7B 25 43-52 69 63 68 17 7 B 25 43
151D:0090  00 00 00 00 00 00 00 00 -00 00 00 00 00 00 00 00
151D:00A0  50 45 00 00 4 C 01 06 00-1F 4D 2D 47 00 00 00 00
151D:00B0  00 00 00 00  E0 00 00 00-0B 01 08 00 00 50 00 00
151D:00C0  00 50 00 00 00 00 00 00 -91 10 01 00 00 10 00 00
151D:00D0  00 10 00 00 00 00 40 00 -00 10 00 00 00 10 00 00
151D:00E0  04 00 00 00 00 00 00 00 -04 00 00 00 00 00 00 00
151D:00F0  00 B0 01 00 00 10 00 00 -00 00 00 00 03 00 00 00
```

linker
Linker assembles the object code into the executable program and writes that file to the disk
Hello.exe

FIGURE 1

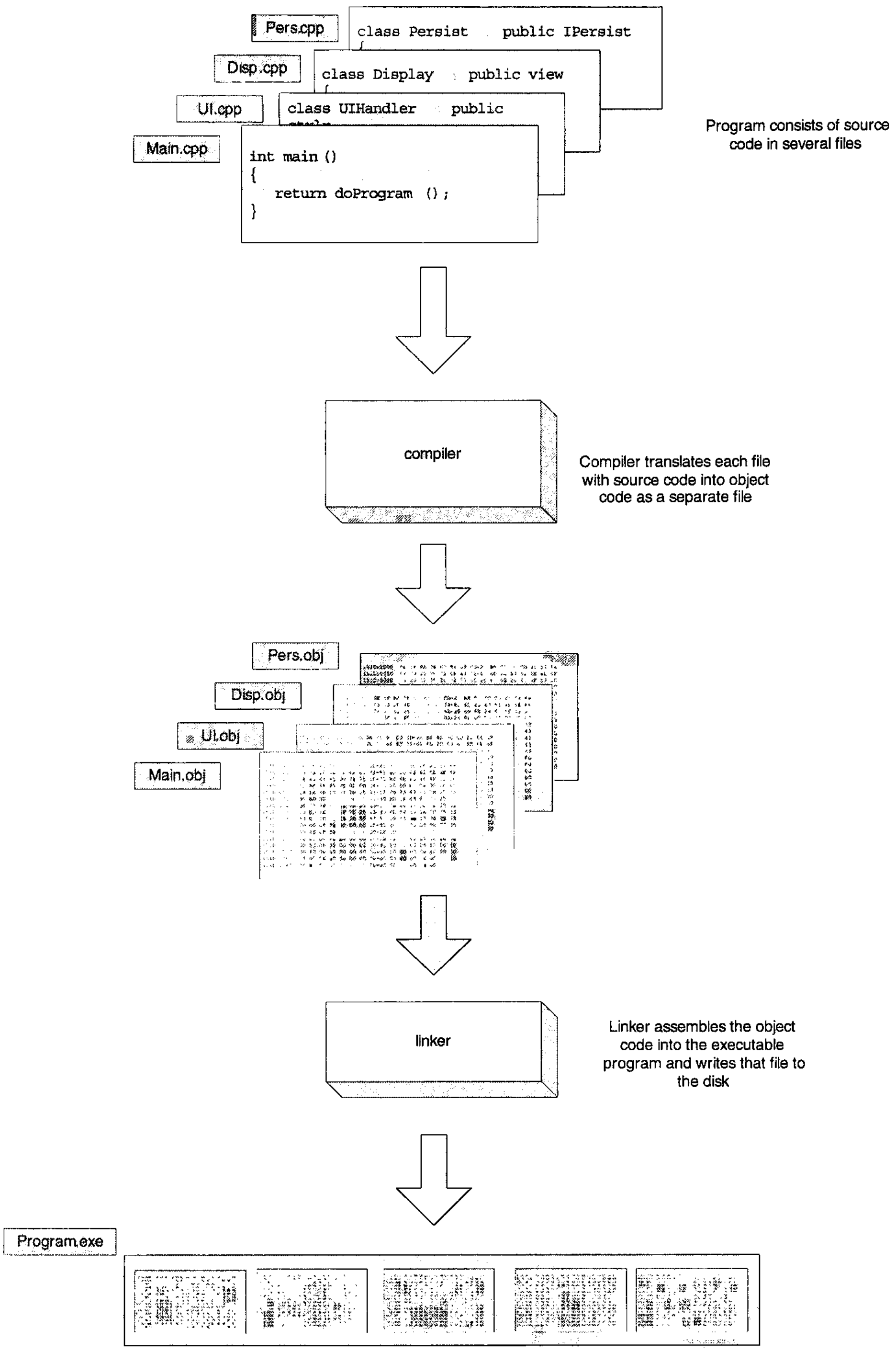

Pers.cpp
class Persist : public IPersist
Disp.cpp
class Display : public view
UI.cpp
class UIHandler : public
Main.cpp
int main ()
{
return doProgram ();
}
Program consists of source code in several files
compiler
Compiler translates each file with source code into object code as a separate file
Pers.obj
Disp.obj
UI.obj
Main.obj
linker
Linker assembles the object code into the executable program and writes that file to the disk
Program.exe

FIGURE 2

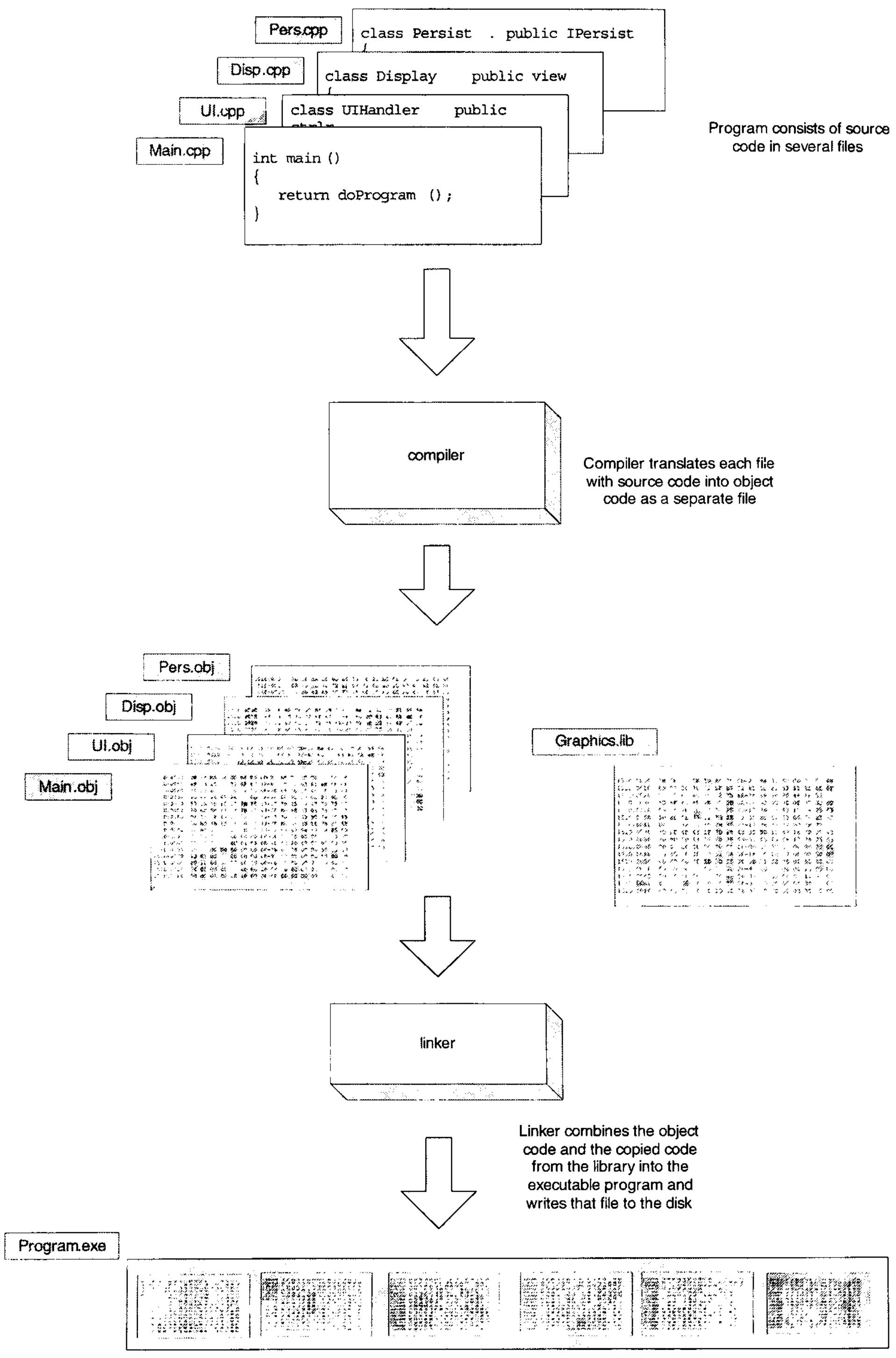

Pers.cpp
class Persist . public IPersist
Disp.cpp
class Display public view
UI.cpp
class UIHandler public
Main.cpp
int main ()
{
return doProgram ();
}
Program consists of source code in several files
compiler
Compiler translates each file with source code into object code as a separate file
Pers.obj
Disp.obj
UI.obj
Main.obj
Graphics.lib
linker
Linker combines the object code and the copied code from the library into the executable program and writes that file to the disk
Program.exe

FIGURE 3

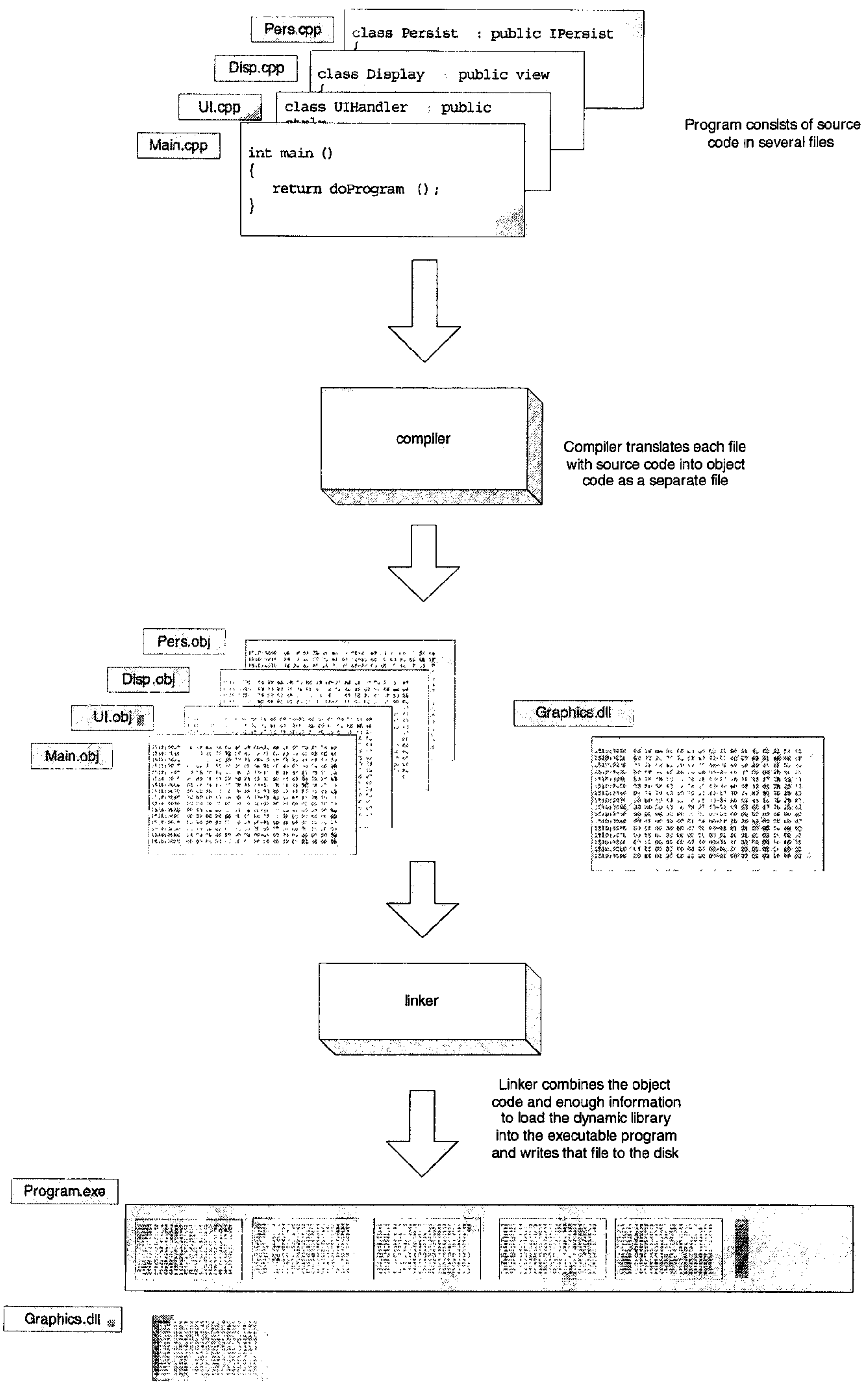

Pers.cpp
class Persist : public IPersist
Disp.cpp
class Display : public view
UI.cpp
class UIHandler : public
Main.cpp
int main ()
{
return doProgram ();
}
Program consists of source code in several files
compiler
Compiler translates each file with source code into object code as a separate file
Pers.obj
Disp.obj
UI.obj
Main.obj
Graphics.dll
linker
Linker combines the object code and enough information to load the dynamic library into the executable program and writes that file to the disk
Program.exe
Graphics.dll

FIGURE 4

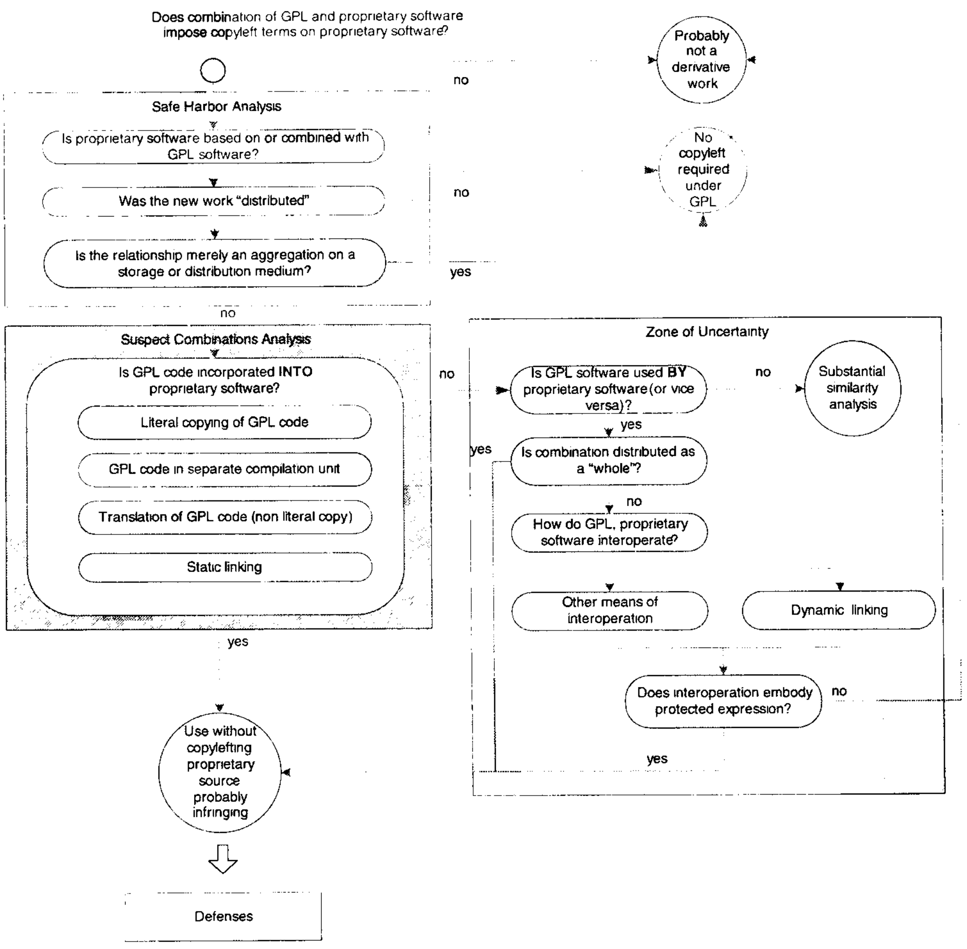
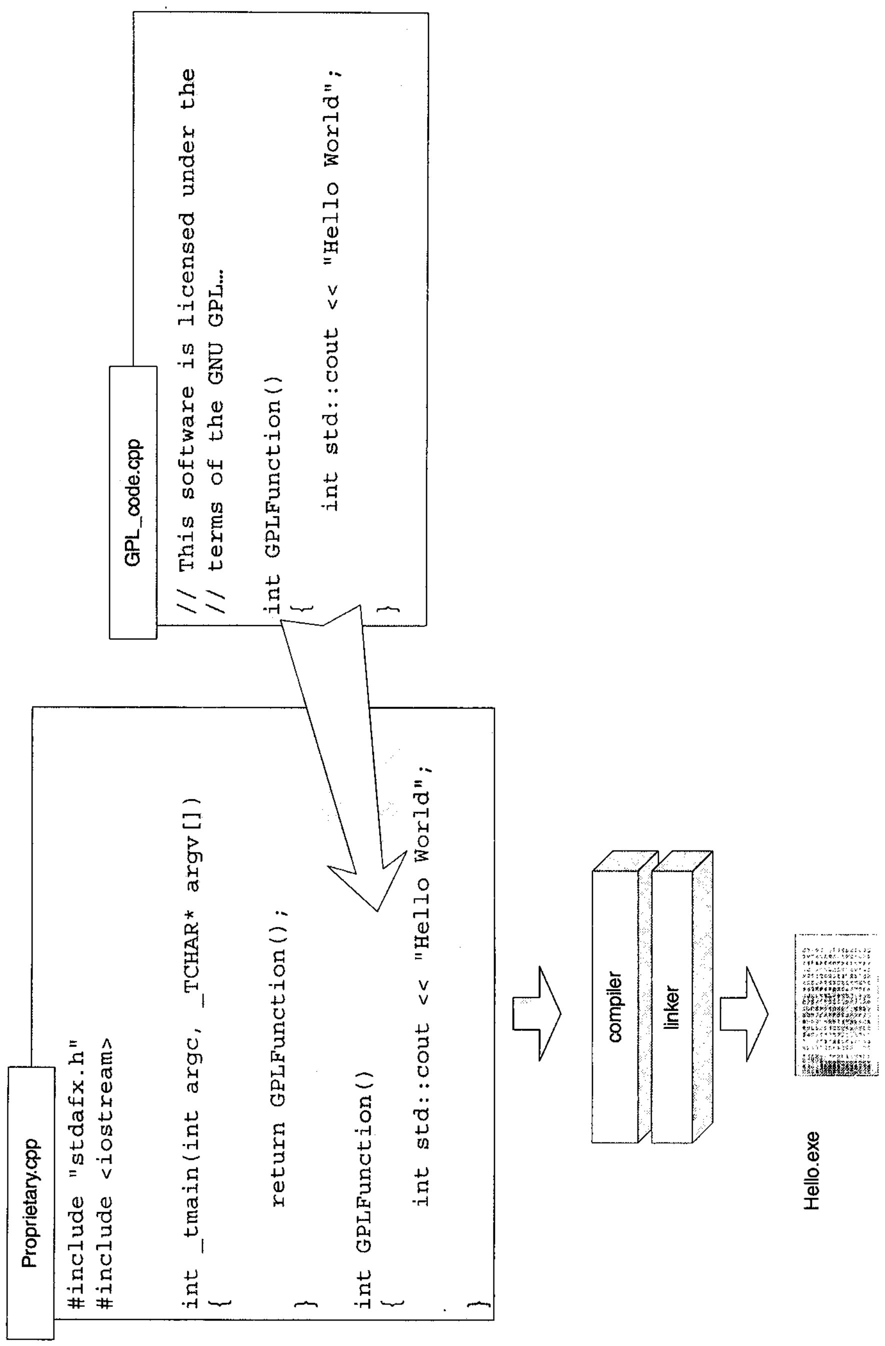

Does combination of GPL and proprietary software impose copyleft terms on proprietary software?
Probably not a derivative work
no
Safe Harbor Analysis
Is proprietary software based on or combined with GPL software?
Was the new work "distributed"
no
No copyleft required under GPL
Is the relationship merely an aggregation on a storage or distribution medium?
yes
no
Suspect Combinations Analysis
Is GPL code incorporated INTO proprietary software?
Literal copying of GPL code
GPL code in separate compilation unit
Translation of GPL code (non literal copy)
Static linking
no
Zone of Uncertainty
Is GPL software used BY proprietary software (or vice versa)?
no
Substantial similarity analysis
yes
yes
Is combination distributed as a "whole"?
no
How do GPL, proprietary software interoperate?
Other means of interoperation
Dynamic linking
Does interoperation embody protected expression?
no
yes
yes
Use without copylefting proprietary source probably infringing
Defenses

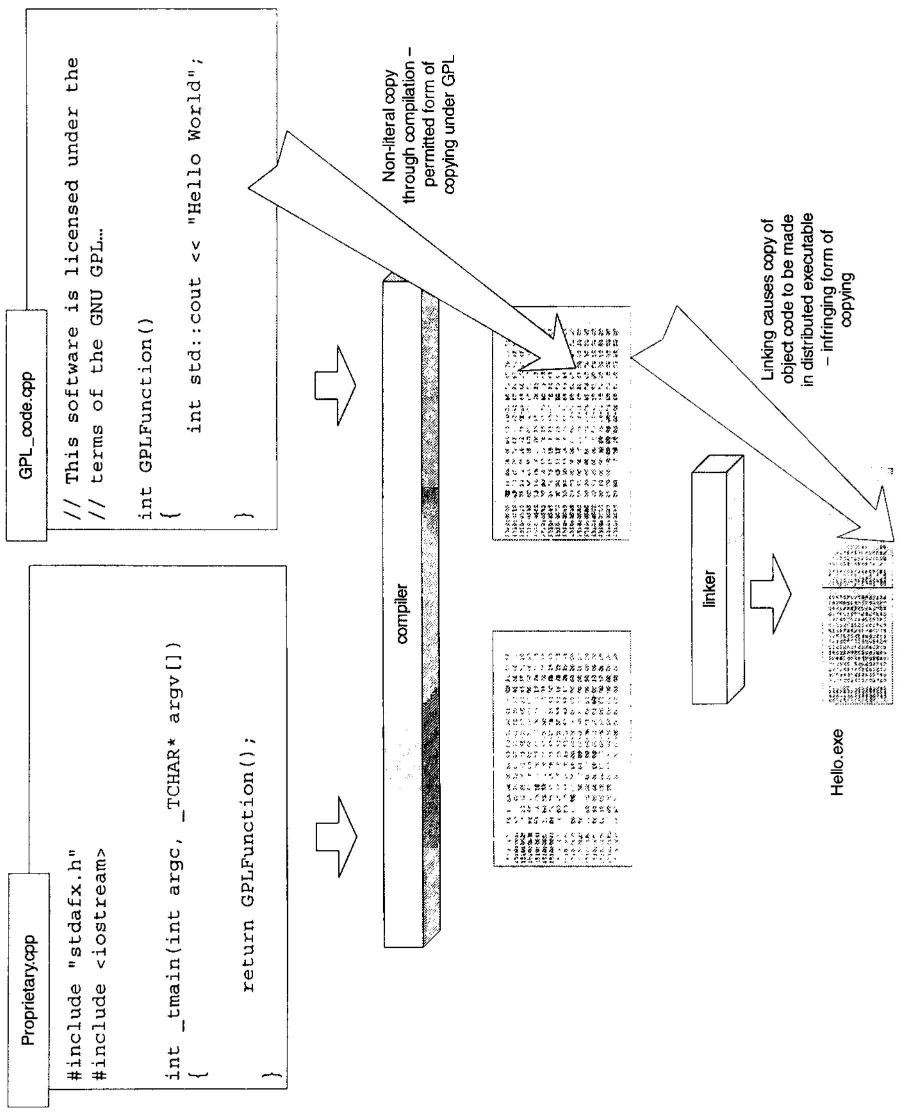
FIGURE 5



```
#include "stdafx.h"
#include <iostream>

int _tmain(int argc, _TCHAR* argv[])
{
     return GPLFunction();
}

int GPLFunction()
{
     int std::cout << "Hello World";
}
```

```
// This software is licensed under the
// terms of the GNU GPL…

int GPLFunction()
{
     int std::cout << "Hello World";
}
```

FIGURE 6


Proprietary.cpp
GPL_code.cpp
compiler
Non-literal copy through compilation - permitted form of copying under GPL
linker
Linking causes copy of object code to be made in distributed executable - infringing form of copying
Hello.exe

```
#include "stdafx.h"
#include <iostream>

int _tmain(int argc, _TCHAR* argv[])
{
        return GPLFunction();
}
```

```
// This software is licensed under the
// terms of the GNU GPL...

int GPLFunction()
{
        int std::cout << "Hello World";
}
```

FIGURE 7

Original source –
C++ programming
language

```
#include <iostream>


int main(int argc, char ** argv)
{
        std::cout << "Hello world" << "\n";
        return 0;
}
```

Translated source –
Java programming
language

```

public class hello
{

        public static void main (String[] args)
        {
                System.out.println("Hello World");
        }
}
```

Object code for
original

```
0000000 457f 464c 0101 0001 0000 0000 0000 0000
0000010 0002 0003 0001 0000 84e0 0804 0034 0000
0000020 0ba4 0000 0000 0000 0034 0020 0008 0028
0000030 001d 001a 0006 0000 0034 0000 8034 0804
0000040 8034 0804 0100 0000 0100 0000 0005 0000
0000050 0004 0000 0003 0000 0134 0000 8134 0804
0000060 8134 0804 0013 0000 0013 0000 0004 0000
0000070 0001 0000 0001 0000 0000 0000 8000 0804
0000080 8000 0804 087c 0000 087c 0000 0005 0000
0000090 1000 0000 0001 0000 087c 0000 987c 0804
00000a0 987c 0804 0128 0000 01c4 0000 0006 0000
00000b0 1000 0000 0002 0000 0894 0000 9894 0804
00000c0 9894 0804 00e0 0000 00e0 0000 0006 0000
00000d0 0004 0000 0004 0000 0148 0000 8148 0804
00000e0 8148 0804 0020 0000 0020 0000 0004 0000
00000f0 0004 0000 e550 6474 0750 0000 8750 0804
0000100 8750 0804 003c 0000 003c 0000 0004 0000
0000110 0004 0000 e551 6474 0000 0000 0000 0000
0000120 0000 0000 0000 0000 0000 0000 0006 0000
0000130 0004 0000 6c2f 6269 6c2f 2d64 696c 756e
0000140 2e78 6f73 322e 0000 0004 0000 0010 0000
0000150 0001 0000 4e47 0055 0000 0000 0002 0000
0000160 0006 0000 0009 0000 0003 0000 0008 0000
0000170 0001 0000 0005 0000 2801 2110 0000 0000
0000180 0008 0000 0009 0000 4bad c0e3 9814 430c
0000190 4979 b66b 0000 0000 0000 0000 0000 0000
00001a0 0000 0000 00eb 0000 0000 0000 0057 0000
00001b0 0012 0000 0010 0000 0000 0000 0000 0000
00001c0 0020 0000 001f 0000 0000 0000 0000 0000
```

Object code for
translation

```
00000 feca beba 0000 2e00 1d00 0007 0102 0500
0000010 6568 6c6c 076f 0400 0001 6a10 7661 2f61
0000020 616c 676e 4f2f 6a62 6365 0174 0600 693c
0000030 696e 3e74 0001 2803 5629 0001 4304 646f
0000040 0a65 0300 0900 000c 0005 0106 0f00 694c
0000050 656e 754e 626d 7265 6154 6c62 0165 0400
0000060 616d 6e69 0001 2816 4c5b 616a 6176 6c2f
0000070 6e61 2f67 7435 6972 676e 293b 0956 0e00
0000080 1000 0007 010f 1000 616a 6176 6c2f 6e61
0000090 2f67 7653 7473 6d65 000c 0011 0112 0300
00000a0 756f 0174 1500 6a4c 7661 2f61 6f69 502f
00000b0 6972 746e 7453 6572 6d61 083b 1400 0001
00000c0 480b 6c65 6f6c 5720 726f 646c 000a 0016
00000d0 0718 1700 0001 6a13 7661 2f61 6f69 502f
00000e0 6972 746e 7453 6572 6d61 000c 0019 011a
00000f0 0700 7270 6e69 6c74 016e 1500 4c28 616a
0000100 6176 6c2f 6e61 2f67 7453 6972 676e 293b
0000110 0156 0a00 6f53 7275 6563 6946 656c 0001
0000120 680a 6c65 6f6c 6a22 7661 0061 0021 0001
0000130 0003 0000 0000 0002 0001 0005 0006 0001
0000140 0007 0000 001d 0001 0001 0000 2a05 00b7
0000150 b108 0000 0100 000a 0000 0600 0100 0000
0000160 0100 0900 0b00 0c00 0100 0700 0000 2500
0000170 0200 0100 0000 0900 00b2 120d b613 1500
0000180 00b1 0000 0001 000a 0000 000a 0002 0000
0000190 0006 0008 0007 0001 001b 0000 0002 001c
```

FIGURE 8

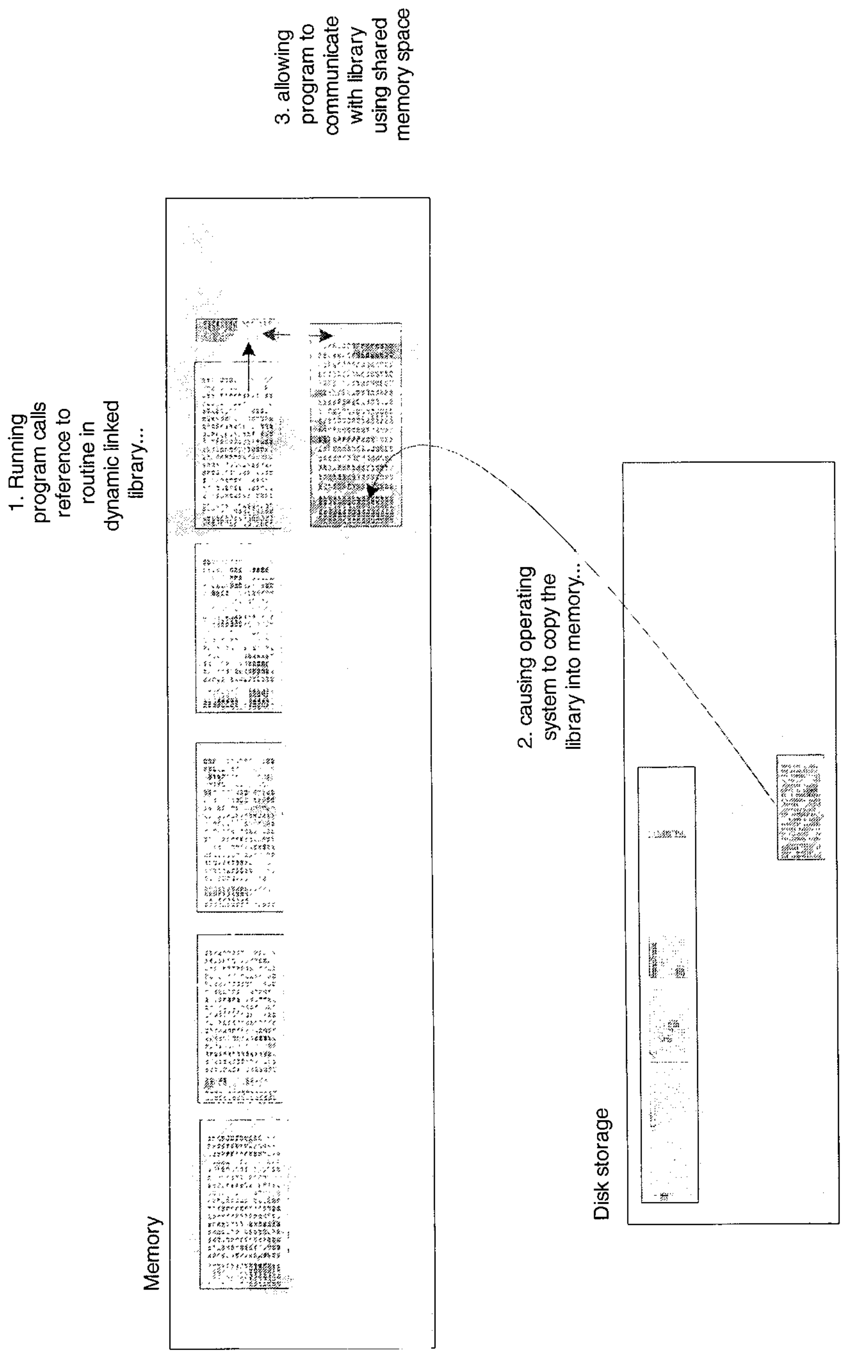

1. Running program calls reference to routine in dynamic linked library...
2. causing operating system to copy the library into memory...
3. allowing program to communicate with library using shared memory space
Memory
Disk storage

FIGURE 9