# Chapter 3
# The Legal Governance Structure

## 3.1. INTRODUCTION

The legislators and the industry arrived at an uneasy accommodation as regards legal protection of computer programmes. Currently, trade secrets, trademarks, copyright, patent, design rights, and moral rights apply to the field of software protection. This is supplemented by technological protection measures and license provisions. The issues as regards legal protection of computer software were again raked up by Open Source Software (OSS); it appears to pose a dilemma, because of its interaction and implications with the various branches of intellectual property.

The legal instrument for propagating the OSS philosophy is the license. The OSS movement designed a counterintuitive licensing system based on the same legal premise as the closed source software but to different ends. Though OSS proponents are in agreement regarding the essential philosophy of the movement, when it comes to transcending policy into practice, licenses tend to diverge on certain terms important to collaboration. To avoid conflicts, and promote modulated growth of OSS, definitional standards have been set by standard setting organizations. Licenses may have different terms, but they must each comply with the core principles propagated by these standard setting bodies in order for the software to be termed OSS.

Section 3.2.1 of this chapter introduces the general approach of OSS towards intellectual property laws. It focuses on issues that emanate from the interaction of the OSS model with the existing intellectual property rights (IPR) structure. How intellectual property impacts on the open source model and how OSS uses intellectual property in a novel way to achieve its ends are discussed. Section 3.2.2 introduces how OSS propagates its model through the licensing medium. The essence of development of the OSS model is based on its marked variance from the proprietary model; this Part etches those differences.

## 3.2. LEGAL GOVERNANCE STRUCTURE OF OPEN SOURCE SOFTWARE

Like the advent of any new technology, computer software too represents several challenges to the established intellectual property law regime. Traditionally, copyright law was available to protect the literal, and patent law, the mechanical. As a written work with a utilitarian purpose, computer programs deride and defy categorization in the present library of intellectual property protection.[1] Despite the paradox, the current intellectual property regime, arrived at an accommodation, and protects various components of a computer software separately through copyright, patent and trade secret laws- copyright law protects the literary aspect, patent law safeguards the utilitarian aspects, and trade secret law theoretically protects the idea.[2] Technological protection measures and licensing provide additional protection.

Interaction of the traditional intellectual property structure and computer software, generated the proprietary model of software development, which was the veritable apotheosis for over two decades. Concurrently, laments increased, that traditional intellectual property protection in software were steadily whittling down the public domain. The issues as regards legal protection of computer software were raked up by the arrival of OSS on the scene. The last couple of decades saw the emergence of OSS as David in this *David v. Goliath* clash.

OSS is not in the public domain. Rather, OSS uses the traditional means to achieve different ends.[3] It depends on an explicit intellectual property regime, to permit its free distribution, via a series of licenses. This new

1. Christina M. Reger, *Let's Swap Copyright For Code: The Computer Software Disclosure Dichotomy*, 24 Loy. L.A. Ent. L. Rev. 215, 221 (2004).
2. *Ibid.*
3. Patrick K. Bobko, *Open-Source Software and the Demise of Copyright*, 27 Rutgers Computer & Tech. LJ 51, 90 (2001) (Open-source software subverts the foundation upon which the commercial software industry is built); Michael J. Madison, *Reconstructing the Software License*, 35 Loy. U. Chi. LJ 275, 285 (2003) (The 'open' (or shared) source code model thus sharply contrasts with the conventional 'closed' (or hidden) source code model at one level but adopts the same underlying legal framework. In the former, both legally and technologically speaking, the program is meant to be distributed and shared among all of its producers and consumers. In the latter, both legally and technologically speaking, the program is meant to be controlled by the original producer. The open source model is ultimately a specialized application of the general purpose conventional software license).

model of software development, instead of relying on the conventional proprietary model of limited access, invites programmers globally, to freely copy, share, and modify the software. 'The open-source software movement reflects the elegant and creative use of property rights and contract law to sustain creative work based on free and transparent distribution rather than the conventional model of exclusion'.[4] Developing upon the mainstream intellectual property regime, OSS, 'shifts the fundamental optic of IPRs away from protecting the prerogatives of an author toward protecting the prerogatives of generations of users'.[5] As *Mann* states:

> Two related features of the open source model are distinctive: the use of collaborative development structures that extend beyond the boundaries of a single firm, and the lack of reliance on intellectual property rights as a means of appropriating the value of the underlying technologies. Firm-level control of intellectual property is replaced by a complex set of relations, both informal and sometimes contractual, among strategic partners not joined by firm boundaries.[6]

The OSS movement, necessitates scrutiny; more than just being a new fangled approach, it catalyzes debate regarding both the mode of software production and its protection. After all, the intention of the intellectual property-software system is to propagate innovation and ultimately serve the society. 'The legal perspective of free software cannot be underestimated. In the end, the whole free software philosophy is only possible through its legal construction'.[7]

### 3.2.1. EXISTING INTELLECTUAL PROPERTY PROTECTION AND OPEN SOURCE SOFTWARE

Essentially, four major types of intellectual property protection apply to software. Trade secret law was the traditional vehicle of software protection; it can protect secrets embodied in or implemented through software. While, copyright chosen as the legislative vehicle, protects the literal expression of software; patent protection for software has grown doctrinally and essentially protects the technological expression of software. Trademark protection too

4. David McGowan, *Intellectual Property Challenges in The Next Century: Legal Implications of Open-Source Software*, 2001 U. Ill. L. Rev. 241, 303 (2001) (Open-source production rests ultimately on the right to exclude. But the point of the right in the open-source community is that it is not used; like the sword of Damocles, the point is not that it falls but that it hangs).
5. Robert A. Hillman and Maureen A. O'Rourke, *Rethinking Consideration in the Electronic Age*, 61 Hastings LJ 311, 313 (2009).
6. Ronald J. Mann, *Commercializing Open Source Software: Do Property Rights Still Matter?*, 20 Harv. J. Law & Tec 1, 21 (2006).
7. Dr Jose J. Gonzalez de Alaiza Cardona, *Open Source, Free Software, and Contractual Issues*, 15 Tex. Intell. Prop. LJ 157, 173 (2007).

applies to software. In addition to these laws, moral rights and design protection laws too are applied. It is not necessary that all the forms of protection would be exercised with every software; developers may choose to rely on none, some, or all of these forms of intellectual property protection.[8] The key critique of the traditional intellectual property protection system is that it is better suited for protecting rights than for facilitating an easy relinquishment of rights; thus the inflexible copyright regimen led to a new movement- the copyleft movement, aimed at giving rights holders the ability to grant certain copyrights, at their discretion, to the entire community.[9]

### 3.2.1.1. Trade Secret Protection

Trade secret law was favoured in the early phases of computing technology, when software was individually distributed under tight contractual control. However, with technological evolution, it no longer remained apt or adequate. Though trade secret protection is still used for software, it is no longer its dominant or sole mode of protection.[10] OSS furthers this rift; its ideology is in complete conflict with Trade Secret law, thus rendering its application to OSS unfeasible. 'The concepts of open source on the one hand, and trade secrets, on the other, seem diametrically opposed to one another'.[11] Trade secret protection of OSS would likely be forfeited with the distribution of open source code.[12] As *Vetter* states:

> [T]rade secret protection is singularly inapplicable to open-source software. The accessible and open source code would almost always defeat the trade secret status by disclosing the secret.[13]

The FSF considers distribution of code as trade secret as a violation of GPL.[14] In fact, the FSF considers distribution under any type of restrictive agreement like a non-disclosure agreement to be a violation of the GPL.[15]

8. David S. Evans and Bernard J. Reddy, *Government Preferences for Promoting Open-Source Software: A Solution in Search of A Problem*, 9 Mich. Telecomm. Tech. L. Rev. 313, 320 (2003).
9. Christopher S. Brown, *Copyleft, The Disguised Copyright: Why Legislative Copyright Reform is Superior to Copyleft Licenses*, 78 UMKC L. Rev. 749, 750 (2010).
10. See *generally* Greg R. Vetter, *The Collaborative Integrity of Open-Source Software*, 563 Utah L. Rev. 563 (2004).
11. Stephen J. Davidson and Stuart D. Levi, *Open Source Software: Critical Issues In Today's Corporate Environment* 169 (Practising Law Inst., 2005).
12. F. Lawrence Street, *Law of the Internet* Ch. 14 § 14.05 (Matthew Bender, 2009).
13. Vetter, *The Collaborative Integrity, supra* n. 10, at 588.
14. The Free Software Foundation, *Frequently Asked Questions about the GNU Licenses*, http://www.gnu.org/licenses/gpl-faq.html (accessed 12 Mar. 2014) (If a company distributes a copy to you and claims it is a trade secret, the company has violated the GPL and will have to cease distribution).
15. *Ibid.* (Does the GPL allow me to distribute copies under a non-disclosure agreement? No. The GPL says that anyone who receives a copy from you has the right to redistribute

Largely, it seems improbable for OSS to contain any subject matter that qualifies as a trade secret, especially as their core ideal is to proliferate source code.[16] However, the GPL allows developing code on a restrictive basis in a client-developer type software development model.[17] Thus, this is one scenario where trade secrets might still be applicable in the OSS context. Still, in such a scenario, the developer would be able to harness the open source community only uptil the period before taking the code as trade secret and any inputs thereafter would be curtailed. Though it may be possible in certain circumstances to maintain trade secrets in OSS, however, the opportunities for doing so appear to be relatively narrow with little benefit.[18]

Trade secrets in the OSS arena have been raised in litigation too, in *Red Hat v. SCO*,[19] where, Red Hat sought to obtain a declaratory judgment arguing that Linux being publicly available could not be protected by trade secrets, and hence, Red Hat had not violated SCO's trade secrets.

### 3.2.1.2. Copyright Protection

OSS proponents disapprove of the current proprietary model of copyright protection for software. They perceive it as a deviant application of the justification for copyright law – to induce innovation; also they challenge the need for monetary incentives itself to induce innovation in software creation. OSS proponents believe that the proprietary software model restrict the benefits to society.[20] In an effort to rectify this perceived erroneous state, the OSS regime introduced their own vision of copyright protection for software, the most prominent application of which is the copyleft concept. 'Copyleft ... offers a dissenting logic based in sharing and proffers an economic model

copies, modified or not. You are not allowed to distribute the work on any more restrictive basis); Davidson and Levi, *supra* n. 11 interpreting similar restrictions under the BSD License, the MIT License and the Mozilla License.

16. Davidson and Levi, *supra* n. 11.
17. The Free Software Foundation, Frequently Asked Questions, *supra* n. 14 (Does the GPL allow me to develop a modified version under a non-disclosure agreement? Yes. For instance, you can accept a contract to develop changes and agree not to release *your changes* until the client says ok. This is permitted because in this case no GPL-covered code is being distributed under an NDA. You can also release your changes to the client under the GPL, but agree not to release them to anyone else unless the client says ok. In this case, too, no GPL-covered code is being distributed under an NDA, or under any additional restrictions. The GPL would give the client the right to redistribute your version. In this scenario, the client will probably choose not to exercise that right, but does *have* the right).
18. Davidson and Levi, *supra* n. 11, at 176.
19. *Red Hat, Inc. v. The SCO Group*, No. 03-772, Plaintiff's Complaint, Count II, ¶ 74-77 (D. Del. 4 Aug. 2003) http://www.groklaw.net/pdf/RedHat-1.pdf.
20. See Shawn W. Potter, *Opening Up to Open Source*, 6 Rich. J.L. & Tech. 24 (2000); see also Nilay Patel, *Open Source and China: Inverting Copyright?* 23 Wis. Int'l LJ 781 (2005).

premised upon giving'.[21] The open source movement reflects the intent of its founders to turn traditional notions of copyright, software licensing, distribution, development and even ownership on their heads, even to the point of creating the term copyleft to describe the alternative approach to these issues.[22]

OSS does not relinquish copyright, infact, 'open source lives or dies on copyright law'.[23] However, OSS innovatively re-interprets the essential legal foundation upon which the proprietary software industry exists. Unlike proprietary software, where copyright law is used to exclude, in OSS, copyright law is used to confer a right to distribute.[24] The OSS *modus operandi* essentially is to copyright software and then mass license it for use, improvement or modification under the condition that the alterations are contributed back to the software community on the same terms. This approach gives subsequent users of the copyrighted code the ability to pass along restrictions that embody open source tenets, resulting in the dissemination of the tenets in proportion to the distribution of the code.[25] Herein lies the innovativeness of the OSS premise, where some traditional copyright benefits are relinquished, thus allowing greater leeway in rights usage. The Preamble of the GNU GPLv1 affirms this ideal by specifically stating that to protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender your rights. The GPL has been called 'a hack on the copyright system' because of this very ideology where it uses the exclusive rights of authors to guarantee rather than restrict public access to software source code.[26] This attempt to allow the author and the public to interact in a way that normal copyright does not is considered as an

21. Severine Dusollier, *Open Source and Copyleft: Authorship Reconsidered?*, 26 Colum. J.L. & Arts 281, 287 (2003).
22. Dennis M. Kennedy, *A Primer on Open Source Licensing Legal Issues: Copyright, Copyleft and Copyfuture*, 20 St. Louis U. Pub. L. Rev. 345, 345 (2001).
23. Marcus Maher, *Open Source Software: The Success of an Alternative Intellectual Property Incentive Paradigm*, 10 Fordham Intell. Prop. Media & Ent. LJ 619, 637 (2000) citing Larry Wall, *Diligence, Patience, and Humility, in Open Sources: Voices from the Open Source Revolution*, 127, 142 (Chris Dibona et al. eds., 1999); Joel West, *Business, Law, and Engineering Perspectives on Open Source Innovation: Policy Challenges of Open, Cumulative, and User Innovation*, 30 Wash. U. J.L. & Pol'y 17, 29 (2009) (The creation of new copyright licenses (open source, free software, and creative commons) shows that the existing copyright law can be used to facilitate, rather than deter, cumulative innovation).
24. Charles R. McManis and Eul Soo Seo, *The Interface of Open Source and Proprietary Agricultural Innovation: Facilitated Access and Benefit-Sharing Under the New FAO Treaty*, 30 Wash. U. J.L. & Pol'y 405, 443 (2009) citing Steven Weber, *The Success of Open Source* 1 (2004).
25. McGowan, *supra* n. 4, at 287.
26. Mitchell L. Stoltz, *The Penguin Paradox: How The Scope Of Derivative Works in Copyright Affects the Effectiveness of the GNU GPL*, 85 B.U.L. Rev. 1439, 1477 (2005).

inherently imperfect solution to what many individuals see as the unreasonableness of contemporary copyright law.[27] Imperfect or not, this concept has spawned a completely divergent policy model of software creation.

The option of donating the work to the public domain is not feasible for apprehension of appropriation and conversion to the proprietary model, thus frustrating the essential OSS agenda:

> The simplest way to make a program free software is to put it in the public domain, uncopyrighted. This allows people to share the program and their improvements, if they are so minded. But it also allows uncooperative people to convert the program into proprietary software. They can make changes, many or few, and distribute the result as a proprietary product. People who receive the program in that modified form do not have the freedom that the original author gave them; the middleman has stripped it away.[28]

Without copyright protection and the ensuing licensing scheme, the OSS development model would be just an honour system.[29] Hence, OSS production has come to rest heavily on property rights.

Scope of copyright protection has an inverse relationship with public access. This in turn variedly impacts on and determines the development of proprietary software and OSS. Consequentially, a need arises to determine the scope of copyright protection for computer software. However, it is easier said than done. Unfortunately, due to their different ideologies, and because it uses copyright law for such a radically different purpose, any change in the law that strengthens the OSS development model also promotes abuses in the proprietary system, and any change that directly addresses those abuses tends to weaken it.[30] Furthermore, as regards the idea-expression dichotomy,

27. Matthew D. Satchwell, *The Tao of Open Source: Minimum Action for Maximum Gain*, 20 Berkeley Tech. LJ 1757, 1782 (2005).
28. The Free Software Foundation, *What is Copyleft?* http://www.gnu.org/copyleft/ (accessed 12 Mar. 2014); Brad Frazer, *Open Source is Not Public Domain: Evolving Licensing Philosophies*, 45 Idaho L. Rev. 349, 350 (2009) (The terms are not necessarily synonymous with either shareware or freeware, and open source software is almost certainly not in the public domain (another misunderstood and misused term of art in copyright law)); Cardona, *supra* n. 7 (In other words, we would take the necessary steps to place the program in the public domain. However, this would be unsatisfactory in order to achieve the goals of the free software movement. First, a program in the public domain can be released as object code, and that impedes the access to its source code. Second, if no copyrights are attached to the program, even being released as source code, everyone could redistribute it as proprietary – that is, turn its source code into object code and release it only as such. If the redistribution as proprietary software takes place after any copyrightable modification is made to the public domain program, the author of the modification will be protected under copyright laws).
29. Robert W. Gomulkiewicz, *How Copyleft uses License Rights to Succeed in the Open Source Software Revolution and the Implications for Article 2B*, 36 Hous. L. Rev. 179, 186 (1999).
30. Stoltz, *supra* n. 26, at 1477.

'nobody has ever been able to fix that boundary, and nobody ever can'.[31] Hence, determining non-literal elements of software remains controversial. Analogously, determining non-literal infringement as against literal infringement is difficult. Judiciary is divided as regards the scope of protection to be afforded to the non-literal elements of a computer program. The broader view provides expansive protection, bringing such elements as the program's structure, sequence, and organization within its scope.[32] On the other hand, the narrower view, confines the scope of copyright protection by excising the program's non-copyrightable elements and then determining its overall copyright protection.[33] Both factions have their share of critics. Whether the broader view or the narrower view is resorted to, both have their iota of positive factors and challenges, and both would cause the development of OSS along different planes. Taking a neutral stand as regards open source development, *Bobko*, states:

> Pragmatically, the 'broad' view is acceptable because it brings more and more intellectual territory within open-source's embrace … critics feared its implications for innovation in the information industries … over-broad copyright protection was wasteful because developers were constantly re-inventing the wheel in order to avoid infringing on a competitor's copyrighted software. Broad copyright protection of OSS obviates these concerns … On the other hand, the 'narrow' view … is equally conducive to the open-source movement, even if it is conceptually inaccurate … The decrease in scope of copyright protection allegedly increases both innovation and efficiency because software developers are not forced to re-invent the wheel. The 'narrow' view that restricts copyright protection to 'a kernel, or possible kernels, of creative expression' aligns itself with the open-source movement because it reduces the pockets of closed code around which hackers must work. Open-source software, coupled with a 'narrow' view of copyright, allays any concern that copyright protection is 'blocking innovation in the software arts'.[34]

A major concern for the OSS arena, and a common allegation of the proprietary arena is the infiltration of open source code in proprietary code and *vice versa*. It could open both sides to allegations of infringement. This issue came to a heated point in the *SCO v. IBM* litigation,[35] where SCO

31. *Nichols v. Universal Pictures Corp.*, 45 F.2d 119, 121 (2d Cir. 1930).
32. See *Whelan Associates, Inc. v. Jaslow Dental Laboratory, Inc*, 609 F. Supp. 1307 (E.D. Pa. 1985).
33. Bobko, *supra* n. 3, at 70, citing, Marci A. Hamilton & Ted Sabety, *Computer Science Concepts In Copyright Cases: The Path to a Coherent Law*, 10 Harv. J.L. & Tech. 240, 250 (1997); see *Computer Assocs. Int'l v. Altai, Inc.*, 126 F.3d 365 (2d Cir. N.Y. 1997).
34. Bobko, *supra* n. 3, at 89–90.
35. *Caldera Sys., Inc. v. Int'l Bus. Machs. Corp.* (D.Utah 2003) (No. 03-CV-0294).

claimed that IBM illegally incorporated SCO's proprietary UNIX code into the open-source Linux operating system, and thus every Linux distributor, developer, and user would become copyright infringers if they did not pay a licensing fee.[36] The court ultimately determined that 326 lines of code in the Linux kernel were potentially infringing.[37] Worse still, for the proprietary arena, if the viral taint of the OSS license is applicable, it could well mean disclosing the code to the community at large.[38] This is what is speculated to have occurred in the case of Microsoft, where it was forced to contribute code (Hyper-V drivers) to Linux to avoid a GPL violation when it was discovered that Microsoft had incorporated GPL-licensed components into its Hyper-V drivers.[39]

### 3.2.1.3. Moral Rights

Moral Rights include the right of disclosure, the right of integrity, the right of withdrawal, and the right of paternity.[40] Moral rights are inalienable and cannot be assigned.[41] Moral rights are considered relevant with respect to OSS,[42] especially that of attribution and reputation, as largely the OSS approach is based on these rewards as opposed to monetary rewards.[43]

36. See *generally* Kerry D. Goettsch, *SCO Group v. IBM: The Future of Open-Source Software*, 2003 U. Ill. J.L. Tech. & Pol'y 581 (2003); David W. Opderbeck, *The Penguin's Paradox: The Political Economy of International Intellectual Property and The Paradox of Open Intellectual Property Models*, 18 Stan. L. & Pol'y Rev 101 (2007).
37. See Charles Babcock, *IBM Argues SCO's Case Comes Down To 326 Lines Of Code* (19 Mar. 2007) http://www.informationweek.com/news/198001720; Dan Goodin, *Only 326 lines of code said to be at issue in SCO-IBM flap* (17 Mar. 2007) http://www.theregister.co.uk/2007/03/17/sco_evidence_mountain/.
38. John Tsai, *For Better or Worse: Introducing the GNU General Public License Version 3*, 23 Berkeley Tech. LJ 547, 556 (2008) (incorporating GPLv2-licensed source code into the source code of a proprietary software product could potentially 'infect' the proprietary software, causing the proprietary source code to be automatically licensed under GPLv2).
39. See Wikipedia, *Linux Kernel*, http://en.wikipedia.org/wiki/Linux_kernel (accessed 12 Mar. 2014); Thom Holwerda, *Microsoft's Linux Kernel Code Drop Result of GPL Violation* (23 Jul. 2009) http://www.osnews.com/story/21882/Microsoft_s_Linux_Kernel_Code_ Drop_Result_of_GPL_Violation.
40. Street, *supra* n. 12, at Ch. 5 § 5.06.
41. Lucie Guibault and O. van Daalen, *Unravelling The Myth Around Open Source Licences: An Analysis From A Dutch And European Law Perspective* 122 (T.M.C. Asser Press, The Hague 2006) (Moral rights are recognized in Art. 25 of the Dutch Copyright Act. These rights are inalienable, in the sense that they remain with the author even after he has assigned his rights to another person).
42. Priti Suri and Associates, *Open Source and the Law* 93 (LexisNexis, India, 2006); Mikko Välimäki, *The Rise Of Open Source Licensing: A Challenge To The Use Of Intellectual Property In The Software Industry* 114 (Turre Publ., Helsinki 2005) (open source can be understood in fact to work largely based on the ideas of moral rights).
43. Axel Metzger and Till Jaeger, *Open Source Software and German Copyright Law*, 32 IIC 52 (2001) (The protagonists of open source software attach considerable importance to

Reputation represents the reward for the developer of open source code; hence, an applicability which violates such moral rights can attract strong reactions considering the sensitivity of damage to their reputation.[44]

While some countries recognize moral rights, others don't; some countries specifically deny moral rights protection to software authors,[45] or permit authors to contractually waive moral rights. Others curtail the moral rights available to software authors. Thus, moral rights in software can be attenuated in two ways- jurisdictionally, and by limits to the rights themselves.[46]

Legal systems that recognize moral rights may thus already provide a degree of latent alternative protection for the OSS approach.[47] Hence, any violation of the open source mandates could amount to violation of the creator's moral rights. Under German law, *Metzger* and *Jaeger* are of the belief that despite the broad scope of the release of OSS by the GPL, the modifier still runs the risk of a prohibition by the author based on section 14, German Copyright Act in exceptional circumstances- even if the authorized user enjoys a right to modify, the author retains a right to prevent distortions or impairments of his work.[48]

The OSS approach implements in a limited manner the spirit of moral rights; both GPLv2[49] and GPLv3[50] recognize the right of attribution subject to contractual provisions. Other licenses too carry similar provisions. Section 6 of the OSL specifically deals with attribution and is intended to protect the

the acknowledgment of authorship, particularly since the remuneration to act as stimulus for the further development of software is replaced in particular by the possibility of becoming well known thanks to a successful problem solution).

44. Suri and Associates, *supra* n. 42, at 59.
45. For example U.S.A.; see Street, *supra* n. 12, at Ch. 5 § 5.06; Paul Edward Geller (eds.) *International Copyright Law and Practice*, USA § 7 (Matthew Bender, New York 2009).
46. Vetter, *The Collaborative Integrity, supra* n. 10, at 664; for e.g., see §§ 52 and 57, Indian Copyright Act, 1957. Indian copyright law recognizes the right of paternity and integrity in context of computer programmes; however, the integrity right is curtailed in context of adaptations of computer programmes for interoperability and back-up copy purposes.
47. *Ibid.* at 571.
48. Metzger and Jaeger, *supra* n. 43.
49. § 1, GNU GPLv2 (You may copy and distribute 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 the notices that refer to this License); § 2(a), GNU GPLv2 (You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change).
50. § 5(a), GNU GPLv3 (The work must carry prominent notices stating that you modified it, and giving a relevant date); § 7(b), GNU GPLv3 (for material you add to a covered work, you may … supplement the terms of this License with terms … Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it); § 7(c), GNU GPLv3 (for material you add to a covered work, you may … supplement the terms of this License with terms … Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version).

reputations of contributors and distributors as their Original Works are copied, modified, and distributed by downstream licensees.[51] *Rosen* opines that there is an inherent conflict between the freedom of each licensee of OSS to change that work any way he pleases, and the desire of the author of the original code to have his creative work properly attributed to him by notices retained in that work.[52] However, the right of modification is certainly not a waiver of the attribution right.[53] In fact, an intentionally low quality modification of an OSS with many programming errors added could infringe the moral rights of the creator.[54]

### 3.2.1.4. Patent Protection

Software patents are a contentious issue. From the CONTU Software Subcommittee in the U.S.,[55] to the Proposed Directive in Europe,[56] opponents of software patents have been extremely vocal. The OSS group though late to arrive at the scene became one of the foremost opponents; their's comprised one of the loudest voices during the Patent Directive discussions in Europe.[57]

Software patents are viewed with a jaundiced eye by the OSS community.[58] *Stallman* elaborates that software patents are the software project equivalent of land mines: each design decision carries a risk of

51. Lawrence Rosen, *Open Source Licensing: Software Freedom and Intellectual Property Law*, 197 (Prentice Hall, New Jersey, 2005).
52. Lawrence Rosen, *OSL 3.0: A Better License for Open Source Software*, http://www.rosenlaw.com/OSL3.0-explained.pdf (accessed 12 Mar. 2014).
53. Guibault and van Daalen, *supra* n. 41, at 122–123 (we must disagree with some commentators who argue that the freedom granted under the GPL to modify the software could be interpreted as a waiver of the author's right of integrity).
54. Välimäki, *supra* n. 42, at 93; Guibault and van Daalen, *supra* n. 41, at 121 (Although the right of a software developer to object to the modification of his programme bears little significance in practice, the right to oppose any impairment of the work that is prejudicial to his name or reputation may still be relevant in the context of the development of OSS).
55. See John C. Phillips, *Sui Generis Intellectual Property Protection for Computer Software*, 60 Geo. Wash. L. Rev. 997, 1012 (1992).
56. The European Commission, Proposal for Directive on the Patentability of Computer-Implemented Invention (CII Directive -2002/ 0047/COD) (2002).
57. Eloise Gratton, *Should Patent Protection Be Considered for Computer Software-Related Innovations?* 7 Comp. L. Rev. & Tech. J. 223, 235 (2003) citing PbT Consultants Ltd., *The Results of the European Commission Consultation Exercise on the Patentability of Computer Implemented Inventions*, Final Report under contract number PRSW/2000/A0-7002/E/98 at 8 (2000), http://europa.eu.int/comm/internal market/en/indprop/comp/softanalyse.pdf; see also Grant C. Yang, *The Continuing Debate of Software Patents and the Open Source Movement*, 13 Tex. Intell. Prop. LJ 171, 203 (2005) (Open source groups, such as Eurolinux, are the loudest voice against software patents).
58. Douglas A. Hass, *A Gentlemen's Agreement Assessing the GNU General Public License and Its Adaptation to Linux*, 6 Chi.-Kent J. Intell. Prop. 213, 274 (2007) (The open source community has traditionally demonized software patents); David S. Evans and Anne Layne-Farrar, *Software Patents and Open Source: The Battle Over Intellectual*

stepping on a patent, which can destroy the project.[59] The arguments against software patents, range from the procedural to the theoretical.[60]

The procedural arguments vilify the patent office's patent grants and procedure in computer software. They challenge the prolonged patent term,[61] relaxed standards of non-obviousness for building-block programs,[62] opaque prosecution process,[63] non-disclosure of source code,[64] abuse of continuation filings[65] etc. These OSS concerns emanate more from ineffective use and regulation of the patent system coupled with procedural issues in context of software patents, rather than inapplicability of the patent system to software. Possible rectification of the patent system addressing these concerns may address the OSS proponents concerns. Instead of complete radical substitution, mere supplementation might do the trick.

The theoretical arguments, perceive software patents as the very antithesis of innovation by creating a 'thicket' – 'an "anticommons" by which large numbers of "building block" programs have become legally

*Property Rights*, 9 Va. J.L. & Tech. 10 (2004) (Patents are particularly insidious because they prevent open-source software programmers from doing what they want).

59. Richard Stallman, *Fighting Software Patents – Singly and Together*, http://www.gnu.org/philosophy/fighting-software-patents.html (accessed 12 Mar. 2014); Evans and Layne-Farrar, *supra* n. 58 (The newest front of the war being waged by the hard-line open-source software proponents is over software patents); see also Daniel Lin, Matthew Sag, and Ronald S. Laurie, *Source Code Versus Object Code: Patent Implications for the Open Source Community*, 18 Santa Clara Computer & High Tech. LJ 235, 236–237 (2002); Stephen M. McJohn, *The Paradoxes of Free Software*, 9 Geo. Mason L. Rev. 25 (2000).
60. See Evans and Layne-Farrar, *supra* n. 58; John R. Allison and Ronald J. Mann, *The Disputed Quality of Software Patents*, 85 Wash. U. L. Rev. 297, 299 (2007) (despite the widespread use of patenting in the modern software industry, the concerns about software patents have continued); see Gratton, *supra* n. 57, at 230 regarding discussion on patents being anti open source; patents representing a threat to interoperability and standards; patents promoting low quality of patents; and, while favouring large organizations, patents have a negative effect on the economy and innovations.
61. Kirk D. Rowe, *Why Pay for What's Free?: Minimizing the Patent Threat to Free and Open Source Software*, 7 J. Marshall Rev. Intell. Prop. L. 595, 617 (2008) (The fact that an entire generation of programs can cycle within a matter of months, in the continuous process of software development, supports the argument that the term for software patents should be shortened).
62. See Evans and Layne-Farrar, *supra* n. 58.
63. Jack George Abid, *Software Patents on Both Sides of The Atlantic*, 23 J. Marshall J. Computer & Info. L. 815, 835 (2005) (These improperly issued software patents are the result of defective PTO examining procedure).
64. See Rowe, *supra* n. 61.
65. Stuart J. H. Graham and David C. Mowery, *The Use of USPTO 'Continuation' Applications in the Patenting of Software: Implications for Free and Open Source*, 27 Law & Pol'y 128 (2005) (strategic 'hold-up' of software adopters and follow-on software innovators … disclosure may be delayed substantially by … 'continuation patent application,' thus eroding the disclosure benefit to open-source developers by creating an IPR that has a pernicious effect in that it offers both monopoly and secrecy).

inaccessible'.[66] While the proprietary model has alternatives such as cross-licensing to navigate this thicket, OSS program's very nature precludes it from these options. OSS proponents seek to revisit patent jurisprudence in context of software programs altogether, citing it as an undeserved reward,[67] and disputing the traditional incentive structure:

> To what extent are the incentives provided by intellectual property protection necessary to promote progress and innovation in the field of computer software development? … the relative success of the open source approach reveals that exclusive intellectual property protection of software as an incentive to promote progress and innovation is unnecessary … under an open source approach, the benefits of collaboration more than make up for any lost economic incentives.[68]

66. Rowe, *supra* n. 61, at 607 citing Ronald J. Mann, *Do Patents Facilitate Financing in the Software Industry?*, 83 Tex. L. Rev. 961, 999 (2005); Deborah Azar, *A Method to Protect Computer Programs: The Integration of Copyright, Trade Secrets, and Anticircumvention Measures*, 2008 Utah L. Rev. 1395, 1400 (2008) (Software cannot be accessed by future generations to build upon it. This is an unfortunate property of copyright that slows down the development of new and better software); Mark A. Lemley & David W. O'Brien, *Encouraging Software Reuse*, 49 Stan. L. Rev. 255, 260 (January 1997) (Software reinvention unnecessarily incurs substantial costs by duplicating effort and failing to incorporate the lessons of past mistakes and inefficiencies into future software designs and implementations); Richard S. Gruner, *Better Living Through Software: Promoting Information Processing Advances Through Patent Incentives*, 74 St. John's L. Rev. 977, 981 (2000) (all patents governing intellectual constructs like software are undesirable because they restrict free access to fundamentally important modes of analysis and information processing tools); *contra* John M. Griem, Jr., *Against A Sui Generis System of Intellectual Property for Computer Software*, 22 Hofstra L. Rev. 145, 156 (1993–1994) (It is axiomatic that an invention becomes valuable and deserving of protection precisely because it eliminates a technological limitation in an art by the novel application of the laws of physics or mathematics); *contra* Jonathan M. Barnett, *Property as Process: How Innovation Markets Select Innovation Regimes*, 119 Yale LJ 384, 434 (December 2009) (Contrary to natural intuitions … the largest resource holders will tend to have the strongest incentives to relax intellectual property protections where incremental transaction costs and associated innovation losses race ahead of incremental innovation gains). But see Evans and Layne-Farrar, *supra* n. 58 (Other industries with longstanding histories of patenting could be categorized as having cumulative and sequential R&D, yet they do not display signs of innovation gridlock. Accordingly, it would seem that patent-induced anticommons are the proper subject of an empirical question, not a theoretical one).
67. Rowe, *supra* n. 61.
68. Matthew D. Stein, *Rethinking UCITA: Lessons From The Open Source Movement*, 58 Me. L. Rev. 157, 169 (2006); Evans and Layne-Farrar, *supra* n. 58 (By providing inventors a temporary right to exclude others from using their invention, patents allow inventors to appropriate the returns to their investment and thus provide incentives both to create more innovations and to make those innovations public. Software patent critics claim that this traditional argument does not apply to software creation, thus software patents are not justified).

Third, unlike copyright, obtaining patents is a tedious and expensive process; hence, a lot of developers would be reluctant to freely release patents to the community.

Moreover, critics question whether the promoters of truly open and mandatory improvement licensing, having spent most of their lives opposed to software patents, can actually attract donors of patents, or would actually participate in a process that they claim to despise.[60]

Furthermore, taking the example of GPLv3's attempt at universalization to achieve a greater market share, where developers shift from other licenses to it, might instead of reducing license proliferation have just the opposite effect. Several open source factionists are concerned regarding such a development and believe that the release of GPLv3 portends the balkanization of the entire open source universe.[61]

Thus, though certain successes have certainly been achieved, in motivating the industry towards patent sharing,[62] still, despite this initiative at correcting patent law in a similar manner as copyright law by the open source faction, it might be more difficult to achieve, and not gain a similar amount of popular support. *Asay* is of the view that with GPLv3's patent provisions, corporations unwilling to risk their patent portfolios may cut-off their input and contributions, and distributions will be forced to 'fork' software packages in order to maintain consistent and acceptable licensing schemes.[63]

60. Wikipedia, *Open Patent*, *supra* n. 2.
61. James E.J. Bottomley et al., *The Dangers and Problems with GPLv3* (2006), http://lwn.net/images/pdf/kernel_gplv3_position.pdf (accessed 12 Mar. 2014).
62. See Open Patents, http://p2pfoundation.net/Open_Patents (accessed 12 Mar. 2014) (On 12 Oct. 2001 the Free Software Foundation and Finite State Machine Labs Inc. (FSMLabs) announced a GPL – compliant open-patent license for FSMLabs' software patent, U.S. Patent No. 5,995,745. Titled the Open RTLinux patent license Version 2, it provides for usage of this patent in accordance with the GPL).
63. Asay, *supra* n. 4, at 296; see also James E.J. Bottomley et al., *supra* n. 61 (This Balkanisation, which will be manifested by distributions being forced to fork various packages in order to get consistent licences, has the potential to inflict massive collateral damage upon our entire ecosystem and jeopardize the very utility and survival of Open Source).

# Chapter 8 The Case for Interoperability

## 8.1 INTRODUCTION

The fair use or fair dealing exception under copyright law is an infringement defence.[1] A party reproducing copyrighted material is not liable for infringement if the reproduction constitutes a fair use.[2] In U.S. the copyright law recognizes fair use[3] and four non-exclusive factors are used to evaluate the 'fairness' of the use: (1) the purpose and character of the use; (2) the nature of the copyrighted work; (3) the amount and substantiality of the portion used in relation to the copyrighted work as a whole; and (4) the effect of the use on the potential market for or the value of the copyrighted work.[4] The

1. Gideon Parchomovsky & Philip J. Weiser, *Beyond Fair Use*, 96 Cornell L. Rev. 91, 92 (2010) (Fair use has long been a venerable part of our copyright system. As one court explained, the doctrine 'permits courts to avoid rigid application of the copyright statute when, on occasion, it would stifle the very creativity which that law is designed to foster'.); see also Michael J. Madison, *A Pattern-Oriented Approach to Fair Use*, 45 Wm and Mary L. Rev. 1525 (March 2004); Eric Allen Engle, *When is Fair Use Fair?: A Comparison of E.U. and U.S. Intellectual Property Law*, 15 Transnat'l Law. 187 (2002); Klaus Lodigkeit, *Intellectual Property Rights in Computer Programs in the USA and Germany*, 101 (Peter Lang, Frankfurt am Main 2006); David Bender, *Computer Law: A Guide to Cyberlaw and Data Privacy Law*, 1 § 9.01 (Matthew Bender, Rev. Ed. 2009).
2. F. Lawrence Street, *Law of the Internet* Ch. 5 § 5.05 (Matthew Bender, 2009).
3. U.S. Copyright Act, 17 U.S.C. § 107 (1976).
4. David Bender *supra* n. 1, at § 7.02; Also generally see David A. Rice, *SEGA And Beyond: A Beacon For Fair Use Analysis ... At Least As Far As It Goes*, 19 Dayton L. Rev. 1131 (1994); S. Carran Daughtrey, *Reverse Engineering of Software for Interoperability and Analysis*, 47 Vand. L. Rev. 145 (January 1994); Allan M. Soobert,

notion of transformative use too has been acknowledged.[5] An individual must possess an authorized copy of a literary work to engage in fair use copying.[6] Several jurisdictions follow the related concept of fair dealing.

A fair use defence allows reverse engineering and decompilation of copyright-protected computer programs to achieve interoperability in certain scenarios, the primary emphasis being on the fair use factors and when disassembly is the only option to gain access to ideas and functions embodied in a computer program and where there exists a legitimate reason for gaining access to those ideas and functions. This has been recognized both under U.S. law[7] and case laws.[8] Similar protections exist under EU law as well.[9]

Technological Protection Measures, technology tweaking and license agreements can be used to restrict reverse engineering.[10] These measures are

*Legitimizing Decompilation of Computer Software under Copyright Law: A Square Peg in Search of a Square Hole*, 28 J. Marshall L. Rev. 105 (1994); Donna L. Lee, *Reverse Engineering of Computer Programs Under the DMCA: Recognizing a 'Fair Access' Defense*, 10 Marq. Intell. Prop. L. Rev. 537 (2006).

5. See *Sony Computer Entertainment, Inc. v. Connectix Corp.*, 203 F.3d 596 (9th Cir. 2000)
6. *Sega Enterprises v. MAPHIA* 857 F. Supp. 679 (N.D. Cal. 1994).
7. U.S. Copyright Act, 17 U.S.C. § 117 (1976) (Limitations on exclusive rights: computer programs).
8. See *Sega Enterprises Ltd. v. Accolade, Inc.*, 977 F.2d 1510 (9th Cir. 1992) (decompilation could be a fair use provided there was no other way to obtain the requisite information and there was a legitimate reason for so doing); *DSC Communications Corp. v. DGI Technologies, Inc.*, 898 F. Supp. 1183 (N.D. Tex. 1995) (where there is a good reason for examining the unprotected aspects of a copyrighted program, disassembly will be a fair use); *Sony Computer Entertainment, Inc. v. Connectix Corp.*, 203 F.3d 596 (9th Cir. 2000) (If reverse engineering using disassembly of object code into source code is the only way to gain access to unprotected functional elements, and there is a legitimate reason for seeking such access, then, copying through disassembly will constitute fair use of the copyrighted work); *Atari Games Corp. v. Nintendo of America Inc.*, 975 F.2d 832 (Fed. Cir. 1992) (reverse engineering of unprotectable portions of a computer program would be a 'fair use' of the program. Beyond the copying necessary to understand the unprotected elements of the disputed program, any other copying would constitute copyright infringement).
9. See Art. 5, European Union, Council Directive 2001/29/EC, On the Harmonisation of Certain Aspects of Copyright and Related Rights in the Information Society, L 167/10 (22 Jun. 2001).
10. Street, *supra* n. 2, at Ch. 5 § 5.01 (A benefit of obtaining an enforceable license agreement is that it can restrict reverse engineering); Anthony J. Mahajan, *Intellectual Property, Contracts, and Reverse Engineering after PROCD: A Proposed Compromise for Computer Software*, 67 Fordham L. Rev. 3297, 3319 (1999) (Where reverse engineering is prohibited by contract, however, there is no similar exception to an action for breach of contract brought against one who reverse engineers while contractually bound. Accordingly, private contract is the overriding force that allows private right owners to maintain control over the balance between public and private rights); see also Diane Rowland and Andrew Campbell, *Supply of Software: Copyright and Contract Issues, International Journal of Law and Information Technology*, 10(I) Oxford University Press (2002).

reinforced through international treaties viz. The WIPO Copyright Treaty, 1996 and legislations such as the U.S. DMCA, E.U. Computer Programs Directive[11] and E.U. Copyright Directive.[12] Analogously, software patents may hinder interoperability by allowing cordoning of file formats, network protocols, interfaces etc.

Several experts have been extremely vocal against this legislative-private governance model.[13] 'Caught up in the tension between the content distributors and changing public expectations, the anticircumvention provisions became a social lightning rod in the clash of ideologies and expectations'.[14]

The incremental nature of programming in the software industry mandates interoperability for best efficacy. Software patents impede this strata creation thus forcing developers to deviate from actual innovation and improvements to designing around. However, in the software industry, designing around and ensuring interoperability is a formidable, if not an impossible task.

As with reciprocal obligation, so also with technological protection measures, the open source movement has attempted to invert the application of traditional restrictions. GPLv3 proponents believe that GPLv3's anti-DRM provisions are crucial for curtailing increasing use of para-copyright measures; otherwise, it would increasingly erode free software's freedoms, and end its dynamic development model.

11. European Union, Council Directive 91/250/EEC Legal Protection of Computer Programs, OJ L 122, (17 May 1991) replaced by European Union, Council Directive 2009/24/EC Legal Protection of Computer Programs, L 111/16 (5 May 2009).
12. European Union, Council Directive 2001/29/EC, On the Harmonisation of Certain Aspects of Copyright and Related Rights in the Information Society, L 167/10 (22 Jun. 2001).
13. See Rod Dixon, *Breaking into Locked Rooms to Access Computer Source Code: Does the DMCA Violate a Constitutional Mandate When Technological Barriers of Access Are Applied to Software?*, 8 Va. J.L. & Tech. 2, 3 (2003) (aspects of computer software, particularly the source code, rarely should be regarded as a category of expression created as a result of independent and, hence, original authorship … approval of locking up access to source code regardless of whether the source code meets the originality requirement may violate copyright's constitutional mandate under circumstances where the technological barrier protects an unoriginal work); see also Mahajan, *supra* n. 10, at 3318 (licenses, as the means by which private prohibitions against reverse engineering are enforced, allow private owners to overrule case law providing that reverse engineering is a permissible means of promoting competition and compatibility … Without a means, such as the right to reverse engineer, to influence this private determination, the public sphere inevitably shrinks).
14. Jon M. Garon, *What Ifs and Other Alternative Intellectual Property And Cyberlaw Story: What If DRM Fails?: Seeking Patronage in The Iwasteland and The Virtual O* 2008 Mich. St. L. Rev. 103, 107 (2008).

## 8.2. THE INTEROPERABILITY SCENARIO: ANGST

This part focuses on the various factors that impact interoperability specifically from the perspective of open source software, viz. the technology, the law and the license.

### 8.2.1. TECHNOLOGICAL PROTECTION MEASURES

Digital rights management (DRM) is not a term of art and can be defined in various ways.[15] The term commonly refers to a system through which content is made available to users in electronic form, pursuant to conditions (such as payment, extent of access or copying) established by the content owner; most DRM systems employ some form of technological controls to prevent unauthorized access to and use of works they contain.[16]

Devices for technological protection measures can primarily be classified under two types – access control[17] and rights control.[18] Access controls can control access, its duration and its frequency. Examples include password protection, IP address controls, scrambling of satellite signals. Examples of copy-controls include controls attached to movies or music DVDs, Serial Copy Management System (SCMS) etc.

Various experts are of the view that DRMs support the traditional model of software development, at the expense of the open source model, by limiting the ability of open source software developers to write programs that increase interoperability and by limiting their ability to engage in peer review.[19] The open source movement and especially FSF oppose the use of digital controls on ideological grounds. GNU relabelled DRM as 'digital

15. June M. Besek, *Anti-Circumvention Laws and Copyright: A Report from the Kernochan Center for Law, Media and the Arts*, 27 Colum. J.L. & Arts 385, 451-2 (2004).
16. *Ibid.* at 452.
17. Michael J. Madison, *Reconstructing the Software License*, 35 Loy. U. Chi. L.J. 275, 289 (2003) ('Access' control technology governs obtaining rights to look at, listen to, or otherwise use the work in the first place. Access control technology receives greater protection under the DMCA than rights control technology); Besek, *supra* n. 15, at 450 (Access controls are measures that prevent someone from viewing, reading, hearing and/or otherwise perceiving the work without authorization from the rightholder).
18. Madison, *supra* n. 17 at 289 ('Rights' control technology governs what the user may do with the work once access is properly obtained); Besek, *supra* n. 15, at 450 (Use controls are technological measures that limit whether and to what extent a work can be copied, communicated, viewed or played).
19. Theodore C. McCullough, *Understanding the Impact of The Digital Millennium Copyright Act on the Open Source Model of Software Development*, 6 Marq. Intell. Prop. L. Rev. 91, 92 (2002).

handcuffs'.[20] According to FSF, technical restrictions that allow other parties to control the user have no legitimate social purpose.[21]

### 8.2.2. TRUSTED COMPUTING

Trusted Computing is a concept that promotes the idea of a secure computing environment founding on hardware based cryptography.[22] It provides a computing platform on which you can't tamper with the application software, and where these applications can communicate securely with their authors and with each other.[23] Trusted Systems are a form of sophisticated

20. The Free Software Foundation, *Words to Avoid (or Use with Care) Because They Are Loaded or Confusing*, https://www.gnu.org/philosophy/words-to-avoid.html (accessed 12 Mar. 2014).
21. See Free Software Foundation, *Opinion on Digital Restrictions Management*, http://gplv3.fsf.org/drm-dd2.pdf (accessed 12 Mar. 2014).
22. See Mark Dermot Ryan, *Trusted Computing: Concepts* (2008), http://www.cs.bham.ac.uk/~mdr/teaching/modules/security/lectures/TrustedComputingConcepts.html (accessed 12 Mar. 2014) (Trusted Computing is a cluster of proposals and ideas for a locked-down PC architecture which can give guarantees about the application software it is running, and which allows applications to communicate securely with other applications and with servers. In its strongest form, pathways between the computer and peripherals such as the keyboard, mouse and monitor are encrypted. The encryption keys are built into the hardware, and are not available to the PC owner. The PC only runs the operating system if it can verify its identity and integrity, and the operating system communicates securely with remote servers to provide guarantees about identity and integrity of application software before running it (attestation)); see also Wikipedia, *Trusted Computing*, http://en.wikipedia.org/wiki/Trusted_Computing (accessed 12 Mar. 2014) (With Trusted Computing, the computer will consistently behave in expected ways, and those behaviours will be enforced by hardware and software. In practice, Trusted Computing uses cryptography to help enforce a selected behaviour. The main functionality of TC is to allow someone else to verify that only authorized code runs on a system. This authorization covers initial booting and kernel and may also cover applications and various scripts); see also Ross Anderson, *'Trusted Computing' Frequently Asked Questions* (2003) http://www.cl.cam.ac.uk/~rja14/tcpa-faq.html (accessed 12 Mar. 2014) (TC provides a computing platform on which you can't tamper with the application software, and where these applications can communicate securely with their authors and with each other); Mark Gimbel, *Some Thoughts on the Implications of Trusted Systems for Intellectual Property Law*, 50 Stanford L. Rev. 1671, 1675 (1998) (a trusted system is a piece of software or hardware that 'can be relied upon to follow certain rules'. In the digital publishing context, a trusted system is a system that will follow the instructions attached to a digital work: specifications written in what is known as a digital rights language).
23. Ross Anderson, *supra* n. 22; see also Wikipedia, Trusted Computing, *supra* n. 22 (Trusted Computing encompasses six key technology concepts, of which all are required for a fully Trusted system, that is, a system compliant to the TCG specifications: Endorsement key, Secure input and output, Memory curtaining / protected execution, Sealed storage, Remote attestation, Trusted Third Party (TTP)).

DRM systems which employ persistent protection and allow transmissions only to particular devices or for particular uses.[24]

The Trusted Computing Group (TCG), successor to the Trusted Computing Platform Alliance (TCPA), is an initiative started by AMD, Hewlett-Packard, IBM, Intel, and Microsoft to implement Trusted Computing.[25] Currently, its members comprise of all the major players in the software and hardware industry.[26] At present, prominent use of Trusted Computing is seen in hardware manufactured by Dell,[27] Lenovo,[28] HP,[29] Sony,[30] and other market leaders; and software manufactured by Microsoft viz. Windows Vista[31] and Windows 7.[32] Many different types of trusted systems are possible: trusted audio or video players, trusted servers, for distributing works over the Internet, and trusted printers.[33]

Trusted Computing has been the subject of great criticism by the open source faction because the architecture it establishes essentially shifts the ultimate control of the computer from the user to the software, and hence the software owner. Treacherous Computing is a corruption of the term 'Trusted Computing' by *Richard Stallman*.[34] Technically, through trusted computing, the computer is secured against all parties, including even its user. Altering the software or hardware in any manner, would potentially block users from accessing their computer. Moreover, the use of hardware based cryptography

24. Besek, *supra* n. 15, at 452.
25. See Trusted Computing Group, http://www.trustedcomputinggroup.org/ (accessed 12 Mar. 2014).
26. *Ibid.*
27. Dell, http://support.dell.com/support/topics/global.aspx/support/kcs/document?c=us&l=en&s=gen&docid=DSN_E75E35123E8AC4D0E030030ABD623A10&isLegacy=true (accessed 12 Mar. 2014).
28. Lenovo, http://www.pc.ibm.com/us/think/thinkvantagetech/security.html (accessed 12 Mar. 2014).
29. HP, http://h20331.www2.hp.com/hpsub/cache/292199-0-0-225-121.html (accessed 12 Mar. 2014).
30. Sony, http://www.sonystyle.com.hk/ss/vaio/product/vgn_tx57gn_b/security.html (accessed 12 Mar. 2014).
31. Windows Vista, http://technet.microsoft.com/en-us/library/cc766159%28WS.10%29.aspx (accessed 12 Mar. 2014).
32. Windows 7, http://technet.microsoft.com/en-us/library/dd835565%28WS.10%29.aspx (accessed 12 Mar. 2014).
33. Mark Gimbel, *supra* n. 22, at 1675.
34. Richard Stallman, *Can You Trust Your Computer?* (2002) http://www.gnu.org/philosophy/can-you-trust.html (accessed 12 Mar. 2014) ('Treacherous computing' is a more appropriate name, because the plan is designed to make sure your computer will systematically disobey you. In fact, it is designed to stop your computer from functioning as a general-purpose computer. Every operation may require explicit permission).

makes it extremely difficult for the user to access or substantially alter the functioning of the computer.[35] As *Anderson* states:

> [D]efinition of 'security' is controversial; machines built according to their specification will be more trustworthy from the point of view of software vendors and the content industry, but will be less trustworthy from the point of view of their owners.[36]

*The Electronic Frontier Foundation* summarizes the situation thus:

> Changing hardware design isn't inherently suspicious, but the leading trusted computing proposals have a high cost: they provide security to users while giving third parties the power to enforce policies on users' computers against the users' wishes – they let others pressure you to hand some control over your PC to someone else. This is a 'feature' ready-made for abuse by software authors who want to anticompetitively choke off rival software.[37]

Hence, by use of Trusted Computing, software not endorsed by specific vendors could be locked out. This could include pirated software, unlicensed software, open source software and even proprietary software. However, there is also a legitimate concern that OS vendors could use these capabilities to restrict what software would load under their OS, thus hurting small software companies or open source providers.[38] It could eventually end up creating customer lock-in. Other than this there would be a restriction on sharing of files, music, games etc. Moreover, Trusted Computing has the

35. See Mark Dermot Ryan, *supra* n. 22 (At manufacture time, a root cryptographic key is generated and stored within the hardware. This key is never transmitted to any other component, and the hardware is designed in such a way that it is extremely difficult to retrieve the stored key by reverse engineering or any other method, even to the owner. Applications can pass data encrypted with this key to be decrypted by the hardware, but it will only do so under certain strict conditions. Specifically, decrypted data will only ever be passed to authenticated, trusted applications, and will only ever be stored in curtained memory, making it inaccessable to other applications and the operating system ... TC requires hardware support, to enable hardware encryption keys, memory curtaining, secure execution, and tamper resistance. At boot time, control can be given to a small ROM program which verifies the hash value of the operating system code before loading and running it. This assures that the operating system is as expected. It in turn can verify the hash of application programs, to check their trustworthiness before running them).
36. Ross Anderson, *supra* n. 22.
37. Electronic Frontier Foundation, *Trusted Computing: Promise and Risk* (2003), http://www.eff.org/wp/trusted-computing-promise-and-risk (accessed 12 Mar. 2014).
38. See Wikipedia, *Trusted Computing Group*, http://en.wikipedia.org/wiki/Trusted_Computing_Group (accessed 12 Mar. 2014) (What a TPM does provide in this case is the capability for the OS to lock software to specific machine configurations, meaning that 'hacked' versions of the OS designed to get around these restrictions would not work. While there is legitimate concern that OS vendors could use these capabilities to restrict what software would load under their OS (hurting small software companies or open source/shareware/freeware providers, and causing vendor lock-in for some data formats), no OS vendor has yet suggested that this is planned).

potential to eliminate fair use entirely, at least within the digital realm.[39] *Woodford* opines that when content becomes intelligent in a sense, as it will under the trusted computing regime, it is never turned over completely to a purchaser; all such restricted content effectively becomes licensed for limited use.[40] It can support remote censorship like applications may be designed to delete pirated music under remote control.[41] This business model is known as 'traitor tracing'. Implications of shifting of user's control can be seen where Microsoft filed a patent application for 'Trusted license removal in a content protection system or the like'; the patent's description manifests the industry's intent thus:

> Typically, a content owner distributing such digital content wishes to restrict what the user can do with such distributed digital content. For example, the content owner may wish to restrict the user from copying and re-distributing such content to a second user, or may wish to allow distributed digital content to be played only a limited number of times, only for a certain total time, only on a certain type of machine, only on a certain type of media player, only by a certain type of user, *etc.*[42]

Hence, to cater to the need that from time to time the user, the computing device, the trusted component, or another entity may wish to remove a license from use in connection therewith, Microsoft seeks to patent the said invention; the nature of restrictions envisaged by the patent are considerable, and non-exhaustive viz.:

> [T]he rules for rendering the content can specify whether the user has rights to so render based on any of several factors, including who the user is, where the user is located, what type of computing device or other playback device the user is using, what rendering application is calling the copy protection system, the date, the time, *etc.* In addition, the rules may limit rendering to a pre-determined number of plays, or pre-determined play time, for example.[43]

39. Chad Woodford, *Trusted Computing Or Big Brother? Putting The Rights Back In Digital Rights Management*, 75 U. Colo. L. Rev. 253, 285 (2004); Mark Gimbel, *supra* n. 22, at 1680 (many of the provisions of the Copyright Act attempt to strike a utilitarian balance by limiting the rights of authors. Predictably, authors have often tried to alter this balance by lobbying to remove these limitations on their rights. Although their efforts have met with some success, important restrictions on authorial rights remain in force. Trusted systems will enable authors to circumvent many of these restrictions, delivering a level of intellectual property protection that the law has been unwilling to provide).
40. Chad Woodford, *supra* n. 39 at 286.
41. Ross Anderson, *supra* n. 22.
42. Trusted license removal in a content protection system or the like, U.S. Patent Application No. 20060265334 (Filed 23 Nov. 2006), http://appft1.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PG01&p=1&u=%2Fnetahtml%2FPTO%2Fsrchnum.html&r=1&f=G&l=50&s1=%2220060265334%22.PGNR.&OS=DN/20060265334&RS=DN/20060265334.
43. *Ibid.*

That such an occurrence happening is not simply theoretical but feasible is evidenced where Amazon removed illegal copies of novels not only from its systems but also from customers' devices without any prior user intimation or consent.[44]

### 8.2.3. Technological Tweaking

Tivoization is a term, coined by *Richard Stallman* to refer to the practice, where devices utilize free software, and meet all the mandates of the license, the most important being providing source code.[45] However, the device hardware is designed in such a manner that it can only function with the originally installed software and none other.[46] The FSF is of the view that by doing this, the device manufacturers are following the free software philosophy only in letter and not in the spirit.[47] Critics regard it as a blatant

44. Matt Hamblen, Amazon CEO Apologizes for Removing Digital Books, (2009) http://www.computerworld.com/s/article/print/9135882/Amazon_CEO_apologizes_for_removing_digital_books (accessed 12 Mar. 2014) (Amazon removed the novels (illegally sold copies of George Orwell's 1984 and other novels) from its Kindle e-book store, as well as any digital trace of the books. That meant the e-books were stricken from users' digital lockers as well as Kindle devices … Amazon refunded the price of the books … and sent e-mails to its affected customers explaining that the deleted books had been added to its bookstore by a third party using Amazon's self-service platform. The third party did not have the rights to the books, Amazon said).
45. See Richard Stallman, *Why Upgrade to GPLv3* (2007) http://www.gnu.org/licenses/rms-why-gplv3.html (accessed 12 Mar. 2014); see also, Free Software Foundation, *iPhone Restricts Users, GPLv3 Frees Them* (2007) http://www.fsf.org/news/iphone-gplv3 (accessed 12 Mar. 2014).
46. See The Free Software Foundation, *Frequently Asked Questions about the GNU Licenses*, http://www.gnu.org/licenses/gpl-faq.html (accessed 12 Mar. 2014) (What is tivoization? How does GPLv3 prevent it? Some devices utilize free software that can be upgraded, but are designed so that users are not allowed to modify that software. There are lots of different ways to do this; for example, sometimes the hardware checksums the software that is installed, and shuts down if it doesn't match an expected signature. The manufacturers comply with GPLv2 by giving you the source code, but you still don't have the freedom to modify the software you're using. We call this practice tivoization; Rowan Wilson, *GPL v3 – What's New* (12 Sep. 2011) http://www.oss-watch.ac.uk/resources/gpl3final.xml (Named after the popular digital video recorder marketed by TiVo Inc, 'tivoisation' refers to the distribution of free software in a device which cannot execute modified versions); Emily Prudente, *Open Source or Open Season?: What Legal Professionals Need to Know About Open Source Software Before Dealing in Corporate Transactions and The Ramifications of GPLv3*, 2010 Syracuse Sci. & Tech. L. Rep. 79, 91 (2010) (Tivoization refers to the practice of employing software licensed under the GPL in a hardware device (such as the popular TiVo digital video recorder used to record television programs) thereby preventing users from sharing the underlying source code).
47. See Premble GNU GPLv3 (Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have

violation that undermines the open source philosophy's freedom to retain, share and allow access to source code.[48] *Linus Torvalds* on the other hand holds a divergent view.[49]

Though the term tivoization is used generically, it has its origin in a specific incident – the TiVo device, from which it derives its name. TiVo is a digital video recorder developed and marketed by TiVo, Inc. It allows a television viewer to record programs for later viewing, among other functions.[50]

TiVo's software incorporates the Linux kernel and GNU software, in addition to proprietary software.[51] The open source software used by TiVo is

designed this version of the GPL to prohibit the practice for those products); see also Richard Stallman, Why Upgrade to GPLv3, *supra* n. 45 (One major danger that GPLv3 will block is tivoization … The manufacturers of these computers take advantage of the freedom that free software provides, but they don't let you do likewise).

48. Prudente, *supra* n. 46, at 91.
49. See Daniel Lyons, *Torvalds on TiVo*, Forbes.com (2006), http://www.forbes.com/2006/03/09/torvalds-linux-licensing-cz_dl_0309torvalds2.html (accessed 12 Mar. 2014) (TiVo … the thing that clashes with some in the FOSS community is that they make it hard to upgrade their box with another version of Linux. Which I personally think is OK – they made the box, they choose how to upgrade it. I only care that they give the source code back, not that they make it easy, or necessarily even possible, to play with their hardware. Again, it's the 'reciprocity of source code' versus the 'freedom of software' thing); see also Stephen Shankland, *Torvalds says DRM isn't necessarily bad*, CNET News (2006), http://news.cnet.com/Torvalds-says-DRM-isnt-necessarily-bad/2100-7344_3-6034964.html#ixzz1 L0GiKUML (accessed 12 Mar. 2014) (Digital signatures and cryptography aren't just 'bad DRM'. They very much are 'good security' too); see also Linus Torvalds, *Dual-Licensing Linux Kernel with GPL V2 and GPL V3*, http://lkml.indiana.edu/hypermail/linux/kernel/ 0706.1/2214.html (accessed 12 Mar. 2014) ([Stallman] calls it 'tivoization', but that's a word he has made up, and a term I find offensive, so I don't choose to use it. It's offensive because Tivo never did anything wrong, and the FSF even acknowledged that. The fact that they do their hardware and have some DRM issues with the content producers and thus want to protect the integrity of that hardware).
50. See TiVo Service Agreementv20080801, http://www.tivo.com/abouttivo/policies/ tivoserviceagreement.html (accessed 12 Mar. 2014) (The TiVo Service. The TiVo service consists of program guide information and the following features: (a) Season Pass® recording – automatically finds and records every episode of a series all season long; (b) WishList® search – finds and records programs that feature your favourite actor, director, team or even topic; (c) Smart Recording – automatically detects program line-up changes for your cable/satellite provider and adjusts recording times so you don't have to worry about the details; (d) TiVo Suggestions – TiVo can be programmed to suggest and auto-record programs that may match your interests; and (e) Parental Controls – lock channels or set ratings limits based on content. Each of these features is part of the 'TiVo service'. The 'TiVo service' means these features and any additional features and functionality of the TiVo DVR that TiVo may, at its discretion and from time to time, offer).
51. See Questions about Linux – TiVo Operating System, http://support.tivo.com/app/answers/detail/a_id/346/kw/linux/session/L3NpZC9UcDhTSHdzaw%3D%3D (accessed 12 Mar. 2014) (The TiVo Digital Video Recorder (DVR) operating system is called Linux. Linux is written and distributed under the GNU General Public License).

licensed under GPLv2. Accordingly, complying with the mandates of GPLv2, TiVo makes available the source code of open source software components used in its device.[52] However, if under the open source software approach, the software is modified or enhancements are added to it, it ceases to function on TiVo. This happens because TiVo has incorporated digital signatures in its system. Hence, once modified, the digital signature of the modified software ceases to match with the digital signature in the hardware and is unable to function.

This strategy of installing DRMs was soon used by a number of other devices.[53]

## 8.2.4. Law and Reverse Engineering

The WIPO Copyright Treaty, 1996, part of the Internet Treaties constitute the first international attempt to prescribe minimum standards of legal protection for DRM technologies.[54] The essence of the treaty is articulated in Article 11[55] which seeks to prevent the circumvention of DRM systems, not mandate the inclusion of DRM in particular classes of technology.[56] The WCT provisions give substantial latitude to adhering countries in deciding how to implement the treaty.[57] Also, in the WCT, the term 'effective

52. See TiVo Service Agreementv20080801, *supra* n. 50 (Open Source Software. Certain components of the software for the TiVo DVR are subject to the GNU General Public License Version 2, or other so-called open source licenses ('Open Source Software') … In compliance with the terms of certain Open Source Software licenses like the GNU General Public License Version 2 ('GPLv2'), TiVo makes certain modifications to Open Source Software that TiVo uses, modifies and distributes pursuant to such licenses available to the public in source code form at www.tivo.com/source. You are free to use, modify and distribute Open Source Software so long as you comply with the terms of the relevant Open Source Software license. In particular, the GPLv2 is available in the product manual or at www.gnu.org/copyleft/gpl.html).
53. See Brett Smith, *It's Not Just TIVO Locking Down Their Hardware*, Free Software Foundation (2010) http://www.fsf.org/blogs/licensing/gplv3-lockdown (accessed 12 Mar. 2014) (When it comes to the embedded devices, I've noticed an upsetting trend: a lot of them are locked down … They take different strategies, too; some of them won't provide you with the tools necessary to build your own firmware, and other times the hardware checks the software for authorized signatures. The result is the same either way – they'll give you the source, but you can't actually modify the software you're running).
54. Dale Clapperton and Stephen Corone, *Locking in Customers, Locking out Competitors: Anti-circumvention Laws in Australia and Their Potential Effect on Competition in High Technology Markets*, 130 St Melbourne Univ. L. Rev. 657, 663 (2006).
55. *The WIPO Copyright Treaty,* opened for signature 20 Dec. 1996, S Treaty Doc No 105-17; 36 ILM 65 (entered into force 6 Mar. 2002) (Art. 11 (Obligations concerning Technological Measures. Contracting Parties shall provide adequate legal protection and effective legal remedies against the circumvention of effective technological measures that are used by authors in connection with the exercise of their rights under this Treaty or the Berne Convention and that restrict acts, in respect of their works, which are not authorized by the authors concerned or permitted by law).
56. Clapperton and Corone, *supra* n. 54, at 663.
57. Besek, *supra* n. 15, at 392.

technological measures' is not further defined, thus creating some uncertainty.[58] This leads to varied levels and approaches by various countries towards protection, some keeping the threshold of protection very high with minimal exemptions while others according a lower level of protection with more exemptions. Accordingly, while in some countries copyright holders acquire a greater level of control, in others the users do so. This inverse relationship creates a constant tussle between copyright holders and users to achieve a balance in their expectations.[59] *Besek* elaborates that the WCT sheds little light on how to achieve adequate protection while accommodating users' privileges.[60]

The United States implemented its anti-circumvention requirement under the WCT in § 1201 of Title I of the Digital Millennium Copyright Act, 1998 (DMCA).[61] § 1201 has three prongs. First, it prohibits circumventing a technological protection measure that effectively controls access to a copyrighted work.[62] However, U.S. law does not prohibit circumventing a technological measure that protects a right of a copyright owner (e.g., a copy-control). Second, it prohibits manufacturing or making available products or services for circumventing technological access controls. This provision essentially prohibits trafficking in devices that circumvent access controls. Products or services that have a commercially significant purpose or use other than to circumvent such controls and are designed and marketed for such other purposes, are permissible.[63] Third, it prohibits manufacturing or making available products or services for circumventing technological

58. Clapperton and Corone, *supra* n. 54, at 663.
59. Besek, *supra* n. 15, at 392 (Copyright owners believe strong protection – against both the distribution of devices and the act of circumvention – is essential … but strong protection for technological measures comes at a cost … [it] allows copyright owners to control all uses of their works and makes it difficult, if not impossible, to benefit from copyright exemptions and privileges).
60. *Ibid.*
61. See Esther C. Roditti, *Computer Contracts* 1 § 3.02 (Matthew Bender) (2009); Clapperton and Corone, *supra* n. 54; Besek, *supra* n. 15.
62. U.S. Copyright Act, 17 USC § 1201 (a)(1)(A) (1998).
63. *Ibid.* at § 1201(a)(2) (No person shall manufacture, import, offer to the public, provide, or otherwise traffic in any technology, product, service, device, component, or part thereof, that:
    (A) is primarily designed or produced for the purpose of circumventing a technological measure that effectively controls access to a work protected under this title;
    (B) has only limited commercially significant purpose or use other than to circumvent a technological measure that effectively controls access to a work protected under this title; or
    (C) is marketed by that person or another acting in concert with that person with that person's knowledge for use in circumventing a technological measure that effectively controls access to a work protected under this title).

measures that protect a right of a copyright owner.[64] This provision essentially prohibits trafficking in devices that circumvent rights controls. The DMCA policy essentially distinguishes between those who actually create a device designed to circumvent copyright protection, and those who traffic in such devices; the DMCA makes this distinction and applies sanctions accordingly.[65]

The DMCA recognizes that legitimate reasons exist for engaging in circumvention.[66] Consequently, it provides seven specific 'Safe Harbor' exceptions to its above-mentioned anti-circumvention requirements.[67] Essentially, the exceptions focus upon reverse engineering;[68] encryption research;

64. *Ibid.* at § 1201(b)(1) (No person shall manufacture, import, offer to the public, provide, or otherwise traffic in any technology, product, service, device, component, or part thereof, that:
    (A) is primarily designed or produced for the purpose of circumventing protection afforded by a technological measure that effectively protects a right of a copyright owner under this title in a work or a portion thereof;
    (B) has only limited commercially significant purpose or use other than to circumvent protection afforded by a technological measure that effectively protects a right of a copyright owner under this title in a work or a portion thereof; or
    (C) is marketed by that person or another acting in concert with that person with that person's knowledge for use in circumventing protection afforded by a technological measure that effectively protects a right of a copyright owner under this title in a work or a portion thereof.).
65. McCullough, *supra* n. 19, at 102.
66. Esther C. Roditti, *Computer Contracts* 1 § 5.01 (Matthew Bender, 2009).
67. *Ibid.*; Rod Dixon, Breaking into Locked Rooms, *supra* n. 13; McCullough, *supra* n. 19.
68. U.S. Copyright Act, 17 USC § 1201(f)(1)-(4) (1998) (Reverse engineering.
    (1) Notwithstanding the provisions of subsection (a)(1)(A), a person who has lawfully obtained the right to use a copy of a computer program may circumvent a technological measure that effectively controls access to a particular portion of that program for the sole purpose of identifying and analysing those elements of the program that are necessary to achieve interoperability of an independently created computer program with other programs, and that have not previously been readily available to the person engaging in the circumvention, to the extent any such acts of identification and analysis do not constitute infringement under this title.
    (2) Notwithstanding the provisions of subsections (a)(2) and (b), a person may develop and employ technological means to circumvent a technological measure, or to circumvent protection afforded by a technological measure, in order to enable the identification and analysis under paragraph (1), or for the purpose of enabling interoperability of an independently created computer program with other programs, if such means are necessary to achieve such interoperability, to the extent that doing so does not constitute infringement under this title.
    (3) The information acquired through the acts permitted under paragraph (1), and the means permitted under paragraph (2), may be made available to others if the person referred to in paragraph (1) or (2), as the case may be, provides such information or means solely for the purpose of enabling interoperability of an independently created computer program with other programs, and to the extent that doing so does not constitute infringement under this title or violate applicable law other than this section.

security testing; technologies involving personal privacy; nonprofit libraries, archives and educational institutions; law enforcement, intelligence and other government activities; and protection of minors.

Of these exceptions, the reverse engineering exception is the most debated. Circumvention and reverse engineering are allowed in the interest of interoperability, but are lawful only if the right to use the program copy has been lawfully obtained, the elements necessary to achieve interoperability are not easily or readily available, and the reverse engineering is lawful under copyright law.[69] Since the DMCA is silent on whether the reverse engineering exception applies to copy-controls, their circumvention is allowed, if the control is identifiable as such, for the purpose of reverse engineering and need not rely upon the DMCA's narrowly drawn statutory definition of reverse engineering.[70]

The scope of the Reverse Engineering Safe Harbor protection has been variously interpreted. One school is of the opinion that the person who creates a circumvention device for another's use is liable under § 1201(a)(2) of DMCA for violating the Ban on Trafficking.[71] The argument is based on that § 1201(f)(1) only provides a safe harbour for those who violate the Basic Provision of the DMCA; hence only the authors of a circumventing device can avail themselves of § 1202(f) protections.[72] Another school holds a contrary view arguing that § 1201(f)(3) of DMCA permits information acquired through reverse engineering to be made available to others only by the person who acquired the information.[73] Both the arguments variedly impact the open source software model. With the former argument, 'open source developers will be barred from providing services that increase the interoperability of programs utilizing encrypted materials'.[74] On the other hand, with the latter argument, 'an open source developer will be able to provide services to write such programs, but his or her ability to engage in

(4) For purposes of this subsection, the term 'interoperability' means the ability of computer programs to exchange information, and of such programs mutually to use the information which has been exchanged).

69. Esther C. Roditti, *Computer Contracts* 1 § 5.01 (Matthew Bender, 2009).
70. Rod Dixon, Breaking into Locked Rooms, *supra* n. 13, at 49.
71. McCullough, *supra* n. 19, at 105 analysing David Nimmer, *A Riff on Fair Use in the Digital Millennium Copyright Act*, 148 *U. Pa. L. Rev.* 673 (2000).
72. *Ibid.* at 106.
73. *Ibid.* analysing *Universal City Studios, Inc. v. Reimerdes*, 111 F. Supp 2d. 294 (S.D.N.Y. 2000) and *United States v. Elcom Ltd.*, No. CR-01-20138 (N.D. Cal. 28 Aug. 2001); see also Hammond et al., *The Anti-Circumvention Provision of The Digital Millennium Copyright Act*, 8 Texas Wesleyan L. Rev. 593 (2002); Stephen M Kramarsky, *Copyright Enforcement in the Internet Age: The Law and Technology of Digital Rights Management*, 11 Depaul J. Art. & Ent. L. 1 (2001).
74. *Ibid.* at 109.

peer review will be restricted'.[75] Thus, either option negatively affects the open source software model.[76]

Critics opine that the anti-circumvention provisions of the DMCA arguably go beyond the requirements of the Internet Treaties.[77] US copyright lobbyists sought implementation of a stronger level of protection with the belief that other countries would emulate their model.[78] However, this has not usually been the case.

In the E.U., the Computer Programs Directive[79] in 1991 laid down the provisions governing computer software. Barring contractual provisions to the contrary, as regards, computer programs, authorization of the right holder is not required for the translation, adaptation, arrangement and any other alteration of a computer program and the reproduction of the results thereof, where they are necessary for the use of the computer program by the lawful acquirer in accordance with its intended purpose, including for error correction.[80] Moreover, the person having a right to use a copy of a computer program can, without the authorization of the right holder, observe, study or test the functioning of the program in order to determine the ideas and principles which underlie any element of the program provided he does so while and being entitled to load, display, run, transmit or store the program.[81]

The European Union implemented the WCT via the Copyright Directive[82] in 2001. The Copyright Directive was implemented to supplement the Computer Programs Directive and was to be read without prejudice to its mandate, in particular, in context of technological measures used in context of computer programs.

The Copyright Directive requires EU Member States to provide adequate legal protection against the circumvention of any effective technological measures, which the person concerned carries out in the knowledge, or with reasonable grounds to know, that he or she is pursuing that objective.[83] Largely, the protections contained in the Copyright Directive are

75. *Ibid.* at 110.
76. *Ibid.*
77. Clapperton and Corone, *supra* n. 54, 657, 664.
78. *Ibid.*
79. European Union, Council Directive 91/250/EEC Legal Protection of Computer Programs, OJ L 122, (17 May 1991) replaced by European Union, Council Directive 2009/24/EC Legal Protection of Computer Programs, L 111/16 (5 May 2009) [Hereinafter, The EU Computer Programs Directive].
80. *Ibid.* at Art. 5(1).
81. *Ibid.* at Art. 5(3).
82. European Union, Council Directive 2001/29/EC, On the Harmonisation of Certain Aspects of Copyright and Related Rights in the Information Society, L 167/10 (22 Jun. 2001) [Hereinafter, The EU Copyright Directive].
83. *Ibid.* at Art. 6(1).

similar to the DMCA.[84] There is a prohibition against trafficking in circumvention devices and services.[85] As opposed to the DMCA, technological measures include access control, copy-control and other protection process, such as encryption and scrambling.[86]

In one respect, however, the EU Directive differs significantly from the DMCA.[87] As far as exemptions are concerned, although it does not contain any specific privilege to circumvent technological protection measures or to deal in circumvention devices. However, it requires Member States, in the absence of voluntary measures taken by right holders, including agreements between right holders and other parties concerned to take appropriate measures to ensure that rightholders make available to the beneficiary of certain exceptions or limitations, the means of benefiting from them, provided, the beneficiary has legal access to the protected work or subject-matter concerned.[88] Essentially, the exemptions entail options of reproduction in certain media for certain acts, ephemeral recordings, teaching purposes, non-commercial uses for people with disabilities, public security, and research into cryptography. Private use for non-commercial purpose too is allowed subject to fair compensation.

84. *Ibid.* at Art. 6(2) (Member States shall provide adequate legal protection against the manufacture, import, distribution, sale, rental, advertisement for sale or rental, or possession for commercial purposes of devices, products or components or the provision of services which:
   (a) are promoted, advertised or marketed for the purpose of circumvention of, or
   (b) have only a limited commercially significant purpose or use other than to circumvent, or
   (c) are primarily designed, produced, adapted or performed for the purpose of enabling or facilitating the circumvention of, any effective technological measures.
85. Klaus Lodigkeit, *supra* n. 1, at 97 (Reverse Engineering is not explicitly cited in s. 69 d of the German copyright act, but it is implied through its wording … In German law the lawful purchaser of a computer program has the right '*to examine the functioning of the program, to test the program, and to discover the ideas and the principles of the program without the consent of the owner of the right, when this is done through acts of loading, notifying, running, transmitting, or saving of the program*'.).
86. Article 6(3), The EU Copyright Directive, *supra* n. 82 (For the purposes of this Directive, the expression 'technological measures' means any technology, device or component that, in the normal course of its operation, is designed to prevent or restrict acts, in respect of works or other subject-matter, which are not authorized by the rightholder of any copyright or any right related to copyright as provided for by law or the sui generis right provided for in Chapter III of Directive 96/9/EC. Technological measures shall be deemed 'effective' where the use of a protected work or other subject-matter is controlled by the rightholders through application of an access control or protection process, such as encryption, scrambling or other transformation of the work or other subject-matter or a copy-control mechanism, which achieves the protection objective).
87. Besek, *supra* n. 15.
88. Article 6(4), The EU Copyright Directive, *supra* n. 82.

Importantly, the Directive makes it clear that it shall leave intact and shall in no way affect existing Community provisions relating to the legal protection of computer programs.[89]

The fair use doctrine developed in an analog world and was not geared for the digital-networked environment that the Internet fuels.[90] Critics opine that there is no perfect solution to the problem of protecting copyrighted works in the digital environment.[91] Seeking to tilt the balance in their favour, as regards the digital world, copyright holders, leveraged a combination of technological and legal means. Hence effectively, the DMCA does not provide for fair use per se as an exception. There have been many criticisms about anti-circumvention legislation, but the overarching concern is that it prevents legitimate uses of copyrighted works.[92] *Madison*, states:

> The DMCA states that 'nothing in this section shall affect rights, remedies, limitations, or defenses to copyright infringement, including fair use, under this title'; but that section has been interpreted as not affording a 'fair use' defense to defendants accused of violating the DMCA. The DMCA ratifies precisely the kind of soup-to-nuts regulatory scheme offered by the software license, effected by control of the artifact as well as control over use of the work of authorship, now encoded in DRM and other technological systems.[93]

Critics argue against provisions protecting works against unauthorized access, citing them as private legislation with inadequate protections.[94]

89. *Ibid.* at Art. 1(2)(a).
90. Parchomovsky & Weiser, *supra* n. 1, at 101.
91. Besek, *supra* n. 15, at 512 (requiring that users have unencumbered ability to use works diminishes the protection for copyrighted works and the ways in which they are made available to consumers in the digital environment); Rowland and Campbell, *supra* n. 10, at 35 (Many software producers would like to be able to control the extent to which, if at all, their software could be decompiled or reverse engineered for other purposes, whether for the production of an interoperable or competing program, whilst users and competitors would like to enjoy the freedom to engage in such activities).
92. Besek, *supra* n. 15, at 394.
93. Madison, Reconstructing the Software License, *supra* n. 17, at 290; *but* see *Storage Tech. Corp. v. Custom Hardware Eng'g & Consulting, Inc.*, 431 F.3d 1374 (Fed. Cir. 2005); *Chamberlain Group, Inc. v. Skylink Techs.*, Inc., 381 F.3d 1178 (Fed. Cir. 2004); *Lexmark Int'l, Inc. v. Static Control Components, Inc.*, 387 F.3d 522 (6th Cir. 2004); see also Timothy K. Armstrong, 'Fair Circumvention', 74 *Brooklyn L. Rev.* 1 (2008).
94. Besek, *supra* n. 15, at 394 (With respect to provisions that protect works against unauthorized access, critics argue that (1) they, in effect, create a new copyright right without the exemptions and limitations that attach to the other rights, and (2) the exemptions in the statute are inadequate); Rod Dixon, *Breaking into Locked Rooms*, *supra* n. 13, at 3 (The DMCA's ostensible approval of locking up access to source code regardless of whether the source code meets the originality requirement may violate copyright's constitutional mandate under circumstances where the technological barrier protects an unoriginal work).

Chapter 12

# Software Protection: The Sui Generis Option

## 12.1. INTRODUCTION

Debate has been raging regarding intellectual property and computer software protection. Suggestions vary from application to non-application of property rights; from correct interpretation, to amendment to outright replacement of property laws with alternative models of software protection. Since 1960s onwards and especially in the 1970s and 1980s there has been extensive discussion on the mode of intellectual property protection for software – copyright, patent, or sui generis. The industry relied primarily on trade secret law and contract law as the mode of software protection initially. Eventually, copyright law was chosen legislatively as the vehicle for software protection. Judicial recognition soon allowed patent protection as well. Thereafter, industry started recognizing technological protection measures as another option and their efficacy was strengthened by legislative and judicial support.

Sui generis proposals have emanated from various arenas viz. academicians, industry, international bodies and government policy initiatives. The sui generis approaches essentially emphasize on attaining a balance between the rights granted by copyright and patent laws. Concurrently, several theories have posited against the adoption of a sui generis model and favoured accommodation of computer software within the ambit of existing intellectual property protection framework.

Several occurrences in the past three decades highlight the developing fissures in the mode of software protection viz. the nature of the software industry and the manner in which it is developing, the increasingly diversified protection of software under law, and the evolving landscape of law and its interaction with technology.

At the same time, certain other developments highlight the increasing acceptance and ease in implementing sui generis protection viz. the increasing sui generis legal recognition being granted to evolving technological arenas and the positive move towards international harmonization in intellectual property laws

Section 12.2 of this chapter generally focuses on the evolution of legal protection of software and Section 12.3 posits a need for a sui generis protection regime. It generally analyses the evolving nature of the software industry, fissures in the existing legal protection, the evolving landscape of technology-law interaction, recognition of evolving technological arenas, and increasing attempts at international harmonization. In light of the earlier discussion, Section 12.4 analyses the evolving property rights jurisprudence.

## 12.2. LEGAL PROTECTION OF SOFTWARE

Software development in the initial phase was usually on an individual scale and not mass-produced. Trade secrets have traditionally been used to protect software. Maintenance of trade secrets is ensured through several avenues such as employee non-disclosure agreements, technological measures, restrictions on distribution etc. Public distribution of goods does not imply a forfeiture of trade secret. *Mahajan* elaborates that as object code can be read only by a machine, the distribution of a program in object code form preserves the distributor's right under trade secret laws to claim misappropriation of the program's source code.[1] Licensing of software with the requisite prohibitions against reverse engineering also assist in ensuring preservation of trade secrets.

Along with trade secrets, contract law has been the oldest form of protection for software. In the initial phases individual contracts used to be tailored. With the evolution and widespread usage of computer technology, the licensing scheme adjusted to mass-market software licenses. From individually tailored contracts or licenses, the licensing scheme graduated to shrink-wrap, click wrap and browse wrap licenses. Licenses supplement intellectual property laws and allow regulation of rights granted under intellectual property laws to suit private interests.

1960s onwards, copyright law too began to be used to protect software.[2] 1970s saw discussions emerge as regards software patents[3] and sui generis proposals. During this period, copyright law was in favour as against patent law or sui generis proposals for software protection.

In the international arena, WIPO made sui generis proposals for software protection.[4] Several academic studies too emerged as regards software protection. *Galbi* was a leading proponent of sui generis software protection in the 1970s.[5] After a brief lull, the late 1980s and early 1990s saw extensive academic sui generis discussions. *Menell*,[6] *Samuelson*,[7] *Stern*[8] and

1. Anthony J. Mahajan, *Intellectual Property, Contracts, and Reverse Engineering after ProCD: A Proposed Compromise for Computer Software*, 67 Fordham L. Rev. 3297, 3308 (May, 1999).

2. See *generally* Jack M. Haynes, *Computer Software: Intellectual Property Protection in The United States and Japan*, 13 J. Marshall J. Computer & Info. L. 245 (1995).
3. See *generally* Nobuhiro Nakayama, *Legal Protection of Software (Sofutowea no hoteki hogo)*, 9-14, Ch. 1, Sect. 2, Legislative History (New ed. 1988).
4. See Model Provisions on the Protection of Computer Software, WIPO No. 814 (E) (1978) [Hereinafter WIPO Model Provisions, 1978]; Draft Treaty for the Protection of Computer Software, WIPO, LPCS/II/3 (1983) [Hereinafter WIPO Draft Treaty, 1983].
5. Galbi, *Proposal for New Legislation to Protect Computer Programming*, 17 (4) Bulletin of the Copyright Society of the U.S.A. 280 (1970); Galbi, *The Prospect of Future Legislation and Court Action Concerning the Protection of Programming*, 13 Jurimetrics Journal 234 (1973).
6. Peter S. Menell, *Tailoring Legal Protection for Computer Software*, 39 Stan. L. Rev. 1329 (July 1987); Peter S. Menell, *The Challenges of Reforming Intellectual Property Protection for Computer Software*, 94 Colum. L. Rev. 2644 (1994).
7. Pamela Samuelson, *CONTU Revisited: The Case Against Copyright Protection for Computer Programs in Machine-Readable Form*, 1984 Duke LJ 663 (1984); Pamela Samuelson, *Creating A New Kind of Intellectual Property: Applying the Lessons of The Chip Law to Computer Programs*, 70 Minn. L. Rev. 471 (December 1985); Pamela Samuelson, *Benson Revisited: The Case Against Patent Protection for Algorithms and other Computer Program-Related Inventions*, 39 Emory LJ 1025 (1990); Pamela Samuelson, Randall Davis, Mitchell D. Kapor & J.H. Reichman, *A Manifesto Concerning the Legal Protection of Computer Programs*, 94 Colum. L. Rev. 2308 (December 1994).
8. Richard H. Stern, *The Bundle of Rights Suited to New Technology*, 47 U. Pitt. L. Rev. 1229 (1986); Richard H. Stern, *A Sui Generis Utility Model Law as An Alternative Legal Model for Protecting Software*, 1 U. Balt. Intell. Prop. LJ 108 (1993); Richard H. Stern, *Solving The Algorithm Conundrum: After 1994 in The Federal Circuit Patent Law Needs A Radical Algorithmectomy*, 22 AIPLA Q.J. 167 (1994); Richard H. Stern, *On Defining The Concept of Infringement of Intellectual Property Rights in Algorithms and other Abstract Computer-Related Ideas*, 23 AIPLA Q. J. 401 (1995).

*Karjala*[9] have done extensive work in this field amongst several other academics.[10]

In the U.S., the Commission on New Technological Uses of Copyrighted Works (CONTU) eventually advised that copyright law be used to protect software leaving the scope of such protection to judicial determination.[11] CONTU also did not eliminate the industry's reliance on other modes of software protection.[12] Pursuant to this, the Copyright law in the U.S. was amended in 1980 to explicitly include computer programs.[13]

In Japan, The Ministry of International Trade and Industry (MITI) Study Committee on Legal Protection of Software issued an interim report in 1972 and found the copyright protection afforded to software as inadequate[14] and made certain proposals for the model for appropriate protection of

9. Dennis S. Karjala, *Lessons from the Computer Software Protection Debate in Japan*, 1984 Ariz. St. LJ 53 (1984); Dennis S. Karjala, *The Limitations on the Protection of Program Works Under Japanese Copyright Law*, 8 Mich. YBI Legal Stud. 25 (1987); Dennis S. Karjala, *Copyright, Computer Software, and The New Protectionism*, 28 Jurimetrics Journal 33 (1987); Dennis S. Karjala, *The Protection of Operating Software under Japanese Copyright Law*, 29 Jurimetrics Journal 43 (1988); Dennis S. Karjala, *Copyright Protection of Computer Documents, Reverse Engineering, and Professor Miller*, 19 Dayton L. Rev. 975 (1994); Dennis S. Karjala, *Misappropriation as A Third Intellectual Property Paradigm*, 94 Colum. L. Rev. 2594 (Decemeber 1994); Dennis S. Karjala, *A Coherent Theory for The Copyright Protection of Computer Software and Recent Judicial Interpretations*, 66 U. Cin. L. Rev. 53 (1997); Dennis S. Karjala, *Copyright Protection of Computer Program Structure*, 64 Brooklyn L. Rev. 519 (1998); Dennis S. Karjala, *The Relative Roles of Patent and Copyright in The Protection Of Computer Programs*, 17 J. Marshall J. Computer & Info. L. 41 (1998); Dennis S. Karjala, *Copyright Protection of Operating Software, Copyright Misuse, and Antitrust*, 9 Cornell J. L. & Pub. Pol'y 161 (1999); Dennis S. Karjala, *Distinguishing Patent and Copyright Subject Matter*, 35 Conn. L. Rev. 439 (2003).
10. Duncan M. Davidson, *The Future of Software Protection: Common Law, Uncommon Software*, 47 U. Pitt. L. Rev. 1037 (1986); Max W. Laun, *Improving the International Framework for the Protection of Computer Software*, 48 U. Pitt. L. Rev. 1151 (1987); Rochelle Cooper Dreyfuss, *Information Products: A Challenge to Intellectual Property Theory*, 20 N.Y.U. J. Int'l L. & Pol. 897 (1988); Tohru Nakajima, *Legal Protection of Computer Programs in Japan: The Conflict Between Economic and Artistic Goals*, 27 Colum. J. Transnat'l L. 143 (1988–1989); Glynn S. Lunney, Jr., *Copyright Protection for ASIC Gate Configurations: PLDS, Custom and Semicustom Chips*, 42 Stan. L. Rev. 163 (November 1989); John C. Phillips, *Sui Generis Intellectual Property Protection for Computer Software*, 60 Geo. Wash. L. Rev. 997 (1992); A. Samuel Oddi, *An Uneasier Case for Copyright than for Patent Protection of Computer Programs*, 72 Neb. L. Rev. 351 (1993); Kenneth W. Dam, *Some Economic Considerations in the Intellectual Property Protection of Software*, 24 J. Legal Stud. 321 (June 1995).
11. See John C. Phillips, *supra* n. 10, at 1012.
12. See *generally* Haynes, *supra* n. 2.
13. Computer Software Copyright Act, 1980, Pub. L. No. 96-517, 94 Stat. 3015, 3028 (codified at 17 U.S.C. 101, 117 (1994).
14. See Kenneth L. Port and Gerald Paul McAlinn, *Comparative Law: Law and the Legal Process in Japan* 670 (Carolina Academic Press, 2nd Ed. 2003) citing Welch and Anderson, *Copyright Protection of Computer Software in Japan*, 11 Computer J.L. 287 (1991).

software.[15] In 1973, the Second Subcommittee of the Copyright Council set up by the Agency for Cultural Affairs (*Bunka-cho*) submitted a report supporting copyright law as the mode of protection for software requiring minimal changes.[16] In 1983, Information Industry Committee, Industrial Structure Council set by MITI submitted an interim report that recommended a sui generis legislation to protect computer software- the Program Rights Law.[17] In 1984, Sixth Subcommittee of the Copyright Council set up by the Agency for Cultural Affairs released an interim report recommending that Japan follow copyright law protection for software.[18] Eventually due to heavy U.S. and European protest and lobbying, the Agency for Cultural Affairs' recommendation was followed.[19] The copyright law of Japan was amended in 1985 to include computer programs and 1987 to include databases.

France in 1985,[20] and Brazil in 1987[21] enacted sui generis models for software protection operating under copyright law but adapted to cater to the unique requirements of computer software. Significantly, the term of protection was twenty-five years.

However, eventually software protection was explicitly brought under the ambit of copyright protection.[22] Furthermore, judicial recognition has ensured that both source and object code are covered.[23] Protection of the non-literal aspects of computer software viz. the structure, sequence or organization of the program and the commands, menus or methods of operation of the program has been a field involving heavy debate. *Flores* opines that regarding such cases, courts have looked further and, in some

15. Nakayama, *supra* n. 3 (the interim report advocated a protection framework that consisted of the registration system, the format examination principle, disclosure of the program outlines, arbitration or conciliation procedures, and a short period of protection (ten years)).
16. Nakayama, *supra* n. 3.
17. *Ibid.*
18. *Ibid.*
19. *Ibid.*; Haynes, *supra* n. 2.
20. Law on Author's Rights and on the Rights of Performers, Producers of Phonograms and Videograms and Audiovisual Communication Enterprises (No. 85-660, of 3 Jul. 1985) http://www.wipo.int/wipolex/en/details.jsp?id=1549; see also J.H. Reichman, *Legal Hybrids between the Patent and Copyright Paradigms*, 94 Colum. L. Rev. 2432 (December 1994).
21. Brazil: Law and Implementing Decree on Software Protection (No. 7646 of 18 Dec. 1987) 27 I.L.M. 993 (1988) (English Translation); see also Theodore G. Bryant, *The History, Development and Changing Environment of Protecting Computer Software Against Copyright Violation in Brazil*, 8 Transnat'l Law. 375 (1995).
22. See U.S. Copyright Act 17 U.S.C. § 101 & § 102 (1976); (U.S.A.) H.R. Rep. No. 96-1307, Part 2, 96th Cong. 2d Sess. 19 (1980); (U.S.A.) H.R. Rep. No. 94-1476, Cong., 2d Sess. 54 (1976); § 31, (Australia) Copyright Act, 1968; Art. 10(1) (ix), Japan Copyright Law (1970); European Union, Council Directive 91/250/EEC Legal Protection of Computer Programs, OJ L 122, (17 May 1991).
23. *Apple Computer, Inc. v. Franklin Computer Corp.*, 714 F.2d 1240 (3d Cir. 1983).

instances, crossed the line that separates copyrights and patents; on the other hand, in other cases courts have left certain non-literary elements without copyright protection.[24] The instantaneous manner of obtaining copyright protection and extensive period of protection are the primary attractions for protection of software under copyright law. However, copyright law only protects against copying, not independent similar creation. 'In a sense, the statutory "exclusivity" afforded ... falls somewhere between the "true" exclusivity which inheres in a patent and the nonexistent "exclusivity" of a trade secret'.[25]

The 1970s saw emergence of patent law protection and by the 1990s they had increasingly become a favoured mode of software protection. Computer software is patentable only if it has a practical technical application or result or effect wherein it is structurally and functionally interrelated or integrated with the other aspects of the invention.[26] Patent law has higher eligibility standards for grant of protection than copyright law. Patent protection has a tedious grant procedure, which escalates the transaction costs considerably. Also, the period of protection is considerably short as compared to copyright law. However, patent law protects against subsequent independent similar creation.

Later developments both in the technological and legal arena recognized the need for adjustments to the protection regime to ensure a proper innovation agenda. This also led to increased methods of protection governing software both by law and technology. Technology is used to protect software through access and copy controls to prevent copying. Methods requiring active registration procedures to be able to use software are present. Encryption procedures and the recent evolution of Trusted Computing prevent unauthorized usage. The possibility of reverse engineering to gain unauthorized access has been countered via law.[27]

Trademark law does not protect the literary and non-literary aspects of software. Trade names and Trademarks are used to protect the mark, brand,

24. Yariel Flores, *The Computer Software 'Dilemma'. The Time for the 'Computer Software Bill' Has Arrived*, 50 Rev. Der P.R. 147, 157 (2010).
25. Roger Milgrim and Eric Bensen, *Milgrim on Licensing*, 1 § 11.04 (Matthew Bender, Rev. Ed. 2010).
26. See United States Patent and Trademark Office (USPTO), The Manual of Patent Examining Procedures § 2106; USPTO Examination Guidelines for Computer-Implemented Inventions, 61 Fed. Reg. 7478 (1996); Arts 52(2) & 52(3), European Patent Convention (1973); European Patent Office (EPO), Guidelines for Examination in the European Patent Office (2001); Japan Patent Office (JPO), Examination Guidelines for Computer Software-Related Inventions (1993); JPO, Examination Guidelines for Computer Software Related Inventions (1997); JPO, Implementation Guidelines for Examinations in Specific Fields, Ch. 1- Computer Software-related Inventions (1999); JPO, Examination Guidelines for Computer Software-related Inventions (2000).
27. Articles 11 & 12, WIPO Copyright Treaty (1996); U.S. Digital Millennium Copyright Act 17 U.S.C. § 1201 (1998); European Directive 2001/29/EC Harmonisation of Certain Aspects of Copyright and Related Rights in The Information Society (2001).

and goodwill of associated entities. They protect the commercial identity of the developer, programmer, and/or the company in charge of project.[28] This is especially important for large corporations viz. Sony, Microsoft, Apple etc. and their software products viz. Sony Creative Software, Windows, Mac, etc.

International treaties, primarily The Agreement on Trade-Related Aspects of Intellectual Property Rights (TRIPS) further fortified copyright law's position by explicitly stating that computer programs, whether in source or object code, shall be protected as literary works.[29] Some theorists are of the view that TRIPS also allows patentability of computer programs.[30] The WIPO Copyright Treaty has reinforced TRIPS position[31] and provided against unauthorized removal of Rights Management Information.[32]

Currently, software is protected by a range of intellectual property rights viz. copyright law, patent law, trademarks law, and trade secret law. Additionally, contract law via licensing and technological measures also play a prominent role in computer software protection. *Thomson* opines that trade secrets, copyrights, and patents differ as to the scope of subject matter protectable as well as the level of protection provided; hence deciding which form of protection is most advantageous is a highly situational determination and no bright-line rule can be followed.[33]

With evolution of technology and software, and its increased adoption, usage and diversity of application, the calls for change have been becoming increasingly strident, and several discussions as regards amendment to existing copyright law[34] and patent law[35] have been posited to better

28. Flores, *supra* n. 24, at 167.
29. Article 10(1), Agreement on Trade-Related Aspects of Intellectual Property Rights (1995).
30. See Eloise Gratton, *Should Patent Protection be Considered for Computer Software-Related Innovations?*, 7 Comp. L. Rev. & Tech. J. 223, 230 (2003) (TRIPS ... has abandoned the exclusion of computer programs from patentability ... Art. 27(1) suggests that computer programs are as patentable as any other product or process).
31. Article 4, WIPO Copyright Treaty (1996) (Computer programs are protected as literary works within the meaning of Art. 2 of the Berne Convention. Such protection applies to computer programs, whether may be the mode or form of their expression).
32. *Ibid.* at Arts 11 & 12.
33. Andy Thomson, *Intellectual Property Protection Of Computer Software: More Effective Than Skywriting on A Windy Day?*, 37:2 Cumberland L. Rev 289, 324 (2007).
34. See Kristina Soderquist, *Yet Another Suggestion for Solving the Computer Program Dilemma*, 6 Va. J.L. & Tech. 14 (2001); Catherine Parrish, *Unilateral Refusals to License Software: Limitations on the Right to Exclude and the Need for Compulsory Licensing*, 68 Brooklyn L. Rev. 557 (2002); Susan Corbett, *What Ifs and other Alternative Intellectual Property and Cyberlaw Story: What If Object Code Had Been Excluded from Protection as A Literary Work in Copyright Law? A New Zealand Perspective*, 2008 Mich. St. L. Rev. 173 (2008).
35. See Vincent Chiappetta, *Patentability of Computer Software Instruction as an 'Article Of Manufacture:' Software as Such as The Right Stuff*, 17 J. Marshall J. Computer & Info. L. 89 (1998); Mark Aaron Paley, *A Model Software Petite Patent Act*, 12 Santa Clara Computer & High Tech. LJ 301 (August 1996); Chad King, *Abort, Retry, Fail:*

accommodate software. Simultaneously, discussions regarding sui generis protection are gaining increasing prominence in academia.[36] There have been dissents too against sui generis protection.[37]

## 12.3. NEED OF A SUI GENERIS REGIME

The three prongs – law, individual and industry determine any innovation and its protection. Patent law being categorized under industrial property tilts the balance more in favour of the industry while copyright which falls

*Protection for Software-Related Inventions in The Wake of State Street Bank & Trust Co. v. Signature Financial Group, Inc.*, 85 Cornell L. Rev. 1118 (May, 2000); Richard S. Gruner, *Intangible Inventions: Patentable Subject Matter for an Information Age*, 35 Loy. L.A. L. Rev. 355 (January 2002); Gratton, *supra* n. 30.

36. Margo A. Bagley, *Internet Business Model Patents: Obvious by Analogy*, 7 Mich. Telecomm. & Tech. L. Rev. 253 (2000–2001); Glynn S. Lunney, Jr., *E-Commerce and Equivalence: Defining the Proper Scope of Internet Patents*, 7 Mich. Telecomm. & Tech. L. Rev. 363 (2000–2001); Toshiko Takenaka, *E-Commerce and Equivalence: Defining the Proper Scope of Internet Patents Symposium: Commentary: International and Comparative Law Perspectives on Internet Patents*, 7 Mich. Telecomm. Tech. L. Rev. 423 (2000/2001); Vincent Chiappetta, *Defining the Proper Scope of Internet Patents: If We Don't Know Where We Want to Go, We're Unlikely to Get There*, 7 Mich. Telecomm. & Tech. L. Rev. 289 (2000–2001); Jean–Paul Smets–Solanes, *Patent or Sui Generis Right: What Protection Should Be Considered for Software and other Intangible Innovations?*, (2001) http://bat8.inria.fr/~lang/ecrits/autres/smets/brevets_plan-en.pdf; Scott J. Hill, *Restructuring The Law: Proposing A New Section of Title 35 and The Effect Of Amazon.Com v. Barnesandnoble.Com on Business Method Patents*, 48 Wayne L. Rev. 1239 (2002); Bruce Abramson, *Promoting Innovation in The Software Industry: A First Principles Approach to Intellectual Property Reform*, 8 B.U. J. Sci. & Tech. L. 75 (2002); Aaron D. Charfoos, *How Far Have We Come, and Where Do We Go From Here: The Status Of Global Computer Software Protection Under The TRIPS Agreement*, 22 NW. J. Int'l L. & Bus. 261 (2002); Richard S. Gruner, *Everything Old is New Again: Obviousness Limitations on Patenting Computer Updates Of Old Designs*, 9 B.U. J. Sci. & Tech. L. 209 (2003); Vincent Chiappetta, *Trip-Ping Over Business Method Patents*, 37 Vand. J. Transnat'l L. 181 (January 2004); Allen Clark Zoracki, *When is an Algorithm Invented? The Need For A New Paradigm For Evaluating an Algorithm for Intellectual Property Protection*, 15 Alb. LJ Sci. & Tech. 579 (2005); Steven B. Toeniskoetter, *Protection of Software Intellectual Property in Europe: An Alternative Sui Generis Approach*, 10(2) Univ. San Francisco Intellectual Property L. Bulletin (2005); Mark H. Webbink, *A New Paradigm for Intellectual Property Rights in Software*, 2005 Duke L. & Tech. Rev. 12 (2005); Matt Flinders, *Protecting Computer Software – Analysis and Proposed Alternative*, 7 J. High Tech. L. 172 (2007); Katherine J. Strandburg, *What if There Were A Business Method Use Exemption to Patent Infringement?*, 2008 Mich. St. L. Rev. 245 (2008).

37. Jane C. Ginsburg, *Four Reasons and A Paradox: The Manifest Superiority Of Copyright over Sui Generis Protection of Computer Software*, 94 Colum. L. Rev. 2559 (December 1994); John M. Griem, Jr., *Against A Sui Generis System of Intellectual Property for Computer Software*, 22 Hofstra L. Rev. 145 (1993–1994); Paul Goldstein, *Comments on A Manifesto Concerning The Legal Protection of Computer Programs*, 94 Colum. L. Rev. 2573 (1994).

outside the purview of industrial property tilts the balance in favour of the individual. Computer software's categorization under copyright law and outside industrial property shifts the focus of the laws on to the individual, while the demands of the industry require it to be recognized as industrial property. This creates unnecessary friction in achieving the correct balance.

Several occurrences in the past three decades highlight the developing fissures in the mode of software protection viz. the nature of the software industry and the manner in which it is developing, the increasingly diversified protection of software under law, and the evolving landscape of law and its interaction with technology.

At the same time, certain other developments highlight the increasing acceptance and ease in implementing sui generis protection viz. the increasing sui generis legal recognition being granted to evolving technological arenas and the positive move towards international harmonization in intellectual property laws

Software protection is going through the cycle of seeking to achieve a balance between different de facto controls. Currently, open source software showcases an example of a de facto element challenging the established controls. This is also an indication of a necessity to seek a recalibration of software protection.

### 12.3.1. THE SOFTWARE INDUSTRY

The manner in which technology has developed and its impact on software development has created several unanticipated challenges for the legal landscape.

The initial years of the computer industry saw hardware being sold and software being bundled along with it.[38] Post unbundling too, the technological penetration was not sufficiently extensive. From single purpose computers to general-purpose computers capable of running several programs saw the transition to a separate market for software.[39] As personal computing evolved and technology became more accessible and affordable, programs have become increasingly user oriented from being developer oriented. This has fuelled further development in supporting applications and tools. Furthermore, technology and simultaneously software is increasingly permeating society. From computers, to mobiles, to cloud computing, software

38. See *generally* Dam, *supra* n. 10.
39. See *generally* Yonatan Even, *The Right of Integrity in Software: An Economic Analysis*, 22 Santa Clara Computer & High Tech. LJ 219, 228 (January 2006).

also pervades the most ubiquitous consumer products, including the automobile (30,000 lines of software), television (500 kilobytes of software), and even the electric shaver (two kilobytes of software).[40]

Emergence of the Internet phenomenon in the last couple of decades has fuelled software variety, penetration and created new avenues for software applications to develop. Moreover, as regards the Internet medium itself, *Weiser* elaborates that it presents a formidable challenge to intellectual property policy, particularly the continuing evolution of the hardware and software infrastructure that supports Internet content gives rise to radically different visions for how intellectual property law should regulate the Internet's software infrastructure.[41] Software's dual nature is increasingly becoming evident – it can be both tangible and intangible and sometimes the dual nature is simultaneously applied or revealed.[42] Moreover, unlike earlier, software has become capable of easy transmission and storage on several mediums like hard disks, CDs, flash drives, Internet etc.

Software industry is largely based on 'network effects' which in turn is based on both de facto and de jure industry standards. This in turn effects another mode of promoting reliance on a particular software and in a manner its indirect protection.

User innovation has presented a new paradigm in software development. User innovators develop technology primarily for their own use, rather than to sell it.[43] Eventually though, considerable user innovation is commercialized. Self use and reputational benefits are the prime motivators for user innovators to invest their efforts; they also often freely reveal their innovations to others because of private benefits they are able to obtain as a result.[44] This contrasts sharply with the intellectual property protection model currently applied to technology, especially software. Additionally, the clear line between the developer and the user is increasingly getting blurred. This in turn affects the manner of protection of software.

40. Alan M. Fisch, *Addressing Copyright and Patent as Software's Legal Aegis: A Review of Software and Intellectual Property Protection*, 5 Cornell J. L. & Pub. Pol'y 119 (1996).
41. Philip J. Weiser, *The Internet, Innovation, and Intellectual Property Policy*, 103 Colum. L. Rev. 534, 536 (April 2003).
42. Duncan M. Davidson, *supra* n. 10, at 1072 (The ontological problems of software are similar to the problems faced early in the century by physicists studying the behaviour of light and of electrons, protons, and other particles. At times, these particles exhibit the characteristics of hard objects; at other times, they exhibit the characteristics of waves spread over an area of space-time. This is puzzling only if one remains in a classical world. In a quantum world, these items are what they are. They are neither particles nor waves. They are something too new to be part of our intuitive training, and therefore difficult to appreciate).
43. Katherine J. Strandburg, *Users as Innovators: Implications for Patent Doctrine*, 79 U. Colo. L. Rev. 467, 469 (2008).
44. *Ibid.* at 470.

User innovation has also shed light on the considerable importance of code reuse and incremental innovation. 'Progress in the field of computer software is accomplished through borrowing and building upon the ideas of others. This practice represents a desire to avoid "reinventing the wheel" with each new program, and is thought to be a crucial element for continued software innovation'.[45] As *Reger* states:

> Reuse improves overall product quality as well as the quality of the detailed code because bugs are identified and fixed. Moreover, reuse increases programmer efficiency and reduces the overall time to develop a new software package. Lastly, reuse aids consumers by allowing them to build on the understanding and knowledge they have already acquired rather than forcing them to learn an entirely new program.[46]

An example of user innovation that has had a huge impact is the development of open source software. Moreover, the open source philosophy has found wide acceptance in other fields based on communal endeavours, prominently Biotechnology, Medicine, Bioinformatics, Genomics, Policy modelling, Artistic Works, Publishing sector etc. Within the Information Technology sector too, it has a spill over effect on ancillary aspects of software creation. Open source is being used to promote and encourage the industry to adopt open standards or specification.[47] Similarly inspired by several similar concerns, there has been a movement in the field of electronic hardware towards design efforts on a collaborative basis: the idea of 'Open Source Hardware'.[48] Thus, the open source philosophy has come to be generally recognized as the broader concept of 'Open Innovation'.[49] Consequently, this raises the issue that there might be several simmering grievances with the application of current intellectual property laws to other emerging fields,

45. John C. Phillips, *supra* n. 10, at 1004.
46. Christina M. Reger, *Let's Swap Copyright For Code: The Computer Software Disclosure Dichotomy*, 24 Loy. L.A. Ent. L. Rev. 215, 236 (2004); see also Mark A. Lemley & David W. O'Brien, *Encouraging Software Reuse*, 49 Stan. L. Rev. 255, 304 (January 1997) (Reusable software is more reliable than reinvented software, and reuse is much cheaper than reinvention).
47. See Christian H. Nadan, *Open Source Licensing: Virus or Virtue?*, 10 Tex. Intell. Prop. LJ 349 (2002); Joseph Scott Miller, *Standard Setting, Patents, and Access Lock-In: RAND Licensing and the Theory of the Firm*, 40 Ind. L. Rev. 351 (2007); Greg R. Vetter, *Open Source Licensing And Scattering Opportunism In Software Standards*, 47 B.C. L. Rev 225 (January 2007) (A typical response to dampen opportunism is norm enforcement, where a community or authority enforces practices that inhibit counterproductive opportunism ... A beneficial technological boundary rearrangement may inherently occur in this semi commons if Software Standards are implemented as free and open source software).
48. John R. Ackermann, *Toward Open Source Hardware*, 34 Dayton L. Rev. 183 (2009).
49. See Shinneman, *supra* n. 123; John Dubiansky, *The Role of Patents in Fostering Open Innovation*, 11 Va. J.L. & Tech. 7 ¶1 (2006) (Open Innovation is a contemporary management theory which espouses the idea that corporations can increase their innovative output by importing ideas from outside the firm).

besides software.[50] Hence, it would be desirable to review the intellectual property system in light of the community development model.

### 12.3.2. Legal Protection of Software

The industry relied primarily on trade secret law and contract law as the mode of software protection initially. Eventually, copyright law was chosen legislatively as the vehicle for software protection. Judicial recognition soon allowed patent protection as well. Soon industry started recognizing technological protection measures as another option and their efficacy was strengthened by legislative and judicial support. *Luettgen* tracing the history of software protection states that copyright law and patent law have experienced a reversal of fortune in the past ten years; copyright protection started off broad, but has since been getting thinner.[51]

Besides issues as regards protection under the specific laws, the multiple and diversified form of protection created several pockets where overlap of protection occurred. Concurrently, it left certain areas unprotected as well. Layering of protection created avenue for mismanagement. Arguments as regards excessive propertization have been raised too.

#### 12.3.2.1. Copyright Law Issues

Copyright law by its very nature protects expressions and not the underlying ideas.[52] In case of computer software, it transitions to protection of the expression but not the underlying functionality.[53] Hence, the essence of

50. See Lisa Mandrusiak, *Balancing Open Source Paradigms And Traditional Intellectual Property Models To Optimize Innovation* 63 Maine L. Rev. 303 (2010); Hafiz Aziz ur Rehman, *Equitable Licensing and Publicly Funded Research: A Working Model for India?*, 16 Southwestern J. of Int'l L. 75 (2010); Stephen M. Maurer, *Open Source Drug Discovery: Finding A Niche (Or Maybe Several)* 76 UMKC L. Rev. 405 (2007).
51. David G. Luettgen, *Functional Usefulness v. Communicative Usefulness: Thin Copyright Protection for The Nonliteral Elements of Computer Programs*, 4 Tex. Intell. Prop. LJ 233, 234 (1996).
52. See Himanshu S. Amin, *The Lack of Protection Afforded Software Under The Current Intellectual Property Laws*, 43 Clev. St. L. Rev. 19, 31 (1995) (Copyright does not protect actual ideas, processes, procedures, methods of operation, systems, concepts, discoveries, principles, or utilitarian aspects of a work).
53. Abramson, *supra* n. 36, at 123 (With respect to functional works, the line between an idea and its expression can be less than obvious); Karjala, *Computer Software, and the New Protectionism*, *supra* n. 9, at 94 (Despite their formal expression as combinations of symbols comprising literary works, computer programs are utilitarian articles created and developed for human use, not human comprehension); Karjala, *A Coherent Theory*, *supra* n. 9, at 57 (In taking 'functionality' as the fundamental distinction between patent and copyright, it is crucial to establish a careful definition of the term. A number of commentators have argued that functionality is not a barrier to copyright protection by equating the term 'functional' to 'useful'. This equation of usefulness with functionality,

computer software is open to analogous development. This in turn allows several expressions of the same innovation to exist.

Software has several similarities to traditional copyrightable works. It is a literary work capable of several different aesthetic expressions and subject to easy, quick and cheap reproduction. However, treatment of computer software in the same genre as books, plays and other such works completely discounts the functional and enormously commercial nature of computer software.[54] In context of computer software, *Amin* opines that functionality and ease of use are far more important to a purchaser of software than its aesthetic appeal or originality.[55] The inordinately long period of protection under copyright law makes the situation worse. 'Once one recognizes that computer programs are machines whose medium of construction is text, it becomes obvious why copyright is an inappropriate vehicle for protecting most program behavior'.[56] *Phillips* succinctly enumerates the current situation as regards software protection by copyright law:

> Arguably, copyright was a useful and adequate form of protection for computer software at the inception of the computer age. Advances in software technology, however, have made copyright an increasingly inapt choice for computer software intellectual-property protection.[57]

#### 12.3.2.2. Patent Law Issues

Granting of patents in the software field has seen dynamic developments. From being unpatentable subject matter, to requiring an actual physical embodiment to reflect integration became a requisite for grant of protection. This slowly made way for a token physical embodiment and now a computer

however, leaves no distinction between patent and copyright subject matter and raises the question of why we should have two very different statutory schemes for protecting the same thing).

54. Karjala, *Software Protection Debate in Japan*, *supra* n. 9, at 61 (the importance of computer programs as economic goods that are widely used in economic and industrial activity and the wide variety of programs and their functions require a more carefully structured system of protection than is available under copyright law).
55. Amin, *supra* n. 52, at 33.
56. Samuelson et al., *Manifesto*, *supra* n. 7, at 2399.
57. John C. Phillips, *supra* n. 10, at 1007; see also Abramson, *supra* n. 36, at 151 (Existing software copyrights are narrow, shallow, and long. They are narrow because behaviour is not protected by copyright. They are probably at least somewhat shallower than a standard copyright because of the courts' growing willingness to allow decompilation by commercial competitors as part of the reverse engineering process. Their length, although formally ninety-five years, is effectively infinite because copyright protection lasts far longer than the useful life of computer code.); see also Jacqueline D. Lipton, *IP's Problem Child: Shifting the Paradigms for Software Protection*, 58 Hastings LJ 205, 246 (December 2006) (Copyright law is a poor fit for the needs of software developers. Additionally, because of its lack of cost, significant formalities, and its lengthy duration, the spectre of over-protection of software code is raised by copyright law).

program on a disk too is considered to be patentable.[58] This dilution in stance grants recognition to the tangible/ intangible and functional/ industrial character of software which does not conform to the traditional notions of subject matter covered by patent law as against copyright law. The scenario has evolved to such an extent that *Mikus* argues that the physical embodiment criteria should be completely done away with:

> [P]atent practice should be altered to embrace the true industrial nature of information technology without resorting to an artificial requirement of physical embodiment. Only those ideas whose level of abstraction would result in inappropriately broad patent protection that would hinder the future development of a competitive marketplace should be excluded from patent protection.[59]

At the same time, various software elements fall outside the purview of patentable subject matter and are hence ineligible for grant of protection. 'Regular patents simply do not fit the basic nature of algorithms and software'.[60] Many programmable processes and programmed machines may not ultimately be patentable subject matter.[61]

Furthermore, software designed to update devices and processes raises distinctive patent law issues because the software is frequently a complex mixture of old and new design elements drawn from multiple design fields.[62] *Gruner* is of the opinion that the distinctive features of computer updates have raised equally distinctive problems in determining the obviousness and patentability of these updates.[63]

Concurrently, the considerable exclusive monopoly period restrains access to building block elements that can precipitate negative business and administrative effects. Finally, obtaining patent protection for software is expensive and involves considerable time.

58. *Supra* n. 26.
59. Jean-Philippe Mikus, *Of Industrious Authors and Artful Inventors: Industrial Works and Software at the Frontier of Copyright and Patent Law*, 18 I.P.J. 187, 253 (August 2004).
60. Paley, *supra* n. 35, at 328.
61. Amin, *supra* n. 52, at 22; see also Paley, *supra* n. 35, at 327 (The Federal Circuit is making valiant attempts to shoehorn or stretch algorithms to fit somewhere between the words 'process' and 'machine'.); Mikus, *supra* n. 59, at 242 (One area where rejection on the basis of subject matter has historically occurred lies precisely at the intersection of copyright and patent law … computer software has been at times projected squarely in that no man's land); Deborah Azar, *A Method to Protect Computer Programs: The Integration of Copyright, Trade Secrets, and Anticircumvention Measures*, 2008 Utah L. Rev. 1395, 1412 (2008) (If computer programs can be reduced to math, there remains the question of whether computer programs are math, an application of math, and whether an application of math falls within the definition of technology).
62. Gruner, *supra* n. 36, at 212.
63. *Ibid.* at 215.

### 12.3.2.3. Diversified Protection Regime

Currently a diversified regime of intellectual property is applied to the field of software. Layering of protection is being done as regards software where the components are protected by more than one set of protection options. Contract law is increasingly being used where software is now licensed instead of sold to avoid exhaustion and first sale liabilities.[64] Similarly, in addition to copyright law, trade secrets or technological measures such as access and copy controls are often used to protect software; these measures, in turn are further protected by laws regulating and monitoring fair use, anti-circumvention, reverse engineering and interoperability.[65] Thus, layering of laws has become a tool that copyright holders use to recalibrate the balance of rights among themselves, their users, and their competitors.[66] The need to have layering of laws questions the efficacy of the existing regime for software protection; 'if copyright were adequate to meet the needs of the software industry, the software industry would not have to engage in a frantic search for multiple layers of protection'.[67] Concurrently, *Lipton* elaborates on the situation stating that when the legislature decided to encourage copyright protection for software, it was not clear what kind of role patent law, trade secret law, contract law, and DRM measures would play in the protection of valuable software; however, that picture is much clearer now.[68]

Usually different elements of a particular software should be protected by different laws. However, lack of uniform coverage under a single law leaves scope for improper protection, overlapping protection or in some instances no available protection at all; herein the conflict ensues.[69] This

64. Stacey L. Dogan and Joseph P. Liu, *Copyright Law and Subject Matter Specificity: The Case of Computer Software*, 61 N.Y.U. Ann. Surv. Am. L. 203, 230 (2005) (As these provisions are inserted into mass-market shrink-wrap licenses, they effectively replace copyright law with what amounts to private legislation).
65. For example, WIPO Copyright Treaty; 17 U.S.C. § 1201 The Digital Millennium Copyright Act, 1998.
66. Dogan and Liu, *supra* n. 64, at 228.
67. Samuelson, Creating A New Kind Of Intellectual Property, *supra* n. 7, at 515; see also Gratton, *supra* n. 30, at 248 (It appears that copyright and trade secret systems are inadequate protections).
68. Lipton, *supra* n. 57, at 241.
69. Bradley W. Grout, *Wobbling on The Shoulders of Giants: The Supreme Court's Failure In Lotus v. Borland*, 4 J. Intell. Prop. L. 77, 90 (1997) (Because patent law is designed to protect the very utilitarian nature of programs that copyright law cannot address, the fact that both are used for the same medium inevitably leads to conflict between the two areas of law); Abramson, *supra* n. 36, at 151 (The current IP system allows software developers to avail themselves of this strong dual protection); Esther C. Roditti, *Computer Contracts* Vol. no. 1 § 5.04 (Matthew Bender, 2009) (while copyright and patent rights are enforceable against the world, trade secret rights are far more limited … they are enforceable against parties in privity of contract or in a fiduciary relationship, such as employment. Also, while patents grant a monopoly, trade secrets do not. As in copyrights, there is no protection against independent discovery or creativity, nor do trade secret rights protect against reverse engineering, such as the disassembly of

according to *Moffat* disrupts both the patent and copyright bargains, each of which falls apart when an alternative form of protection is available for the invention or creative work.[70] 'To permit overlapping protection may increase the protections for intellectual property owners, but it also deprives society of some of the public benefits that otherwise would flow from copyright and patent law'.[71]

Further confusing the scenario is the completely different characters of the laws being applied to software and their interaction with computer software. 'The fundamental difference between traditional patent and copyright subject matter is best captured by the term "functionality"'.[72] Computer software presents a doctrinal conflict; it looks like a writing, even though it behaves like an invention.[73] *Amin* succinctly captures the essence of the argument:

> Copyright does not protect the ideas, algorithms and logic underlying software programs. Patent protection is very expensive and often difficult to obtain. Trade secret protection requires the owner to monitor the use of the program by licensees to make sure they are not misusing or disclosing the secret, which is a near impossible task when software is mass-licensed to hundreds or thousands of users.[74]

Lastly, overlapping protection and layering of laws also increases transaction costs, where investment has to be made in each form of protection separately.

The above-mentioned conflicts have only become enlarged with the passage of time and evolution of technology and software. Critics are of the opinion that the existing legal regimes cannot evolve to provide appropriate protection for software innovations because of a fundamental mismatch

the product and the building of a clone, provided that the information was obtained without a breach of faith); Andy Thomson, *supra* n. 33, at 324 (it is imperative to understand the nuances of each form of protection as applied to software in order to fully appreciate the scope and limitations of each).

70. Viva R. Moffat, *Mutant Copyrights and Backdoor Patents: The Problem of Overlapping Intellectual Property Protection*, 19 Berkeley Tech. LJ 1473, 1512-13 (2004).
71. *Ibid.* at 1515.
72. Karjala, *Relative Roles*, *supra* n. 9, at 45.
73. Peter S. Menell, *An Analysis of the Scope of Copyright Protection for Application Programs*, 41 Stan. L. Rev. 1045, 1050 (May, 1989).
74. Amin, *supra* n. 52, at 39.

between them.[75] They further affirm that the present equilibrium is considerably more fragile than it seems.[76]

## 12.3.3. EVOLVING LANDSCAPE OF TECHNOLOGY-LAW INTERACTION

Information products are increasingly becoming hybrid where distinguishing between the real and virtual is becoming complicated. Furthermore, traditional roles of hardware and software are increasingly becoming functionally interchangeable. Besides the increasing hybridization and diversity of information products, options of customization to individual needs are also increasing. This in turn complicates traditional notions of application of intellectual property laws. Certain prominent examples of the complexity and dichotomy arising out of this technological integration are online storage, cloud computing, interface protection, and business method patents.

Online file storage allow an Internet hosting service to host user files; this leads to a transition of storage option from the hard disk on to the Internet, which in turn challenges the notion of physicality as well as ownership of the storage. Similarly, cloud computing allows the transition of computing including programs from the computer to the online medium. These transitions are indications of increasing hybridization of hardware and software. They reflect a challenge to the traditional notion of what was earlier considered to be under the strict domain of hardware or software. They also showcase the dynamism and widening grasp of software in the computing arena.

75. Samuelson et al., *Manifesto*, *supra* n. 7, at 2421; see also Samuelson, *Creating A New Kind of Intellectual Property*, *supra* n. 7, at 530–531 (Although the software industry initially may be hesitant to acknowledge it, it too would be better off with something more certain, and more carefully crafted to address its needs, than the uncertain hodgepodge of protection currently available); Galbi, *Prospect of Future Legislation*, *supra* n. 5, at 239 (court cases which will develop … will merely be trying to define boundary areas … until Congress passes new legislation establishing some better form of protection); Toeniskoetter, *supra* n. 36, at 65 (struggled since the late 1970s to place software into two existing regimes of intellectual property ('IP') protection: copyright and patent law … Each of these approaches is an attempt at putting a square peg into a round hole); *contra* Ginsburg, *supra* n. 37, at 2572 (A well-known ungrammatical adage warns against repairing that which, even if imperfect, nonetheless works reasonably well); *contra* Arthur R. Miller, *Copyright Protection for Computer Programs, Databases, and Computer-Generated Works: Is Anything New Since CONTU?*, 106 Harv. L. Rev. 977, 1035 (March 1993) (excluding them [program aspects] from copyright protection in favour of some – as yet undefined – form of sui generis protection is unnecessary and potentially mischievous); *contra* Griem, Jr., *supra* n. 37, at 150 (Although computer software is unique, each aspect of its dual nature can be understood separately in the context of prior technologies and forms of expression).
76. Samuelson et al., *Manifesto*, *supra* n. 7, at 2421; see also Lipton, *supra* n. 57, at 250 (The status quo may seem comfortable, but it is unstable).

User interfaces are especially important to the computing arena. Video game interfaces and spread sheet interfaces showcase two polar aspects of user interfaces. While video game interfaces are closer to copyright law's concept of literary or audio-visual works, spread sheet programs are more like accounting systems. Yet both involve considerable investment. The creative effort involved in the conception of how a user interface should look and perform represents the majority of development expense and commercial value; the cost of reducing that idea to a form understandable by the computer is significantly less.[77] Even if competitors do not have the source code to copy particular modules, they could easily recreate the sensory elements of the user interface so that their version would perform in a similar, or even identical, manner.[78] Courts recognize the inadequacy in legislations to deal with this problem.[79]

Business method patents have been an extremely dynamic field. First, they were recognized in an extremely limited manner. After the *State Street Bank* decision[80] there was wide acceptance of business method patents in U.S. Then again, after the *Bilski* decisions,[81] a balanced and cautious path is being advocated.[82] The real issue however, is again increasing hybridization:

> [T]he clamor over business methods as patent subject matter basically misses the real issue. The principle of hardware/software equivalence – that is, the principle that for every general purpose computer running under the control of computer software there is an equivalent device consisting solely of hardware that is indistinguishable – basically eliminates the subject matter issue for the programmed machine itself.[83]

### 12.3.4. Legal Recognition of Protection for Evolving Technological Arenas

The past decades have seen legal recognition being accorded to several arenas that would ordinarily have been attempted to be accommodated under the traditional intellectual property laws. 'Several "carve out" statutes have been enacted for the special protection of certain intellectual property that

77. John C. Phillips, *supra* n. 10, at 1007.
78. *Ibid.*
79. *Ibid.* at 1008.
80. *State Street Bank & Trust Co. v. Signature Financial Group* 149 F.3d 1368 (Fed. Cir. 1998).
81. *In re Bilski*, 545 F.3d 943, 88 U.S.P.Q.2d 1385 (Fed. Cir. 2008); *Bilski v. Kappos*, 130 S. Ct. 3218 (2010).
82. See *generally* Joshua I. Miller, *Unknown Futures and The Known Past: What Can Patent Learn From Copyright in The New Technological Age?*, 21 Alb. LJ Sci. & Tech. 1 (2011).
83. Dennis S. Karjala, *Distinguishing Patent and Copyright Subject Matter*, *supra* n. 9, at 444.

shares both functional and expressive characteristics'.[84] *Janis* opines that utility model laws and design protection laws, quite plainly, constitute prototypical examples of hybrid intellectual property regimes.[85] In addition to utility model laws[86] and design protection,[87] other 'carve-outs' include semiconductor chip protection,[88] vessel hull design protection,[89] and database protection.[90] Even within copyright law and patent law greater clarity has been attempted to be achieved as regards computer software. This can be seen by virtue of the EU Directive on Legal Protection of Computer Programs[91] and debate on EU Proposal for a Directive on the Patentability of Computer-Implemented Inventions.[92] These have furthered the arguments to recognize sui generis protection for software.[93]

The major impetus for semiconductor design protection was that their designs, like software, are relatively easy to transform from expression into useful embodiments and they represent a significant and expensive amount of intellectual effort.[94] Semiconductor Chip protection laws blended elements of patent law and copyright law.[95] The criteria for grant of protection are considerably relaxed than under patent law but stricter than copyright

84. Flinders, *supra* n. 36, at 195.
85. Mark D. Janis, *Second Tier Patent Protection*, 40 Harv. Int'l LJ 151, 217 (1999); Nimmer, Nimmer on Copyright § 8A.01 (Matthew Bender 2011) ([the Semiconductor Chip Protection Act of 1984] stands alone as a new and sui generis form of intellectual property, 'separate from and independent of the Copyright Act'.).
86. Article 1, Paris Convention for the Protection of Industrial Property (WIPO, 1883); (Germany) Utility Model Law, 1986; (Japan) Utility Model Act (Act No. 123 of 1959); (Korea) Utility Model Law (No. 952 of 1961).
87. Article 25, Agreement on Trade-Related Aspects of Intellectual Property Rights (1995); (U.S.) Patent Act, 35 U.S.C. § 171 (2000); EU Directive on the Legal Protection of Designs 98/71/EC (1998).
88. See Washington Treaty on Intellectual Property in Respect of Integrated Circuits (1989); Art. 35, Agreement on Trade-Related Aspects of Intellectual Property Rights (1995); U.S. Semiconductor Chip Protection Act of 1984 Act of 8 Nov. 1984, Pub. L. 98-620, 98 Stat. 3347 (17 U.S.C. §§ 901-14); EU Directive on The Legal Protection of Topographies of Semiconductor Products (87/54/EEC 16 Dec. 1986).
89. U.S. Vessel Hull Design Protection Act (Act of 28 Oct. 1998), Pub. L. 105-304, 112 Stat. 2860 (17 U.S.C. §§ 1301-1332).
90. Article 10, Agreement on Trade-Related Aspects of Intellectual Property Rights (1995); EU Directive on the Legal Protection of Databases 96/9/EC (1996).
91. EU Directive on the Legal Protection of Computer Programs (91/250/EEC) (1991).
92. The European Commission, Proposal for Directive on the Patentability of Computer-Implemented Invention (CII Directive -2002/ 0047/COD) (2002).
93. Zoracki, *supra* n. 36, at 604 (A comparison of the situation for semiconductor chips and computer software may provide some insight for deciding if sui generis law for computer software will be helpful); Flinders, *supra* n. 36, at 195–196 (Should a similar type protection [Design Patents] be extended to mixtures of expression and function within software?).
94. Flinders, *supra* n. 36, at 195–196 (2007).
95. Kristen Osenga, *Information May Want to Be Free, But Information Products Do Not: Protecting and Facilitating Transactions in Information Products*, 30 Cardozo L. Rev. 2099, 2140 (May, 2009).