OPEN SOURCE BUSINESS MODELS

David Ma
Blaney McMurtry LLP
416.596.2895
dma@blaney.com

OPEN SOURCE BUSINESS MODELS

by David Ma[1]

1. INTRODUCTION

This paper will: (a) review some of the more common business models used to exploit intellectual property; (b) describe, in brief, what open source is; and (c) identify characteristics of open source licenses as they pertain to those business models.

It is oriented primarily to owners or developers of intellectual property that are contemplating the alternatives available to them in the commercial exploitation of that IP. The general context on which this paper focuses is the development and exploitation of software. However, some or all of the principles described below may be applied in other contexts, and we describe some of these briefly toward the end of the paper.

The intent of this paper is not to advocate open source business models as the definitive way to undertake such a venture. Rather, it is to familiarize the reader with the underpinnings of what is becoming an increasingly prevalent approach to exploiting IP which warrants serious consideration as an alternative to more traditional methods - namely, a proprietary licensing model which emphasizes the treatment of underlying source code as a trade secret. It may well be that the particular circumstances of a business undertaking do not lend themselves to such models. However, it would be, in the author's opinion, inadvisable not to give them due consideration.

2. WHAT IS OPEN SOURCE?

Perhaps the key principle of the open source is denial - more specifically, denying any one person the right to exclusively exploit software. In this regard, the Free Software Foundation, in its Free Software Definition, specifies that there must be no restrictions on "the users' freedom to run, copy, distribute, study, change and improve the software".[2] The FSD sets out four essential freedoms that users should have: (1) freedom to run the program for any purpose; (2) freedom to study how the program works, through access to the source code, and change it to make it do what the user wishes; (3) freedom to redistribute copies; and (4) freedom to distribute copies of the modified versions to others.

This list is not necessarily exclusive, nor would everyone agree on all the relevant factors that qualify a given project as being "open source". For example, the Open Source Initiative sets out ten criteria in its Open Source Definition, which adds the following criteria: integrity of the author's source code, no

---

[1] Partner, Blaney McMurtry LLP. I would also like to express my gratitude for the assistance and diligent efforts of Morgan Borins, student at law.

[2] Free Software Foundation, *The Free Software Definition*, online: The GNU Operating System http://www.gnu.org/philosophy/free-sw.html.

discrimination against persons or groups, no discrimination against fields of endeavour, distribution of the license, the license must not be specific to a particular product, the license must not restrict other software, and the license must be technology-neutral.[3]

While it is important to keep in mind the varying definitions, this paper will focus on certain key elements relating to the business models described.

3. OPEN SOURCE LICENSES

At present, the Open Source Initiative has qualified just under 70 licenses as meeting its criteria for open source licenses,[4] each with its own particular set of rights and obligations. A comprehensive review of all the characteristics of a given license is beyond the scope of this paper. Instead, we focus primarily on how one element is treated – more specifically, the manner and extent to which they permit or restrict the subsequent redistribution of modifications and combinations with other code. Although there are other important provisions in all licenses, an understanding of the provisions related to modifications and combinations in these licenses is key to understanding and operationalizing the open source business models described below.

*(a) **Least restrictive licenses***

The first category of licenses is those that impose no restrictions on the redistribution of modifications or combinations of code that are licensed under them. Two illustrative examples of such licenses are the Apache and Berkeley Software Distribution licenses. The Apache license was created for the distribution of software produced by the Apache Software Foundation, a non-profit corporation responsible for the development of the most predominant website server software in use today.[5] The BSD License was used for UNIX operating system derivatives developed by the University of California at Berkeley beginning in the late eighties.

There have been three versions of the Apache license to date, the most recent of which is the Apache License 2.0 released in January 2004.[6] One of the ASF's objectives in drafting this form of license was that software licensed under its terms would be widely adopted. In order to achieve this objective, they

---

[3] "The Open Source Initiative (OSI) is a non-profit corporation formed to educate about and advocate for the benefits of open source and to build bridges among different constituencies in the open source community." The OSI certifies licenses as being open source if they qualify within the ODI. See Open Source Initiative, online: http://www.opensource.org/.

[4] See http://www.opensource.org/licenses/alphabetical/.

[5] The most recent Netcraft survey shows Apache powering 58.07% of the websites it surveyed as of October, 2010. See http://news.netcraft.com/archives/2010/10/12/october-2010-web-server-survey.html.

[6] See *Apache License Version 2.0*, online: The Apache Software Foundation http://www.apache.org/licenses/LICENSE-2.0.

permitted users to distribute modifications and combinations under whatever terms they wished to, including proprietary licenses that did not require the user to disclose the source code to such modifications or improvements. [7] The relevant provisions in the Apache license 2.0 are straightforward:

> You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:
>
> > 1. You must give any other recipients of the Work or Derivative Works a copy of this License; and
> >
> > 2. You must cause any modified files to carry prominent notices stating that You changed the files; and
> >
> > 3. You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
> >
> > 4. If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.
>
> You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use,

[7] As stated by the ASF, "The result is a license that is supposed to be compatible with other open source licenses, while remaining true to the original goals of the Apache Group and supportive of collaborative development across both non-profit and commercial organizations."

> reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

These provisions allow for distribution of copies of an original work or any derived works thereof by licensees of the original work, while ensuring that recipients of those copies or derivative works are made aware of the original license terms, any modifications that were made to the original work, all legal rights (copyright, patent, trademark) attached to the original work and any notices that were distributed along with the original work. The final paragraph in the redistribution clause allows for licensees of the original work that create derivative works to copyright their modifications and license them under different terms, so long as the conditions of the original license are complied with. The result is that derivative works may be licensed proprietary licenses and without the provision of source code.

The current version of the BSD license, in use since 1999, is relatively short and contains only three clauses:

> Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
>
> > Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
> >
> > Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
> >
> > Neither the name of the <ORGANIZATION> nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.
>
> THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)

> HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.[8]

Unlike the Apache License 2.0, the BSD license is silent on whether licensees must impose the same conditions on any derivative works. However, the absence of any such obligation is generally interpreted to mean that there is no such obligation under the BSD license.[9]

(b) ***Very restrictive licenses***

The second category of licenses are those that impose a positive obligation to redistribute modifications and combinations under the same terms as the work on which they are based. Examples of these licenses include the GNU[10] General Public License and Affero General Public License.

The GPL was first written in 1989, and has become the most frequently used form of license for open source software.[11] The current version, Version 3 or GPLv3,[12] was officially released by the FSF in 2007 after four drafts and an extensive public consultation process. The GPL is intended, as stated in its preamble, to "guarantee your freedom to share and change all versions of a program—to make sure it remains free software for all its users." The fundamental difference in the GPL from the least restrictive licenses such as Apache or BSD is that licensees, if they wish to distribute derivative works of the work licensed under the GPL, must do so under the GPL as well, thereby ensuring that derived works (including their source code) remain open and accessible and not become proprietary and closed.

---

[8] See http://www.opensource.org/licenses/bsd-license.php. A further simplified version also drops the attribution prohibition and adds the following disclaimer: "The views and conclusions contained in the software and documentation are those of the authors and should not be interpreted as representing official policies, either expressed or implied, of <copyright holder>."

[9] Andrew M. St. Laurent, *Understanding Open Source and Free Software Licensing* (Sebastapol, CA: O'Reilly Media, Inc., 2004) at 24. See also http://www.freebsd.org/doc/en/articles/bsdl-gpl/article.html: "This new BSD license is intended to encourage product commercialization. Any BSD code can be sold or included in proprietary products without any restrictions on the availability of your code or your future behavior."

[10] GNU is a recursive acronym for "GNU's Not Unix", the name of an operating system consisting entirely of free software that was developed by the GNU Project beginning in the early eighties. See http://www.gnu.org/gnu/gnu-history.html.

[11] Black Duck Software, a company which specializes in open source licensing compliance, undertakes an ongoing automated survey of the licenses utilized by open source projects. In its most recent survey, it has calculated that the 46.50% of projects use GPL 2.0 and 6.54% use GPL 3.0. See http://www.blackducksoftware.com/oss/licenses.

[12] See *The GNU General Public License v3.0*, online: GNU Operating System http://www.gnu.org/licenses/gpl.html.

The relevant provisions are found in Section 5 of the GPL:

> 5. Conveying Modified Source Versions.
>
> You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:
>
> * a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
>
> * b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to “keep intact all notices”.
>
> * c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
>
> * d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.
>
> A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an “aggregate” if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation’s users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

The specific provision imposing the obligation to use the GPL is set out in clause (c). It is, however, important to note that this requirement only applies in respect of “anyone who comes into possession of a copy” and not if and when modifications are created. Thus, there is no positive obligation either to make modifications (whether in source code or otherwise) available at all or to do so under the terms of the GPL, unless and until those modifications are distributed to a third party. It is only at that point which the obligation arises. This is particularly relevant in situations where modifications are only to be

used internally and not distributed, or where such modifications are being used to provide a network accessible service without actual delivery of the software.

Attention should also be drawn to the last provision, which clarifies the extent to which the GPL must be applied where the GPL licensed work is merely aggregated with other works. The short answer is that it does not. However, the FSF does make it clear that the GPL is intended to apply to larger works if it is linked to or compiled with such larger work.[13]

The GNU Affero General Public License[14] was written with the intention of adapting the GPL to apply to software that would typically be run as a network service or "software as a service", where end users would not need to receive or install a copy of the software in order to use it - rather, they would be able to use its functionality through a browser or similar interface. Essentially, the AGPL consists of all of the terms of the GPL, with the additional following clause:

> Notwithstanding any other provision of this License, if you modify the Program, your modified version must prominently offer all users interacting with it remotely through a computer network (if your version supports such interaction) an opportunity to receive the Corresponding Source of your version by providing access to the Corresponding Source from a network server at no charge, through some standard or customary means of facilitating copying of software.

This clause was added to address the perceived "loophole" of the GPL described above: Companies could modify and enhance GPL licensed code in order to provide proprietary service offerings, but would have no obligation to release those modifications or enhancements because they were never "distributed". The additional clause above closes that loophole by including an additional trigger for that obligation - namely, the provision of services using such modifications.

(c) ***Somewhat restrictive licenses***

In between the most and least restrictive licenses are moderately restrictive licenses, such as the GNU Lesser General Public License and the Mozilla Public License. These licenses do require that modifications to any code licensed under their terms be made available in source code form, similar to the most restrictive licenses described above. However, they also permitting the licensing of a new work resulting from combination or linking with other proprietary code under different terms.

The original intent of the LGPL was to focus on the licensing of software libraries, permitting them to link with non-GPL software and not requiring the whole to be licensed under those same terms, in order to

---

[13] See http://www.gnu.org/philosophy/why-not-lgpl.html.

[14] See *GNU Affero General Public License*, online: GNU Operating System, http://www.gnu.org/licenses/agpl.html.

encourage the widest possible adoption of such libraries, so that they become (or at least have the potential to become) de facto standards.[15]

Thus, the LGPL allows for modifications to an LGPL licensed work, provided such modifications are licensed only under either the LGPL or the GPL:

> 2. Conveying Modified Versions.
>
> If you modify a copy of the Library, and, in your modifications, a facility refers to a function or data to be supplied by an Application that uses the facility (other than as an argument passed when the facility is invoked), then you may convey a copy of the modified version:
>
> * a) under this License, provided that you make a good faith effort to ensure that, in the event an Application does not supply the function or data, the facility still operates, and performs whatever part of its purpose remains meaningful, or
>
> * b) under the GNU GPL, with none of the additional permissions of this License applicable to that copy.

The LGPL licensed work may also be combined, linked or compiled with other code. In contrast to the GPL (as well as the requirements of the LGPL in respect of modifications), doing so does not require that the whole be licensed under the LGPL, so long as the licensee is provided both with the technical materials as well as the rights to modify the LGPL licensed work and recreate the combined work using such modifications:

> 4. Combined Works.
>
> You may convey a Combined Work[16] under terms of your choice that, taken together, effectively do not restrict modification of the portions of the Library contained in the Combined Work and reverse engineering for debugging such modifications, if you also do each of the following:
>
> * a) Give prominent notice with each copy of the Combined Work that the Library is used in it and that the Library and its use are covered by this License.

---

[15] This is expressly set out in the preamble to the LGPL.

[16] "Combined Work" is defined as a work produced by combining or linking an Application (a work that makes use of an interface of the original work licensed under the LGPL) with the Library (the original work licensed under the LGPL) .

* b) Accompany the Combined Work with a copy of the GNU GPL and this license document.

* c) For a Combined Work that displays copyright notices during execution, include the copyright notice for the Library among these notices, as well as a reference directing the user to the copies of the GNU GPL and this license document.

* d) Do one of the following:

o 0) Convey the Minimal Corresponding Source under the terms of this License, and the Corresponding Application Code in a form suitable for, and under terms that permit, the user to recombine or relink the Application with a modified version of the Linked Version to produce a modified Combined Work, in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.[17]

o 1) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (a) uses at run time a copy of the Library already present on the user's computer system, and (b) will operate properly with a modified version of the Library that is interface-compatible with the Linked Version.[18]

* e) Provide Installation Information, but only if you would otherwise be required to provide such information under section 6 of the GNU GPL, and only to the extent that such information is necessary to install and execute a modified version of the Combined Work produced by recombining or relinking the Application with a modified version of the Linked Version. (If you use option 4d0, the Installation Information must accompany the Minimal Corresponding Source and Corresponding Application Code. If you use option 4d1, you must provide the Installation Information in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.) [19]

---

[17] Put simply, this means that one must provide all materials that would enable a user to modify the LGPL-licensed work and then create an updated Combined Work with that modification, all on terms that would permit the user to do so. Depending on how they are combined, this may potentially require the provision of some source code for the non-LGPL work.

[18] Another mechanism to enable the use of modifications of the LGPL licensed work within the Combined Work.

[19] Installation Information means information required to install and execute modified versions of LGPL-licensed work. This clause is tied to the “anti-Tivoization” provisions in the GPL to ensure that consumers who purchase

Along similar lines, the MPL permits modifications, but any such modifications must be made available in source code form under the terms of the MPL:

> 3. Distribution Obligations.
>
> 3.1. Application of License.
>
> The Modifications which You create or to which You contribute are governed by the terms of this License, including without limitation Section 2.2. The Source Code version of Covered Code may be distributed only under the terms of this License or a future version of this License released under Section 6.1, and You must include a copy of this License with every copy of the Source Code You distribute. You may not offer or impose any terms on any Source Code version that alters or restricts the applicable version of this License or the recipients' rights hereunder. However, You may include an additional document offering the additional rights described in Section 3.5.
>
> 3.2. Availability of Source Code.
>
> Any Modification which You create or to which You contribute must be made available in Source Code form under the terms of this License either on the same media as an Executable version or via an accepted Electronic Distribution Mechanism to anyone to whom you made an Executable version available; and if made available via Electronic Distribution Mechanism, must remain available for at least twelve (12) months after the date it initially became available, or at least six (6) months after a subsequent version of that particular Modification has been made available to such recipients. You are responsible for ensuring that the Source Code version remains available even if the Electronic Distribution Mechanism is maintained by a third party.
>
> 3.3. Description of Modifications.
>
> You must cause all Covered Code to which You contribute to contain a file documenting the changes You made to create that Covered Code and the date of any change. You must include a prominent statement that the Modification is derived, directly or indirectly, from Original Code provided by the Initial Developer and including the name of the

---

devices that have LGPL licensed code installed on them have information necessary to bypass hardware or similar restrictions (such as locked bootloaders, digital signatures, or encryption) to install modified code on those devices if they wish to do so.

> Initial Developer in (a) the Source Code, and (b) in any notice in an Executable version or related documentation in which You describe the origin or ownership of the Covered Code.

And, as with the LGPL, the MPL generally permits code licensed under its terms to be combined with other code (which may be proprietary) and the whole licensed under terms other than the MPL, so long as the original code licensed under the MPL and any modifications thereof are made available in source code form:

> 3.7. Larger Works.
>
> You may create a Larger Work[20] by combining Covered Code with other code not governed by the terms of this License and distribute the Larger Work as a single product. In such a case, You must make sure the requirements of this License are fulfilled for the Covered Code.

In the case of either the LGPL or the MPL, developers who wish to exploit their contributions under a proprietary, closed licensing regime could structure their development efforts so that improvements or additional functionality are created as separate elements combined into either a Combined Work or a Larger Work (using the LGPL and MPL terminology, respectively), rather than as direct modifications of the components licensed under either license.

(a) ***License compatibility***

One further licensing consideration which warrants brief mention is license compatibility, particularly in situations where existing work is being incorporated into either new work or combined with other existing work. In some cases, the licenses under which each work is licensed may make it difficult or impossible to comply with one or more of them. Thus, as a simple example, modifications of a work licensed under a very restrictive license such as the GPL could not be licensed under a less restrictive license such as the Apache license. On the other hand, work licensed under the Apache license could be incorporated into another work that was licensed under the GPL.[21]

License compatibility issues may become quite complex as the number of disparate works combined increases. In addition, much is dependent on exactly how the various elements are combined. A full examination of this issue is beyond the scope of this paper, but it should be borne in mind and considered when considering the business models described below.

---

[20] The MPL defines "Larger Work" as a work which combines Covered Code (the original code and any modifications) or portions thereof with code not governed by the terms of the MPL.

[21] Given the terms of the GPL, this would likely require that the whole work be licensed under the GPL. However there may be ways to avoid this – see for example http://www.softwarefreedom.org/resources/2007/gpl-non-gpl-collaboration.html for one possible strategy as well as general recommendations regarding license compatibility when the "least restrictive" license types are being used.

4. COMMON OPEN SOURCE BUSINESS MODELS

(a) ***Dual licensing***

Dual licensing involves licensing the same code under two different licenses - one typically being a very restrictive license (such as the GPL) and the other being a less restrictive license (typically a proprietary license) that does not mandate the release of source code on any modifications or combinations. While the developer can continue to engage members of the open source community, who can take advantage of the free version of the code (and are required to contribute back their modifications under its terms), commercial developers wishing to create their own proprietary derivatives or combined works will be required to pay a fee in order to ensure they may maintain the proprietary nature of those modifications or the combined work. In effect, dual licensing attempts to convert potential users' perceived risks and disadvantages of more restrictive open source licenses into a business opportunity.[22]

Given the objectives of dual licensing, the choice of license is significant. Utilizing a less restrictive license in place of the GPL, such as Apache, would obviate the need for a proprietary license (given the ability under the Apache license to commercially exploit modifications) and therefore the motivation to pay a fee for same. Similarly, utilizing a moderately restrictive license, such as the LGPL, only partially restricts the ability of licensees to build proprietary extensions – they may, if technically feasible, attempt to build upon your work by incorporating into a larger work or combined work rather than direct modifications which, as previously noted, would enable them to avoid the obligation to license their contributions under the provisions of the applicable open source license.

The nature of the technology that is to be developed also has a bearing on the appropriateness of this model. Products intended primarily for direct use by end users in production, rather than as a value-added component or function within a larger product, would be less suited to this model in comparison to infrastructure, functional component and library type developments.

As an illustrative example, MySQL employed a dual licensing regime for its product – a database, and has stated the reasons for doing so in no uncertain terms:

> **MySQL Commercial License for OEMs, ISVs and VARs**
>
> Oracle provides its MySQL database server and MySQL Client Libraries under a dual license model designed to meet the development and distribution needs of both commercial distributors (such as OEMs, ISVs and VARs) and open source projects.

---

[22] Apart from commercial strategy, dual licensing can also be used to offer different licenses to enable compatibility under other licenses, or to accommodate user preferences. For example, Mozilla is licensed under the MPL, the LGPL and the GPL. See http://www.mozilla.org/MPL/.

For OEMs, ISVs, VARs and Other Distributors of Commercial Applications:

OEMs (Original Equipment Manufacturers), ISVs (Independent Software Vendors), VARs (Value Added Resellers) and other distributors that combine and distribute commercially licensed software with MySQL software and do not wish to distribute the source code for the commercially licensed software under version 2 of the GNU General Public License (the "GPL") must enter into a commercial license agreement with Oracle.

For Open Source Projects and Other Developers of Open Source Applications:

For developers of Free Open Source Software ("FOSS") applications under the GPL that want to combine and distribute those FOSS applications with MySQL software, Oracle's MySQL open source software licensed under the GPL is the best option.

For developers and distributors of open source software under a FOSS license other than the GPL, Oracle makes its GPL-licensed MySQL Client Libraries available under a FOSS Exception that enables use of the those MySQL Client Libraries under certain conditions without causing the entire derivative work to be subject to the GPL.[23]

Database software is particularly well-suited to dual licensing given that databases are common used to "power" other applications, such as CRM, ERP and accounting software packages. Perhaps then, it is not surprising that MySQL has experienced a fair degree of commercial success using such a strategy. This success was likely in no small part bolstered by MySQL's success as an open source platform for web-based applications, becoming to some extent a de facto standard as part of the very popular, "LAMP" software stack.[24]

The basic economic concept which underpins the dual licensing strategy is market segmentation. A developer can, through dual licensing, require potential customers to self-identify the value they place on a product based on its use (i.e. proprietary or non-proprietary) and is therefore able to charge a fee to those who expect to derive greater economic benefit from higher value forms of usage.

---

[23] See http://www.mysql.com/about/legal/licensing/oem/.

[24] Short for Linux (operating system), Apache HTTP Server, MySQL (database software) and PHP (scripting language). See http://en.wikipedia.org/wiki/LAMP_%28software_bundle%29.

Dual licensing, as with most open source business models, allows developers to benefit from an entire community of developers beyond the boundaries of their immediate organizational structure, which may lower development costs.[25] At the same time, some in the open source community may look upon such a model with suspicion, insofar as the developer has, in effect, permitted some of its licensees to "close" the code and not contribute back to the community. And, as with all open source models, developers will give up some measure of control over their intellectual property. If community developers become discontent with the primary developer's business model, development direction, or anything else for that matter, there is always a risk that community developers will attempt to "fork" the code - taking the existing code provided under an open source license, and continuing development in a significantly different direction (such as evolving a new data model, or creating distinct interfaces) which could ultimately result in a distinct and incompatible code base and fracturing the product's user base, taking away one of the significant benefits of an open source model - namely, the possibility of becoming a de facto standard.

Dual licensing regimes may be more susceptible to this due to the need to use contributor agreements in such a model: Developers must ensure that any contributions which are adopted as a part of the "official" code base that the developer wishes to license under both licenses are provided under an agreement that permits the developer to do so. In the absence of such an agreement, community contributions received by the developer would also be subject to the restrictive terms of the open source license. The developer would therefore not be permitted to license their product under a separate proprietary license. Using a properly drafted contributor agreement overcomes this issue.

For example, the following is language from Digium's contribution agreement in respect of contributions to its Asterisk open source project:

> The rights granted may be exercised in any form or format, and Digium may distribute and sublicense to others on any licensing terms, including without limitation: (a) open source licenses like the GNU General Public License (GPL), or the Berkeley Science Division license (BSD); or (b) binary, proprietary, or commercial licenses. If Your Submission is derived from software released by Digium under the GPL, Digium as licensor thereof waives such requirements of the GPL as applied to that software to the limited extent necessary to allow you to provide the Submission and the foregoing license to Digium.[26]

Note in particular the waiver of the requirements of GPL. Along similar lines, the following is an excerpt from the Sun Microsystems, Inc. Contributor Agreement, which it uses for its OpenOffice open source project:

---

[25] *Fink*, *supra* note 10 at 181.

[26] See https://issues.asterisk.org/view_license_agreement.php.

> With respect to any worldwide copyrights, or copyright applications and registrations, in your contribution: you hereby assign to us joint ownership, and to the extent that such assignment is or becomes invalid, ineffective or unenforceable, you hereby grant to us a perpetual, irrevocable, non-exclusive, worldwide, no-charge, royalty-free, unrestricted license to exercise all rights under those copyrights. This includes, at our option, the right to sublicense these same rights to third parties through multiple levels of sublicensees or other licensing arrangements; you agree that each of us can do all things in relation to your contribution as if each of us were the sole owners, and if one of us makes a derivative work of your contribution, the one who makes the derivative work (or has it made) will be the sole owner of that derivative work;…[27]

Note the different approaches taken in each. Digium provides for a fairly broad license from the contributor, while Sun requires joint ownership. In either case the primary objective of such agreements, at least for the purposes of this model, would be to enable the licensing of a contributor's content under the provisions of a proprietary form of license.[28]

(b) ***Open core or hybrid***

The open core or hybrid model again exploits the economic basis of market segmentation to create a commercial opportunity, but instead of segmenting on the basis of the utility derived by licensees based on the manner of use, it segments the market based on the feature set of the product. An open core model basically involves the development and maintenance of a base product with some level of functionality that is made freely available under an open source license. The developer may use existing open source components and may also in turn contribute back some of its improvements or modifications under an open source license.

At the same time, the developer also develops additional functions, extensions and plugins which can be used in conjunction with the base product to enhance the base product or provide for additional or improved functionality in one form or another. However, these components are developed on a proprietary basis and not contributed to the community or licensed under open source licenses. They are developed and exploited using a traditional, closed-source, proprietary model. The developer then either integrates both the open source base and the proprietary extensions as a consolidated whole

---

[27] See http://www.openoffice.org/licenses/sca.pdf.

[28] Apart from dual licensing considerations, contributor agreements may also be used to provide greater certainty regarding the provenance of contributions and used as a defence against infringement claims. In such cases, the rights granted could be narrowed accordingly. In this regard it would preferable to always have major contributors sign contributors agreements, irrespective of the business model adopted. See for example http://www.apache.org/licenses/icla.txt and http://www.gnu.org/prep/maintain/maintain.html#Legal-Matters.

(with corresponding open source and proprietary licenses) in consideration for a fee, or offers the extensions separately as add-on components that licensees may license, also for a fee.

The open core model is often touted as being the "best of both worlds" since it combines the strengths of open source business models with the strengths of traditional closed and proprietary business models. It also addresses some of the weaknesses. One such weakness in open source business models is the reduced barrier to entry (as compared with proprietary models) – anyone with sufficient skill can study, learn and compete with an incumbent developer. And as suggested by Mårten Mickos (former CEO of MySQL), "for an open source company to become commercially successful, it needs to have an unfair advantage against its competition - something that they cannot copy, use, modify or provide to their customers."[29]

Perhaps the most well-known company which has effectively utilized an open core strategy is Apple.[30] Both Apple's OSX (for Mac) and iOS (for its mobile devices, including the iPhone) are built upon a core set of operating system components (named "Darwin") and other components, all of which were built on, and which Apple continues to make available, through open source licensing.[31] While Darwin remains open and can be modified and redistributed, Apple has developed a number of key proprietary components[32] which it then adds to Darwin in order to arrive at the end product, for which it charges a handsome price.

As with a dual licensing regime, an open core business model allows for the leveraging of open source advantages, such as lowered development costs and community contributions, while providing the benefit of retained control over key value-added components that a specific customer base values highly and is willing to pay for. Thus, while a dedicated hacker may be content in installing a Darwin kernel and using it with an open source user interface, a typical iPhone user expects convenience, simplicity and ease of use out of the box, and is willing to pay accordingly.

For new developers, one element in particular is imperative to successfully using an open core model – that of balance – which features are incorporated into the open core and freely available base, and

---

[29] Simon Phipps, "Open Source Needs to Have an Unfair Advantage to Succeed" *Computerworld UK* (June 30, 2010), online: *Blogs* http://blogs.computerworlduk.com/simon-says/2010/06/open-source-needs-to-have-an-unfair-advantage-to-succeed/index.htm.

[30] It is readily acknowledged by the author that there is much more to Apple's business than a pure open core model. Apple can also be considered to be in the hardware sales business the music business, and so on. However, it cannot be denied that one of Apple's fundamental and strategic assets is its operating system software which is, very clearly, using an open core strategy.

[31] See http://developer.apple.com/opensource/, http://www.apple.com/opensource/ and http://www.opensource.apple.com/.

[32] These include proprietary application program interfaces, the windowing system and graphic user interface. See http://developer.apple.com/technologies/.

which are kept proprietary. Too much in the former, and what remains will likely not offer a sufficient value proposition for a developer to earn a reasonable profit. Conversely, too little in the former will lead to few people adopting or contributing to the open core and a corresponding reduction in the benefits that would otherwise accrue to the developer. In addition, while some in the open source community have openly embraced open core business models, others have also been strongly critical, asserting that open core is in essence a closed source model, a view that has been affirmed by some of the individual board members of the OSI[33]. Some open source advocates claim that the non-open features reduces the number of freedoms available to customers, and consider themselves exploited by developers who are able to benefit from these circumstances.

Offering proprietary extensions and components also comes with all the comparative disadvantages of a the traditional proprietary software development regime – higher up front development costs (due to the need to author or purchase proprietary code), higher marketing costs (as potential customers will need convincing to purchase your product, rather than finding out themselves by using it) and a longer time to market, albeit proportionately lower as compared to a wholly proprietary product.

Lastly, care must be taken in the development of proprietary elements, particularly those that may link into or incorporate with open source components which are licensed under more restrictive licenses such as the GPL.

*(c) **Value added products or services***

At the most basic level, a value added model can be thought of as a variation of the "loss-leader" strategy. Just as how grocery stores will advertise a staple for a low cost, in the hopes of attracting shoppers into their stores to see (and hopefully buy) higher margin products, a value added business model attempts to familiarize potential customers with a base product provided at no charge, in the hopes of selling them on value added products[34] or services. That being said, the value of such a model as applied to software is somewhat more compelling, given the upfront investment that users would otherwise be required to make in order to gain familiarity and proficiency with most software products.

The types of value added products and services are only limited by the imagination of the developer, but have traditionally included: (a) consolidated, accelerated and fully-tested and stabilized releases; (b) better (proprietary) mechanisms for updates (e.g. on-line automatic updates); (c) add on tools or features (e.g. management tools, clustering, load-balancing); (d) support, maintenance, customization or consulting services; (e) training and certification services; or (f) warranties - for example, on

---

[33] See Russ Nelson, "Why open core has a problem and is not a problem" (July 17, 2010), online: *Open Source Initiative* http://opensource.org/node/537 and Andy Coliver, "A Simple Declaration About 'Open Core'" (July 20, 2010), online: *Open Source Initiative* http://opensource.org/blog/OpenCore. The OSI however has not released an official position on this.

[34] To the extent that value added products consist primarily of other software products, the comments set out above in respect of the open core business model would apply equally here, and vice-versa.

performance of the product or non-infringement of third party IP rights. Each of these elements is intended to offer users some measure of value, whether through increased or improved functions or utility, convenience and ease of use or use of expert and efficient resources. Amongst all the business models used under an open source regime, the value added approach appears to be predominant. [35]

Interestingly, many of the earlier entrants into commercial open source that ultimately adopted a value-added services approach looked to the legal services industry for inspiration. Bob Young, the founder of Red Hat Software, described how he looked to various industries in contemplating the approach Red Hat would take:

> If we do not own intellectual property the way almost all of today's software companies do, and if those companies insist that their most valuable asset is the intellectual property represented by the source code to the software they own, then it is safe to say that Red Hat is not in the Software Business. Red Hat is not licensing intellectual property over which it has ownership. That's not the economic model that will support our customers, staff, and shareholders. So the question became: What business are we in?
>
> The answer was to look around at other industries and try and find one that matched. We wanted an industry where the basic ingredients were free, or at least freely available. We looked at the legal industry; you cannot trademark or patent legal arguments. If a lawyer wins a case in front of the Supreme Court, other lawyers are allowed to use those arguments without requesting permission. In effect, the arguments have become public domain.[36]

And this, from Michael Tiemann, currently the President of the Open Source Initiative, but in this excerpt discussing his experiences in founding Cygnus Solutions, another open source software company using a value added services model:

> It is all very well and good to have wonderful theories about how to make the world a better place. It is another thing entirely to get those theories funded to the point that they are self-sustaining. Although

---

[35] A recent study suggests that just under 50% of all companies using open source business models use a value added service approach. See http://guide.flossmetrics.org/index.php/6._FLOSS-based_business_models. However, this should not necessarily lead one to be believe that it is either the best or most successful model to use – rather, as with most software companies, the provision of services for a fee is typically a proposition that poses little in the way of risk and can be combined with any number of other strategies and is therefore quite likely to be adopted by most open source enterprises.

[36] See http://oreilly.com/catalog/opensources/book/young.html.

> service-based companies were rare in the world of software products, there were many cases to study in other areas.
>
> Consider the practice of law in America (or Great Britain). Common law is freely available to all who wish to use it. One need not license a decision such as Roe v. Wade in order to use it for arguments. Indeed, the decisions, once made, and at whatever cost, are free to all. Yet for all this freedom, lawyers are among the most expensive professionals to be found. How can a practice of law, which has no primary proprietary code, command such value?
>
> It is not just the act of prosecuting law that people value so highly. It is also the cumulative value of that prosecution. If you hire a good lawyer, and in the course of the prosecution, a decision is made in your favor, that precedent becomes a new part of the law. Justice is not blind; it is full of history.
>
> This is somewhat analogous to the concept of creating and maintaining standards with open-source software. It is very expensive to create a standard and get it right. But it is far more expensive to work without standards or to try to maintain a standard if the standard is bogus. There is great value in having good people working on software whose precedents will set the standards of tomorrow. We believed at the beginning that people would understand this value proposition, and would value the opportunity to pay us to create high-quality, open-source programs that would become the de facto standard of the software world.[37]

Perhaps the best example of this business model in operation is Redhat Software. Redhat licenses most of its enterprise software under GPL licenses, but offers value-added services under proprietary agreements. The services in question include access to supported versions as well as patches, upgrades and bug fixes, through their portal, support services, and software assurance programs, which offer some degree of protection against IP infringement claims. While licensees can freely copy, modify and redistribute the code, they cannot redistribute or provide the benefit of the proprietary services to third parties. They also cannot copy, modify or redistribute any of Redhat's trademarks that may be present in the code or documentation, over which Redhat retains ownership.

Amongst the business models presented in this paper, a value-added model relies the least upon the types of open source licenses used, at least to the extent the value add takes the form of services. Instead, it relies largely upon the ability of the developer to persuade users of its open source offering of

---

[37] See http://oreilly.com/catalog/opensources/book/tiemans.html.

the value of its proprietary offerings. In this regard, developers using such a model may find themselves facing the same branding, marketing and advertising challenges that more traditional software companies using proprietary models face.

(d) ***Variation and mixes***

The business models described above describe some of the more common approaches to exploiting IP through the use of open source business models. It is not intended to be comprehensive. In addition, although we have attempted to describe open source business models within specified categories, there is nothing that would preclude a developer from adopting more than one of the models described above, or taking certain elements from one model and combining them with another. A basic example of this would be the combination of a dual licensing model with value added services. The fact that a proprietary license is offered does not preclude an opportunity for a developer to sell value added services, such as consulting services to assist the user in its integration efforts, or to those who choose to license under an open source license (as they may have no desire to integrate the product into their own commercial offering) but wish to have an expert develop additional functionality specific to their use.[38]

There are also variations possible on any of the approaches described above, each with their own strengths and weaknesses. Thus, for example, a developer may adopt an open core model, but limit the lifespan of its proprietary extensions and release them under open source licenses after a predefined period of time, enhancing their contributions to the open source community, as well as the attractiveness of their open source offering by contributing new functionality on a regular basis, while still delivering value to its paying customer by providing them the "latest and the greatest". A converse approach may also be taken within a value-added service model: In some instances, progress on open source versions may progress too rapidly for those using them in critical production environments, such that the product evolving more quickly than a licensee wishes to migrate. Community support and development may be limited on older versions, providing an opportunity for a developer to offer, as a value added service, extended maintenance and support services, ensuring that older versions of the product can continue to be used reliably over a time frame more suited to such users.

Although we have discussed the above business models within the context of open source licensing, there is nothing that would preclude the application of some or any of these models, or aspects thereof, without the application or use of open source licenses. Examples of this are abundant, such as: (a) a "free but proprietary" model, where one or more products are licensed at no charge, but without disclosing or providing source code[39]; (b) where open source software is used to provide software as a service, for which a charge is levied or otherwise monetized (e.g. through advertising), either with or

---

[38] Michael Olson, "Dual Licensing" in Chris DiBona, Danese Cooper & Mark Stone, *Open sources 2.0: the continuing evolution* (Sebastapol, CA: O'Reilly Media, Inc., 2006) 71 at 84.

[39] For example, Adobe Acrobat Reader or VMware Vsphere Hypervisor.

without[40] releasing source code;[41] (c) "badgeware" where use rights are similar to moderately restrictive open source licenses, but which require that the developer be clearly identified in the user interface; or (d) free non-commercial use licenses with a restriction on commercial redistribution or use.

Although we have discussed the business models above within the context of the software industry, many open source concepts and approaches are also portable to other industries which focus other forms of intellectual property. To some extent, this has already occurred: Open source concepts have been applied to data standards by OpenID and to patents by organizations such as the Open Invention Network and Allied Security Trust. In computer chips and CPUs, Sun Microsystems started the OpenSPARC project and has open sourced the design specifications of its microprocessors under the GPL.[42] Along similar lines, Arduino has released its designs for electronic circuits on an open source basis,[43] while OpenCores has released, through open source licenses, almost 300 types of chip designs, ranging from CPUs to memory cores to communication controllers.[44] Within the realm of research and content, Wikipedia offers a free online encyclopaedia, the content of which is generated through user collaboration and may be modified and redistributed under the Creative Commons Attribution/Share-Alike License 3.0 (Unported) and GNU Free Documentation Licence;[45] O'Reilly has released books under various open source licenses through its Open Book initiative;[46] MIT has released notes, exams and lectures through its OpenCourseWare program under the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 United States license;[47] and the Khan Academy has released over 1600 education lectures on an open source basis.[48] Closer to home, Goldcorp, a Toronto-based mining company, revealed all of its proprietary geology data to the global community of geologists, and in turn received contributions that helped the company excavate over 8 million ounces of gold from its supposedly exhausted mine.[49] In the entertainment industry, the Electronic Frontier Foundation encourages musicians to license their music under the Open Audio License (or more recently the

---

[40] Unless the open source components have been licensed under the AGPL.

[41] Google, Yahoo, Facebook and many others.

[42] http://www.opensparc.net/

[43] http://www.arduino.cc/

[44] http://opencores.org/projects

[45] http://wikimediafoundation.org/wiki/Terms_of_Use

[46] http://oreilly.com/openbook/

[47] http://ocw.mit.edu/terms/

[48] http://www.khanacademy.org/

[49] See Don Tapscott & Anthony D. Williams, *Wikinomics: How Mass Collaboration Changes Everything* (Toronto: Penguin Group, 2006) 7-10.

Creative Common Attribution Share-Alike license) and to list the music on the Open Music Registry as a means to encourage reworking and reissuing of songs to achieve viral distribution. Numerous firms are attempting to apply open source concept to automobile design. [50] Even the food and beverage industry has been influenced by open source, with OpenCola and Free Beer offering their recipes to customers and encouraging them to make their own improvements.[51]

5. CONCLUSION

The intent of this paper was to give a brief overview of open source, open source licenses and the business models through which they may be applied within the context of a commercial undertaking. Open source practices should not be considered either a panacea or something to be blindly adopted, but rather a possible alternative that is, at a minimum, worthy of contemplation in determining how one may wish to exploit software (or for that matter any intellectual property). Given the increasing growth in the use of open source business models and the highly-disruptive impact they sometimes have, it would, in the author's opinion, be highly unwise to ignore them.

---

[50] Such as Trexa (http://www.trexa.com/), Oscar (http://www.theoscarproject.org/), C,mm,n (http://www.cmmn.org/nc/en/home.html), Riversimple (http://www.riversimple.com/) and Local Motors (http://www.local-motors.com/) to name a few.

[51] Although these products were intentionally created to raise awareness of and advocate open source coding, they both received some attention and were able to attract sales.