A License to Swil

Kris Kowal

4th March 2006

Abstract

Software developers have a gamut of options for software licensing. These options range from propriety to permissiveness, including many open source options. This article introduces the attributes and categories of common licenses and presents their strengths and weaknesses. There follows an analysis of how each license attribute and licensing model would serve the author, the public, and authors of derivative works. In particular, this article addresses the licensing needs of a programming language called Swil. The article concludes that a dual licensing scheme including a commercial license and a viral free software license would provide the best compromise for all parties.

1 Facts

I created a programming language I call Swil.[1] As the author of Swil, I implicitly hold a copyright for the text of the program. With that copyright, I can dictate the conditions of my work's distribution and use.[2] The conditions I provide can range in restrictiveness or permissiveness, from retention of complete propriety, to dedication toward the public domain. To keep all distribution

[1] Information about the Swil language and project are available at *cixar.com/swil*.

[2] However, under copyright law, the public can breach those conditions for cases like citation and parody. These cases are called "fair use".

rights of Swil for my own, I may write "all rights reserved" in Swil's copyright notice. In a similar fashion, I can donate Swil to the public domain. Choosing the correct license makes the difference between success or regret. If I chose a highly permissive license, large corporations could easily co-opt my control of the source code, market the results. If I chose a highly restrictive license, large corporations and the software development community could ignore Swil, and could even produce a comparable product to meet their needs.

To provide complex combinations of permission or restriction on Swil's distribution and use to the public, I can specify a license in the copyright notice. Rather than become a legal expert and author my own license, there exist a large number of software licenses that I can employ. These licenses include the MIT, BSD, Academic, Apache, GPL, LGPL, MPL, QPL, and proprietary licenses.[3] I can also apply a proprietary or commercial license. For sake of argument, I consider a commercial license a license for which rights to the use and distribution of the code are retained, but the source code may be available to the public.

Open source licenses provide three primary attributes.

1. Modification is permitted and source code is provided.
2. Distribution is permitted.
3. No warranties are provided. [SSC, paragraph 10]

Additionally, open source software can either be "viral" or "permissive". Permissive software grants the user non-exclusive rights to use, modify, and sell the software without providing strong constraints on how that software is redistributed. Viral software, like the GPL, imposes its own terms to all works

[3] I do not consider the Perl Artistic and Creative Commons licenses contenders for Swil's license. The Perl Artistic license contains verbiage specific to Perl. The Creative Commons Licenses declare that they do not apply to software.

derived from the licensed software. This property is called "viral" by its proponents, to describe the potential exponential growth of the body of GPL licensed software. The property is also called "share-alike" by proponents of the Creative Commons, since it promotes the pro-social attitude of sharing embodied in their licenses. However, the GPL is called "infectious" by agents bent on its demise, and academics trying to avoid the notion that open source software has an association with a computer virus.

> Some free or open software infect other software with its licensing terms. Popularly, this is called a viral license, but the software is not a computer virus. [Vetter]

Cautionary tales of using Open Source software enumerate the softwares that viral licenses can "infect".[4]

1. the original software if re-distributed
2. any modification of the original software if redistributed
3. software containing or integrated with the original software, if redistributed
4. software used in conjunction with the original software to provide a web based service. [SSC, paragraph 11]

I call softwares that include the first and second attribute "Exclusively Open Source". The first, second, and third attribute, I label "Exclusively Open Object".

[4] I prefer the term "viral" because "infectious" bears a strong negative connotation and implies that viral licenses can passively infect software without the author's knowledge or consent, when in fact the author of derivative works must do so knowing the conditions of the underlying software licenses.

1.1 The Contenders

The MIT license provides no restrictions on use and distribution and includes a disclaimer. [AML, 14]

The BSD license augments the MIT license. The license includes a provision that the author's name cannot be used to endorse derivative works [AML, 16]. I call this attribute "non-endorsement". Some version of the BSD license require that all derivative works and advertisements for those works include a notice that their product includes software developed by the author.

The Academic Free License introduces a provision that grants users and authors derivative works permission to exercise any patents embodied in the original work for free, as long as the original author holds those patents. However, the intricacies of patent law may dictate that this is implicit in all licenses [AML, 25].

The Apache license (version 2.0) applies not only to source code, but all other artifacts packaged with the software, including documentation and configuration [AML, 21].

The GNU Public License, known as the GPL, introduces the notion of viral open source. The GPL imposes that any derivative work also be licensed by the GPL. GPL works and derivative works must be provided with source code and permission to modify that source code. However, distributors may package GPL licensed software with proprietary software; the rule of reflexive application of the GPL only applies to works that build on GPL software, not mere neighbors. The GPL also requires that authors of derivative works prominently notify users that they have made modifications from the original. [AML, 38] [GPL] These attributes, I collectively call "Exclusively Open Object", since no proprietary project may use source and object code of this kind.[5]

[5] I will define how this differs from "Exclusively Open Source" in relation to the Mozilla Public License on the following page.

The Lesser GNU Public License, known as the LGPL, relaxes the exclusive open source requirement that all works that use the licensed software be bound by the LGPL. This relaxation applies only to software that uses but does not include the LGPL licensed software. derivative works that include LGPL code, or modify LGPL code, are reflexively bound by LGPL.

The GPL and LGPL implicitly subscribe to a "non-endorsement" policy, since authors of derivative works are required to note that they have modified the original work.

The Mozilla Public License, or MPL, provides a middle ground between the GPL and BSD licenses by requiring that all derivative works must provide the source code for the original work, but does not restrict the licensing terms of the derivative work's object code. [AML, 74] [MPL] This is "Exclusively Open Source" since proprietary vendors may use the object code if they do not modify it and distribute its source. The MPL also provides a "moving target clause", meaning that any software licensed with the MPL is bound by the "current" version of that license at the time of its creation or modification.

License	MIT	BSD	Academic	Apache	GPL	LGPL	MPL	QPL	Proprietary
Disclaimer of Warranty	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Max
Attribution		Yes	Yes	Yes	Yes	Yes	Yes	Yes	NA
Attribution in Advertising		May							May
Non-endorsement	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	NA
Commutes Patents			Yes	Yes	Yes	Yes	Yes	Yes	May
Exclusively Open Source					Yes	Yes	Yes	Yes	
Exclusive Open Object					Yes	May		Yes	
GPL Compatible	Yes	May			Yes	Yes			
Protects Related Documents				Yes					May
Moving Target							Yes		May

[GNU]

Swil is built on Python and two Python libraries, Egenix mx.TextTools, and the Reportlab PDF library. The TextTools library is licensed with the eGenix.com Public License, which I will call the EPL. The EPL is GPL compatible.[6] The Reportlab PDF library is released with a FreeBSD license.[7]

[6]The EPL is available at <http://egenix.com/files/python/mxLicense.html#Public>

[7]Information about Reportlab's licensing is at <http://www.reportlab.com/docs/rlcore-ds.pdf>

2 Issue

How should I license Swil?

3 Arguments

In the following sections, I will consider the supporting and opposing arguments regarding the generic attributes of these software license contenders, and then each licensing model including "permissive", "viral", "proprietary", and "dual" licensing.

3.1 License Attributes

As noted in the table on the page before, the attributes include warranty, attribution, attribution in advertising, non-endorsement, commute of patents, exclusive open source, exclusive open object, GPL compatibility, application to related documents, and a moving target clause.

3.1.1 Warranty

All of the contending licensing options include broad disclaimers of warranty. However, the Software Engineering Code of Ethics states:

> Software engineers shall, as appropriate, accept full responsibility for their own work. [SECOE, 1.01]

Ideally, I would produce a license that communicates that I would take responsibility for failure of my product for its implied purpose. However, Swil is a general programming language which implies that it is fit for whatever purpose you find use for it. This over-broad purpose could invite litigation against me for failing to meet many particular implied purposes. If I release Swil to the

public free of any charge, I would not have earned any remuneration from which to repay users. In this sense, draconian licenses, even when inapplicable, serve to protect the author from generalist attacks.

The Software Engineering Code of Ethics promotes an honest warranty. The code contends that I should not include broad statements that suggest that I am indomitable for the actions of my software.

> Software engineers shall, as appropriate, be fair and avoid deception in all statements, particularly public ones, concerning software or related documents, methods and tools. [SECOE, 1.06]

All of these warranties deter legal action by staking claim that litigation can not be brought against the author. This is not true, in extenuating circumstances.

Rather than claim that it is the user's responsibility to determine whether software is safe or fit for their purpose, the Code of Ethics recommends that the license should be forthright about any dangers of using the software.

> Software engineers shall, as appropriate, disclose to appropriate persons or authorities any actual or potential danger to the user, the public, or the environment, that they reasonably believe to be associated with software or related documents. [SECOE, 1.04]

However, the nature of offering software to a developer community is one of admission that the software will be used in components that are beyond the scope of the original intent, requirements, and design. Having these draconian clauses reminds developers that they are responsible for reviewing the code and performing appropriate testing to assure that it fits their projected need.

3.1.2 Attribution

Many licenses require that all derivative works credit the original author and other contributors for the portion of the code that they developed. This often

takes the form of a clause that insists that no credits be removed from the source code in redistributions or derivative works. Also, these licenses include a clause that the names of the original authors cannot be used to endorse derivative works.

I should use a license with an attribution clause.

An attribution clause would protect the reputations of Swil's contributors including myself, and guarantee that I will be credited for my work.

I should not use a license with an attribution clause.

Attribution clauses pose an inconvenience for developers of derivative works. Swil's proliferation might be hindered by software developers who are unwilling to use Swil because they cannot "clean" the headers of the source code.

3.1.3 Attribution in Advertising

Some licenses include a clause that require the logo of the author or original organization be prominently displayed in all advertisements for redistributions and derivative works.

I should use a license with an attribution in advertising clause.

Using an attribution in advertising clause would permit me to propagate the Swil brand and would possibly promote proliferation of the Swil source code.

I should use a license with no attribution in advertising clause.

Attribution in advertising clauses also are inconvenient for authors of derivative works. Considering Kant's categorical imperative, we must analyze the world that would exist if all software licenses included an attribution in advertising clause. All advertisements would be required to include the logos of every

organization that contributed to a derivative work. After prolonged application of this rule, advertisements for software would be covered in logos. Such noise would render the advertisement ineffective. Because the advertisement would no longer advertise, there is a contradiction and the test of the categorical imperative fails. While this is tenuous theoretically, the pragmatic implications are strong. Authors of derivative works would be reluctant to use software licensed with such a clause because of the burden on advertising.

3.1.4 Commute of Patents

Some licenses commute the right to exercise patents the author holds.

> *Whether I use a license that includes a commute of patents clause is irrelevant.*

I hold no patents, but I may in the future. The Academic license explicitly commutes patent claims, while other open licenses can implicitly do so under the conditions of the patent. Proprietary licenses sell the right to use patented behavior. This renders whether I use a license that commutes the right to exercise my patents irrelevant.

3.1.5 Exclusively Open Source and Code

I will address these attributes of a license in section 3.2.3 on page 16, regarding the licensing model implied by these attributes.

3.1.6 GPL Compatibility

GPL compatibility would permit other software developers to combine derivative works of Swil with GPL licensed code.

Licensing is a two way street. I need to be sure that my licensing decisions are compatible with the licenses of libraries that my program uses. This is clear

in the Software Engineering Code of Ethics.

> Software engineers shall, as appropriate, not knowingly use software that is obtained or retained either illegally or unethically. [SECOE, 2.02]

I should use a GPL compatible license.

Using a GPL compatible license would permit Swil's source code to be used by developers of free software.

I should not use a GPL compatible license.

Regarding the licenses the libraries that Swil builds on, both are GPL compliant. However, the Egenix people concede that Richard Stallman could contest that the Egenix mx.TextTools library is not GPL compliant because it contains a "choice of law" clause, meaning that the license could be interpreted in the context of the laws of a hypothetical state, "Unfreedonia", where the provisions of the GPL are not admissible. If this is the case, I would not be able to use the GPL.

3.1.7 Application to Related Documents

Some licenses include a clause that explicitly applies the license to all code related documents including the manual.

Whether I use a license that applies to related documents is irrelevant.

Since I can separately apply any license to all of the components of the Swil software package, the provision that applies a license to all artifacts of the software package is irrelevant.

3.1.8 Moving Target Clause

Some licenses include a clause that provides that redistributions of the licensed software must be licensed with the most current version of that license. This means that the authors of derivative works would have to consider the most current version of the code every time they release their work.

I should use a license that includes a moving target clause.

Using a moving target license clause would permit to me to revise my license to account for emerging legal details. If a major court ruling made my original license ineffective, I could effect a change to the license to account for the new legal precedent.

I should not use a license that includes a moving target clause.

Providing a moving target license would permit me to further constrain the conditions of redistribution and derivative works at whim. This attribute would potentially deter proliferation of the Swil source code in derivative works, by intimidating prospective contributors.

3.2 License Models

I am considering three licensing models: a permissive license, a viral license, a proprietary license, or a combination of a viral license and a commercial license.[8] I could use a permissive license like the BSD or MIT software licenses. I could use a viral license the GPL or MPL. I could also use a combination like the QPL, providing a commercial license at cost for complete use and a viral license for the open source community to develop a body of free software.

[8] I am not considering a commercial license combined with a permissive license because the availability of a permissive license would eliminate the motivation to purchase the commercial version.

3.2.1 Open Source Licenses

Permissive, viral, and dual licensing schemes are open source. A viral license tends to favor the growth of a body of viral open source software. Permissive licenses favor commercial and open source software equally. Dual licenses would provide a commercial and viral option for users.

I should use an open source license.

Eric Raymond argues that open source software makes a business more healthy by enriching the market. Even though providing open source software will help your competitors, it will help you proportionally and will attract more activity in your field. Raymond calls this "Growing the Pond".

> Sometimes the smartest way to become a bigger frog is to make the pond grow faster. This, of course, is the reason technology firms have participated in public standards–and it's useful to think of open-source software as an executable standard. [ESR, 150]

The Software Engineering Code of Ethics promotes the authoring of open source software.

> Software engineers shall, as appropriate, be encouraged to volunteer professional skills to good causes and contribute to public education concerning the discipline. [SECOE, 1.08]

By providing Swil to the public with an open source license, viral or otherwise, I would be contributing to the profession's body of knowledge, permitting the Swil code base to be augmented and analyzed in research.

I should use a proprietary license.

Proprietary vendors argue that using open source licenses opens the door for three legal risks:

1. Exposure to faults and intellectual property claims.

2. Disclosure of confidential code.

3. Inability to commercialize. [SSC, paragraph 5]

If I use an open source license, users of Swil could catch legal action that could not be taken against me because of disclaimer of warranty. For example, if Swil code embodies the ideas expressed in a current patent, people that use Swil could be sued. If I use a viral license, companies could be required to expose secret algorithms in their software if they unscrupulously distribute Swil as component. Also if I use a viral license, companies would not be able to use my code in a proprietary product.

This could lead players in industry to be hesitant to use Swil, which would hinder its proliferation. Like many services, Swil's value is likely to stem from the size of the network of users and tools.

3.2.2 Permissive Licenses

Permissive licenses include the MIT and BSD licenses. They are called permissive because all entities, from proprietary software developer to GPL proponent, have permission without restriction to use the licensed software.

In an interview with Linux Today, Guido van Rossum discussed why he chose a BSD license over a GPL license. Regarding whether using the GPL would attract or deter developers, Guido suggests that it would have both effects. Proprietary developers would not be inclined to contribute to a GPL code base because they would lose the right to use their contributions in proprietary software. Idealistic proponents of "Free as in Freedom Software" would be less inclined to contribute because their code would no longer contribute to their growing political arm.

> [Using a BSD-style license] may deter GPL hardliners (but there seem to be few of these in the Python world). But it attracts developers from the proprietary world like I mentioned above. Many of these "proprietary" companies are major contributors to Python and other open source products. For example the new Unicode support and regular expression engine, as well as several existing core library modules, were contributed by people who also develop proprietary Python software.[GVR]

If I use a permissive license, I fail proponents of building the body of free software and keeping it free for everyone. If I only use a viral license, proprietary vendors would be reluctant to contribute relevant code to the main code base because they would no longer be able to use it in their proprietary products.

I should use a permissive license.

Andrew St. Laurent contends that permissive licenses would help proliferate Swil. A "Research-style" license would engender widespread adoption. Companies intending to build tools that use Swil languages could freely adopt an MIT or BSD licensed Swil implementation. The license would be compatible with any business model, from providing Swil related services, through branding and packaging Swil or related products, to writing proprietary software built on Swil and selling it on the shelves of the local office supply store. Assuming that Swil is worthy of use, companies with disparate agendas would have little reason to hesitate embracing the technology.

> Research-style licenses, like the BSD and MIT Licenses, are ideal for situations in which you want wide deployment of your ideas and do not care whether this results in open source software or proprietary software.[AML, 31]

Again, using a permissive license would invite vendors to contribute to the main project. Vendors would be motivated to back-contribute like this to provide a standard way to interact with applications that would happen to be compatible with theirs. It would also alleviate the costs of keeping a customized version of the code, and would promote peer review.

I should not use a permissive license.

If I use a permissive license, I could easily lose control of the code base. Another party could take the code wholesale at no expense and start developing it in a new direction without my consent.

Also, proponents of Free software would not be able to corner the vendors into the open source market with Swil.

3.2.3 Viral Licenses

Viral licenses, like the GPL, LGPL, MPL, and QPL, constrain all derivative works to their terms. In this section I will present the opposition first because it segues into the supporting argument.

I should not use a viral license.

The primary attribute of the GPL is described three different ways. It is commonly called "viral" by its proponents, to describe the potential exponential growth of the body of GPL licensed software. This attribute is called "share-alike" by proponents of the Creative Commons, since it promotes the pro-social attitude of sharing embodied in their licenses. Richard Stallman calls the attribute "Copyleft".

> Copyleft uses copyright law, but flips it over to serve the opposite of its usual purpose: instead of a means of privatizing software, it becomes a means of keeping software free. [RMS]

However, the GPL is called “infectious” by agents bent on its demise, and academics trying to avoid the notion that open source software has an association with a computer virus.

> Some free or open software infect other software with its licensing terms. Popularly, this is called a viral license, but the software is not a computer virus. [Vetter]

Governments may be reluctant to adopt software based on the GPL. Pamela Jones cites the New Zealand State Services Commission.

> A "strongly infectious" open source licence will infect any redistributed piece of software that contains or is derived from software licensed under it. It is generally very difficult to modify or integrate software licensed under a strongly infectious open source licence without the resulting product, when redistributed, becoming "open source" on the same terms as the original. The GPL is an example of a strongly infectious open source licence. [SSC, paragraph 15]
>
> *I should use a viral license.*

In response, Pamela Jones argues that the SSC’s recommendations are irrelevant because proprietary software does not provide a better alternative.

> At least the GPL allows you to use the code, modify it and redistribute at all. Let’s think for a minute, something FUD purveyors hope you will never do. Suppose you redistribute modified Microsoft software instead. What will happen to you? That’s right. They’ll sue your pants off.
>
> So I’d say if FOSS is "infectious," Microsoft is flesh-eating bacteria.[PJ]

Jones appears to assume that the SSC's recommendations favor proprietary solutions. Permissive software does not have the risks of viral software that the SCC mentions, nor the restrictions of proprietary software that Jones mentions.

Viral "share-alike" software harmonizes with the pro-social Software Engineering Code of Ethics. Richard Stallman notes that the "free" in "free software" means "freedom", not "absolutely cheap", the former being a *human value* and the later being *valued by humans*. The Code of Ethics infers promotion of causes that increase freedom.

> Software engineers shall, as appropriate, temper all technical judgements by the need to support and maintain human values. [SECOE, 1.04]

By making Swil available with a viral software license, I would provide a tool for human kind, which could aid economically disadvantaged people and developing nations. These people would be able to begin a business and improve their welfare. By providing Swil, I could provide a means for leveling the economic playing field. Building open source software is like throwing money in the air. Anybody can catch it, but those who have the least benefit the most.

3.2.4 Proprietary Licensing

A proprietary license would permit me to sell all rights to use and duplication of Swil.

> *I should use a proprietary license.*

A proprietary license would permit me to have exclusive rights on monetization of Swil.

> *I should not use a proprietary license.*

In an online conversation among Ryan Paul, Ryan Witt and myself, Ryan Paul[9] enumerated my constraints succinctly:

1. I don't have a marketing budget.
2. I don't have an innovative product. I have a vast improvement on existing ones.
3. Swil does not have inherent hype value.
4. No company will buy it or invest in Swil because they can produce something comparable at substantially lower cost.
5. Swil cannot be packaged and sold on shelves. [KPW]

The second point highlights that Swil's success operates in a narrow margin. While I'm sure that people will find it a pleasure to work with Swil, its economic portent is shadowed by numerous existing solutions, many of which are ubiquitous, free, open standards. Swil's success depends on its own merit and cannot be sold.

> Distributing Swil commercially would stunt its growth. [KPW]

3.2.5 Dual Commercial and Viral Licenses

A dual licensing scheme involves a commercial license and a viral license. The viral license would provide the free software community an opportunity to develop a body of free software as long as that software and all derivative works remained free. I would sell the commercial license for non-exclusive rights to use Swil in proprietary applications. Both licenses would be open source, differing only in cost and permissions granted.

[9] The Ryans Paul and Witt are personal friends and co-conspirators. Ryan Witt is an undergraduate in Computer Science at CIT. Ryan Paul is a journalist for the online journal, Ars Technica.

I should use dual licenses.

A dual licensing scheme exhibits that notion inherent to the biblical quote:

> Render therefore unto Caesar the things which be Caesar's, and unto God the things which be God's. [KJV, Luke 20:25]

Or, rendered analogously:

> Give to commerce what belongs to commerce, and give to open source what belongs to open source.

Using a dual license would permit me to play by commercial rules and by the rules of viral open source simultaneously. All of the arguments for and against using viral licenses apply. The licensing scheme is not strictly proprietary, since the source code is still open.

Using a dual license, I would require proprietary vendors to pay for the privilege of creating proprietary derivative works.

I should not use dual licenses.

Using a QPL license would encourage the open source world to "fork" Swil. Anybody could take the free, GPL version of the code for the purpose of an "embrace and extend" attack on the original branch of Swil. These developers would add desirable or critical features which would subsequently be virally licensed under the GPL. The authoring organization would not be permitted to include these features in their commercial distribution.

However, since we're dealing with copyright and not patents, it would be perfectly legal for me to duplicate the effort of the forked community or organization to add these features independently and so avert loss of my business model. This does not preclude that I would have to combat "embrace and extend" attacks against my branch of the Swil source code. I believe this would provide motivation to continuously innovate.

4 Analysis

Considering the attributes of open source software, I find that they should include an honest warranty, an attribution clause, and should not include an attribution in advertising clause, nor a moving target clause. I desire credit for my work and that credit does not pose a significant cost to the public or developers and contributors. Since these licenses imply a commute of patents clause, whether the license explicates that notion is irrelevant. The license should be GPL compatible if possible, but this is not extremely important. I can afford to lose the support of an occasional free software zealot, but I would rather have it. An application to related documents clause is irrelevant since I can apply licenses to those documents independently, including a Creative Commons.

Since I require an attribution clause, I strongly prefer the BSD over the MIT license. This I further support because the particular BSD license does not require an attribution in advertising clause, and can be GPL compatible.

Swil does not fit in the niche provided by the LGPL; I do not have object code that would better be protected by the LGPL than the GPL.

Considering the remaining licenses, the GPL and BSD are the only GPL-compliant options, but the importance of this is tenuous. Choosing between the GPL and BSD hinges on whether I would like to use a viral license model. If I forgo this requirement, the Apache, Academic, MPL, QPL and proprietary licenses are options.

I've assigned some scores, from zero to ten, for how each licensing model ("permissive", "viral", "dual commercial and viral", and "proprietary") serve the interests of various parties ("self", "public", and "permissive", "viral", and "proprietary" developers).

	Self	Public	Permissive Developers	Viral Developers	Proprietary Developers
Permissive	3	10	10	5	10
Viral	3	9	4	10	0
Dual Commercial and Viral	7	9	4	8	8
Proprietary	10	1	0	0	5

I've avoided considering game theory to truncate circular reasoning. For example, using a proprietary business model serves in my self interest very well, but serves me not at all if I consider that the public and other developers would shun this model and I would in the end not be served at all.

Considering the Software Engineering Code of Ethics, it would be unethical to reserve too much value to my own well being at the expense of providing software that would benefit the profession as a whole.

> Software engineers shall, as appropriate, not promote their own interest at the expense of the profession, client, or employer. [SECOE, 6.05]

For this reason, I will consider the values to self as half of their stated merit.

Again in accordance with the Code of Ethics, I consider other developers equally important to the value of the public since most of the value to the public would be through other developers translating Swil's functionality for their particular needs. In most cases, Swil will not benefit the public directly.

> Software engineers shall, as appropriate, moderate the interests of the software engineer, the employer, the client, and the users with

> the public good.[SECOE, 1.02]

Again, considering the requirement that Swil be compatible with the underlying licenses of its libraries, Swil can be GPL compatible because the EPL and FreeBSD are GPL-compatible. Since all of these licenses are permissive, I can apply any additional licensing models. Python's source code is not GPL compatible because it contains a "choice of law" clause, but Swil does not build on Python's source.

I consider adoption of Swil for personal and commercial use sufficient acknowledgement of my work. To maximize this effect, I need a license that would permit widespread proliferation of the software. Authors of proprietary and commercial software, permissive and viral open source software alike need feel free to take up Swil as a component of their projects.

Authors of proprietary and commercial software require software that they can integrate in their products. Permissive open source licenses and commercial licenses meet this end. Authors of viral software require that Swil's license be compatible with their viral license, albeit GPL-compliant. Authors of permissive software require that their projects remain permissive.

5 Conclusion

A dual licensing scheme, including a commercial license and the GPL, provides the best compromise among the interests of the public, developers, and myself. The public would benefit from a de-facto standard implementation available for free use that would encourage the development of free derivatives. The scheme would only discourage development of permissive derivatives. Proprietary software vendors, presumably having the capital to spend, would be able and required to pay for the right to incorporate Swil in proprietary derivative

works.[10]

Baring a dual license scheme, a BSD license provides the best net value for the interests of the public, developers, and myself. I would favor this license scheme if I deemed it more important to proliferate the software than promote free software and my own interests.

References

[AML] *Understanding Open Source and Free Software Licensing* by Andrew M. St. Laurent (O'Reilly 2004). Andrew M. St. Laurent describes the concepts of open source and free software licensing and analysis several licenses in detail.

[BP] *The emerging economic paradigm of Open Source* by Bruce Perens. First Monday, Special Issue #2: Open Source (October 2005), URL: http://firstmonday.org/issues/special10_10/perens. Bruce Parens describes how open source software drives a capitalistic business model.

[ESR] *The Cathedral & the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary* by Eric S. Raymond. O'Reilly (January 2001). Eric Raymond presents a collection of essays about Linux and Fetchmail to illustrate how and why open source software works.

[GNU] *Various Licenses* by the GNU project. URL: http://www.gnu.org/philosophy/license-list.html

[GPL] *The GNU Public License v2.0* by the Free Software Foundation. A viral software license designed to build a body of free software.

[10] This conclusion is tentative and not legally binding. If you want to know the final legal provisions of Swil, please consult the copyright notice of the version you hold.

[GVR] *Guido van Rossum Responds to Python Licensing Issues*. Kevin Reichard. Linux Today. 2000. Guido response to questions about Python's licensing scheme and the issues that have shaped it.

[KJV] *The Catholic Bible*, King James Version. Provides illumination for millions of Christians, and apt quotes for at least one atheist.

[DOS] *Open Sources: Voices of the Revolution* by Chris DiBona, Sam Ockman, Mark Stone. O'Reilly (1999). Chris DiBona et al introduce a series of essays by the prolific heralds of open source software.

[KPW] *Cixar Meeting Transcript*. Kris Kowal, Ryan Paul, Ryan Witt. 2004. URL: http://cixar.com/~kris.kowal/doc/meeting-transcript-2004-10-19.html. The Ryans and I discuss ways to monetize Swil.

[LGPL] *The Lesser GNU Public License*. A version of the GPL that makes a concession for proprietary software that could use the licensed software as a shared library.

[MPL] *Mozilla Public License*. A viral software license that make a compromise for proprietary software developers.

[OSD] *Open Source Definition* by Bruce Parens. URL: http://opensource.org/definition.php

[PJ] *Groklaw* by Pamela Jones. URL: http://www.groklaw.net/article.php?story=20060301213210833. Pamela Jones discusses a conflict of interest in the State Services Commission in New Zeland regarding consultation about open source software.

[RMS] The GNU Operating System and the Free Software Movement by Richard M. Stallman. Richard Stallman discusses the history and purpose of the Free Software Foundation.

[SECOE] *Software Engineering Code of Ethics* by the ACM and IEEE.

[SSC] *Guide to Legal Issues in Using Open Source Software*. State Services Commission, New Zeland. URL: http://www.e.govt.nz/policy/open-source/open-source-legal. A consulting group which lists both Microsoft and the New Zeland government as clients, makes cautions the New Zeland government of the perils of open source software.

[Vetter] *"Infectious" Open Source Software: Spreading Incentives or Promoting Resistance?* by Greg R. Vetter. Rutgers Law Journal Vol. 36:53.