

Title:

Title:

Title: A2.4-D1 Software distribution strategies and business models

Author(s)/Organisation(s):

Michael Printzos / PRO; Armantas Ostreika / KTC; Guillermo Schwartz / LOGCMG

Working Group:

2 (Cost and Process Analysis)

References:

Short Description:

Report on "Software distribution strategies and business models" gives an overview of different software distribution strategies and analyzes their suitability to HUMBOLDT infrastructure. The main attention is paid to the open source business models as the open source methodic is expected to be mostly acceptable solution for the ESDI software distribution strategy.

Keywords:

Open/closed source software distribution methods

History:

Version	Author(s)	Status	Comment
001	Michael Printzos, Armantas Ostreika	new	First draft of A2.4.D.1 report
002	Guillermo Schwartz	rfc	Section 5 supplemented; conclusions extended
003	Armantas Ostreika	rfc	Minor corrections made

Title:

Table of contents

1	Objectives of This Report	4
2	Introduction.....	4
3	Open source models emphasize on use value	5
4	The Open Source/Closed Source Trade-off.....	6
5	Software Licensing	7
5.1	Non-Open Source license.....	7
5.2	License free software.....	7
5.3	Copyleft free software license.....	8
5.4	GNU General Public License	8
5.5	Free linkable software license.....	8
5.6	BSD License and BSD-style Licenses	8
5.7	Artistic License	9
5.8	Mozilla Public License	10
5.9	Non copyrighted licenses.....	10
5.10	Public domain software.....	10
6	Open Source Business Models.....	11
6.1	Support Sellers Business Model	11
6.2	Loss Leader Business Model.....	11
6.3	Widget Frosting Business Model	12
6.4	Accessorizing Business Model	13
6.5	Service Enabler Business Model	13
6.6	Sell It, Free It Business Model	14
6.7	Brand Licensing Business Model.....	14
6.8	Software Franchising Business Model.....	15
6.9	Hybrid Business Models	15
7	Can be HUMBOLDT as an Open Source?	18
8	Conclusions	19
9	References	20
10	Annexes	22
10.1	Annex 1: Comparison of software licenses	22

Title:

10.2 Annex 2: Cornerstones of Intergraph's Humboldt Software Distribution Strategy25

1 Objectives of This Report

In order to make HUMBOLDT infrastructure to be self-sustaining for a long period proper software distribution and support strategies should be selected. Objectives of this report are to make an overview of well known software distribution strategies, licensing and business models, to make preliminary conclusions and suggestions about suitability of particular software distribution strategies for HUMBOLDT project software products and for the whole framework.

As final set of HUMBOLDT software components is not determined at this stage of the project just preliminary recommendations for software distribution strategies are given. Adjective attention is paid to the open source business models as the open source methodic is expected to be closest to a fast-track solution for the ESDI software distribution strategy.

Supplemented by new data obtained during ongoing developments of the project this report will be included as a separate section in the final report of WP2: A2.1-D2 Report on Processes Analysis. Notes and suggestions derived in this overview will serve as starting point for further software distribution strategies development in work package 11 „Exploitation and Dissemination“.

2 Introduction

Software distribution strategies have many aspects and can be analysed by reviewing different incisions of a strategy. We can distinguish three main groups of aspects: technical aspect, logistical aspect or cost effectiveness and licensing aspect.

Technical aspect involves networking, distribution technologies e. g. server infrastructure, distribution methodologies (online, physical: CD creation, printing material), distribution software automation technologies, etc.

Logistical aspect involves inventory track of the distribution line, distribution resources (online and physical), marketing and publicity issues and organisational aspects.

Licensing aspect involves licensing strategy, licensing risks, licensing enforcement costs. Furthermore, by formulating licensing strategy main technical and logistical aspects are predicted also.

The mentioned technical and logistical aspects specifically to this project are related not only to software products but also to HUMBOLDT infrastructure as the main product of the project and will be developed in „Exploitation and Dissemination“ work package. Therefore in this report we will concentrate on software distribution strategies in licensing context.

To go for open or a closed source software distribution method is currently one of the most debated topics in the software industry. At a most basic level of definition the term “open source” software simply means software for which the source code is open and available. The term “open” refers to software where its source code can be seen and be modified by everybody. Availability implies that anybody can have access to this code either free of charge or at a nominal fee (usually the latter covers media and shipping charges and/or online charges).

For systems that are already in operation for few years open system solutions should be considered when equipment or software is becoming obsolete or when maintenance costs are becoming unaffordable or when a more price-competitive solution is required [1]. However, within recent years there has been a considerable demand and a considerable boost in the production of new open source software [2, 3]. The key factor of aroused interest to such products is lower costs (total cost of ownership and lower acquisition costs). Sometimes the open source concept is also pushed by

Title:

political decisions which tend to reduce the presence of proprietary owned software in governmental agencies and other public bureaus in order to reduce administration costs.

The power and success of open source software is often seen when the users addressed are the so called “sophisticated users” and not the general public. With this term we mean the system and server administrators or more generally those that are experienced enough in order to handle different services that computers nowadays offer.

A really important factor that needs to be taken into account when deciding which software distribution method to follow for HUMBOLDT is the different segments and typologies of users that HUMBOLDT will be addressing after it is implemented and integrated. As pointed out above the open source software has been particularly successful in those segments of the market where the potential demand is made of sophisticated users. On the contrary in the mass-diffusion segments where the demand typically comes from less sophisticated customers who are generally not informed about the existence and/or the characteristics of the various packages, open-source software have hardly gained relevant market shares. A typical example is OpenOffice, a suite which has been created in direct competition with Microsoft Office in the market for office applications. According to different reviews it was very difficult for OpenOffice to gain a bit significant share of the market in spite of the fact that its performances are comparable with those of the rival software and that it is supplied for free.

3 Open source models emphasize on use value

Free and open-source software (known as FOSS) is dramatically affecting the evolving information and communications technology landscape [7]. The main cause is its unusual economical behaviour. To understand the economic model of FOSS development let's have a look at the example provided in [4]:

“If developer X devotes 5 days a month to developing FOSS product A, perhaps she/ he (or the employer) lost \$5,000 of opportunity cost in consulting work or salary. But she/ he knows that 100 other developers are doing the same. So this \$5,000 investment has just yielded a return in software now worth \$500,000. The ROI of the FOSS model should make an VC envious.

As a side effect, even people who don't develop the software, that is who don't contribute directly, can get to use the software as well, most often without any payment. Proponents of proprietary software find this “free-rider” aspect most disturbing. But a rational economic model requires that one only calculate the benefit to oneself. The \$5,000 investment yields the \$500,000 software needed today, which without that investment would not have existed. It is irrelevant to the investor that tomorrow others who didn't invest the \$5,000 also benefit. In fact, investors do benefit from the “free-riders”, since these find bugs and test the software in varied environments, helping it become more secure and robust over time. So “free-riders” contribute value, although of a lesser sort. Finally, many investors start as “free-riders,” so they are an important part of the growth of the model.”

In summary, proponents of open source software business models argue that, at least sometimes, it is possible to extract equal or greater economic benefit from a tool's value as an intermediate good (its use value) as from its value as a final good. In other words open source business models tend to maximize use value by gaining input of many users and second to extract economic benefit from that increased use value.

Offering HUMBOLDT as open source solution is based on the fundamental assumption that harnessing the input of many users will at the end improve the use value of the HUMBOLDT infrastructure itself.

Title:

A list of factors contributes to the use value of any tool developed under the open source concept. Improvement in the value of a tool to its users can result from improved understanding how the tool works or from improved skill on the part of the user. These can further influence improvements in the tool's quality, including improvements to accuracy, reliability, versatility and/or suitability for a specific task, interoperability, or adaptability/robustness to changes in the working environment. The importance of any of these factors of use value in relation to a particular tool will depend on the nature of the tool and the specific circumstances of its users.

To put in other words apart from improved understanding, quality and availability, improvements in use value sometimes result directly from an increase in the number of users of a given tool. By guaranteeing unrestricted access and attracting new users through improvements in availability and quality, the open source approach for HUMBOLDT may maximize the positive network externalities associated with the platform itself.

4 The Open Source/Closed Source Trade-off

When there are no opportunities for distribution improvement the open source approach is likely to bring any benefit. Even where starting conditions are favorable and there is a potential to make gain or avoid losses by adopting an open source approach, open source is not necessarily the best way to go. There are costs and risks associated with the open source strategy, including the costs of providing a community infrastructure, launching and maintaining the project, and the opportunity cost of foregoing license fees and/or the competitive edge that may be gained by denying competitors free access to an important new technology.

In one sense the risk associated with open source is arguably higher than the associated with more standard business models because of the uncertainties involved in a novel approach. On the other hand there are also costs and risks associated with the traditional approach. The decision whether to go for open source is thus a trade-off, the outcome of which will depend on the specific circumstances of the case.

In order to define these circumstances should be answered to the series of questions which are effectively more or less the same as in any business plan, and can be answered by much the same kind of research. For example and in our case the exact and unique value proposition of HUMBOLDT has to be investigated and then clearly defined to all parts. In the case of software development the final tool developed is more likely to do well under open source conditions if:

- The tool establishes a common infrastructure. In other words the network effects in relation to the particular tool are strong
- The tool is business critical to its users. This means customers will place a high value on the prospect of avoiding dependence on different suppliers of the same line of services.
- The reliability, availability and scalability of the tool are very important. The assumption here is that open source development methods tend to produce better tools than do traditional proprietary development methods.
- Peer review is needed to verify the correctness of design or even implementation
- Key methods of functional equivalents of key methods implemented by the tool are part of a common technical knowledge in the field. This suggests the closed source approach would not necessarily be highly profitable in any case.

Title:

- The tool is an enabling technology or a scientific resource that represents a non-substitutable standard.

The final point shows another interesting characteristic as well. It shows that the outcome of the trade-off may well change over the lifetime of a tool: early on in some cases it may be better from a purely economic perspective to keep it proprietary; later it may be better to let it go open source. In this regard it is worth noting that many owners of patented research tools allow patent protection to lapse over the economic life of the tool; open source licensing could be seen as an alternative to allowing the tool to return straight into the public domain, and the question then to the owner would be whether the advantages expected from an open source approach would make up for the cost of the yearly patent fees.

5 Software Licensing

Business strategies on software distribution are directly correlated to the license associated with the software. Depending on the business model that the owner of the software chooses, and hence the granted license, the “buyer” will have more or less freedom on the way he can manipulate the code, distributed the new versions, and sometimes even sell them as a new product. There are two types of limit strategies one were the acquisitioner of the software does not have any manipulation or distribution rights over the software, proprietary license, and the opposite where there is no restriction whatsoever as to distribution or manipulation of the software. In between those extremes there is a myriad of licenses with different levels of restriction (see Annex 1).

In any case what we also have to remember that licenses are in effect a document by which the acquisitioner of the software accepts losing his rights as described by the copyright law where he got the software in exchange for the rights and duties described in the license.

In this section we have tried to give a classification on the types of licenses. Albeit imperfect since it does not take into account all the subtle differences between them, at least it gives a general idea of the main characteristics of most of them.

5.1 Non-Open Source license

This is the well known restrictive license used exclusively until the start of the open source movement in the 1990s. It excludes all manipulation, transformation, distribution or reselling of the code.

The better known of these licenses is the proprietary license, but softening some of the definition terms of the license we can include other types. For example the freeware license that allows redistribution but not transformation or reselling. Or the Microsoft reference license which allows view of the code but not its modification or redistribution.

5.2 License free software

Contrary to what its name might indicate the software without license does not lack vendor’s associated rights over it. By not giving any license to the software but claiming copyright over it, the vendor’s rights fall back to the copyright law of the country where the software was sold or given. So the amount of things that can be done to or with the software depend on how restrictive the copyright law of the country is.

5.3 Copyleft free software license

The copyleft free software license, classification under which falls the GNU General Public License [5, [i11](#)], allows free access to the code, allows its manipulation and its redistribution either of the original or the transformed code. This license is to a certain extent also restrictive since it does not allow either linking the code with proprietary software or re-packaging it under another license. So for example repackaging the GPL License software in proprietary software is forbidden.

One of the draw backs from the point of view of the open source movement, of most of this type of licenses is that they do not force the developer that changes the code into a new product to publish those changes. In the traditional distribution world this would not be a problem since to run the software first you would have to compile in a local machine, or at least in principle you would have the right to back engineer and redistribute the new software. But in the age of the Internet big companies can use open source software to provide services without disclosing the code (e.g. Google, Yahoo).

5.4 GNU General Public License

The GNU General Public License (GNU GPL or plain GPL) [5, [i11](#)] was created by Richard Stallman and the Free Software Foundation [[i12](#)]. The GPL in many ways occupies a place at the opposite end of the spectrum from BSD-style licenses (see BSD License below): Where BSD-style licenses permit essentially unlimited commercial use of open-source software and essentially unrestricted creation of proprietary derivative works, the GPL is explicitly designed to prevent open-source software from being used to create proprietary derivative works. It does this through "copyleft" [5, [i13](#)] provisions in the GPL that require:

- that programs licensed under the GPL must be distributed without a license fee and with source code made available; and
- that derivative works of a program licensed under the GPL must also be licensed under the GPL.

Additionally we can note that GNU GPL licenses are very often used for GIS related software projects [[i36](#)]. Such license use leading solutions like GRASS (Geographic Resources Analysis Support System) [[i37](#)], UMN Mapsrever (an Open Source development environment for building spatially-enabled internet applications) [[i38](#)], etc.

5.5 Free linkable software license

The free linkable software licenses main characteristics is that it allows the code to be used as part of proprietary software or to link the open source software to proprietary software. The proponents of the Open Source software are philosophically against this practice because they think it goes against the basic principle that would allow the efficient creation of robust software by a public group of developers. Examples of these licenses are the Apache [[i6](#)] license or the BSD License.

5.6 BSD License and BSD-style Licenses

The BSD License [5, [i1](#)] was originally used for the UNIX distributions released by the University of California at Berkeley. ("BSD" stands for "Berkeley Software Distribution.") Since then the BSD License or licenses adapted from or similar to it (including the original MIT and X Consortium License [[i2](#)]) have been used for several other open-source projects, including FreeBSD [[i3](#)], NetBSD [[i4](#)], and OpenBSD [[i5](#)] (Unix-based operating systems), Apache [[i6](#)] (a web server), SLAPD (an LDAP-based

Title:

directory service) [i7], XFree86 (an implementation of the X Window System) [i8], and SATAN (a network security analysis tool) [i9], to mention only a few. The BSD License has the following main features:

- an explicit grant of the right to unlimited use in source or binary form
- a requirement that the developers' copyright notices (and related material) be retained
- a requirement that the developers be credited in "advertising material"
- legal boilerplate to limit the developers' liability

Some licenses derived from the BSD License (like the original X Consortium License) omit the advertising requirement (which some have criticized [i10]); others add a provision requiring that the software be provided "at cost."

As noted above, the original BSD License was developed in order to release non-commercial software developed as a by product of university research, and its legal provisions reflect this heritage: they affirm the academic tradition of giving proper credit to researchers (i.e., the developers) and safeguard the basic legal interests of the university (i.e., the organization employing the developers), but otherwise impose no real restrictions on use of the software. From the point of view of a commercial software company BSD-style licenses contain the minimum terms and conditions that an open-source license would need to have in order to be an effective license at all; from the point of view of open source developers BSD-style licenses allow the maximum freedom in using the source to create derivative works. This includes the freedom to take open-source software under a BSD-style license and use it to create a proprietary product for which source code is not made available. As a result many open-source advocates recommend not using BSD-style licenses, preferring licenses that require (to a greater or lesser degree) that derivative works of open-source software also be made available as open source.

5.7 Artistic License

The Artistic License [i14] was originally created by Larry Wall for use with Perl [i15] (a scripting language interpreter); it has also been used for open-source Ada library software (as the Ada Community License [i16]). The Artistic License is perhaps best thought of an attempt to create an open-source license that eliminates or mitigates the more controversial aspects of the GPL. In particular the Artistic License differs from the GPL in the following ways (among others):

- The Artistic License encourages users to make modifications freely and publicly available, but allows exemptions from such a requirement in the cases where the derived works are used only within an organization and are not publicly distributed, or where the derived works are not represented as being the original work, i.e., the derived work's executables have different names and differences from the original work are documented. The GPL has no such exemptions.
- The Artistic License allows the original work or derived works to be embedded "invisibly" in a proprietary program, i.e., where the proprietary program does not expose direct interfaces to the functionality of the original or derived work. Under the GPL the proprietary program would be considered a derived work also subject to the GPL.
- The Artistic License explicitly declares that input and output data submitted to or produced by the original work (or derived works) does not fall under the terms of the license. This is presumably intended to address a feature of the GPL whereby output from such GPL-ed

Title:

works as GNU flex (a program to generate lexical analyzer code) was considered to fall under the GPL, since such output contained program fragments embedded as data in the original GPL-ed source code and was thus considered a derived work under the GPL.

Unfortunately the Artistic License is not as general-purpose as it might be; in particular, some of its provisions (e.g., Sections 5 and 7) seem to assume that the product being licensed is a language interpreter (like Perl) or similar program. However the Artistic License (or a variant thereof) is a possible candidate license for anyone looking for an open-source license more encouraging of code sharing than BSD-style licenses but less restrictive than the GPL or GPL-like licenses.

5.8 Mozilla Public License

The Mozilla Public License (MozPL or MPL) [i17] and the related Netscape Public License (NPL) [i18] were created by Netscape as part of the project to release Netscape Communicator source code. Where the BSD License was created by a university and the GPL and Artistic License were created by free software developers (albeit with somewhat different philosophies), the MozPL is notable as an open-source license created by a commercial software company. As one of the newest open-source licenses the MozPL was influenced by and to some extent incorporates features from a number of older licenses, including the GPL and LGPL; however the MozPL is a distinctive license in its own right and has a number of interesting and even innovative features not found in other open-source licenses.

5.9 Non copyrighted licenses

To finalise overview of licenses we will comment on the fact that all these licenses are copyrighted themselves, so to modify them into a new license is forbidden or at least the organisation has to look for permission to do so. An exception to this rule is the MIT license also called the X11 license.

5.10 Public domain software

The remaining possibility from the licensing point of view is to use a license that liberates the software from all vendors' rights. One way to do this is to release the software into the Public Domain. Although the term "public-domain software" is often used loosely to refer to open-source or free software in general, public-domain software is strictly speaking software which has no copyright (e.g., the copyright has expired, or the copyright holder has explicitly waived copyright); since there is no copyright there is no "owner" of the software to grant licenses, and hence anyone may use the software in any way without any restrictions.

Putting software into the public domain grants the maximum freedom possible to end users and developers. However at the same time it opens the possibility that one or more developers may take the software and use it as a base to create proprietary programs; if those programs become dominant in the market then from a practical point of view the software is no longer open-source, even if one form of it remains available in the public domain. (In fact, users may not even be aware that the proprietary products use public-domain code.)

Because of this potential problem most open-source advocates recommend not making software public-domain; even developers who do not believe in the concept of "intellectual property" still advocate using the mechanism of copyright, if only to be able to use a formal open-source license to promote certain beliefs and practices.

6 Open Source Business Models

More specifically, there are a number of business models that have been attempted with open-source source, and a number of others that are possible in theory and may be workable in practice. As discussed above, all of the models assume the absence of traditional software licensing fees. We discuss these models in more detail below, including:

- which vendors (if any) are using this model
- what general types of open-source license might be appropriate for the model
- what opportunities exist for vendors to differentiate themselves
- what opportunities exist to set prices based on perceived value rather than actual cost

(Note: The names and basic definitions for the first four models below are from the OpenSource.Org web site.)

6.1 Support Sellers Business Model

In the "Support Sellers" model software-related revenue comes from media distribution, branding, training, consulting, custom development, and post-sales support instead of from traditional software licensing fees. It is the most common model today for companies involved with open source and in the Linux market is used by several vendors, of which Red Hat Software [i19] and Caldera Software [i20] are perhaps best known. For historical reasons all these companies use the GNU General Public License (GPL), but most if not all open-source licenses would work for this model.

Revenue is generated in this model by selling two broad categories of items: physical goods, e.g., media and hard-copy documentation, and/or services, e.g., technical support. Vendors can differentiate themselves by providing more complete and easier-to-use software distributions, in essence simplifying and improving the user's experience with the open-source software in question; they can also differentiate themselves by the quality and pricing of their service offerings.

In this model there is only limited ability to use value-driven pricing; more typically the pricing is determined primarily by the cost to provide goods and services, as there is price competition with other vendors offering comparable goods and services and definite limits to what users are willing to pay for them. However if a vendor's reputation is good then that can be used to justify higher prices than for a "commodity" offering.

6.2 Loss Leader Business Model

In the "Loss Leader" model a no-charge open-source product is used as a loss leader for traditional commercial software; the open-source product generates little or no revenue, but providing the product makes it more likely that customers will buy other products that are sold using the traditional software business model. To some extent this is the model now being used by Netscape with the Netscape Communicator product (but see also the discussion of the "Service Enabler" business model below); it will also be used by Sendmail, Inc. [i21], which plans to release a commercial product based on the open-source version of the popular "sendmail" [i22] mail system.

If the open-source product shares any source code with the company's proprietary products (for example, they use common libraries) then the open-source license chosen must allow distribution of such source code for the open-source product while allowing the same source code to be used in

Title:

proprietary products distributed using standard licenses. The company should therefore avoid the use of the GPL or other licenses that aggressively "taint" products incorporating code under the license; a BSD-style license would work fine, and the Mozilla Public License or variants thereof might be appropriate if the common source code is accessed via a set of defined APIs.

Generation of revenue (if any) from the open source product could be done as in the "Support Sellers" model, i.e., by selling media and services. However typically the bulk of revenue generated would be through sales of other software products; having the open-source product could increase sales of such traditional products in various ways:

- by helping build the overall vendor brand and reputation
- by making the traditional products more functional and useful (in essence adding value to them)
- by increasing the overall base of developers and users familiar with and loyal to the vendor's total product line

Vendors have ample scope to differentiate themselves based on the products in the traditional product line, and can also employ value-driven pricing where appropriate with such products.

6.3 Widget Frosting Business Model

The "Widget Frosting" business model is intended for companies that are in business primarily to sell hardware ("widgets") but which use the open-source model for enabling software such as driver and interface code ("frosting") distributed at no charge along with the hardware. The hardware could be anything from an individual chipset to a controller board to a peripheral device to a complete computer system. The enabling software could be driver code (e.g., for a graphics board or peripheral), compilers and linkers (e.g., for microprocessors and related chips), or complete applications or operating systems (e.g., for a workstation or network computer). Companies could use any of a number of open-source licenses.

COREL is using this model for the Netwinder [\[i23, i24\]](#) product line to be offered by its Corel Computer subsidiary; Netwinder is essentially a network computer using Linux as its operating system kernel. Corel Computer is porting the Linux kernel to Netwinder and will release all Netwinder driver code as open source under the GPL; it will also release a set of Netwinder development tools based on open-source applications. Another example of companies using this model to some degree is VA Linux Systems [\[i25\]](#) and other companies which sell PCs designed and preconfigured to run Linux; unlike Corel VA Linux and its competitors are selling standard PC hardware, but are still using Linux to maximize the utility of their systems (e.g., as network servers or high-end workstations).

In the "Widget Frosting" model most if not all revenue would be generated through sales of the hardware itself. Using open source for the enabling software could increase hardware sales as in the "Loss Leader" scenario, e.g., by increasing the base of developers familiar with the hardware and able to make it perform to full capacity, thus making the hardware more functional, reliable, and useful to its end users.

In this model vendors can differentiate themselves based on the attributes of the underlying hardware, e.g., functionality, performance, reliability, flexibility, and cost; the function of the open-source software is to enable the hardware to fulfill its full potential with regard to these qualities. Since the vendor is selling physical goods for which competitive products exist in most cases, the pricing is typically much more cost-driven than value-driven.

6.4 Accessorizing Business Model

The "Accessorizing" business model is for companies which distribute books and other physical items (as opposed to software and services) associated with and supportive of open source software. Here the company does not typically participate in open-source development personally, but rather for the most part piggybacks on open-source software developed and maintained by others. A good example of a company pursuing this model is O'Reilly & Associates [i26], which publishes books documenting and explaining various open software products (including Linux, Perl, GNU Emacs, etc.). The only license feature required for this model is the ability to bundle free software with the item being sold (where this makes sense); by definition this is permitted by any true open-source license.

Vendors can differentiate themselves based on the quality of the goods themselves, and can also build some brand loyalty among people who use and like open-source software and who are appreciative of vendors that support it in some way. Since what is being sold here is a physical item as opposed to intellectual property of some sort, pricing will tend to be cost-driven rather than value-driven for the most part, although in some cases brand reputation can justify somewhat higher prices than otherwise possible.

6.5 Service Enabler Business Model

In the "Service Enabler" business model a company creates and distributes open-source software primarily to support access to revenue-generating on-line services. (The services may generate revenue, e.g., through subscription fees or advertising.) To a certain extent Netscape can be seen as pursuing this model in addition to the "Loss Leader" model, given that Netscape Communicator can be used as a front-end to access Netscape's Netcenter services (from which Netscape derives revenue from advertising and related sources). Another example would be a company providing a fee-based on-line game service which created special-purpose software to access the service and then distributed the software as open source so that users could not only download the software at no charge but could also modify the software, e.g., to customize it for their use or to support special input or output devices.

For this model a company would likely use an open-source license that minimized the possibility that its open source software could be turned directly into proprietary software by competitors; examples of such licenses include the GPL and the Mozilla Public License.

Vendors can differentiate themselves based on the attributes of the services themselves; some of these attributes are a function of the open-source software serving as the front end, but as much or more would be a function of the back-end systems. Those back-end systems could be based on proprietary software or on a combination of open-source and proprietary software; in either case, by creating unique and useful services which could not be easily duplicated by competitors a company could justify pricing the service more commensurate with the value experienced by the user. (For example, an on-line gaming service could price itself based on the entertainment value it provides to users relative to alternatives like traditional video games, movies, and cable TV.)

The following business models are more theoretical, in the sense that no companies appear to be actively using them today. However they appear to be at least theoretically feasible, and it may be that in the future one or more companies will be able to implement them successfully.

Title:

6.6 Sell It, Free It Business Model

The "Sell It, Free It" model is essentially the "Loss Leader" model repeated and extended through time. In this model a company would deliberately structure its development and licensing practices so as to release software products first as traditional commercial products and then convert them to open-source products when they reach an appropriate point in their life cycle where the benefits of developing them in an open-source environment outweigh the direct software license revenue they produce. The newly freed open-source products would still add value to the remaining proprietary products, as in the "Loss Leader" model; in fact, if the proper open-source license were chosen (e.g., a BSD-style license or the Mozilla Public License) then newer proprietary products could be based in part on code from older products now released as open source.

There would be two main tactical problems to overcome in successfully implementing a "Sell It, Free It" strategy. The first would be deciding exactly the right time in the product's life cycle to convert it to open source; doing it too early might mean unnecessarily forgoing significant license revenues, while doing it too late might lessen the interest of open source developers in contributing to the product's further development. The second problem is related to the first: If customers think the product is likely to be converted to open source at some point then it is more difficult to justify to them why they should buy it now rather than waiting. The problem becomes more tractable if the product's sales cycle is short relative to the product's life cycle (otherwise price reductions or conversion to open source are likely before customers complete their buying decisions) and if product pricing is managed so as to plan for scheduled reductions in license price and at least partially compensate for such reductions with revenue from other sources, such as technical support or professional services.

In a fully-realized "Sell It, Free It" model customers buying the product near the beginning of its life cycle are in essence paying a premium for the value of having the software earlier rather than later, and license pricing can reflect this value. Once the product converts to open source it becomes in essence a commodity, but it can still add value to newer proprietary products, whose license pricing can reflect that added value.

6.7 Brand Licensing Business Model

In the "Brand Licensing" model a company makes the software product itself open source but retains the rights to its product trademarks and related intellectual property, and charges other companies for the right to use those trademarks in creating derivative products distributed under the exact same brand name. This of course requires that the product exist in at least two different forms with two different names: the "official" product referred to by a trademarked name, and the "unofficial" product referred to by a separate name.

Using this model was not possible with traditional free software products because the product "brand names" (as it were) were typically not formally registered as trademarks. (In the case of Linux this caused a dispute [\[127\]](#) requiring legal action to resolve.) However a commercial product being converted to open source will almost always have associated with it trademarks in the form of product names and logos, and these can remain under the control of the company. For example, even though Netscape released source code for the Netscape Communicator product, only Netscape (or authorized Netscape licensees) can use the source code to create a product called "Netscape Communicator;" products built by others from the source are typically referred to by the name "Mozilla" (from the original code name for Netscape Navigator 1.0).

In general the two product versions (with and without associated trademarks) could be built from identical or near-identical source code; however from the perspective of the market they would be two

Title:

different products with possibly different perceived value. (For example, the branded product may have undergone additional testing and validation not done with the non-branded product.) If you convert a product to open source then a third party wishing to distribute a product based on that source (for example, for distribution as part of a special on-line service) may also wish to separately pay you for a license to use your trademarks in association with that product. The price of that license can reflect the brand value and reputation that your company (and by association, your trademarks) have in the marketplace.

(I should note that even though I've used Netscape as an example here, Netscape has not publicly announced any plans to license its trademarks as described above.)

6.8 Software Franchising Business Model

"Software Franchising" is a possible business model based on taking to their logical conclusion the ideas of some of the preceding models, in particular "Brand Licensing" and "Support Sellers." The "raw materials" underlying a support seller's business are available to anyone, but it's reasonable to assume that some may be better at the business than others and as a result may build value in a particular brand associated with the company's services. If the company then wishes to expand, one possibility would be to grow not through direct hiring and acquisition but rather through franchising; in other words, the company would authorize other developers to use its brand names and trademarks in creating associated organizations doing open-source support and custom software development in particular geographic areas or vertical markets.

In this model the "software franchisor" would not only license brands and trademarks but also supply franchisees with training (e.g., in specific aspects of running an open-source development effort and support seller business) and services (e.g., centralized support for contracting and procurement, advertising and marketing campaigns). Revenues would come from sources such as sales of franchises and royalties based on franchisees' revenues.

6.9 Hybrid Business Models

The business models discussed in the previous section are all based on open-source software in one way or another, with that software licensed under a true open-source license along the lines of that recommended by the Open Source Definition; if a business also involves software that is not open-source (as for example in the "Loss Leader" model), then the software is assumed to be totally proprietary or "closed-source," i.e., the company does not make its source available except under very restrictive licenses.

However there are possible business models that involve software distributed under licenses that are not quite open-source in the strict sense, but are also not as restrictive as traditional proprietary licenses. One way to explore the space of possible "hybrid" business models is to go back to the Open Source Definition and look at relaxing one or more of its requirements; here are some changes that might be made:

- change the availability of source code
- change the treatment of different users
- change the treatment of different types of use

Let's consider these in more detail, along with two examples of organizations that have pursued such hybrid models: Troll Tech [\[i28\]](#) and The Open Group [\[i29\]](#).

Title:

- **Changing availability of source code**

An open-source license grants at least four separate rights with regard to source code:

- the right to view (i.e., to see the code in the first place and possess a copy of it)
- the right to use (i.e., to compile the source into executable form and run the resulting application)
- the right to modify (i.e., to make changes to the source code)
- the right to redistribute (i.e., to give the source code to a third party, potentially in either modified or unmodified form)

All these rights come bundled together and are available "for free;" i.e., the developer does not charge the user any license fees to have them.

However at least in theory these rights could be separated and could be made conditional on paying some sort of license fee. (Some might object that there are no technical means to enforce such separation; for example, if someone has a copy of the code and can use it to build an executable, there is no way to prevent them from modifying it. This is true, but in practice such considerations have not been a major problem in commercial software licensing; for example, although many software products are now available over the Internet for downloading (e.g., for evaluation use) and thus there is no technical way to enforce payment for a right-to-use license, nevertheless almost all commercial and government organizations do in fact "pay up" and officially acquire licenses if their users end up using the product -- particularly if the organizations are prompted to do so.)

Thus, for example, users could be given a low-cost or even no-cost license to possess a copy of product source code, compile it for internal use, and make bug fixes as necessary, but would be charged a higher license fee if they wished to redistribute modified versions. (Returning bug fixes to the original vendor could be exempted from this restriction.) Such licensing terms would likely not be attractive to many if not most noncommercial developers, but might be acceptable and of interest to many commercial organizations. If the product serves a very specialized market then it is quite possible that only commercial users would be interested in the source code anyway, so that a lack of participation by noncommercial developers would not necessarily always be a drawback.

As a real-life example, Troll Tech distributes a free version of its Qt GUI toolkit [\[i30\]](#), including source code; the Qt Free Edition license originally allowed developers to view, use, and modify the source code, but prohibited them from distributing modifications. (As discussed below, Troll Tech later changed licensing of the Qt Free Edition to loosen this restriction somewhat.)

- **Changing treatment of different users**

An open-source license is available to all potential users no matter who they are. (The Open Source Definition specifies that an open source license may not discriminate between users based on "fields of endeavor," which in essence is saying the same thing.) However in general a source license could be offered on different terms to one group of users versus another; the most common and probably the most useful distinction is made between commercial and noncommercial users. (Note that this distinction is not always easily made in general; for example, there are many instances of large organizations which are technically nonprofits but which are in many ways indistinguishable from for-profit entities performing similar functions. In practice it would be up to the software vendor to decide to which class a particular user would be assigned.)

Title:

Thus, for example, noncommercial users could be given a full set of rights to the source (rights to view, use, modify, and redistribute), but the vendor might charge commercial users license fees to obtain all or some of those rights. A real-life example of this was the licensing originally adopted for the X11R6.4 release of the X Window System [i31] from The Open Group; X11R6.4 source code was made available under a noncommercial license at no charge; however users wishing to distribute binaries for commercial purposes were required to pay a fee for a separate commercial license. (As discussed below, the Open Group later changed the licensing of X11R6.4, and dropped the distinction between commercial and noncommercial use.) Troll Tech also originally prohibited commercial use of the Qt Free Edition, and sold a separate Qt Professional Edition for that purpose. (As discussed below, Troll Tech subsequently loosened this restriction somewhat.)

- **Changing treatment of different types of use**

An open-source license does not discriminate between different uses of the software; a given user may use the software in any context and for any purpose they wish. However in general a source license could have restrictions limiting the use of the software for certain purposes, or could be offered on different terms (including different prices) depending on the end use.

Thus, for example, a source license could allow personal use or internal use within an organization at low cost or no cost, but prohibit or require higher license fees for use of the software to provide a service to other users (e.g., on the Internet or as part of an extranet). Alternatively, the license could allow no-charge use of the software on particular operating system and/or hardware platforms (e.g., Linux) but require that a license fee be paid to use the software on other platforms (e.g., Windows 95 or Solaris). As a real-life example, Troll Tech originally restricted the Qt Free Edition to being used only under the X Window System (e.g., on Linux and various Unix-based systems); developers wishing a version of Qt for Windows 95 or other platforms had to license Qt Professional Edition. (The current Qt Free Edition license does not contain this restriction, but as a practical matter there is still only an X version of Qt Free Edition.)

- **Viability of hybrid models**

A company using a hybrid business model can derive revenue and profits directly from the sale of licenses for the software in question as opposed to deriving revenue and profits only through more indirect means, as in an open-source business model. The question is whether such a company can also gain the other benefits of open-source business models such as leveraging outside developers to help improve its software and drive its adoption in the market. Although commercial developers presumably would have no qualms about working with hybrid software, many noncommercial developers would likely object to the more restrictive terms of hybrid licenses versus open-source licenses.

For example, The Open Group aroused widespread controversy when they changed the licensing terms for X11R6.4 to use separate commercial and noncommercial licenses as opposed to the single BSD-style license previously used; among other things this resulted in the threat of a split in X11 development, with the XFree86 Project [i8] deciding to not use X11R6.4 or other future releases from The Open Group, instead committing to continuing development based on X11R6.3. Subsequently The Open Group decided to change the licensing of X11R6.4; the current X11R7.0 license is a true open-source license and explicitly allows commercial sale without paying royalties. In return the XFree86 project decided to employ X11R6.4 as the basis for their 4.0 release, and later formally joined X.Org, the group established by The Open Group to oversee X development.

Similarly, although Troll Tech attempted to justify its business model to the open-source developer community and established a foundation to help ensure that Qt Free Edition would never become a

Title:

proprietary product, nevertheless many in that community still saw Troll Tech's licensing policies as inconsistent with the goals of the free software movement. This resulted in a rivalry of sorts between proponents of Qt and the Qt-based K Desktop Environment (KDE) [i32] and proponents of the GTK+ [i33] toolkit and GNOME [i34] desktop environment, which are distributed under the LGPL and GPL. It also resulted in the formation of a separate project (known as Harmony) to build a Qt-compatible library to be distributed under the LGPL. Subsequently Troll Tech changed its licensing of the Qt Free Edition to use a new Q Public License (QPL) [i35]. The QPL is a true open-source license, and eliminates some of the restrictions of the original Qt Free Edition license; for example, the QPL now allows distribution of modified versions, as well as commercial use of the library. However, Troll Tech does require that modifications to Qt Free Edition source code be distributed separately from the base Troll Tech distribution, and also still attempts to make some distinctions between use of Qt in noncommercial and commercial contexts in an effort to support Troll Tech's chosen business model. For example, the QPL requires that applications developed to link to the Qt Free Edition should themselves be open-source applications; development of traditional proprietary software still requires licensing of the Qt Professional Edition.

7 Can be HUMBOLDT as an Open Source?

The open source software distribution strategy offers an opportunity that is difficult to gain otherwise. That is it offers access to resources outside the creators of the software in order to further develop it. In the case of the HUMBOLDT platform it is foreseen that within the next years HUMBOLDT will be operative in various locations offering answers under specific scenarios simulations. This will be only the beginning. The first versions of software tools and application examples will probably require a lot of updating and maybe the inclusion of extra added components that nobody has seen when HUMBOLDT was envisioned. The above will require a lot of resources mainly on time and money. This is difficult to achieve by offering HUMBOLDT as closed source software. On the other hand if the infrastructure itself is based on open source it will be easy to further develop it and insert new tools within it by offering its source code to different users (users in this context are service providers or software developers) and giving them possibility to make modifications they want.

To achieve this two main factors are required. The first is to offer HUMBOLDT core components as an open source. It is crystal clear that the HUMBOLDT users cannot become code developers of the HUMBOLDT tool unless they have access to that tool in a form that they can understand and modify. So in a software context this is satisfied only if HUMBOLDT is offered in an open source software distribution methodology. The other important factor is the users' incentive to contribute to a cooperative effort. In other words if potential contributors expect to be prevented from using this tool that they helped to create, they will eventually be reluctant to contribute in first place. Seen in this light, the open source prohibition on terms restricting use, redistribution and modification of licensed subject matter is a way of shoring up the motivation of potential contributors.

On the other hand there is a downside easily spotted when dealing with open source software distribution strategies. Clearly open source prohibitions on restrictive licensing terms are incompatible with standard proprietary business models because they prevent intellectual property owners (in our case the whole partners of the HUMBOLDT contributing to each tool) from charging for access and thereby eliminate what may be a business's primary source of revenue. It is therefore not a surprise that the idea of licensing proprietary tools on open source terms tends to meet resistance from business audiences.

Besides mentioned considerations there are more aspects (denoted by the project partner Anette Breu, Intergraph Deutschland GmbH) that shows some places where open source strategy can not be

Title:

accepted because of the specific applications that software products serve, for example border security, rescue forces etc. (see Annex 2). Thus we can see that different distribution strategies will be needed for different HUMBOLDT related software products.

8 Conclusions

Considering one of the main objectives of the project to make HUMBOLDT infrastructure to be self-sustaining an open source distribution strategy for core framework software components should be used.

Offering HUMBOLDT in an open source contributes to quality improvements in two major ways. First giving access to a large group of users guaranteed to a certain extent that design flaws will be eliminated and enhancements will be introduced very rapidly.

Second the existence of an informal development community that includes both users and owners of the tools allows user to communicate their needs and priorities to owners so that overall development efforts are more likely to be redirected towards the most useful tasks.

However selection of the open source strategies as main form for HUMBOLDT software framework does not guaranty sustainability by itself. Appropriate infrastructure development, exploitation and dissemination strategy should be developed.

HUMBOLDT ESDI involves different segments of large European geospatial community and services provided by the ESDI participants are rather different. Therefore different distribution strategies will be needed for different HUMBOLDT software components.

At the current stage of the project it is early to state witch particular licensing type should be adopted, or used as pattern for HUMBOLDT type license, however some trends can be denoted:

Open source GPL type of license could be used for the main infrastructure in order to make sure that every part of the core infrastructure remains commercial free.

BSD style License could be used for the most peripheral sides of the applications. It would allow the combination of open source services infrastructure with more specific service post-developments which would be proprietary. This would allow the service providers to resell there added value software developments to third parties and cover some of their costs. This type of license will also allow us to combine services with proprietary applications or legacy systems that evolved around a proprietary license.

9 References

1. Dargan, P.A. Open Systems and Standards for Software Product Development. Norwood, MA, USA: Artech House, Incorporated, 2005.
2. Study on the Economic impact of open source software on innovation and the competitiveness of the Information and Communication Technologies (ICT) sector in the EU. Final Report. Nov. 20, 2006. R.A. Ghosh, UNU-MERIT, NL. et al., 287 pp. Available Online: <http://ec.europa.eu/enterprise/ict/policy/doc/2006-11-20-flossimpact.pdf>
3. Koch, Stefan. Free/Open Source Software Development. Hershey, PA, USA: Idea Group Publishing, 2005.
4. ITRG. Free and Open Source Software: A Field Guide. London, , Canada: ITRG, 2004.
5. Dixon, Rod. Open Source Software Law. Norwood, MA, USA: Artech House, Incorporated, 2003.
6. Christopher M. Kelty. Free software/ free science. First Monday, volume 6, number 12 (December 2001). Available Online: http://www.firstmonday.dk/issues/issue6_12/kelty/.
7. Koch, Stefan. Free/Open Source Software Development. Hershey, PA, USA: Idea Group Publishing, 2004. p 259.

Internet references

- i1. The BSD License. <http://www.opensource.org/licenses/bsd-license.php>
- i2. The MIT License. <http://www.opensource.org/licenses/mit-license.html>
- i3. The FreeBSD Project. <http://www.freebsd.org/>
- i4. The NetBSD Project. <http://www.netbsd.org/>
- i5. The OpenBSD project. <http://www.openbsd.org/>
- i6. The Apache Software Foundation. <http://www.apache.org/>
- i7. The SLAPD and SLURPD Administrators Guide. <http://www.umich.edu/~dirsvcs/ldap/doc/guides/slapd/>
- i8. Free86® Home to the X Window System. <http://www.xfree86.org/>
- i9. SATAN (Security Administrator Tool for Analyzing Networks). <http://www.porcupine.org/satan/>
- i10. The BSD License Problem. <http://www.gnu.org/philosophy/bsd.html>
- i11. GNU General Public License. <http://www.gnu.org/copyleft/gpl.html>
- i12. The Free Software Foundation. <http://www.fsf.org/>
- i13. What is Copyleft? <http://www.gnu.org/copyleft/copyleft.html>
- i14. Artistic License. <http://www.perl.com/language/misc/Artistic.html>
- i15. Perl.com: The Source for Perl -- perl development, conferences. <http://www.perl.com/>

Title:

- i16. The Ada Community License. <http://www.rivatech.com/booch/acl.html>
- i17. Mozilla Public License. <http://www.mozilla.org/MPL/MPL-1.0.html>
- i18. Netscape Public License. <http://www.mozilla.org/MPL/NPL-1.0.html>
- i19. Red Hat Software. <http://www.redhat.com/>
- i20. The SCO Group, Inc. (Caldera Software) <http://www.caldera.com/>
- i21. Sendmail, Inc. <http://www.sendmail.com/>
- i22. Sendmail.org - Sendmail Home. <http://www.sendmail.org/>
- i23. NetWinder.org - Netwinder News Page. <http://www.netwinder.org/>
- i24. Product Review: Corel's NetWinder. <http://www.linuxjournal.com/article/3288>
- i25. VA Software. <http://www.vasoftware.com/>
- i26. O'Reilly & Associates. <http://www.oreilly.com/>
- i27. Linux International Announces New Direction. <http://www.li.org/News/trademark.html>
- i28. Trolltech. <http://www.troll.no/>
- i29. The Open Group: Enterprise Architecture Standards, Certification and Services. <http://www.opengroup.org/>
- i30. TrollTech Qt GUI toolkit. <http://www.trolltech.com/products/qt/>
- i31. X.Org Foundation. <http://www.x.org/>
- i32. K Desktop Environment - Conquer your Desktop! <http://www.kde.org/>
- i33. GTK+ - The GIMP Toolkit. <http://www.gtk.org/>
- i34. GNOME: The Free Software Desktop Project. <http://www.gnome.org/>
- i35. Q Public License v1.0 – Trolltech. <http://www.trolltech.com/products/qt/licenses/licensing/gpl>
- i36. Index of Open Source / Free GIS related software projects. <http://opensourcegis.org/>
- i37. GRASS (Geographic Resources Analysis Support System). <http://grass.itc.it/>
- i38. UMN Mapsrever (an Open Source development environment for building spatially-enabled internet applications) <http://mapserver.gis.umn.edu/>

Title:

10 Annexes

10.1 Annex 1: Comparison of software licenses

Note: The following tables are taken from Wikipedia, the free encyclopedia website (http://en.wikipedia.org/wiki/Comparison_of_software_licences).

General comparison

The following table compares various features of the license. This should not be taken as legal advice, but instead a general guide to the terms a license contains.

Licence	Author	Latest version	Publication date	Link from code with a different license	Release changes under a different license
Academic Free License	?	?	?	?	?
Apache License	Apache Software Foundation	2.0	2004	Yes	Yes
Apple Public Source License	?	?	?	Yes	No
Artistic License	?	2.0	?	Yes	With restrictions
Berkeley Database License	?	?	?	?	?
BSD license	?	?	?	Yes	Yes
Common Development and Distribution License	?	?	?	?	?
Common Public License	?	?	?	?	No
Cryptix General License	?	?	?	?	?
Eclipse Public License	?	?	?	?	?
GNU General Public License	Free Software Foundation	2.0	June 1991	No	No
GNU Lesser General Public License	Free Software Foundation	2.1	February 1999	Yes	No
Hacktivism Enhanced-Source Software License Agreement	?	?	?	?	?
IBM Public License	?	?	?	?	?
Intel Open Source License	?	?	?	?	?

Title:

LaTeX Project Public License	?	?	?	?	?
License of Python	?	?	?	?	?
MIT license	?	?	?	Yes	Yes
Mozilla Public License	?	?	?	Yes	No
Netscape Public License	?	?	?	?	?
Open Software License	?	?	?	?	?
OpenSSL license	?	?	?	?	?
PHP License	?	?	?	?	?
Q Public License	?	?	?	No	No
Sun Industry Standards Source License	?	?	?	?	?
Sun Public License	?	?	?	Yes	No
W3C Software Notice and License	?	?	?	Yes	Yes
X11 license	?	?	?	Yes	Yes
XFree86 1.1 License	?	?	?	?	?
zlib/libpng license	?	?	?	?	?
Zope Public License	?	?	?	?	?

Approvals

This table lists for each license what organizations have approved it, and how those organizations categorize it. Organizations usually approve specific versions of software licenses.

Licence and specific version	<u>FSF</u> approval	<u>OSI</u> approval
Academic Free License	Yes ¹	Yes
Apache License	Yes ¹	Yes
Apple Public Source License version 1.x	No	Yes
Apple Public Source License version 2.0	Yes ¹	Yes
Artistic License	No ²	Yes
Clarified Artistic License	Yes	Yes

Title:

Berkeley Database License	Yes	Yes
BSD license	Yes	Yes
Common Development and Distribution License	Yes ¹	Yes
Common Public License	Yes ¹	Yes
Cryptix General License	Yes	No
Eclipse Public License	Yes ¹	Yes
GNU General Public License	Yes	Yes
GNU Lesser General Public License	Yes	Yes
Hacktivismo Enhanced-Source Software License Agreement	No	No
IBM Public License	Yes ¹	Yes
Intel Open Source License	Yes	Yes
LaTeX Project Public License	Yes ¹	No
License of Python	Yes	Yes
MIT license	Yes	Yes
Mozilla Public License	Yes ¹	Yes
Netscape Public License	Yes ¹	No
Open Software License	Yes ¹	Yes
OpenSSL license	Yes	No
PHP License	Yes ¹	Yes
Q Public License	Yes ¹	Yes
Sun Industry Standards Source License	Yes ¹	Yes
Sun Public License	Yes ¹	Yes
Sybase Open Watcom Public License	?	Yes
W3C Software Notice and License	Yes	Yes
XFree86 1.1 License	No	No
zlib/libpng license	Yes	Yes
Zope Public License version 1.0	Yes ¹	?
Zope Public License version 2.0	Yes	Yes

Note 1: The FSF considers this licence to be free, but incompatible with its GNU General Public Licence.

Title:

Note 2: The original version of the Artistic License is defined as non-free because it is overly vague, not because the substance of the licence in question. The FSF encourages you to use the Clarified Artistic License instead.

10.2 Annex 2: Cornerstones of Intergraph's Humboldt Software Distribution Strategy

(See below)

Cornerstones of Intergraph's Humboldt Software Distribution Strategy

Intergraph's current business model is to sell software product based on licensing fees and costs for maintenance and support. Additionally, individual or customized solutions require payment for the necessary development effort. The software itself is commercial and only limited components like Intergraph's OGC Viewer are available as OpenSource (the source code is available here: <http://www.intergraph.com/interoperability/ogcviewer.asp>).

Nonetheless, Intergraph Offices throughout the European Union are fully aware, there is a growing demand within the European Union on OpenSource solutions in general (see <http://ec.europa.eu/enterprise/ict/policy/doc/2006-11-20-flossimpact.pdf>) and a significant part of the Humboldt software development will be done under OpenSource regulations. From the very beginning of the Humboldt project preparations Intergraph made clear, that a 100% adoption of the OpenSource approach will not be accepted by the company, but a strict sticking to the common software distribution strategies without any adaptation to the special Humboldt needs is not intended. This means in detail:

- Intergraph has committed itself to share its software products with the Humboldt partners involved in software development for the runtime and the scope and objectives of the project – for more information please refer to the Humboldt Consortium Agreement, chapter 10 (Intellectual Property Rights) and Annex C (Pre-existing Know-how).
- Beyond the project, the approach shall be similar to an “Open-Source – model” regarding *pricing*:
 - Developing licences will include the Core Software (GeoMedia) and the Humboldt scenario application for free. These developing licences will also include free maintenance and hotline support
 - Productive licenses (without any development part) shall include the Core Software product (GeoMedia) for free. The scenario application has to be paid regularly – the same applies – where required - for maintenance of core software and scenario applications.
- Any developments within Humboldt especially those of the scenario application that touch other Intergraph Core Products such as the Command & Control Software (I/CAD) will be distributed according to the common licensing models of Intergraph. Due to the fact that this software is used in very sensitive areas (Command & Control Center, Rescue Forces etc.) and according to given requirements in the area of Public Safety, there will be only limited access to this software package within Humboldt.