DEFENCE DÉFENSE

Free and Open Source Software

Overview and Preliminary Guidelines for the Government of Canada

Robert Charpentier

Richard Carbone

The authors will gratefully accept feedback and comments at:
FOSS@drdc-rddc.gc.ca

Unlimited Distribution
DRDC ECR 2004-232

Defence R & D Canada – Valcartier
December 2004

Executive Summary

Free and Open Source Software Evolution During the past two decades, the software market has been dominated by Commercial Off-the-Shelf (COTS) products such as MS Windows and Oracle database management systems that offer a myriad of functionalities at a reasonable price. However, the intrinsic limitations of COTS software (e.g. closed source code, lock-in effect, expensive upgrades, security weaknesses etc.) have emerged over time. This led to the development of a parallel 'economy' based on Free and Open Source Software (FOSS). FOSS refers to programs whose source code is made available for use and modification without the expensive license fees imposed by COTS software editors. FOSS is developed either by volunteers or through development sponsored by large computer firms who want to include 'commodity' software to give a competitive advantage to their hardware products. Over the past ten years, the FOSS phenomenon has been constantly growing in importance: thousands of FOSS projects carried out via Internet collaboration; hundreds of high-quality applications available for use or modification at no (or small) cost and tens of FOSS products now widely considered to be as mature and secure as their COTS equivalents.

The good reputation of free and open source software has attracted the attention of many governments around the world and they are now considering the systematic migration of their servers and their workstations to FOSS. The leading countries, currently migrating to FOSS, are the United Kingdom, Germany and France but it is estimated that more than 20 other countries are preparing policies and action plans to adopt FOSS systematically in their government and industrial systems. The strategic rationale for migrating to FOSS is typically related to three main factors: 1) the expectation of direct cost savings, 2) the reduction of economic loss at the national level caused by commercial software imports and 3) the hope to better develop national IT expertise by means of access to source code (and development of original components) which is not really possible with COTS packages.

Canada appears to be behind the curve in FOSS adoption. The lack of clear business cases and the underestimation of the strategic value of FOSS partly explain this situation. However the Government of Canada (GoC) has recently endorsed a pro-active position on FOSS to ensure that GoC staff are aware of the options available and that no barriers to procurement remain. Some comprehensive open source initiatives can be found in the education and health sectors and an increased awareness is now being expressed by the GoC, who see FOSS as a viable alternative to COTS software and expensive custom code development.

Proposed Way-Ahead for GoC FOSS is not a panacea, but it does offer a concrete and credible technological opportunity. GoC could benefit from an improved diversity in software supplies (custom code vs. FOSS vs. COTS), augmented security by source code auditing (and enhancement) and higher compliance with open standards and specifications that contribute to system interoperability.

Specific actions are proposed to increase awareness/use in GoC such as: to promote FOSS by means of publications, workshops and conferences; to consider FOSS-based solutions in contractual work when they are technically competitive with other development strategies; to support GoC departments in assessing this emerging technology. This report also includes various navigation aids to help identify suitable FOSS products, a comparison spreadsheet that facilitates side-by-side comparisons of FOSS and COTS software and some practical guidelines to help project leaders to determine the suitability of FOSS in their specific project contexts.

Table of contents

1 Introduction

After a slow beginning in the late 1990s, Free and Open Source Software (FOSS) has been constantly growing in importance and expanding in many software architectures all over the world. This impressive growth has been supported by the numerous successes, the high-quality reputation of FOSS-based systems and, of course, by the expectation of cost savings.

In Fall-2003, Defence Research & Development Canada (DRDC) initiated a special study to determine the role of FOSS in the evolution of our information system architectures and was later expanded to the whole Government of Canada (GoC). This report summarizes our findings in four main chapters. In the first part of this report, we offer a general introduction to this technology followed by a preliminary way-ahead for GoC (Parts I and II). We also tried to identify/categorize FOSS by technical application areas (Part III). Finally, some preliminary guidelines are offered to GoC project leaders in assessing usefulness of FOSS in their specific project contexts (Part IV of this report).

2 How to read and navigate this report

It is recommended that the readers start their navigation by reading the overview report. If further information is required, hyperlinks may be followed to the specific references in the bibliographies. In most cases, the complete reference can be found on the internet by pasting the title (and main author name) into a search engine like Google.

Links are color-coded. A blue link points within the report. A magenta link points to a web page, indicating that access to the Internet is required. A cyan links points to a file on the CD-ROM. To get back from a followed link, use the 'Go to previous view' arrow or the ALT-Left arrow. The left/right arrows can be used to go to the previous/next page. It is also possible to zoom in and out using the following keyboard shortcuts:

- Fit in window: CTRL-0
- Actual size: CTRL-1
- Fit width: CTRL-2
- Fit visible: CTRL-3
- View in full-screen mode: CTRL-L

Using the navigational bookmarks provided by Portable Document Format (PDF), it is easy for the reader to obtain a quick overview of the report. Use the F-5 function key on the keyboard to display/hide the bookmarks (F-6 on Acrobat 6).

3 Report Validation Process

Members of the FOSS advisory group
Cycle 1 – Valcartier

- ***Robert Charpentier (DRDC, lead scientist)***
- ***Richard Carbone (DRDC, technical editor)***
- ***Paul-André Côté (DRDC, secretary)***
- ***Martin Salois (DRDC, report editor)***
- ***David Demers (DRDC)***
- ***2Lt Stéphane Fortin (DRDC)***
- ***Dr Denis Poussart (Université Laval)***
- ***Max Blanchet (CGI)***
- ***Bertrand Couture (DMR Consulting)***

With helpful comments from:

- ***Micheline Bélanger (DRDC)***
- ***Yves van Chestein (DRDC)***
- ***Julie Couët (DRDC)***
- ***Louis Bastarache (IEEE section - Québec)***

Cycle 1 – Valcartier Technical Evaluation

To support the writing process, a group of Information Technology (IT) experts was formed at DRDC Valcartier to review and validate the contents of this report (January/February 2004). See list of names.

Cycle 2 – DRDC Corporate review

After this first review, an advanced draft was circulated in DRDC Headquarters (HQ) to allow a critical review of the report in a Corporate perspective. The authors wish to thank specifically those who offered comments in this second validation cycle and Mr. Brian Cheers for his patience in reviewing the English of this report an Mrs Caroline Lemelin for the French translation.

Cycle 2 – DRDC Corporate HQ Review

- ***Gavin Hemphill (DRDC Atlantic)***
- ***Bruce Skinner (DRDC Atlantic)***
- ***Dave Hazen (DRDC Atlantic)***
- ***Bill Page (DRDC Corporate)***
- ***Lawrence Kemdirim (DRDC Corporate)***
- ***Ingar Moen (DRDC Corporate)***
- ***Eric Fresque (DRDC Corporate)***
- ***Delmar Permann (DRDC Corporate)***
- ***Philip Staal (DRDC Corporate)***
- ***Dr Robert Walker (DRDC Corporate)***
- ***Mark Daniels (DND DIMR 4-6)***
- ***Ken Heaton (DND J2 DSI)***
- ***Karine Pellerin (DND DDCEI 3-5-3)***

Cycle 3 – DND/CF and Other Government Department (OGD) review

During the months of June, July and August 2004, an advanced draft of this report was circulated in DND/CF and in OGD. Comments and suggestions were included in this updated version prepared in September 2004.

You can contribute too!

Given the fast pace of information technology evolution and the diversity of topics covered in this report, any contributions to the technical content for future editions would be greatly appreciated. To bring to our attention relevant references or general comments about the contents of this report, readers are invited to submit citable references to: FOSS@drdc-rddc.gc.ca

Part I

Free and Open Source Software (FOSS) Executive Introduction

4 Main Definitions

Free Software (FS) refers more to the concept of freedom (liberty) than to the concept of no-cost (gratis)

According to the Richard Stallman's Free Software Foundation (FSF) [1], 'free' should not be understood as "free-of-charge" but rather as the **user's freedom**:

- **To run** the program for any purpose.
- **To study** how the program works and **to adapt** it to a specific need.
- **To redistribute** copies of the original or of the modified program.

A glossary is available at the end of this report for a more formal definition of FOSS related terms and a graphical taxonomy is available in Figure 1.

Open Source Software encompasses more than just the access to source code

The Open Source Initiative (OSI) requires that Open Source Software (OSS) complies with the following criteria [2]:

- Free distribution of components or aggregate programs.
- Source code must be included.
- Derived works must be possible and distributable at least as a patch files.
- Discrimination against persons or groups is not allowed.
- Discrimination against field of endeavor is not allowed.
- No additional license should be imposed in OSS redistribution.
- The same rights should be granted as for the original software distribution.
- The license must not restrict other software.
- No provision of the license may be predicated on any individual technology or style of interface.

Some contributors to FOSS find these requirements (FSF & OSI) difficult to satisfy in the competitive context of software production, but most of them try to adopt the OSI philosophy which is considered to be more pragmatic than the "FSF Idealism"

Closed Source Software (or Proprietary code) designates software for which the source code is not available

Commercial firms tend to restrict access to their source code in order to protect their intellectual property. They compile their proprietary source code and then distribute executable code (i.e. binary form of the program) that can essentially be understood only by the computer Central Processing Unit (CPU). Most COTS software is sold with this strategy (e.g. MS Windows, MS Word, Symantec or McAfee virus scanner etc.)

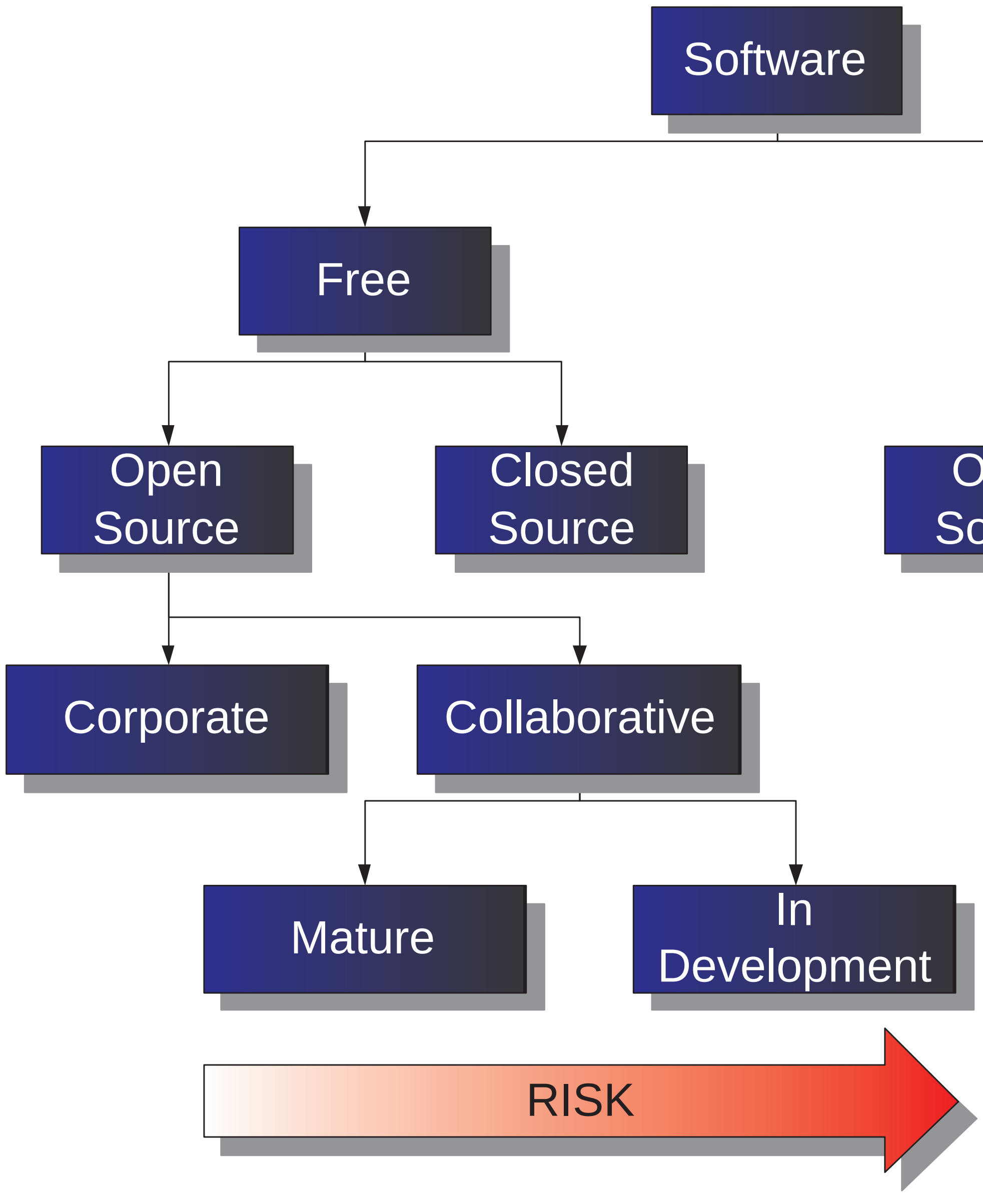

Figure 1: *A Software Taxonomy*

5 FOSS Legal Background

Licenses attached to FOSS provide basic rights to the licensee and the user but they vary a lot depending on the originator's preferences

Even if a lot of freedom is given to the users of FOSS, the programmers can impose some constraints on the exploitation of the components that are being released. For example, a license can state that the released component can only be integrated into a pure FOSS system (i.e. purist approach) or, alternatively, may be linked with some proprietary library (more practical approach). System architects must pay attention to the terms of the license when selecting a product in order to avoid legal pitfalls. License comparison can be found in many references [3, 4, 5].

The most common license is the GNU General Public License (GPL) which is used with 65% of the FOSS

The GNU's Not Unix (GNU) General Public License (GPL) has been adopted by most programmers in the past 15 years and has been the reference model for many other license agreements. This licensing model demands that all complementary code development be integrated with open source software only and published under GPL compatible license. It is expected however that less restrictive license models (such as Mozilla and BSD) will be more popular in the future [4], since hybrid proprietary/FOSS systems are more appropriate to most modern hybrid IT infrastructures.

The Free Software Foundation proposed the term 'Copyleft' to describe the privilege of using FOSS freely

As opposed to the concept of a copyright, a 'copyleft' describes the case where the owner forfeits intellectual property and private licensing. Not all FOSS are copylefted but most of the code under the GPL and Lesser General Public License (LGPL) licenses is. Compatibility of the various license models and with the "Copyleft Ideal" is discussed in [3].

Some complementary tactics were developed to prevent appropriation of FOSS by commercial firms

FOSS is intrinsically exposed to the risk of appropriation by commercial vendors. So, in addition to licensing, developers may legally incorporate and/or transfer their property rights to a non-profit corporation and/or trademark the brand and logos of their software product etc. [6].

Patenting algorithms is intrinsically incompatible with FOSS development

According to Richard Stallman, the worst threat faced by FOSS comes from software patents, which can put algorithms and features off-limits to free software for up to twenty years. The FSF has a petition in circulation against software patents [7]. In [8], the author indicates that Microsoft increased its patent rate by more than 30% in 2002, which may confirm that the threat is real. Steps to prevent patents from interfering with software freedom are proposed in [9]

6 Key Findings on FOSS Evolution

Over the years, many very useful software products have been distributed in an open-source paradigm

Some famous ones are:

LaTeX Text editor (and typesetting) used for scientific publications.

Linux Popular Unix-like operating system.

Apache Very reliable and secure web server.

MySQL Fast, precise and full-functioned database.

When this report was published (May 2004), it was estimated that 115 FOSS applications [5] have achieved maturity comparable or superior to their equivalent COTS products.

FOSS also evolved in a very efficient "development process"

The strength of FOSS development is the ability to recruit and motivate communities of competent programmers to develop, debug, and optimize code on a volunteer basis. Coordination is assumed by a delegate leader who is responsible for the assessment of the various solutions offered by the programmers and for the integration of the best code into the next FOSS updates that are rapidly put on-line [10].

By its simplicity and efficiency, the FOSS development model has repeatedly demonstrated many benefits

- Huge diversity of software [11].
- High flexibility and scalability of software solutions through source code editing.
- High reliability and high security through source code review and validation [12].
- One-order of magnitude faster release rate than equivalent COTS software.
- Rapid development of custom solutions to meet specific requirements through code reuse and extension.
- Lifetime extension of FOSS-based systems through source code upgrades [13].
- High degree of compliance with open standards leading to more interoperability between information systems.
- Leaner and meaner systems compared to COTS equivalents that often suffer from "marketing feature bloating".

For more documentation on FOSS advantages, the following references are recommended [14, 8, 12].

7 FOSS Risks and Drawbacks

Some criticisms can be found in scientific literature

When this report was being prepared (i.e. winter 2004), the following criticisms were found in the technical literature:

- Version control may be more complex with FOSS than COTS (evolving).
- System maintainability requires more local resources (debatable in the long-term).
- Higher technical skill needed from system administrators.
- May offer less integration within an application suite and less user-friendliness (evolving).

Microsoft is sponsoring many studies against FOSS

Some other criticisms can be found in various reports but in many instances the perspective is clearly biased. For example, the National Economic Research Associates (NERA) report claims that there is nothing wrong with closed source software since it allowed “a very fast growth of the software industry over the last few decades, providing ever more powerful, easy-to-use software to ever more users” [15]. Even if NERA’s impartiality can be questioned (since their study was sponsored by Microsoft), the report offers quite complete counter-arguments against FOSS adoption in government. This report is not unique! A recent article from Todd Bishop indicates that many studies on Linux performed by IDC, Giga and Meta Group were in fact sponsored by Microsoft. Critics question how independent the analyses were [16]. Some of these studies are listed below:

- Veritest study on Windows 2003 vs. Red Hat Linux [17].
- Meta Group study on Linux Benchmark [18].
- Veritest study on Windows 2003 vs. Linux server performance [19].
- Veritest study on Windows 2003 vs. Linux Web Server [20].
- IDC study on Windows 2000 vs. Linux in Enterprise computing [21].
- 7-Eleven business case for MS Windows 2003 instead of Linux [22].
- Total Economic Impact MS vs. Linux [23].
- Counterpoint from Linux Insider about Windows vs. Linux [24].

Interestingly, the January/February 2004 edition of IEEE Software includes a series of very positive articles on FOSS, which are preceded by a guest editor’s introduction signed by Szyperski (Microsoft Research) and by Spinellis (Athens University of Economics and Business) [11].

8 FOSS Adoption Around the World

The European community is actively adopting FOSS; mostly in the public service

The United Kingdom (UK), France, and Germany are the leading countries in the migration of the public sector IT infrastructure to open standards and FOSS. The UK government has adopted a policy to consider open-source solutions alongside proprietary ones in IT procurement [25]. At the end of 2003, a migration guide was issued by NetProject [26], which gave very detailed guidelines for the migration to FOSS for desktops and servers in general, and principal applications (office suite, mail, databases, Operating Systems etc.) in particular. KBSt (Germany) also has prepared an extremely detailed technical guide, which facilitates an aggressive migration strategy to FOSS for servers and workstations [27]. European countries share this knowledge through the Free/Libre and Open Source Software (FLOSS) project that is financed by the European Commission under the Information Society Technologies program. Their reports cover, among other subjects: policy for the European Union, FOSS business models, survey of developers and source code [28, 29, 30, 31, 32, 33, 34, 35, 36]. The Swedish Agency for Public Management (Statskontoret) has an excellent series of rigorous publications on FOSS-adoption in Sweden including [37, 38]. At a conference held in Washington in June 2003, it was estimated that twenty-four countries are currently reviewing policies, including Denmark [39], the Netherlands, Italy, Norway and Sweden [40, 41]. The Open Source Observatory (OSO) maintains a very informative Web site on FOSS evolution in Europe [42].

Latin American, African, Oceanian and Asian countries are also moving toward FOSS to varying degrees

The reasons for migrating to FOSS are typically related to three main factors:

- The expectation of direct cost savings.
- The reduction of national economic loss caused by commercial software imports.
- The hope to better develop national IT expertise via access to source code (and development of original components) that is not really possible with COTS packages.

A recent overview of FOSS policy in different countries is available at `http://www.csis.org/tech/OpenSource/0408_ospolicies.pdf`. Steven Weber (University of California-Berkeley) offers an interesting analysis of the reasons for developing countries to migrate to FOSS [43]. Some strategic decision announcements were released during our study by different countries, such as Israel [44], Japan, South Africa [45] and Australia [46, 47]. The AFUL Web site [48] also offers a periodically updated list of countries adopting FOSS policies or legislation [49] and [50] is an interesting complementary web site. Gartner analysts expect that developing countries will increasingly adopt FOSS through 2004. The business case for emerging nations is also reviewed in [51].

9 FOSS in the USA

FOSS originated largely in the United States of America (USA) and remains a very strong movement

A plethora of reports discusses the growth of FOSS in various fields of the USA economy. A large portion of this information is incomplete and/or biased – written to support a specific perspective. Almost unanimously however, it is recognized that FOSS is expanding rapidly in most IT infrastructures. The well-known Linux Operating System and Apache (Web server) are the most often cited as fast growing because of their recognized maturity and their technical qualities compared to their commercial equivalents [52, 53].

Many large American corporations contribute to the FOSS networks

In addition to the software developed by groups of volunteers, a substantial contribution to FOSS is made available by large firms who wish to experiment with a different business model based on collaborative development. ***Netscape*** has had one of the most famous success stories in adopting an openness strategy that is described in this very interesting article [54]. ***IBM, Hewlett-Packard, Sun Microsystems, Novell and Silicon Graphics*** are just a few of the better known IT leaders who host/contribute/sponsor/support a large number of open source projects [55]. IBM made a formal commitment to speed Linux deployment in the banking industry [56] and in government [57]. Dell Computer is also turning to FOSS and expects it to be widely adopted [58].

Some USA government initiatives contribute to FOSS

Government sponsoring of FOSS is not common, although some examples are reported in [5] including the famous Security Enhanced Linux (SELinux) which can be downloaded directly from the National Security Agency (NSA) Web page [59]. In geomatics, the National Technology Alliance (NTA) has sponsored the impressive Open Source Prototype Research project which had a significant impact on geospatial information organizations in the USA government [60] including the Department of Defense (DoD). More recently, a mission-critical development with FOSS has been reported in IEEE Software [61] and describes how FOSS has been used very efficiently in NASA JPL project.

Adopting a strong FOSS policy may be problematic for the American government since the proprietary software industry strongly supports the USA economy

The software business is estimated to $70B (US) [10] and so it is not surprising to see a vigorous reaction from COTS editors against FOSS [62]. For example, the reference [15] is a Microsoft sponsored study that is trying to counter the FOSS business case and a more formal article [63] gives the official Microsoft perspective on FOSS. A well-balanced essay on the FOSS economy is proposed by David Adams in [64].

10 FOSS in Canada

Canada appears to be behind the curve in adopting FOSS, especially in the public sector

The lack of clear business cases and the underestimation of the strategic value of FOSS partly explain this situation [65]. An excellent assessment of open source software in Canada was conducted by e-Cology for Industry-Canada [40] in 2003. The report also presents the profile of 17 Canadian companies involved in FOSS in varying degrees and offers an enlightening discussion of the revenue strategies that these firms are utilizing to maximize the return on their investment.

A FOSS policy for the Government of Canada has been recently endorsed

In June 2004, the Government of Canada announced a new position on FOSS. It is based on a balanced approach to ensure that governmental policies and guidelines do not bias one software business model over another (FOSS vs COTS vs custom development). Some government departments will address a series of next steps to support the national policy on FOSS including: to review federal procurement practices to ensure a level playing field; to provide advices on software quality and security best practices; to develop a strategy for property rights, patent protection and technology transfer; to provide advice on licensing and other legal issues; etc. [66]. More information is available at `http://www.cio-dpi.gc.ca/fap-paf/oss-ll/oss-ll_e.asp`.

Software development tools, Apache and Linux are the dominant FOSS products in use in Canada

At the time that this report was being written, the use of FOSS in Canada was mostly in software development and in the back-office environment (i.e. servers and network management). It is expected that this trend will remain dominant for the next 1-3 years [65]. Analysts often describe this phenomenon as the horizontal market penetration of FOSS (i.e. one layer of service offered by FOSS).

Some more comprehensive initiatives can be found in the education and health sectors

In British Columbia and Québec, vigorous FOSS projects attempt to integrate a complete software suite for school and other educational uses [40, 67]. McMaster University and the Department of Family Medicine are collaborating to Electronic Health Record (EHR) for family physicians [40]. Most analysts consider that such vertical penetration of FOSS (i.e. through the multiple layers in a specific application domain) is required to support a more widespread penetration of FOSS technology.

11 FOSS and Software Security

Access to source code greatly eases security enforcement

When software is created, it has a level of quality that depends directly on the programmer's competence, experience and professional methodology. To increase the reliability and security of code, it is essential to use some complementary mechanisms such as peer review, testing, quality audits, alpha and beta versioning etc. FOSS and proprietary software rely essentially on the same processes (probably at similar levels) during the main development period. However, after the first public release, FOSS offers the very significant advantage of keeping access to source code. This encourages more peer reviews, testing, and quality audits by a much larger community of users/developers than what would be possible with proprietary code. For closed source software, flaws and code defects are often discovered by some subversive exploits which can lead to some destabilization in large corporations that rely on such COTS packages (i.e. patch and repair). On the contrary, confidence in FOSS may be built faster and, potentially, to a higher degree than with a proprietary equivalent [68, 69]. A myriad of statistics on software vulnerabilities are available in Chapter 6 of reference [8] and they seem to confirm the general perception that open source software is often superior to proprietary code. Reference [70] gives a comparison of the vulnerabilities contained in Red Hat Linux (160) and Windows NT (1200) that appears to be more scientific but great care should be taken to avoid extrapolating this study beyond its original scope. In short, FOSS is not intrinsically more secure than COTS software but the openness of source code makes security enforcement more ubiquitous and less disruptive.

The dilemma on security through obscurity vs. openness was the subject of a heated debate in the cryptographic community in the 80's. The final decision was to make the cryptographic algorithms generally available so as to provide for security assessment and validation by the widest scientific community possible. Whitfield Diffie, the inventor of public key cryptography, and now chief security officer at Sun Microsystems, has repeatedly said that "openness is essential for trust" in software as it was for cryptographic protocols twenty years ago [71, 72]. "Sunshine kills bacteria" [68].

FOSS has three other key advantages

Other advantages for FOSS include:

1. "Leaner and meaner" software systems than COTS equivalents that often suffer from feature bloating. Since they are smaller, open source systems are expected to provide fewer opportunities for exploits.
2. Source code can be enriched with assertions, complementary safety checks etc.
3. Increased code diversity in the "software ecosystem" that could reduce the speed and the proliferation of cyber attacks.

And some increased risks to manage

FOSS is often perceived as a return to more reliance on internal resources for system development and maintenance. For security enforcement, high-quality expertise is scarce and may often have to be developed to adequately cope with the increased responsibilities that FOSS-based systems will require. COTS software has a significant advantage over FOSS by virtue of the intrinsic imputability in the commercial world (often greatly restricted by license agreements!). Access to source code can also be an advantage to an attacker who can try to develop more elaborate attacks on the open source code [63]. Some authors are also concerned about potential infiltration into collaborative development projects by malicious developers who could install backdoors or other undesirable functionalities [73]. At any rate, neither COTS nor custom software are immune to malicious or programming defects that result in information system vulnerabilities. FOSS proponents consider these threats to be exaggerated [74, 75, 76]. As noted later in this report, advantages and disadvantages can only be balanced in a specific project context.

12 Authors' Synthesis

FOSS should not be considered as a panacea, nor ignored as a marginal irrelevant phenomenon

All major forecasting firms predict that FOSS-based systems will continue to expand to the detriment of their COTS equivalents. It seems obvious that the advantages greatly outweigh the disadvantages in many application contexts. Many FOSS programs have achieved a level of maturity and of recognition that raises them to a position of superiority over their commercial equivalents. With the migration of many governments around the world, it is expected that FOSS quality and diversity will continue to improve.

The intrinsic limitations of closed source software may be too stringent for many GoC systems in the future

Even though the closed source strategy appears to be appropriate for the mass market (e.g. domestic/personal uses with no/little programming skills), for military systems and government computing in general, the access to source code and the adoption of open standards are obvious advantages. The need for higher reliability/security, more flexibility/scalability, more competition in software supplies and, finally, direct cost savings will always tend to justify considering FOSS in the next decade.

The R&D communities should demonstrate leadership in FOSS adoption

It seems that the R&D community has an important responsibility to activate projects that could demonstrate the strategic value of FOSS and that would help make clearer business cases for the GoC. One of our prime responsibilities is to perform exploratory activities that could lead to a risk reduction in the technological evolution of our respective department.

Part II

Proposed Way-Ahead for GoC

13 Guiding Principles for a Way-Ahead

FOSS offers a concrete and credible opportunity for R&D communities

The business case for FOSS in Research and Development (R&D) projects was recently studied by some research laboratories, including the National Aeronautics and Space Administration (NASA) Ames Research Centre [77, 78], the National Nuclear Security Administration [79] and by a number of Universities [80]. These studies concluded that FOSS offers an attractive third option to the "build or buy" dilemma, with clear advantages in terms of expertise development, creativity and productivity. In R&D projects, the traditional disadvantages of FOSS (such as technical complexity of software development and long-term maintenance) are less of an issue, since the technical expertise is typically available in labs and many projects aim at building demonstration prototypes.

Diversity in supplies is preferable (Custom Software vs. FOSS vs. COTS)

FOSS offers the flexibility of building a specialized system in an accelerated development process that is at least as efficient as buying COTS components. For R&D projects, the use of FOSS can assure a rapid development (i.e. code reuse and modification) of a high quality code (i.e. well debugged), which will be very difficult to achieve through custom code developed from scratch. In some instances, FOSS-based development is the only reasonable alternative when COTS products are not available (e.g. High Performance Computing [79]) and when custom development is too expensive for the available budget [61]. FOSS helps in avoiding lock-in to proprietary IT products and services and in reducing our dependence on monopolistic technologies.

Open Standards and specifications contribute directly to system interoperability

FOSS implements open standards and specifications that are shared among developers during the design, coding and testing processes. This is generally recognized as a strategic advantage in enforcing interoperability policies between independently developed systems [81].

Evaluation of FOSS must be done on a case-by-case basis

While very attractive in general, FOSS must be evaluated in the context of each project on a case-by-case basis in order to determine if the advantages outweigh the disadvantages in practice. In the case of GoC, special attention must be paid to the protection of classified technologies, the protection of intellectual property and the selection of a license suitable for the specific activity. Some preliminary guidelines are available in this document (Parts III & IV).

14 Proposed Way-Ahead for GoC

Adopt FOSS progressively

Adoption of FOSS development methods can have fundamental and far-reaching consequences on engineering practices, especially if the objective is to contribute actively to an open source project. It is recommended that experience be gained with FOSS as a passive user first, then to become progressively more involved by reporting bugs, suggesting new features, and modifying existing code before engaging in active development within a collaborative project. Figure 2 illustrates the evolution schema of a FOSS user/developer.

Consider FOSS-based solutions in contractual work when technically competitive with other development strategies

GoC should consider FOSS solutions alongside proprietary ones in IT procurements especially in large development contracts such as Technology Demonstration Project (TDP). According to Industry-Canada [82], contracts are awarded on a value-for-money basis and no Public Works Government Services Canada (PWGSC) rules restrict FOSS uses in federal government contracting and no Treasury Board rules restrict FOSS use in our internal programs. The Canadian position on FOSS that has been endorsed in June 2004 confirms that no barriers to procurement should be maintained.

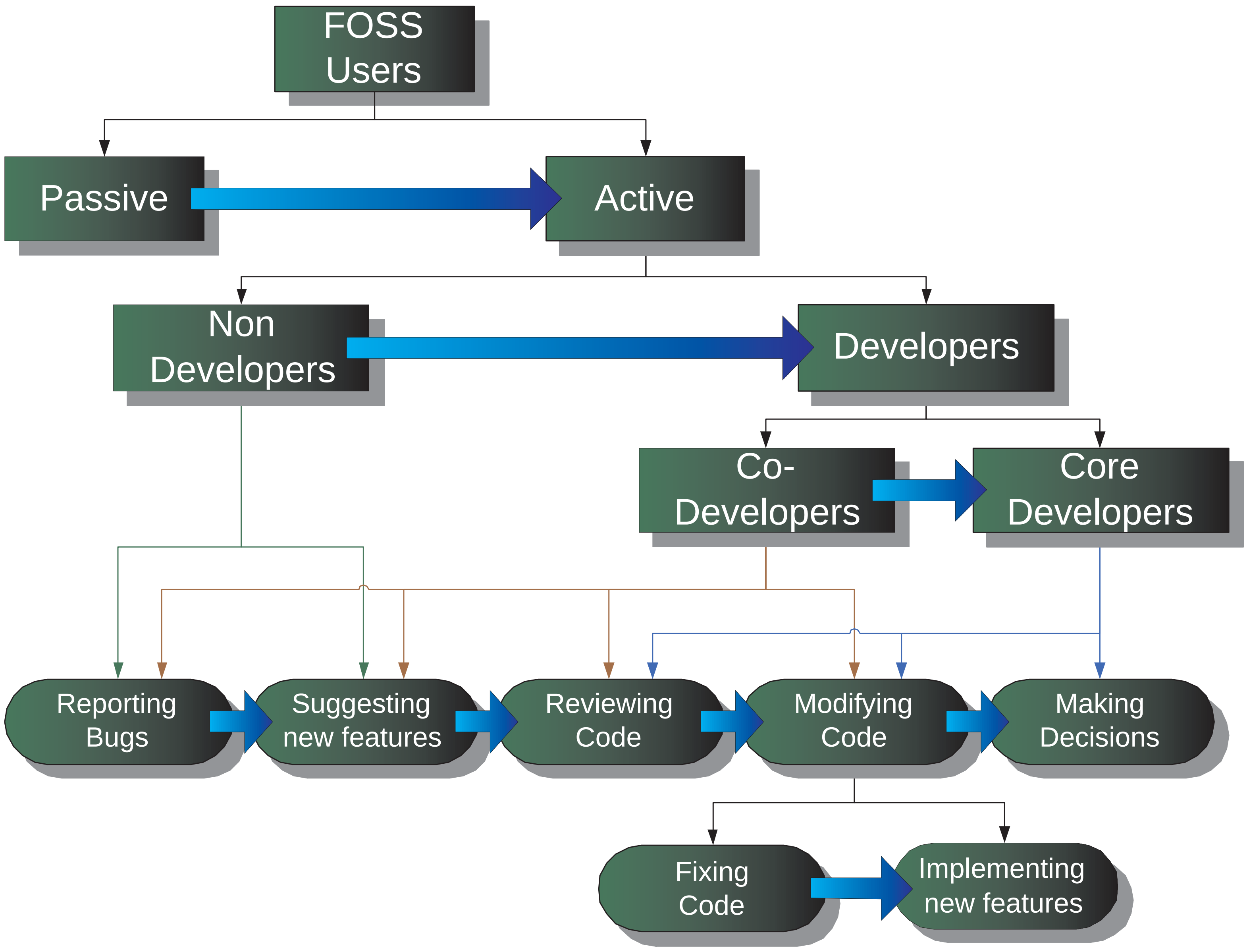

Figure 2: *The evolution schema of a FOSS user/developer*

Part III

Catalogue of Selected FOSS that could be considered in GoC Projects

15 Overview of Available FOSS

Of the tens of thousands FOSS projects, only a few hundreds have achieved enough maturity to be considered for inclusion within GoC systems

According to Spinellis and Szyperski [11], more than 115,000 open source projects were registered at the main four open source forums (30,000 at `http://www.freshmeat.net`, 70,000 at `http://www.sourceforge.net`, 5,400 at `http://www.cpan.org` and 10,000 ports distributed with FreeBSD). Note that some projects are registered twice and many projects are inactive/dead. To select the best-of-breed code, it is recommended that reference be made to credible lists of mature or secure FOSS such as the ones that appear below. These lists provide navigation aids to help identify suitable FOSS for one specific application, but should not be taken as an official government position, policy or decision about the value of each specific software component.

MITRE has compiled a list of 115 FOSS applications that offer an excellent starting point for identifying high-quality software

MITRE has focussed on "Generally Recognized As Safe" (GRAS) open source software. GRAS FOSS requires that the software program be: (a) commercially supported, (b) widely used and accepted, (c) have a proven track record for security and reliability. The complete list is available in [5] and is included in our list proposed in Appendix A.

The "Generally Recognized As Mature" (GRAM) is another interesting list of 39 FOSS

The GRAM list is maintained by David Wheeler, a professed computer security guru and is accessible through [83].

The Software Development Magazine identifies 27 high-profile FOSS projects

Rosalyn Lum proposes a list of high-profile open source projects that she considers ready for primetime in Software Development Magazine (March 2004) [84].

The Interchange of Data between Administrations (IDA) migration guide lists multiple FOSS of high quality

The Interchange of Data between Administrations (IDA) migration guidelines include a comparison of many FOSS alternatives to COTS software including: operating systems, office suites, mail servers, groupware, web services, document management and databases etc. [26]

16 FOSS Relevant to GoC

Contributions to FOSS from large IT companies are among the best quality software available

Many large IT companies contribute massively to open source software. In [85], some 137 high-quality FOSS has been identified from the following manufacturers: Sun Microsystems (28), Silicon Graphics (15), Hewlett-Packard (33), Red Hat (2), AT&T (24) and IBM (35).

DRDC has prepared a more comprehensive FOSS list that includes general purpose and scientific software FOSS (approximately 392 FOSS)

Based on the lists introduced above and on independent research, a more comprehensive list was constructed by the authors. It offers guidance on the selection of scientific software programs in addition to general purpose computing. Our objective was to give a more comprehensive picture of the richness and diversity of currently available software.

Appendix A provides navigation aids to help identify suitable FOSS

In Appendix A, the reader will find a list of representative free and open source software at the time this study was conducted (winter 2004). This list could facilitate the identification of candidate products for evaluation within the GoC. Many of them have been used in the past in some DRDC projects but no systematic testing or rigorous evaluation has been carried out on the FOSS listed below. Therefore they should not be considered as "government approved" or officially recommended by the GoC. It is strongly recommended that each software component be assessed before its integration. A rigorous methodology is proposed in Part IV.

Part IV

GoC Guidelines to Assess FOSS

17 Guideline Principles

FOSS and COTS software should be evaluated side by side

The process to evaluate FOSS or COTS software is essentially the same and a side-by-side comparison remains the best approach to identify the pros/cons of each option [86, 87, 38]. The evaluation process can vary a lot in duration and in technical depth depending on the application context and the project requirements.

Advantages / disadvantages should be compared in each specific project context

It is to be noted that most COTS packages are designed for a very broad client spectrum and typically include a huge diversity of functionalities and potential configurations. On the other hand, FOSS tends to be more specialized since it is often designed to meet the requirements of a specific user community. A direct comparison of both types of software against a well-defined application context is recommended to determine the best option. In short, the main evaluation steps include:

1. Understand the requirements and the application context.
2. Prioritize the selection criteria.
3. Identify COTS and FOSS candidates.
4. Compare the best candidate options.
5. Analyze the best products in depth (e.g. performance, security audit, cost), if needed.
6. Seek approval from local management and from the project client.
7. Document lessons learned.

Special attention must be paid to the license model

At this time, it does not seem appropriate for GoC to select one license model and to impose it on all projects. It seems preferable to identify the most suitable license model in the context of each project including due consideration of:

1. Intellectual Property (IP) protection,
2. National and international partnership constraints,
3. and client preferences.

18 Recommended Evaluation Steps

Step #1 - Define the application context

1.1. Clarify objectives and client expectations.

1.2. Document project constraints such as classification level, partners' demands, compatibility with development/execution environment, compatibility with legacy systems and existing information formats, mandatory standards to comply to, etc.

1.3. Prioritize evaluation criteria to compare software including functionality, cost, required support/maintenance, reliability, security, performance, flexibility, scalability, user-friendliness, legal/license issues and other issues specific to the applications.

1.4. Estimate internal (and external) resources available to the project (including money, time and technical expertise which may be more demanding for FOSS development).

1.5. Seek support from an experienced colleague that would 'mentor' the evaluation process and help in avoiding pitfalls.

Step #2 - Identify candidates

2.1. Look in Appendix A of this report for your application domain.

2.2. Perform complementary search on the Internet, including specialized sites: `http://www.sourceforge.net`, `http://www.gnu.org/directory`, `http://www.freshmeat.net`, `http://www.debian.org`, `http://www.savannah.gnu.org`, `http://www.icewalkers.com`, `http://www.cpan.org`.

2.3. Gather technical reviews and product comparisons.

Step #3 - Compare the best 3-4 options side by side

3.1. Consult existing lists of 'reliable' FOSS such as the Generally Recognized As Safe (GRAS), Generally Recognized As Mature (GRAM), and IDA.

3.2. Read/assess technical product reviews (COTS and FOSS). Remain vigilant concerning excessively biased evaluations (encountered for both COTS and FOSS!)

3.3. Consider compatibility of the software with existing libraries and your development and execution environments.

3.4. Assess maturity and technical risk through "download counts" (and other popularity measures), product longevity (often revealing maturity) and market penetration.

3.5. Summarize your findings in a spreadsheet that includes your criteria as prioritized in Step 1.3 FOSS evaluation spreadsheets.

Step #4 - If appropriate, perform an in-depth code analysis

4.1. If time permits, download evaluation versions to confirm performance, compatibility, user-friendliness etc.

4.2. Clarify details with suppliers/developers.

4.3. Evaluate licenses and seek advice from your local Business Development Service (BDS) for IP protection if needed.

4.4. If appropriate, perform detailed code analysis with software analysis tools to detect flaws and other types of defects. See [88].

4.5. If appropriate, evaluate the feasibility of adding new functions.

Step #5 - Seek approval from client and local management

5.1. Even if software packages are used unchanged (no code development), it is recommended to inform your local management (and possibly the project client) of the use of FOSS.

5.2. If FOSS is used to build a research prototype involving substantial code development, seek approval from your local management and project client (if any). See advice on "Licenses in GoC software development" below.

5.3. If a GoC development is considered for distribution in one of the FOSS networks, estimate the additional effort required to clean up the code, to improve the documentation and to support the community in a timely fashion once released in one of the FOSS networks. Seek approval from your local management and project client (if any). See advice on "Licenses in GoC software development" below.

5.4. If a GoC development project is to be carried out in a collaborative open source paradigm, it could be necessary to build a comprehensive business case to justify this approach. Seek approval from your from local management and project client (if any). See advice on "Licenses in GoC software development" below. Guidance can be found on the FSF web site and in this book by Jan Sandred [89].

Step #6 - Document lessons learned

6.1. Summarize lessons learned from your evaluation in a brief tech note to share your experience with GoC communities.

6.2. Keep track of FOSS usage and of the changes made to the original software by a rigorous software revision control throughout the development process. The revision control data must remain available to the Crown after the development has finished. Without a complete history of the code development in a software revision control system, the new code may fall under other license models such as the GPL by default.

19 Licenses in GoC Software Developments

Licenses can be grouped into two main categories

A detailed analysis of FOSS licenses goes beyond the scope of this document. For the purposes of this report, it is sufficient to say that the multiple licenses currently registered at the OSI (roughly 48) can be classified into two main categories: (1) those inspired by the GPL, which demand that all complementary code developments be integrated with open source code only and published under a GPL-compatible license (e.g. LGPL, Zope etc.); (2) those which allow a combination of FOSS with proprietary code (e.g. Berkeley Software Distribution (BSD), Mozilla etc.) [90].

The end-user usually prefers the GPL strategy

MITRE looked at the various licenses from the perspective of the end-user (the USA DoD in this instance) and they concluded that the GPL license is the best one by virtue of the diversity of source code and the 'total' visibility of implementations. This guarantees faster and more autonomous responses to cyber threats [5]. It is expected that the most end-users will also prefer the GPL model.

Research establishments typically prefer BSD or Mozilla models

Alternatives to the "rather strict GPL licensing" were developed to accommodate organizations, which need to integrate open source software with some proprietary components. This is the case of many research organizations that want to protect their innovative IP while demonstrating it efficiently with a FOSS-based prototype. NASA compared the various licenses and selected the Mozilla model in 2002 [77] and they are currently moving toward the development of their own license [78] inspired by this licensing model. The Canadian National Research Council (NRC) analysis concludes that BSD and Academic Free License (AFL) licenses are often the two most appropriate license models for them [91]. Gartner believes that licenses inspired by BSD and Mozilla will increase in popularity since they offer more flexibility [4]. Given the complexity of license selection and the multiple options available, researchers should obtain assistance from their local BDS when in doubt concerning legal implications.

The legal background is currently challenged

It can be observed that companies whose business model of proprietary software is somewhat on a collision course with FOSS are not fighting back solely on technical grounds but also use legal weapons as well as Fear, Uncertainty, and Doubt (FUD). The developer of FOSS who is targeting substantial deployment of his/her work should be well aware of the legal pitfalls and issues which might occur as a result of events which are occurring in the software community, some of them viewed as clearly abusive by various observers [92]. Examples include the current legal battles between Santa Cruz Operation (SCO) and IBM. To keep track and be better aware of such undercurrents, see for instance `http://www.groklaw.net`, `http://www.osriskmanagement.com`, and `http://www.pubpat.org`. The Open Source Development Labs `http://www.osdl.org`, home to Linus Torvalds, the creator of Linux, is also a good source of information on issues and current news related to intellectual property management and FOSS. Monitoring of this “legal war” between the commercial world and FOSS communities is recommended in GoC for the next 24 months at least.

Many good references on FOSS licenses can be found on the internet

The reality of the various licenses is much more complex than the overview presented above. A complementary study is being initiated by the Departments of Justice and Industry to provide legal opinions on licensing , property rights, technology transfers and other legal issues. Therefore, it is expected that more precise guidelines on FOSS licensing will be available soon. For readers interested in more comprehensive license comparison, we recommend the following references:

- Free Software Foundation - GPL “school of thoughts” (rather idealistic) [84] .
- Open Source Initiative - multiple license models (some more pragmatic) [1].
- European guide to choosing free software licenses [93].
- Detailed analysis of FOSS licenses [94].
- Comparison of FOSS licenses from USA DoD perspective [5].
- Comparison of FOSS licenses from a developer perspective (i.e. NASA) [77].
- Good overview of FOSS licenses [95].
- A practical look at FOSS integration into commercial products [96].
- A rather theoretical look at licenses, giving interesting trends [97].
- GPL copylefting vs. non-copylefting licenses [98].
- Complementary norms and legal tactics to FOSS licensing (good!) [6].
- A review of software patents issues [99, 9].

20 Migration to FOSS

In some projects, a complete migration to FOSS may be envisaged

In some cases, it could be appropriate to consider the complete migration of a GoC project to open source software. Even if many recent experiences have been very successful at NASA [61], in health care [100], in industry [96, 101], and in some universities [102] , it must be understood that the FOSS collaborative process is a new development paradigm that involves significant cultural changes. The South African Government recommends in [45] the VCS model (i.e. demonstrating Value–building Capacity–mobilizing Support) that appears to be a great conceptual framework for FOSS migration studies. Hewlett-Packard has defined an innovative software engineering paradigm for large corporations, called Progressive Open Source, that appears to be very well done [103]. More about the cultural changes can be found in the following references:

- Performance of scientific applications on Linux clusters [104].
- Back-office migration supports anti-terrorism (in OSO) [105].
- Software requirement understanding with FOSS [106].
- FOSS configuration management [107].
- Linux in government - white paper [57].
- Good summary article on Linux market penetration [108].
- Desktop Linux technology and market overview [52].
- IDC white paper on Linux Total Cost of Ownership (TCO) (IBM sponsored) [109].
- IDC white paper on expanding Linux in enterprise (IBM sponsored) [110].
- IDC white paper on accelerating Linux adoption (IBM sponsored) [111].
- Bloor Research assessment of Linux maturity [112].
- Impact of FOSS adoption in government [113].
- Perceived benefits of FOSS in public sector [114].
- The growing popularity of Linux on Wall Street [115].
- Gartner recommendation to use FOSS in government [116].
- Excellent migration guide from QinetiQ (UK) [117].
- Very detailed migration guidelines from IDA [26].
- Another very impressive migration guide from the German Ministry of the Interior [27].
- Free/Libre and Open Source Software (FLOSS): Survey and Study [28].
- FOSS in European firms and public institutions [29].

- Commercial motivations and policy implications [30].
- Policy within the European Union [31].
- FOSS markets and business methods [32].
- Survey of developers of FOSS [33].
- Software source code survey [35].

Multiple tools exist to perform cost analysis

Many cost models have been developed for software systems. Some are rather simple and easy to use and could be appropriate for small projects of partial migration to FOSS technology:

- Notes on a cost comparison spreadsheet [118].
- Cost Comparison Spreadsheet: A cost comparison model.
- A basic introduction to FOSS with cost estimation [119].
- TCO for Linux in the Enterprise [120].
- A collection of statistics on TCO in chapter 7 [8].

Scientists leading a major project or considering a comprehensive migration to FOSS could prefer more complex cost models such as:

- A top view article from Gartner which leads to TCO analyses [121].
- MITRE study on FOSS business case (see chapter 3) [122].
- A Forrester report on total economic impact of Microsoft vs. Linux/Java 2 Enterprise Edition (J2EE) [23].
- Linux vs Windows TCO comparison [123].
- Danish desktop evaluation model for FOSS migration [124].

References

1. Stallman, Richard (2003). Basic Idea Behind Open Source. Paper. Open Source Organization,. http://www.gnu.org.
2. OpenSource.org. Web Site. Open Source Organization,. http://www.opensource.org.
3. Wu, Ming-Wei and Lin, Ying-Dar (2001). Open Source Software Development: An Overview. (Technical Report IEEE 0018-9162/01). National Chiao Tung University, Taiwan.
4. Driver, M. (2001). The Future of Open-Source Software. (Technical Report SPA-13-7536). Gartner.
5. Bollinger, Terry (2003). Use of Free and Open-Source Software (FOSS) in the U.S. Department of Defense. (Technical Report MP 02W0000101 v1.2.04). MITRE.
6. O'Mahony, Siobham (2003). Guarding the Commons: How Community Managed Software Projects Project Their Works. Paper. Harvard University Graduate School of Business Administration.
7. Casamento, Gregory. Petition Against Software Patents. Web Page,. http://www.petitiononline.com/pasp01/petition.html.
8. Wheeler, David A. (2003). Why Open Source Software / Free Software (OSS/FS)? – Look at the Numbers!. Paper. Personal Web Page.
9. Rosen, Lawrence (2004). Patents in an Open Source World. *newsforge.com*.
10. United Nation Conference on Trade and Development Secretariat (2003). Free open-source software : Implications for ICT policy and development, Ch. 4. United Nation.
11. Spinellis, Diomidis and Szyperski, Clemens (2004). How Is Open Source Affecting Software Development?. Paper. IEEE SOFTWARE.
12. James W. Paulson, Giancarlo Succi and Eberlein, Armin (2004). An Empirical Study of Open-Source and Close-Source Software Products. (Technical Report IEEE 0096-5589/04). General Dynamics Canada, University of Bozen, University of Sharjah.
13. Dowling, Ted (2000). Software COTS Components – Problems, And Solutions?. In *RTO SCI Symposium on "Strategies to Mitigate Obsolescence in Defense Systems Using Commercial Component"*, pp. 28–1—28–8.
14. Calvin, James B. and Rodgers, Steven L. (2003). The Case for Open Source Tools is Compelling. *COTS Journal*, pp. 25–29.
15. Evans, David S. and Reddy, Bernard (2003). Government Preferences for Promoting Open-Sources Software: A Solution in search for a problem. (Technical Report 9 Mich. Telecomm. Tech. L. Rev. 313 (2003)). National Economics Research Associates.
16. Bishop, Todd (2004). Studies on Linux help their patron: Microsoft. *Seattle Post-Intelligencer*.

17. VeriTest (2003). Microsoft Windows Small Business Server 2003 vs. Red Hat Enterprise Linux ES 2.1 Deployment. Test Report. VeriTest.

18. Rubin, Howard (2003). Mainframe Linux Benchmark Project Audit Report. Audit Report. META Group, Inc.

19. VeriTest (2003). Microsoft Windows Server 2003 vs. Linux Competitive File Server Performance Comparison. Test Report. VeriTest.

20. VeriTest (2003). Microsoft Windows Server 2003 with Internet Information Services (IIS) 6.0 vs. Linux Competitive Web Server Performance Comparison. Test Report. VeriTest.

21. Bozman, Jean, Gillen, Al, Kolodgyand, Charles, and al (2002). Windows 2000 Versus Linux in Enterprise Computing. Test Report. IDC.

22. Morrow, Keith (2004). 7-Eleven Upgrades Its Strategic Retail System, Lowers Costs, and Improves Customer Service. Case study. Microsoft Corporation.

23. John R. Rymer and Bob Cormier (2003). The Total Economic Impact of Developing and Deploying Applications on Microsoft and J2EE/Linux Platforms. Technical Report. Forrester Research.

24. Murphy, Paul (2004). Getting the Facts About Windows and Linux. *LinuxInsider*.

25. Office of the e-envoy. Leading the drive to get the UK online. UK online,. `http://www.e-envoy.gov.uk/Home/Homepage/fs/en`.

26. netproject (2003). The IDA Open Source Migration Guide. (Technical Report OSPL/EEC-01.10). Interchange of Data between Administrations.

27. KBSt (2003). Migration Guide – A guide to migrating the basic software components on server and workstation computers. (Technical Report 1.0). Bundesministerium des Innern, Germany.

28. Ghosh, Rishab Aiyer, Krieger, Bernard, Glott, Ruediger, and Robles, Gregorio (2002). Free/Libre and Open Source Software: Survey and Study – Deliverable D18: FINAL REPORT. (Technical Report Part 0: Table of Contents and Executive Summary). International Institute of Infonomics, University of Maastricht.

29. Wichmann, Thorsten (2002). Free/Libre and Open Source Software: Survey and Study – FINAL REPORT. (Technical Report Part 1: Use of Open Source Software in Firms and Public Institutions – Evidence from Germany, Sweden and UK). Berlecon Reasearch.

30. Wichmann, Thorsten (2002). Free/Libre and Open Source Software: Survey and Study – FINAL REPORT. (Technical Report Part 2: Firms' Open Source Activities: Motivation and Policy Implications). Berlecon Reasearch.

31. Ghosh, Rishab Aiyer, Krieger, Bernard, Glott, Ruediger, and Robles, Gregorio (2002). Free/Libre and Open Source Software: Survey and Study – Deliverable D18: FINAL REPORT. (Technical Report Part 2B: Open Source Software in Public Sector: Policy within the European Union). International Institute of Infonomics, University of Maastricht.

32. Spiller, Dorit and Wichmann, Thorsten (2002). Free/Libre and Open Source Software: Survey and Study – FINAL REPORT. (Technical Report Part 3: Basics of Open Source Software Markets and Business Models). Berlecon Reasearch.

33. Ghosh, Rishab Aiyer, Robles, Gregorio, Krieger, Bernard, and Glott, Ruediger (2002). Free/Libre and Open Source Software: Survey and Study – Deliverable D18: FINAL REPORT. (Technical Report Part 4: Survey of Developers). International Institute of Infonomics, University of Maastricht.

34. Ghosh, Rishab Aiyer, Robles, Gregorio, Krieger, Bernard, and Glott, Ruediger (2002). Free/Libre and Open Source Software: Survey and Study – Deliverable D18: FINAL REPORT. (Technical Report Part 4A: Survey of Developers – Annexure on validation and methodology). International Institute of Infonomics, University of Maastricht.

35. Ghosh, Rishab Aiyer, Robles, Gregorio, and Glott, Ruediger (2002). Free/Libre and Open Source Software: Survey and Study – Deliverable D18: FINAL REPORT. (Technical Report Part 5: Software Source Code Survey). International Institute of Infonomics, University of Maastricht.

36. Ghosh, Rishab Aiyer (2002). Free/Libre and Open Source Software: Survey and Study – Workshop on Advancing the Research Agenda on Free / Open Source Software. (Technical Report Workshop Report). International Institute of Infonomics, University of Maastricht.

37. The Swedish Agency for Public Management (2003). Free and Open Source Software.

38. The Swedish Agency for Public Management (2003). Free and Open Source Software – a feasibility study.

39. Ministry of Science (2003). The Danish Software Strategy. Software Strategy. Danish Ministry of Science, Technology and Innovation.

40. e-cology Corporation (2003). Open Source Software in Canada – A Collaborative Fact Finding Study. Technical Report. e-cology Corporation.

41. e-cology Corporation (2003). Open Source Software in Canada – Appendices. Technical Report. e-cology Corporation.

42. Open Source Observatory (OSO). Web Site. Open Source Observatory,. http://europa.eu.int/ISPO/ida/jsps/index.jsp?fuseAction=showChapter&chapterID=452&preChapterID=0.

43. Weber, Stephen (2002). Open Source Software in Developing Economies. Technical Report. University California, Berkeley.

44. Enav, Peters (2003). Israel stops buying Microsoft software. *The Associate Press*.

45. South African Government (2003). Using Open Source Software in the South African Government – A Proposed strategy compiled by the Governmen Information Technology Officers Council. (Technical Report Version 3.3). Government Information Technology Officers Council.

46. Riley, James (2004). Tax to Open Up Software Policy. *Australian IT*.

47. Varghese, Sam (2003). ACT set to adopt open source bill. *IDG News Service, West Africa bureau*.

48. AFUL's Site. French speaking Linux and Libre Software Users' Association. AFUL,. http://www.aful.org/index.html.

49. AFUL's Site on Peru OSS. French speaking Linux and Libre Software Users' Association. AFUL Web Page on Peru,. http://www.aful.org/politique/perou/english/referencias.html.

50. proposicion.org.ar. Government's documents about Free Software. Open Source Organization Web Page,. http://proposicion.org.ar/doc/gob/.

51. Logan, D., Arevolo, W., and Bittinger, S. (2004). Open-Source Software Means Benefits for Emerging Nations. (Paper SPA-19-4429). Gartner.

52. Decrem, Bart (2003). Desktop Linux Technology & Market Overview. Technical Report. Open Source Application Foundation.

53. Horstmann, Jutta and Muehlig, Jan (2003). Linux Usability Study. (Paper Version 1.01). Relevantive.

54. Hecker, Frank (2000). Setting up Shop: The Business of Open-Source Software. Paper. Netscape.

55. Fricke, Pierre (2003). Linux Strategies and Solutions 2003: Linux Server Suppliers Contend for Leadership. Technical Report. D.H. Brown Associates, Inc.

56. Jacob, Bart, Janson, David, Mark, Oliver, and Marras, Fabio L (2002). Linux and Branch Banking. (Technical Report SG24-6909-00). ibm.com/redbooks.

57. Fisher, Mary Ann (2002). Linux in Government White Paper. Paper. IBM.

58. Lowery, J. Craig (2003). Dell's Open Source Software Philosophy. In *Open Standards/Open Source for National and Local eGovernment Programs in the U.S. and EU: Conference*, eGovOS.

59. National Security Agency (NSA). Security-Enhanced Linux. NSA Web Page,. http://www.nsa.gov/selinux/index.html.

60. TYBRIN (2002). Open Source Prototype Research – Open Source Process. Technical Report. NTA – NCAT.

61. Norris, Jeffrey S. (2004). Mission-Critical Development with Open Source Software: Lessons Learned. (Technical Report vol 21 no1). IEEE Software.

62. Frye, Emily (2003). Open-Source Software, Proprietary Software: Implications for National and Economic Security. Technical Report. THE CIP REPORT.

63. Mundie, Craig (2002). Security: Source Access and the Software Ecosystem. Technical Report. Microsoft Corporation.

64. David Adams (2004). Free Can Mean Big Money: The Open Source Economy. Technical Report. OSNews.

65. Patrick, Ryan B. (2003). Linux deepening its Canadian foothold. *Computerworld*.

66. Branch, Chief Information Officer (2004). GoC Proposed Position on Open Source Software and Next Steps. Presentation Power Point. Treasury Board Secretariat of Canada.

67. Dumais, Michel (2004). Le logiciel libre en éducation : le projet MILLE. *Accération du CRIM*, **4**, 26–27.

68. Hissam, S.A., Plakosh, D., and Weinstock, C. (2002). Trust and Vulnerability in Open Source Software. (Technical Report 20020208). IEE Proceeding online.

69. ITsecurity (2002). The Strengths and Weaknesses of Open Source Software – And Its Role in the Security Model. Paper. ITsecurity.com:.

70. Jiwnani, K. and Zelkowitz, M. (2002). Maintaining Software with a Security Perspective. (Technical Report IEEE). International Conference on Software Maintenance (ICSM'02).

71. Judge, Peter (2002). Diffie defends open-source security. *ZDNet, UK*.

72. Diffie, Whitfield (2003). Perspective: Decrypting the Secret to Strong Security. *news.com*.

73. Jones, A. Russell (2004). Open Source Is Fertile Ground for Foul Play. *DevX.com*.

74. Stone, Mark (2004). Is Open Source Secure?. *www.devx.com*.

75. Anderson, Ross (2002). Security in Open versus Closed Systems – The Dance of Boltzmann, Coase and Moore. Paper. Cambridge University, England.

76. Angelius, Ladd (2004). Who's Guarding the Guards? We Are.. Technical Report. DevX.com.

77. Moran, Patrick J. (2003). Developing An Open Source Option for NASA Software. (Technical Report NAS-03-009). NASA Ames Research Center.

78. Geurts, Bryan A. (2004). NASA's Open Source Licensing. *news.osdir.com*, Vol. 1.1.

79. Advanced Simulation and Computing (ASCI) (2003). Request for Information: Open Source Software Development Acceleration (OSSODA). (Technical Report UCRL-MI-153295). National Nuclear Security Administration (NNSA).

80. Dongarra, Jack, Malony, Allen, Hollingsworth, Jeffrey, and al. (2003). Response to the Request for Information: Open Source Software Development Acceleration. Technical Report. National Nuclear Security Administration (NNSA).

81. Office of Government Commerce (2002). Open Source Software – Guidance on Implementing UK Government Policy. Technical Report. Cabinet Office Minister of State.

82. Gagné, Claude (2003). Innovation in ICT Innovation in Canada. Presentation Power Point. Industry Canada.

83. Wheeler, David A. (2003). GRAM List. Paper. Personal Web Page. http://www.dwheeler.com/gram.html.

84. Ravella, John and Lum, Rosalyn (2004). Free as in Freedom. *Software Development Magazine*.

85. Carbone, Richard (2004). Contributions to FOSS from the Large IT Companies. Internal Document.

86. Wheeler, David A. (2003). How to Evaluate Open Source Software / Free Software (OSS/FS) Programs. Paper. Personal Web Page. http://www.dwheeler.com/oss_fs_eval.html.

87. Surman, Mark and Diceman, Jason (2003). Multimedia Training Kit – Choosing Open Source Software Handout. Paper. The Commons Group, for the Association for Progressive Communications (APC).

88. Crispin Cowan (2003). Software Security for Open-Source Systems. Technical Report. IEEE Computer Society.

89. Sandred, Jan (2001). Managing Open Source Projects, Robert Ipsen.

90. GNU Project. Various Licenses and Comments about Them. GNU Project web server,. http://www.fsf.org/licenses/license-list.html.

91. Désilets, Alain, Jenkins, Stephen, Kark, Anatol W., and al (2004). Open Source Software in NRC. Guidebook. National Research Council Canada – Information Management Services Branch.

92. Moglen, Eben (2004). Now They Own It, Now They Don't: SCO Sues Novell to Stay Afloat. *LinuxDevices.com*.

93. ATICA (2002). Guide to choosing and using free software licences for government and public sector entities. Technical Report. ATICA – Agency for Information and Communications Technologies in the Civil Service.

94. ATICA (2002). Guide to choosing and using free software licences for government and public sector entities, Appendix: Detailed analysis of licences. Technical Report. ATICA – Agency for Information and Communications Technologies in the Civil Service.

95. Webbink, Mark (2003). Understanding Open Source Software. *GROKLAW*.

96. Ruffin, Michel and Ebert, Christof (2004). Using Open Source Software in Product Development : A Primer. Study. IEEE Software.

97. Lerner, Josh and Tirole, Jean (2002). The Scope of Open Source Licensing. Paper. Harvard University and NBER; University of Toulouse and MIT.

98. Aigrain, Philippe (2002). A Framework for Understanding the Impact of GPL Copylefting vs. non Copylefting Licenses. Paper. European Commission.

99. McOrmond, Russel (2003). A Review of Software Patent Issues. *Flora, Ca.*

100.Fitzgerald, Brian and Kenny, Tony (2004). Developing an Information Systems Infrastructure with Open Source Software. Study. IEEE Software.

101.Lussier, Stepane (2004). New Tricks: How Open Source Changed the Way My Team Works. Study. IEEE Software.

102.Serrano, Nicolas, Calzada, Sonia, Sarriegui, Jose Mari, and Ciordia, Ismael (2004). From Proprietary to Open Source Tools in Information Systems Development. Study. IEEE Software.

103.Dinlelacker, Jamie, Garg, Pankaj K., Miller, Rob, and Nelson, Dean (2001). Progressive Open Source. Study. Hewlett-Packard Compagny.

104.Dressler, Jean-Marc and Kandadai, Swamy N. (2001). Performance of Scientific Applications on Linux Clusters. Performance Technical Report. IBM.

105.Open Source Observatory (2003). Back-office migration supports anti-terrorism. Technical Report. Bundeskartellamt.

106.Scacchi, Walt (2001). Understanding the Requirements for Developing Open Source Software Systems. Paper. Institute for Software Research, University of California.

107.Asklund, Ulf and Bendix, Lars (2001). Configuration Management for Open Source Software. Paper. Lund Institute of Technology and Aalborg University.

108.Macvittie, Lori (2004). Inside Linux. Study. Network Computing.

109.Gillen, Al, Kusnetzky, Dan, McLarnon, Scott, and Perry, Randy (2003). Linux and Intel-Based Servers: A Powerful Combination to Reduce the Cost of Enterprise Computing. WhitePaper. IDC.

110.Gillen, Al, Kusnetzky, Dan, Melenovsky, Mark, and North, Bill (2003). Expanding Linux System Configurations for Enterprise Deployment. White Paper. IDC.

111.Gillen, Al, Kusnetzky, Dan, and Rosen, Michele (2003). Accelerating the Adoption of Enterprise Linux Through IBM Software Solutions. White Paper. IDC.

112.Claybrook, Bill (2002). Linux Is Ready Scalability, Reliability, Security, Flexibility, and Total Cost of Ownership Considerations. Technical Report. Bloor Research North America.

113.Schmidt, Klaus M. and Schnitzer, Monika (2002). Public Subsidies for Open Source? Some Economic Policy Issues of the Software Market. Paper. University of Munich, CEPR and CESifo.

114.Taylor, Graham (2001). Open Source – Coming of Age. Paper. OpenForum Europe.

115.Shahrawat, Dushyant (2002). Wall Street Romances the Penguin: The Growing Popularity of Linux. Research Notes. TowerGroup.

116.Drakos, Nikos, Mai, Andrea Di, and Simpson, Robin (2003). Open-Source Software Running for Public Office. (Technical Report AV-19-5251). Gartner.

117.Briggs, Julie and Peck, Dr Matthew (2003). QinetiQ Analysis of Open Source Solution Implementation Methodologies – QOSSIModo. (Technical Report Version 1). QinetiQ.

118.netproject (2003). Notes on the use of the Cost Comparison Spreadsheet. (Technical Report version 1, see also Excell spreadsheet in References). Interchange of Data between Administrations.

119.Meng, Tan Tze (2003). The Case for Open Source: OSS vs Proprietary Software. (Technical Report Version: 1.2). MNCC OSSIG Awareness Sub-Group Paper.

120.Robert Frances Group (2002). Total Cost of Ownership for Linux in the Enterprise. Study. Robert Frances Group.

121.Smith, David Mitchell, Simpson, Robin, Silver, Michael A., and Fiering, Leslie (2003). Linux on the Desktop: The Whole Story. (Technical Report AV-20-6574). Gartner.

122.Kenwood, Carolyn A. (2001). A Business Case Study of Open Source Software. (Technical Report MP 01B0000048). MITRE.

123.Cybersource (2002). Linux vs. Windows – Total Cost of Ownership Comparison. (Technical Report Version 1.0.1). Cybersource.

124.Offentlig Information Online (OIO) (2003). Desktop Evaluation Model. (Technical Report 0.8). Danish Government.

List of Acronyms

AFL	Academic Free License
AFPL	Aladdin Free Public License
AFUL	Association Francophone des Utilisateurs de Linux et des Logiciels Libres
AOL	America On Line
ANSI	American National Standards Institute
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
BBS	Bulletin Board System
BDS	Business Development Service
BSD	Berkeley Software Distribution
CAD	Computer Aided Design
CF	Canadian Forces
CGI	Common gateway interface
CORBA	Common Object Request Broker Architecture
COTS	Commercial Off-the-Shelf
CPAN	Comprehensive Perl Archive Network
CPU	Central Processing Unit
CVS	Concurrent Versions System
DBMS	Database management system
DLL	Dynamic Link Library
DND	Department of National Defence
DNS	Domain Name Server/Service
DoD	Department of Defense
DRDC	Defence Research & Development Canada
EHR	Electronic Health Record
EJB	Enterprise JavaBeans
EPS	Encapsulated PostScript
ERP	Enterprise resource planning
FAQ	Frequently-Asked Questions
FLOSS	Free/Libre and Open Source Software
FOSS	Free and Open Source Software
FS	Free Software
FSF	Free Software Foundation
FTP	File Transfer Protocol
FUD	Fear, Uncertainty, and Doubt
GIS	Geographic Information System
GNU	GNU’s Not Unix
GoC	Government of Canada
GPL	General Public License
GPS	Global Positioning System
GRAM	Generally Recognized As Mature
GRAS	Generally Recognized As Safe
GRASS	Geographic Resources Analysis Support System

GSF	Generalized Satellite Format
GUI	Graphical User Interface
HQ	Headquarters
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	HTTP over SSL
IDA	Interchange of Data between Administrations
IDE	Integrated Development Environment
IDS	Intrusion Detection System
IEEE	Institute of Electrical and Electronics Engineers
IIS	Internet Information Server
IP	Intellectual Property
IT	Information Technology
J2EE	Java 2 Enterprise Edition
J2ME	Java 2 Micro Edition
J2SE	Java 2 Standard Edition
JDK	Java Development Kit
JIT	Just-in-time
JLS	Java Language Specification
JPL	Jet Propulsion Lab
JVM	Java Virtual Machine
JVMS	Java Virtual Machine Specification
KBSt	Federal Government Co-ordination and Advisory Agency — Germany
KDE	K Desktop Environment
LAN	Local Area Network
LGPL	Lesser General Public License
LPPL	LaTeX Project Public License
MFC	Microsoft Foundation Classes
MIT	Massachusetts Institute of Technology
MPI	Message Passing Interface
MPL	Mozilla Public License
MS	Microsoft
NASA	National Aeronautics and Space Administration
NERA	National Economic Research Associates
NIMA	National Imagery and Mapping Agency
NIST	National Institute of Standards and Technologies
NRC	National Research Council
NSA	National Security Agency
NTA	National Technology Alliance
ODBC	Open Database Connectivity
OGD	Other Government Department
OGSI	Open Grid Services Infrastructure
ORB	Object Request Broker
ORDBMS	Object-Relational Database Management System
OS	Operating System
OSI	Open Source Initiative

OSO	Open Source Observatory
OSPR	Open Source Prototype Research
OSS	Open Source Software
PDA	Personal Digital Assistant
PDF	Portable Document Format
PGP	Pretty Good Privacy
PHP	Hypertext Preprocessor
PIM	Personal Information Manager
PKI	Public Key Infrastructure
PS	PostScript
PWGSC	Public Works Government Services Canada
PDF	Portable Document Format
RBAC	Role-Based Access Control
R&D	Research and Development
RDBMS	Relational Database Management System
RFC	Request for Comments
RPC	Remote Procedure Call
RTF	Rich Text Format
SCO	Santa Cruz Operation
SELinux	Security Enhanced Linux
SNMP	Simple Network Management Protocol
SSH	Secure Shell
SSL	Secure Socket Layer
SQL	Structured Query Language
TCO	Total Cost of Ownership
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
TDP	Technology Demonstration Project
UDP	User Datagram Protocol
UK	United Kingdom
UML	Unified Modelling Language
URL	Uniform Resource Locator
USA	United States of America
VCS	demonstrating Value–building Capacity–mobilizing Support
VM	Virtual Machine
VPN	Virtual Private Network
VRML	Virtual Reality Markup Language
WYSIWYG	What-You-See-Is-What-You-Get
XML	Extensible Markup Language

Glossary of Relevant Terms

(Excerpt from[45])

Commercial software A software program is commercial if it is developed as a business activity. Commercial software can be free or non-free, depending on its license. Likewise, a program developed by a school or an individual can be free or non-free, depending on its license. The two questions, "what sort of entity developed the program?" and "what freedom do its users have?", are independent. 'Commercial' and 'proprietary' are not synonymous — most commercial software is proprietary, but there is commercial free software, and there is also non-commercial non-free software.

Compatibility The term compatibility, in the software context, is closely related to interoperability. A product is compatible with a standard but interoperable with other products that meet the same standard (or achieve interoperability through a broker).

Copylefted software Copyleft (as opposed to 'copyright') is the idea and the specific stipulation when distributing software, that the user will be able to copy it freely, examine and modify the source code, and redistribute the software to others (free or priced) as long as the redistributed software is also passed along with the copyleft stipulation. The term was originated by Richard Stallman and the Free Software Foundation (FSF). Copylefted software is free software whose distribution terms do not allow re-distributors to add any additional restrictions when they redistribute or modify the software. This means that every copy of the software, even if it has been modified, must be free software. Copyleft is a general concept; to actually copyleft a program, you need to use a specific set of distribution terms (see reference below). Reference: The FSF definition of Copyleft: `http://www.gnu.org/copyleft/copyleft.html`

Free Software (FS) Free Software (FS) is software that comes with permission for anyone to use, copy, and distribute, either verbatim or with modifications, either gratis or for a fee. In particular, this means that the source code must be available. "Free software" is a matter of liberty, not price. Within this context, 'free' should be understood as in "free speech", not as in "free beer". Free software deals with the users' freedom to run, copy, distribute, study, change and improve the software. More precisely, it refers to four kinds of freedom for the users of the software:

- Freedom 0 - The freedom to run the program, for any purpose.
- Freedom 1 - The freedom to study how the program works, and adapt it to your needs - access to the source code is a precondition for this.
- Freedom 2 - The freedom to redistribute copies so you can help your neighbour.
- Freedom 3 - The freedom to improve the program, and release your improvements to the public, so that the whole community benefits - access to the source code is a precondition for this.

Reference:

The FSF definition of Free Software: `http://www.gnu.org/philosophy/free-sw.html`

Free Software vs. Open Source There is significant disagreement in the software community about these two (largely synonymous) concepts - to some extent, the Free Software movement and the Open Source movement are like two political camps within the free software community. The official definition of "open source software", as published by the Open Source Initiative, is very close to the definition of "free software" used by the Free Software Foundation, though it is a little 'looser' in some respects. We won't go further into this debate, except to acknowledge it as a contentious issue. More information is available at: `http://www.gnu.org/philosophy/free-software-forfreedom.html`.

Freeware The term 'freeware' has no clear accepted definition, but it is commonly used for software packages that permit redistribution but not modification (and their source code is not available). Freeware is offered at no cost, but it is typically copyrighted so that you can't incorporate its programming into anything you may be developing.

Interoperability Institute of Electrical and Electronics Engineers (IEEE) defines interoperability as the ability of two or more systems or components to exchange information and to use the information that has been exchanged. Interoperability is the ability of a system or a product to work with other systems or products without special effort on the part of the customer. The term is widely used in product marketing descriptions. Products achieve interoperability with other products using either or both of two approaches:

- By adhering to published interface standards
- By making use of a 'broker' of services that can convert one product's interface into another product's interface "on the fly"

A good example of the first approach is the set of standards that have been developed for the World Wide Web. These standards include TCP/IP, HTTP, and HTML. The second kind of interoperability approach is exemplified by the Common Object Request Broker Architecture (CORBA) and its Object Request Broker (ORB).

Liteware Liteware is a term for software that is distributed freely in a version having less capability than the full for-sale version. It is usually designed to provide a potential customer with a sample of the 'look-and-feel' of a product and a subset of its full capability. Liteware can be considered a type of shareware (where shareware also includes products distributed freely, usually on a trial basis, that do not have full capability).

Open Source Software (OSS) In general, Open Source Software (OSS) refers to any program whose source code is made available for use or modification as users or other developers see fit. Open source software is usually developed as a public collaboration and made freely available. In a stricter sense, OSS refers to software that complies with the "Open Source Definition".

Open Standards Open Standards are characterized by the fact that the specifications on which they are based are owned by a vendor-neutral organization rather than by the original developers. Anyone is free to build software according to the specifications without infringement of intellectual property rights, though typically there are several freely available implementations (commercial or Open Source). Their real virtue is that they have been adopted by the industry and are "future proof". An open standard is more than just a specification. The principles behind the standard, and the practice of offering and operating the standard, are what make the standard 'open':

Availability Open standards are available for all to read and implement.

Maximize End-User Choice Open standards create a fair, competitive market for implementations of the standard. They do not lock the customer into a particular vendor or group.

No Royalty Open standards are free for all to implement, with no royalty or fee. Certification of compliance by the standards organization may involve a fee.

No Discrimination Open standards and the organizations that administer them do not favour one implementer over another for any reason other than the technical standards compliance of a vendor's implementation. Certification organizations must provide a path for low and zero-cost implementations to be validated, but may also provide enhanced certification services.

Extension or Subset Implementation of open standards may be extended, or offered in subset form. However, certification organizations may decline to certify subset implementations, and may place requirements on extensions (see Predatory Practices).

Predatory Practices Open standards may employ license terms that protect against subversion of the standard by embrace-and-extend tactics. The licenses attached to the standard may require the publication of reference information for extensions, and a license for all others to create, distribute, and sell software that is compatible with the extensions. An open standard may not otherwise prohibit extensions.

An important aim of adhering to open standards is to achieve and promote interoperability.

A second set of open standards is typically created by a consortium of industry leaders (institutions or individuals) that determined that there is a general requirement for a specific standard. It is also important to note the influence of general acceptance of open standards. If a standard is not widely adopted, its development will probably stop and it will end up supporting only a very limited number of proprietary products' interaction.

Reference: Principles and Practice of Open Standards: `http://perens.com/OpenStandards/Definition.html`

OSI The Open Source Definition Open Source is a software certification mark owned by the Open Source Initiative (OSI). Developers of software that is intended to be freely shared, possibly improved, and redistributed by others can use the Open Source trademark provided that their distribution terms conform to the OSI's Open Source Definition. To summarize, the Definition model of distribution terms require that:

- The software being distributed must be redistributed to anyone else without any restriction
- The source code must be made available (so that the receiving party will be able to improve or modify it)
- The license can require improved versions of the software to carry a different name or version from the original software

Reference: The OSI definition of Open Source: `http://www.opensource.org/docs/definition_plain.php`

Postcardware Postcardware is freeware (no-charge software that is freely shared) that requires only that the user send the software provider a postcard as a form of payment. The idea is to humanize the transaction, to remind the user that someone else shared something freely, and to remind the provider that someone is actually using the creation.

Proprietary Software Proprietary software describes software that is owned exclusively by a single company that carefully guards knowledge about the technology used and the software's inner workings. Some proprietary products can only function properly, if at all, when used with other products owned by the same company. Proprietary software is software that is not free or semi-free. Its use, redistribution or modification is prohibited, or is restricted so much that you effectively cannot do it freely.

Public domain software Programs that are uncopyrighted, because their authors intended to share them with everyone else, are in the public domain. The Unix community has developed a number of such programs over the years. Programs in the public domain can be used without restriction as components of other programs. The simplest way to make a program free is to put it in the public domain, uncopyrighted. This allows people to share the program and their improvements, if they are so minded. However, it also allows people to convert the program into proprietary software. They can make changes, many or few, and distribute the result as a proprietary product, thus removing the freedom that the original author provided. Public domain software is software that is not copyrighted. If the source code is in the public domain, it is a special case of non-copylefted free software, which means that some copies or modified versions may not be free at all. In some cases, an executable program can be in the public domain but the source code is not available. This is not free software, because free software requires accessibility of source code.

Semi-free software Semi-free software is software that is not free, but comes with permission for individuals to use, copy, modify and distribute (including distribution of modified versions) for non-profit purposes. Pretty Good Privacy (PGP) is an example of a semi-free program.

Shareware Shareware is software that is distributed free on a trial basis with the understanding that the user may need or want to pay for it later. Some software developers offer a shareware version of their program with a built-in expiration date (e.g. after 30 days, the user can no longer get access to the program). Other shareware (sometimes called liteware) is offered with certain capabilities disabled as an enticement to buy the complete version of the program. Shareware comes with permission for people to redistribute copies, but no-one who continues to use a copy is required to pay a license fee. Shareware is not free, or even semi-free software, for two reasons:

- For most shareware, source code is not available; thus, you cannot modify the program in any way.
- Shareware does not come with permission to make a copy and install it without paying a license fee, not even for individuals engaging in non-profit activities. (In practice, people often disregard the distribution terms and do this anyway, but the terms do not permit it).