Advances in Technology Project Management: Review of Open Source Software Integration

Maurice Dawson
Brian Leonard, Alabama A&M University
Emad Rahim, Oklahoma State University

Available at: http://works.bepress.com/maurice_dawson/25/
Chapter 16

Advances in Technology
Project Management:
Review of Open Source Software Integration

Maurice Dawson
University of Missouri – St. Louis, USA

Brian Leonard
Alabama A&M University, USA

Emad Rahim
Oklahoma State University, USA

ABSTRACT

As organizations must continually drive down costs of software-driven projects, they need to evaluate the Systems Development Life Cycle (SDLC) and other software-based design methodologies. These methodologies include looking at software-based alternatives that could save a significant amount of money by reducing the amount of proprietary software. This chapter explores the use and integration of Open Source Software (OSS) in software-driven projects to include in enterprise organizations. Additionally, the legalities of the GNU General Public License (GPL), Lesser General Public License (LGPL), Berkeley Software Distribution (BSD), and Creative Commons are explored with the integration of these OSS solutions into organizations. Lastly, the chapter covers the software assurance and cyber security controls to associate with OSS to deploy a hardened product that meets the needs of today’s dynamically evolving global business enterprise.

DOI: 10.4018/978-1-4666-6473-9.ch016
APPRAOCH

The authors reviewed multiple Linux distributions and their uses. Reviewed in depth were the copyrights and open sourcing legal implications.

IMPRESSION

As indicated through legal case reviews, there are some very valuable benefits to open source software, in that it allows for collaboration in the development of new software and technology that can undoubtedly spur innovation and improve many processes and functions that individuals and businesses in our society rely on every day. Consequently one purpose of the GNU GPL is to protect and preserve individual rights and the creativity of others while at the same time providing a benefit and contributing to society at large. OSS must be considered in the development process as it is essential in overall license cost reduction with the ability to reuse already constructed software.

PROJECT MANAGEMENT

There are numerous perspectives regarding the concept of project management as this is a field with many employment opportunities in various industries such as defense or aerospace (Dawson & Rahim, 2011). Thus, the definitions generated by these perspectives also vary, according to the context in which it is discussed. However, the purpose of most project management activities is generally similar. Project management is a way of managing and organizing corporate resources so the available resources can generate the completion of a project within given scope, time, and resource constraints (Wideman, 2001).

The understanding behind project management also accounts for the definition of a project. A project is a unique endeavor performed to create certain products, services, or results (Project Management Institute, 2009). This definition is dissimilar to the definitions of process and operation due to several factors. The easiest to define is the time-constraint factor. A project performs the work necessary to complete activities within a limited amount of time, while processes and operations generally account for ongoing continuous effort. A project aims to produce a single or a group of products, services, or results and the chain of activities are terminated once these are produced. Thus it is important to understand the acquisition of Information Technology (IT) and Information Systems (IS) in project management (Rahim & Dawson, 2010).

Software Design Methodologies

The SDLC is a process for planning, creating, testing and deploying ISs (Avison & Fitzgerald, 2003). Requirements are an impact factor as they feed the development and serve as an important prerequisite to development. The SDLC is a modified waterfall method as when objectives are not met then the process is to move backward but the goal is to continually move forward into the next process steps such as system deployment. Another design methodology is agile software development. Agile is based on iterative and incremental development, in which requirements and solution evolve through collaborating teams (Cockburn, 2002). In agile it is essential to understand the people factor to ensure success (Cockburn & Highsmith, 2001). A modified agile methodology is Scrum which is an interactive and incremental software development framework (Rising & Janoff, 2000). All methodologies described allow for code reuse and the integration of OSS. As design methodologies continue to grow so does the need for quicker development. To do this effectively one would need to consider using the option of code reuse.
Details of Linux

The definition, terms, and understanding of open-sourcing have been synonymous with the World Wide Linux is an Unix like OS that is built on the Linux kernel developed by Linus Torvalds with thousands of software engineers. As of 2012 there are over two hundred active Linux distributions. The majority of the kernel and associated packages are free and OSS. This type of software provides licenses which allows users the right to use, copy, study, change, and improve the software as the source code is made available. Providing source code allows an organization’s developers or engineers to understand the inner workings of development. Imagine being able to study Mac or Windows by viewing all the source code to replicate similar developments. This exercise would be great for a new developer to learn low level coding techniques, design, integration, and implementation. Students and faculty could actively participate in design groups in which they would contribute code or design guidance for the upcoming software releases. However some distributions require a cost for updates or assistance that related to specific needs such as OS modifications for server hosting. In software, there is a packet management system that automates the process of installing, configuring, upgrading, and removing software packages from an OS. In the Linux OS builds the most common packet management systems are Debian, Red Hat Package Manager (RPM), Knoppix, and netpkg.

Since Linux does not have redistribution limits it can be used to replace proprietary OSs in computer labs to save costs. The cost that would be associated with the proprietary labs can be redirected towards additional hardware instead. With the many variations of Linux one can find the appropriate distribution for their targeted use. Table 1 displays the different distributions to include the potential uses.

Fedora is an OS based on the Red Hat Package Manager (.rpm) (Proffitt, 2010). Fedora has a side development project known as Fedora Spins which contains multiple spin off versions of the Fedora OS. These spins allow academics, researchers, and students the ability to perform tasks such as cyber security, forensics, electronics design, and more (Petersen, 2013). Two of the spins are lightweight distributions which are key to reviving older systems. Kitten Lightweight Kernel (LWK) and other similar kernels allow individuals the ability to practice development on lightweight OSs (Brightwell, Riesen, Underwood, Hudson, Bridges, & Zaharia, 2003). The possibilities are endless for encouraging low level development, integration, and increasing overall lifecycle expertise.

Why Use Open Source

Using OSS such as Linux allows for a significant reduction in the cost of proprietary licensing. Additionally, when coupled with virtualization OS capabilities can be replicated in a virtualized layer (Dawson & Al Saeed, 2012). Much software today is too complex to be developed from scratch thus reuse adds competitiveness allowing for immediate code reuse (German, & González-Barahona, 2009). In addition, OSS provides the ability to allow developers to perform static code analysis on source code as it is readily available (Louridas, 2006). As over half of the vulnerabilities are found in the application using OSS could provide beneficial to all that are stakeholders in the SDLC (Paul, 2011). Additionally, benefits include the ability to capture all known and unknown risks that allow the use of sound software engineering practices (McGraw, 1999).

Argument against the Use of Open Source

One of the most well-known arguments against the use of OSS is that individuals with malicious intent can find flows within the code and exploit them (Carrier, 2002). The issue with this particular
Table 1. Linux distributions and uses

<table>
<thead>
<tr>
<th>Linux Distributions</th>
<th>Description and Potential Use</th>
<th>Packet Management System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ubuntu</td>
<td>One of the most popular Linux OS developed to be a complete OS that can be an easily replacement for other comparable OSs.</td>
<td>Debian-based</td>
</tr>
<tr>
<td>Edubuntu</td>
<td>OS targeted for grades k-12. Contained in OS are tons of software applications that is useful to those who are education majors.</td>
<td>Debian-based</td>
</tr>
<tr>
<td>Damn Small Linux</td>
<td>This OS is designed to as a small OS to be utilized on older hardware. This OS is great for institutions that have old computers and want to revitalize them for use. OS is also great for VMs as DSL requires a low amount of memory.</td>
<td>Knoppix-based</td>
</tr>
<tr>
<td>BackTrack</td>
<td>OS based on Ubuntu for digital forensics and penetration testing. Great tool for students majoring in technology fields. As cyber security is becoming a hot topic around the world this tool provides students the ability to learn from over thirty software applications that aid in penetration testing and more.</td>
<td>Debian-based</td>
</tr>
<tr>
<td>Kali Linux</td>
<td>OS based BackTrack that is a continuation of the popular penetration testing distribution.</td>
<td>Debian-based</td>
</tr>
<tr>
<td>Red Hat Enterprise Linux</td>
<td>This OS serves as the standard for many enterprise data centers. OS was developed by Red Hat and targeted for commercial use. Red Hat has a policy against making nonfree software available for the system through supplementary distribution channels. This is different and why this OS is listed as an exception in terms of OSS.</td>
<td>RPM-based</td>
</tr>
<tr>
<td>Fedora</td>
<td>This OS is supported by the Fedora Project and sponsored by Red Hat. This OS provides a great resource for learning Red Hat Enterprise Language (RHEL). As there are thousands of jobs requiring expertise specifically with Red Hat this OS is a great tool to prepare students for employment in IT. Fedora has over six Fedora Spins such as Design-suite, Scientific-KDE, Robotics, Electronic-lab, Games, and more.</td>
<td>RPM-based</td>
</tr>
<tr>
<td>CentOS</td>
<td>This OS derived entirely from RHEL. The source code is developed from Red Hat which allows a student to learn RHEL with a small number of differences. CentOS can be used for teaching IT students on how to setup, administer, and secure a server.</td>
<td>RPM-based</td>
</tr>
<tr>
<td>SUSE Linux</td>
<td>OS is of German origin with the majority of its development in Europe. Novell purchased the SUSE brand and trademarks.</td>
<td>Debian-based</td>
</tr>
<tr>
<td>Xubuntu</td>
<td>Xubuntu is based upon Ubuntu however it uses the light weight Xfce desktop environment.</td>
<td>Debian-based</td>
</tr>
<tr>
<td>Ubuntu Studio</td>
<td>This OS is derived from Ubuntu. This OS is developed specifically for multimedia production such as audio, video, and graphics. Departments for multimedia could use this OS for multimedia instruction and the development of projects. As many of the tools for multimedia production are expensive these alleviates large license costs for institutions.</td>
<td>Debian-based</td>
</tr>
<tr>
<td>Lubuntu</td>
<td>OS is based on Ubuntu and uses the LXDE desktop environment. It replaces Ubuntu's Unity shell and GNOME desktop.</td>
<td>Debian-based</td>
</tr>
<tr>
<td>Chromium OS</td>
<td>An open source light weight OS that is targeted for netbooks and mobile devices.</td>
<td>Portage-based</td>
</tr>
</tbody>
</table>

The argument is that proprietary software packages or closed OSs are being exploited such as Windows. More importantly more research needs to be understood on the accessibility and human use of the different operating environments (González, Mariscal, Martínez, & Ruiz, 2007). The misconceptions surround ease of use in which individuals feel they need to have a mastery of command line
among other system administrator like abilities to navigate the desktop based OSs. Thus more attention needs to be provided to system usability for industry and home use on the modern open source Linux OS such as Ubuntu 14.04 Long Term Support (LTS) and others (Brooke, 1996).

**COPYRIGHTS AND OPEN-SOURCING: ECONOMIC AND LEGAL IMPLICATIONS**

**Basis of Legal Protection**

**Copyright Law**

The U.S. Constitution provides that, “The Congress shall have Power… [t]o promote the Progress of Science and useful Arts, by securing for limited Times to Authors and Inventors the exclusive Right to their respective Writings and Discoveries…” A copyright gives the author, provided certain legal requirements are met, exclusive rights to distribute, sell, license, produce, and publish the copyrighted material (Cheeseman 2013). Moreover, the Copyright Revision Act of 1976, establishes the legal requirements for copyright protection and provides for protection for copyright infringement (Cheeseman 2013). In addition, in 1989, the U.S. and several other countries signed the Berne Convention, which is an international copyright treaty.

Establishing copyright protection is only half the battle. In order to prove copyright infringement, one must show that a party has copied a substantial and material part of the owner’s copyrighted work without permission (Cheeseman 2013). If successfully proven, an owner of a copyrighted work may recover profits from the infringement, damages suffered by the owner, and even a court order requiring destruction of the infringing material and/or an order preventing such infringement in the future (Cheeseman 2013).

**License Agreement**

In its most basic sense, a licensing agreement is a detailed agreement which indicates the terms between a licensor, the owner and/or creator of intellectual property, and a licensee, the party who has granted limited rights in or access to the intellectual property (Cheeseman 2013). A licensing agreement may be a contract, but all licensing agreements may not meet the requirements of a contract. (Stein 2006). The extent to which a licensing agreement meets the legal requirements of a contract may provide greater or lesser legal protection of the intellectual property (Stein 2006).

**Traditional Contract Law**

Basic contract law requires that in order for an agreement to be legally enforceable, it must contain at minimum, an offer, an acceptance, and must be supported by consideration. An offer is generally a manifestation of intent to be bound, and an acceptance requires an unequivocal assent to the terms of the offer. Consideration is generally defined as bargained for exchange, where each side of the agreement receives some legal value. Generally courts will not inquire into the adequacy of consideration, or in other words the sufficiency of consideration (Cheeseman 2013). If an agreement is a legally enforceable contract, the law provides for several remedies in the event that a party to the agreement does not fully perform the agreement or violates its terms in some way.

**Unique Challenges of Open Source Software**

**Copyleft and Free of Charge**

One of the major challenges of “open source software,” is the fact that the author or creator of the original source code, makes their source code available to other users to distribute and modify,
free of charge, and in many cases requires that any modification of that source code must also remain free and available to other users down the line, or the concept of copyleft work (Stallman, 2013). The problem for traditional copyright law is that it has primarily been focused on protecting and restricting the use and distribution of copyrighted work rather than the free and open distribution thereof. Furthermore, the fact that the software is made available free of charge creates some problems when it comes to determining whether and to what extent the author or creator has suffered any damages, and is therefore entitled to any compensation as is usually the case in the traditional copyright infringement lawsuit.

**Enforceability**

Furthermore, given the usual form of the open source licenses utilized by creators and authors of open source software, there is some question as to the availability of proof that the licensee of open source software is even aware or ever assents to the terms of the license. (Stein 2006). For example, if a licensee of open source software is not aware that there is a license or what the terms of the license provide, it may be difficult, in the event that the user modifies the licensed source code and then attempts to restrict its availability to others, for the author or licensor to enforce the license against the licensee. This would not be the case for example, if the licensee was required to download the source code and in doing so had to accept the terms and conditions of the license (Stein 2006).

**Lack of Consideration**

Another challenge for licensors of open source software is whether and to what extent the license is actually a contract or merely a bare license (Stein 2006; Mandrusiak 2010). If an open source license was considered to be more than just a bare license, but a contract, the author or creator may enjoy greater protection. (Stein 2006; Mandrusiak 2010). The problem however is that in order to be considered more than just a bare license, but a contract, the license would have to meet the traditional contract law requirements, which includes consideration. In order for an open source license to be enforceable as a contract, the author or creator must show that it is supported by consideration. Thus the author or creator or licensor must be able to show some legal value provided and received for the use of the license and source code. Since many of these open source code license are made available free of charge, it may be more difficult for a licensor to make such a showing. There are other arguments that a licensor may have such as that the promise to abide by the terms of the license could be sufficient consideration, but it may be unclear whether courts would agree with this rationale. If not, then the open software license would not be considered a legally enforceable contract and therefore, would not receive the traditional contract law protections or remedies for breach.

**Illustrative Cases**

There have been a few cases to consider the legal protections afforded to some open source software code materials. For example, in *Computer Associates International v. Quest Software, Inc., et al.*, the court recognized the validity of the GPL involved in that case, and specifically found that any user of the GNU GPL was bound by its terms. Furthermore, the court noted that no copyright protection could be afforded to the modified version of the source code provided based on the terms of the GNU GPL. However, the Court further found that where the GNU GPL provided an exception for the commercial use of the output of that program, the GNU GPL would not be violated and copyright protection may exist for that output.
In addition in Progress Software Corporation, *et al.*, v. Mysqlab, *et al.*, the Court recognized and considered, but did not rule upon at that stage of the case, a GNUGPL that was at issue in that case.\footnote{4} Furthermore, in *Planetary Motion, Inc.*, v. *Techplosion, Inc.*, Michael Gay A.K.A. Michael Carson, the Court recognized and reiterated that the GNU GPL utilized in that case, “…allows users to copy, distribute and/or modify the Software under certain restrictions, e.g., users modifying licensed files must carry “prominent notices” stating that the user changed the files and the date of any change.”\footnote{5}

Lastly, but certainly not least, probably the most notable case which dealt with legal protection, specifically copyright protection and open source software, is the *Jacobsen, v. Katzer, et. al.*\footnote{6} In *Jacobsen*, the U.S. Court of Appeals for the Federal Circuit, considered whether and to what extent a copyright holder, could use copyright laws to enforce an open source license, with respect to software that that had been made free and available to the public. The District Court held that while the defendant’s actions may have been in breach of the nonexclusive Artistic License granted to them, they did not rise to the level of copyright infringement, and thus did not allow the copyright holder to use copyright laws to enforce the open source license.

The U.S. Court of Appeals for the Federal Circuit, on the other hand not only recognized the existence of the Artistic License, but held that despite its nonexclusivity, the Artistic License, did prevent certain other actions from being taken with regard to the source code, specifically use of the information without compliance with the Artistic License, such as indicating the source of the material, and including appropriate notices with any subsequent distribution of the material.\footnote{7} The Court specifically held, that “Copyright holders who engage in open source licensing have the right to control the modification and distribution of copyrighted material.”\footnote{8} The court also held that the mere fact that open source licenses like the Artistic License at issue in the case are free of charge, does not render them devoid of economic value, and does not entitle them to any less protection than other forms of copyrighted material. The Court stated, “[t]he choice to exact consideration in the form of compliance with the open source requirements of disclosure and explanation of changes, rather than as a dollar-denominated fee, is entitled to no less legal recognition.”\footnote{9}

Thus, the *Jacobsen* case appears to have provided at least one example of where a Court has provided copyright and it appears probably contract protection open source software utilizing open source licenses, in a way that some had believed was not possible given their unique nature. However it should be noted, that it is not clear whether other federal circuits will follow suit, and/or if the U.S. Supreme Court will ultimately agree with the *Jacobsen* Court in its analysis of this issue. As with many issues in the law, we will have to wait and see.

**UCITA**

In addition to the cases previously discussed, the Uniform Computer Information Transactions Act (UCITA) may also provide some legal protection to open source software and code, in the states where it has been enacted, and except where federal law controls, such as in the area of copyright law. The “…UCITA is a model act that establishes a uniform and comprehensive set of rules governing the creation, performance and enforcement of computer information transactions.” (Cheeseman 2013).

**Review of the Specific Licenses**

**GNU GPL v3**

After a review of the terms and conditions provided by this license it appears to be more comprehen-
sive in its requirements for use of the licensed software. It contains several more terms and appears to contain many more prohibitions that the previous version of the license terms contained. It contains the requirement to include appropriate notices for distribution of the code. It also contains specific prohibitions regarding restriction on the subsequent use of the code, including modified versions, by downstream users (Kumar, 2006).

**GNU GPL v2**

After review of the terms and conditions of this license, this version’s license does not appear to have as many requirements and certainly is not as long as the newest version of this software’s license appears to be. While considerably shorter than the subsequent version’s license, this license does still maintain and include the requirement that appropriate notices accompany the distribution of the code (Kumar, 2006).

**LGPLv3**

After review of the terms and conditions of this license, this version’s license does not appear to have as many requirements as either of the licenses under the GNUGPLv3 or v2, but it does maintain several requirements for compliance. Of note, is this license includes an exception to the GNUGPL license, namely that the work produced under this license may be reproduced without compliance with Section 3 of the GNUGPL, which relates to Protecting Users’ Legal Rights from Anti-Circumvention Law.

**LGPL v2**

After review of the terms and conditions of this license, this version’s license appears to somewhat longer than the terms and conditions of the subsequent version’s license, but it appears to be closer to the GNUGPLv2’s license terms than the LGPLv3’s terms and conditions, and noticeably does not include the exception to the GNUGPL license as is contained in the subsequent version of this license.

**LLGPL**

After review of the Lisp Lesser General Public License (LLGPL), this version’s license is like the LGPL but with a prequel. This prequel defines the effect in terms more typically used in Lisp programs. This license is grounded in the C programming language as the license specifically calls out functions not present in other languages that are not traditionally compiled (Greenbaum, 2013).

**Creative Commons**

After review of the terms and conditions of this license, it appears that this license is very similar to that of Modified BSD. It is interesting of note that the license begins by indicating that the company is not a law firm. Additionally, this license appears to include a waiver of copyrights and related rights, and a fall-back in the event that the waiver is invalidated, which appears to be based upon the purpose of promoting the overall ideal of free culture. In addition this license includes a limitation to make sure that neither patent or trademark rights are being waived by this license.

**Artistic License 2.0**

After review of the terms and conditions of this license, this license appears to be very similar to that at issue in the *Jacobsen* case discussed above. Moreover, it appears that this license makes clear that the copyright holder intends to retain some creative control over the copyrighted work overall,
while still trying to ensure that the copyrighted material remains as open and available to others as possible under the circumstances.

**Modified BSD**

After review of the terms and conditions of this license, these terms and conditions appear to be the shortest list of terms and conditions of all of the licenses reviewed in this paper. Additionally this license appears to allow reproduction and modification of the copyrighted material provided certain conditions are met, which if subject to legal challenge, a court might construe as being subject to only protection as a contract, at best, and a bare license at worst. Moreover, based upon the legal authorities cited in this paper, it may be unclear whether this license may provide sufficient copyright protection.

**Clear BSD License**

After review of the terms and conditions of this license, this license appears to be very similar to the Modified BSD License, in that it is very short, and appears to allow reproduction only if certain conditions are met. This license does make clear that no patent rights are granted by this license.

**CYBER SECURITY AND SOFTWARE ASSURANCE**

As malicious intent is an issue with OSS it is important to deploy software security in the development lifecycle to ensure proper security posture (McGraw, 2004). To do this effectively while minimizing the effort for developing controls, organizations can adopt government cyber security controls from the National Institute of Standards and Technology (NIST) Special Publications (SP) 900 Series to include the Department of Defense (DoD) (Dawson Jr, Crespo, & Brewster, 2013). On April 26, 2010, the DoD released the third version of the Application Security and Development Security Technical Implementation Guide (STIG) provided by the Defense Information Systems Agency (DISA). This STIG can be used as a baseline for software configuration and development. DISA provides STIGs for other system components that can allow for full system hardening that will provide the OSS additional security through defense in depth. This process allows for Availability, Integrity, and Confidentiality (AIC) of the entire system.

In the event of a vulnerability finding within the OSS, the software code may require redesign and implementation. This iterative cycle is costly in time and resources. To truly understand security threats to a system, security must be addressed beginning with the initiation phase of the development process. For an organization this means they must allow the Information Assurance (IA) controls and requirements to drive design and influence the software requirements. Therefore, any identified security threats found during the requirements and analysis phase will drive design requirements and implementation. Security defects discovered can then be addressed at a component level before implementation. The cost of discovery and mitigation can then be absorbed within the review, analysis and quality check performed during the design, and implementation of our SDLC. The resultant product is one with security built in rather than security retrofitted. Figure 1 displays the Secure-SDLC (S-SDLC) process in which OSS can be implemented into the development process. For Agile or Scrum this process must be modified to be aligned with that specific design process.

**CONCLUSION**

As indicated in the Jacobsen case, there are some very valuable benefits to open source software,
in that it allows for collaboration in the development of new software and technology which can undoubtedly spur innovation and improve many processes and functions that individuals and businesses in our society rely on every day. Consequently one purpose of the law is to protect and preserve individual rights and the creativity of others while at the same time providing a benefit and contributing to society at large. How courts will interpret and protect and/or enforce open source licenses will depend greatly on how well the case can be made that this form of software and use can be beneficial and still comports with the overall interests that copyright law was intended to accomplish in the first place. As with any new development, the law will have to endeavor to ensure that it strikes a delicate balance between the good of the many and the good of the few or the individual. The use of OSS proves to be a positive and viable option with the addition of appropriate cyber security controls to mitigate risks of use in projects.

REFERENCES


Cockburn, A., & Highsmith, J. (2001). Agile software development, the people factor. Computer, 34(11), 131–133. doi:10.1109/2.963450


German, D. M., & González-Barahona, J. M. (2009). An empirical study of the reuse of software licensed under the GNU General Public License. In Open Source Ecosystems: Diverse Communities Interacting (pp. 185-198). Springer. doi:10.1007/978-3-642-02032-2_17


**KEY TERMS AND DEFINITIONS**

**Authentication:** Security measure designed to establish the validity of a transmission, message, or originator, or a means of verifying an individual’s authorization to receive specific categories of information.

**Availability:** Timely, reliable access to data and information services for authorized users.

**Confidentiality:** Assurance that information is not disclosed to unauthorized individuals, processes, or devices.

**GNU Public License:** A widely used free software license that is managed under the GNU Not Linux Project (Stallman, 1991).

**Integrity:** Quality of an IS reflecting the logical correctness and reliability of the operating system; the logical completeness of the hardware and software implementing the protection mechanisms; and the consistency of the data structures and occurrence of the stored data. Note that, in a formal security mode, integrity is interpreted more narrowly to mean protection against unauthorized modification or destruction of information.

**Linux:** An open source version of the UNIX OS (Perens, 2009).

**Open Source Software:** Software that allows the original source code to be freely available which may be freely redistributed or modified (Perens, 2009).

**Project Management:** The discipline of planning, organizing, motivating, and controlling resources to achieve project success (PMI, 2009).

**ENDNOTES**

1. Article I, Section 8, U.S. Constitution.
3. See Id.
5. 261 F.3d 1188, 1191 (11th Cir. 2001).
6. 535 F.3d 1373 (Fed. Cir. 2008).
7. 535 F.3d 1373, 1382 (Fed. Cir. 2008).
8. See Id.
9. See Id.