Open Source Software in Computer Science and IT Higher Education: A Case Study

Dan R. Lipşa
Visual and Interactive Computing Group
Department of Computer Science, Swansea Univ.
Swansea, UK
d.lipsa@swansea.ac.uk

Robert S. Laramee
Visual and Interactive Computing Group
Department of Computer Science, Swansea Univ.
Swansea, UK
r.s.laramee@swansea.ac.uk

***Abstract*— The importance and popularity of open source software has increased rapidly over the last 20 years. This is due to a variety of advantages open source software has to offer and also the wide availability of the Internet in the early nineties. We identify and describe important open source software characteristics and then present a case study using open source software to teach three Computer Science and IT courses for one academic year. We compare fulfilling our educational requirements and goals with open source software and with proprietary software. We present some of the advantages of using Open Source Software (OSS). Finally we report on our experiences of using open source software in the classroom and describe the benefits and drawbacks of using this type of software over common proprietary software from both a financial and educational point of view.**

Keywords-open source software (OSS), free software

I. INTRODUCTION

Open source software (OSS) has become widely used in IT departments, with large software vendors making a significant amount of revenue from activities that use OSS \cite{idc-oss}. The emergence of the Internet in the early nineties has enabled collaboration between programmers at different locations in the world and easy distribution of software. That together with distinct advantages OSS offers has resulted in an increasing popularity of this type of software.

We briefly introduce of open source software, and describe its main proponents. We describe the main OSS licenses and explain how some licenses protect users' freedom and the ability to use OSS in the future. We describe the impact open source software has on the computer industry. We believe this knowledge is important for fully appreciating the value offered by open source software.

We present a case study in using open source software in teaching three Computer Science and IT classes for one academic year. We compare satisfying our educational requirements with open source software and with proprietary programs. We describe open source software used for infrastructure, user applications and development applications and compare it with proprietary software that achieves the same goals. We evaluate the two categories of software for cost, student appeal and ease of use and we conclude with the main reasons we believe open source software should be more broadly integrated in Computer Science and IT education.

We believe our study presents a balanced comparison between open source and commercial products relevant to an educational environment. We contribute to a better awareness of the relative benefits and drawbacks of open source software versus commercial software and we help educators make informed decisions regarding the software used in their classrooms and in the infrastructure that supports classroom activities.

II. OPEN SOURCE SOFTWARE BACKGROUND

Open source software has a rich history with great achievements, spectacular falls, powerful players and colorful stories. We believe knowledge of the history of open source software is useful to understanding the software business and the computer industry as a whole. We present a brief introduction to open source software describe its achievements and introduce its main proponents. We describe common open source licenses and present the impact open source software has on the computer industry.

A. History of Open Source Software

When discussing open source software, two prominent figures stand out as what we call the creator and the enabler of today's events in this area.

Richard Stallman can be rightfully considered the father of Open Source Software (or Free Software as he calls it). He is the founder of the Free Software Foundation (FSF) a tax-exempt charity that raises funds for work on the GNU (Gnu's Not Unix) Project [4]. The GNU project started in 1983 with an email to a Unix newsgroup, in which Richard Stallman, states that he is going to write a complete Unix-compatible software system and share it with everybody. He asks for contributions of time, money, programs and equipment. With the help of thousands of programmers from around the world, and with the arrival of Linux, an operating system kernel, Richard Stallman succeeded in doing just that. He is the initial developer for many popular Open Source projects such as GNU C Compiler, GNU Emacs, GNU debugger, and GNU Make and FSF

developed Bourne Again Shell (bash) and GNU C library. The name GNU, comes from a recursive acronym for "Gnu's Not Unix", which was designed to show its relationship with the Unix operating system. The GNU operating system is a Unix compatible system but in the same time it was written from scratch so it is different than the proprietary Unix systems.

A more recognizable name than Stallman's is Linus Torvalds and the Linux operating system kernel. By contributing the original version of Linux to the Open Source pool, Torvalds added the last piece missing from completing Stallman's vision: a free Unix like system, and so, he enabled the widespread of the GNU/Linux systems as we see it today. Linux was initially created by Linus Torvalds when he was a student at University of Helsinki in Finland. The first version 0.01 was released in September 1991, and version 1.0 was release in March 1994 [9].

Open Source Software is a term promoted by the Open Source Initiative (OSI) [20], a non-profit organization launched in 1998. In their own words, the movement, is a marketing campaign to turn around the negative image Free Software had outside the hacker community. They argue for Free Software on pragmatic grounds of reliability, cost and business risks. They claim that development of Open Source Software happens at an astonishing pace compared with the conventional software [22]. This is a consequence of the fact that source code is available to everyone and can be changed, so anyone can find and fix bugs, and add their own improvements to the source code. This rapid evolutionary process produces better software than the traditional closed model. For this reason, and because of the lower development costs it makes business sense to choose open source software, and contribute to its development.

The official definition of Open Source Software is very close to how FSF defines Free Software. Still the two movements differ in the reason they argue why people should adopt Open Source/Free Software. For FSF, the reason is that people want and deserve freedom, as defined in Section II-B1. For OSI the motivation is that software produced in an open source environment is technically superior.

From now on, we will use the term Open Source Software because it appears to be much more popular than Free Software in the general press.

B. Open Source Licenses

This section describes the various license agreements under which open source software is available.

1) Free Software

Richard Stallman sees software as information, and he believes everyone should have the freedom to use it and to learn from it. In particular, for a program to be Free Software, everyone should be able to run it for any purpose, to study how the application works, adapt it to their needs, redistribute copies so that the user may assist others, and improve the program and release the improvements, so that the whole community benefits. A common misconception that FSF tries to clarify, is that Free Software means no money for your work. Free refers to freedom, as in "free speech" not as in "free lunch".

2) Copyleft

Copyleft is the use of a license to protect the rights of free software (as defined in Section II-B1) such that remains free software.

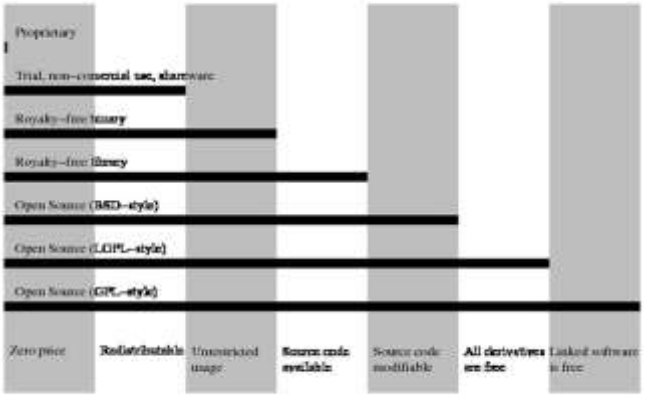

Figure 1: Software Licenses Classification. On the X axis we show possible license features. On the Y axis we show possible types of software.

X Windows is a good example of what happens when free software is not protected by copyleft. X Windows is a windowing system for Unix, developed at MIT, which was released as free software with a permissive license (without copyleft). It was adopted by many software companies, which shipped their improved versions of X Windows without the source code. In those releases, X Windows was no longer Free Software. The users lost the freedom they had for the initial release of X Windows. Copyleft was introduced to prevent this.

Copyleft uses copyright law, but flips it over to serve the opposite of its usual purpose. Instead of keeping the software proprietary, it becomes a mean to keep the software Free Software. Copyleft, gives everyone permission to run, copy, modify a program, and distribute modified versions, -- but not permission to add restrictions of their own. The term copyleft comes from a letter sent to Richard Stallman by Don Hopkins, in which the following phrase appears: "Copyleft – all rights reversed" [23].

3) Software Licenses Classification

An extended classification of software licenses, proprietary and Open Source, adapted from [6], is presented in Figure 1.

Proprietary software does not have any of the seven properties listed at the bottom of Figure 1.

Trial software, non-commercial software and shareware, do not cost anything and they are redistributable but they all have restricted usage. For trial software the time it may be used is

restricted or the features available limited. Non-commercial software cannot be used for any purpose, and shareware has an unenforced limited time usage (for instance WinZip [27]).

A royalty-free binary allows unrestricted usage and a royalty-free library is usually distributed with the source code.

Open Source (BSD-style, where BSD stands for Berkeley Software Distribution) license allows you to modify the source of the program and redistribute the improved version. This is the non-copyleft open source software distributed before the apparition of the Free Software Foundation. Some examples of projects distributed under this kind of license are the X Windows windowing system [29], the FreeBSD operating system [5] and the Apache web server [1].

The software protected by the last two licenses is copyleft-ed, so it is guaranteed to remain Free Software. GPL stands for General Public License and LGPL stands for Library (Lesser) General Public License. Both were created by the Free Software Foundation. The difference between the two licenses is that only a library protected by GPL requires that all programs that link with it, should be GPL programs as well. LGPL protected libraries allow proprietary programs to link with it as well. Most of libraries on GNU/Linux system are protected by LGPL, or less strict licenses, which means that the user may release proprietary programs on GNU/Linux, and link with the libraries available. Many companies have done so (see [25]). Linux, GNU Compiler Collection and Emacs are example of programs protected by GPL.

C. Impact of Open Source Software

International Data Corporation (IDC) forecasts that revenues from open source software will grow at a 22.4% rate to reach $8.1 billion by 2013 [11].

In June 2000, Netcraft Web Server Survey~\cite{netcraft} found that GNU/Linux runs on about 29.9% of the active websites, Microsoft OS runs on about 28.32%, and Solaris is third with 16.33%. Companies like IBM, Oracle and Intel fully support GNU/Linux systems.

Apache is a powerful, full-featured and efficient open source web server. Apache is also the most popular web server on the Internet The July 2009 Netcraft Web Server Survey [17] found that over 66\% of the million busiest sites on the Internet are using Apache, thus making it more widely used than all other web servers combined. Apache Web Server is based on National Center for Supercomputing Applications (NCSA), University of Illinois, Urbana-Champaign public domain HTTP daemon, and the first release was in 1995. It is released under a simple, non-copyleft open source software license. Examples of sites which run on Apache are: Apple (http://www.apple.com), Financial Times (http://www.ft.com), Sony (http://www.sony.com), Palm (http://www.palm.com), Cnet (http://www.cnet.com) and Amazon (http://www.amazon.com).

With success comes competition. The company that has the most to lose from a wide acceptance of GNU/Linux systems is Microsoft. Their monopoly on the Operating System market is threatened. So, they have increased the propaganda against GPL and GNU/Linux.

Windows operating-system chief Jim Allchin has declared in 2001 that Open Source (GPL-style) will result in "the demise of both intellectual property rights and the incentive to spend on research and development" [15]. The Initiative For Software Choice organization [24] was created to fight against governments that mandate use of Open Source in government agencies and against licensing publicly funded projects with GPL.

On the other hand many companies have found that GNU/Linux fits well in their business plans. Linux is certified on all IBM Systems [10]. Oracle is the first commercial database on Linux in 1998 and it invests significant resources in developing, optimizing and testing many open source technologies [10]. Intel works on a wide variety of open source projects to enable a broad range of programs and environments to run best on their hardware [12].

A recent attack on GNU/Linux and GPL is the SCO Group (SCO stands for Santa Cruz Operation) lawsuit accusing IBM of adding copyrighted Unix code into Linux. SCO is asking for 1 billion dollars in damages, and credible speculations surfaced recently that Microsoft is financing SCO through a third party venture capital firm (see [7]).

An interesting use of GPL in promoting a proprietary product is that of QT library by Trolltech [21] which was later acquired by Nokia. QT provides a platform-independent interface to all central computer functionality: GUI, database access, networking, file handling, etc. The library became popular with its use in the KDE desktop, and is included in Suse, a German distribution of Linux which is currently owned by Novell. The Open Source community started a campaign against KDE (because of their use of proprietary QT library) and Red Hat didn't include KDE desktop in their distribution of Linux. In 2000, QT on Unix was release under dual-license GPL and proprietary, ending the quarrel with the Open Source community. By releasing their library under GPL, Trolltech continues to receive the free marketing from the use of the library in KDE. In the same time, they don't lose any business because GPL won't allow a proprietary program to link with QT. This is just one example of successfully combining open source and generating a profit.

III. Open Source in Computer Science and IT Higher Education: A case Study

We present the infrastructure, user and development applications used for one academic year in teaching three classes: Data Structures and Algorithms using Java, Rapid Java Application Development (an advanced Java class) and Design and Analysis of Algorithms. For these classes our goals were to:

- Present information about classes, present assessment methods and post student grades. Teach web applications development (web server recommended)
- Use a database to store students enrolled in classes and grades assigned to students and teach database access from Java (database server needed)
- Use both a desktop and a laptop (a method to synchronize between the two needed)
- Maintain the security of the two computers (a method to encrypt the communication and a firewall is needed)
- Maintain history of changes to course files and web site (source control system needed)

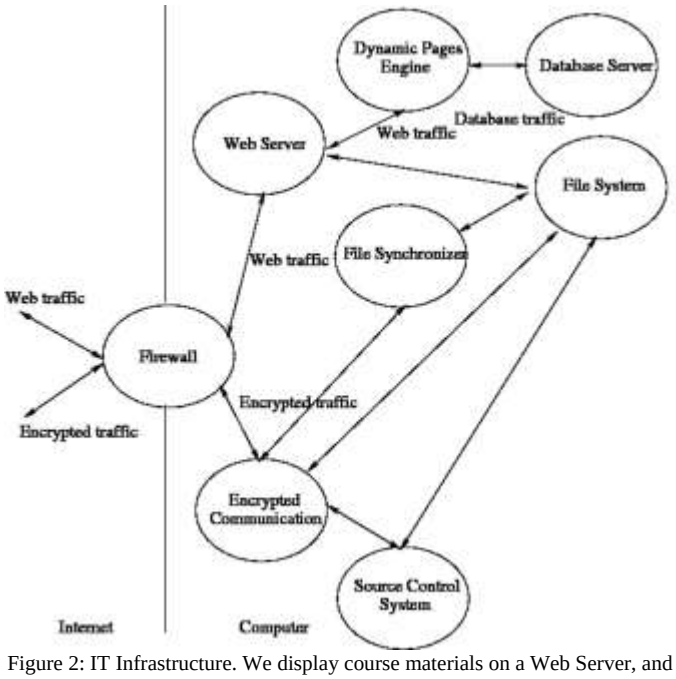

Figure 2: IT Infrastructure. We display course materials on a Web Server, and we use a Dynamic Page Engine and a Database Server to keep track of students grades. We use a File Synchronizer to maintain the same content on our desktop and laptop and we use a Source Control System to keep track of changes made to our classes. Our server is protected by a Firewall and all communication though the Internet is encrypted.

- Browse the Internet (browser needed)
- Read/write email (email client needed)
- Create documents containing mathematical formulas for the Algorithms class (word processor with support for mathematical formulas needed)
- Create presentations for classes (presentation program needed)
- Create diagrams for use in classes (drawing tool needed)
- Use an IDE for writing and debugging Java programs

A. Infrastructure

We used the server infrastructure described in Figure 2 for providing information about the educational institution and the classes taught, for using a database server to store grades for students and display them once the proper credentials were provided, for allowing students to submit homework through the website, and for allowing us to synchronize our work between the laptop and the server. A Firewall is protecting the server allowing only two types of communication: Web traffic for serving the website and Encrypted traffic for remote sessions on the server and file copying and synchronizing. A Web Server provides information about the professor and classes taught (static pages) and information about the grades assigned (dynamic pages). The static pages are read from the File System and the dynamic pages are built by programs run by a Dynamic Pages Engine which uses information stored in the Database Server. The Encrypted Communication server is used to encrypt any communication with the server. We can either synchronize files between the laptop and the server or access the Source Control System.

Figure 3 shows the Open Source implementation of the abstract infrastructure presented in Figure 2. We used the same setup on our laptop and our desktop. An identical setup on both computers enables the mobility of the teacher as they work either on their desktop or on their laptop. Almost all the applications used to perform the desired functions come standard in most GNU/Linux distributions (We used RedHat 9.0). The exception is Unison File Synchronizer a tool built at University of Pennsylvania~\cite{unison}. The GNU Head (mascot of the GNU Project) and the Penguin (the mascot of Linux) show the source of the components used in our setup.

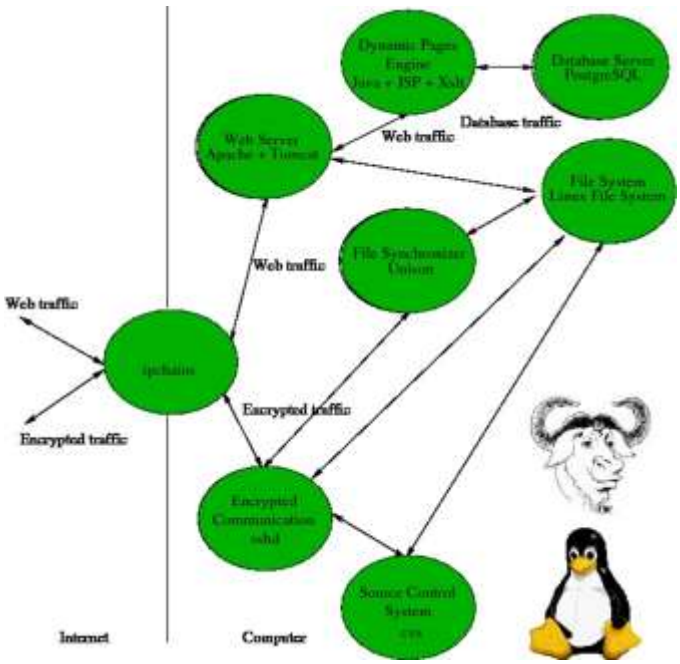

Figure 3: IT Infrastructure using Open Source Components. We use Apache as a web server, PostgreSQL as a database engine, Java, Java Server Pages (JSP) and XSL Transformation (XSLT) to generate dynamic web pages. We use Unison as a file synchronizer and cvs as a source control system. Communication is encrypted using ssh and we use ipchains as a firewall. All these components come standard on a Linux operating system.

We categorize the applications used in two groups: user applications and developer applications.

B. User Applications

- *Web Browser*: We used Mozilla which allows us to browse the Internet and to read newsgroup content.
- *Email*: We used Evolution [3] which allows email, address book and calendar management, and synchronizes with a Palm compatible device.
- *Word Processor*: We used LaTeX [13], a powerful formatting and typesetting processor with strong capabilities for writing mathematical formulas. For writing LaTeX files we used Emacs. Another option is Open Office Writer, which is especially useful for reading or writing Microsoft Word files.
- *Presentation*: We used Prosper [14], a LaTeX based presentation package with features such as: incremental display, overlays, transition effects, predefined slide styles. Another choice is Open Office Impress which is useful for reading Microsoft PowerPoint presentations.
- *Drawing Tool*: We used Xfig [28], a drawing tool for creating vector graphics.

C. Development Applications

The development tools used reflect the programming language taught (Java). However powerful tools exist for C/C++ development and many other languages.

- *Text Editor*: We used Emacs [2], a text editor offering intelligent text editing for Java, HTML, LaTeX, XSLT and many other languages.
- *Integrated Development Environment (IDE)*: We used the NetBeans IDE [16] for editing and debugging Java, JSP, HTML, XSLT and other.

IV. Open Source versus Proprietary Software

This section compares open source software we used in our experiment with an equivalent setup using proprietary software. We did not compare individual features of open source software versus proprietary software. Our benchmark for listing an individual piece of software was to satisfy our educational objective. Both open source and proprietary software satisfied that criterion. We compare the open source versus proprietary software for cost, appeal to students and ease of use.

A. Cost

The Open Source programs used do not cost anything, so we calculate the cost of using common proprietary programs for an equivalent setup. This is presented in Table I. In parenthesis we present a package that contains the listed program. Microsoft uses a licensing model where Client Access Licenses (CALs) - which can be users or devices - are used to regulate access to their server programs. We used server programs with 5 CALs as they serve to make our point and they were the cheapest alternative. Wherever possible we applied an Education discount to the prices. We didn't used a volume discount as we were interested in the price that a student would get if he wants to install the given software on their own machine. We did not use the Express editions for certain pieces of software that Microsoft make available at no cost. While those versions can be used for education, they have reduced functionality that prevents their use in developing a business. In recent years, many companies including Microsoft began offering limited functionality of their products at no cost, we believe as a direct consequence of the strong competition open source products provide.

Using Open Source products may result in increased administration costs, but that cost is difficult to calculate and depends on individual circumstances. The extra administration cost may range from zero if the system administrator is familiar with the particular open source product to being prohibitive if significant training is required. Administration cost is influenced by the maturity of the open source product and also by the number of products that the administrator has to manage. If open source products are used alongside with commercial products the system administrator has more work to do just as a result of the number of software products she administers.

TABLE I. COST OF PROPRIETARY PROGRAMS USED TO ACHIEVE OUR EDUCATIONAL GOALS

Function	Application	Cost (USD)
Operating System	MS Windows XP Professional	300
Web Server	IIS (Windows Server 2003, Std.)	1000
Firewall	(Windows Server 2003, Std.)	0
Encrypted Communic.	(Windows Server 2003, Std.)	0
Database Server	MS SQL Server	1500
Source Control	(Visual Studio .NET 2003 Enterprise)	0
Web Browser	MS Internet Explorer	0
Email Client	Outlook (MS Office 2003)	0
Word Processor	Word (MS Office 2003)	150
Presentation Program	Powerpoint (MS Office 2003)	0
Drawing Tool	MS Visio Standard 2003	200
IDE	Visual Studio .NET 2003 Enterprise	1800
	Total	4950

B. Student Appeal

There are several reasons for which Open Source might appeal more to students than proprietary programs.

First many Open Source projects have their roots in academia. Great examples are X Windows which was an MIT project, the Unison File Synchronizer which was created at University of Pennsylvania, the BSD Unix which was created at University of California at Berkeley and Linux which started at the University of Helsinki.

Second, the availability of source code and documentation for the programs students work with, and the possibility for them to improve those programs could be very beneficial in attracting them to the IT field. In this respect the quote from [8] is revealing: "I'm a poorly skilled UNIX programmer but it was immediately obvious to me how to incrementally extend the DHCP client code (the feeling was exhilarating and addictive)."

Third, the costs detailed in the previous section would affect not only the professor and the school but the student as well. Students like to install the software used in school and work with it on their home computer. They may even want to start a business using the same software. When using open source software no additional costs are required. This is a big advantage for students and for promoting entrepreneurship.

Proprietary software appeals to students because they get direct experience with something they might use at their work place. While this might be beneficial, the computer industry is notorious for fast changes and for many competing products on the same market segment. It is impossible for a school to train students in all competing products, so we believe market share should not be the main criteria for selecting the software product to be used in class.

C. Ease of Use

Individual open source applications are comparable with proprietary applications when trying to achieve common tasks.

However, open source software operating systems are not as user friendly as their commercial software conterparts. This is the case mainly because of lack of hardware drivers from the hardware manufacturers. It is still difficult to use GNU/Linux on a laptop because of lack of wireless drivers and missing support for suspend and hibernate functionality. We see this as the major reason why we have not seen a widespread of open source software in the consumer market.

V. CONCLUSIONS: WHY OPEN SOURCE SOFTWARE?

We present our experience in using entirely open source tools for teaching and research, and we examine at why open source projects might appeal to students and professors.

Some advantages in using open source software in Computer Science and IT education that we have seen from our experience are:

- The cost for the university and the cost for students may be lower.
- Open source projects are advantageous for research as the user can get the source and is free to implement new ideas. They are great for teaching as students have the opportunity to make a difference in a project used by many people.
- Open source software allows the developer to port an application to another operating system. Proprietary software, is usually shipped on specific platforms. In our experience we used open source software on Linux as students used it on Windows. No problems were observed.
- In many cases, an open source project is the de facto standard for that particular type of application. So, the user is working with the best application possible. Some examples are Apache Web Server, Linux Operating System, sendmail Mail Server.
- Open source encourages entrepreneurship, as students can directly use open source tools in order to develop a business idea without the up-front costs of proprietary programs.

The main disadvantage in using open source software is the fact that Linux usability on laptops is seriously affected by the lack hardware drivers especially for wireless, graphic cards and suspend/sleep functionality in laptops.

Student feedback from this experiment was mixed, many students were excited to use Linux and open source tools some students thought that learning a proprietary tool will give them a better chance to get a job. A student who worked in an IT department commented that he is glad that we use and cover some Linux and open source software because he is using it at his job and he had to learn it all by himself. He thought we should cover open source software in other classes as well.

Adopting open source software for your courses is a challenge. Here are a few misconceptions and challenges that have to be overcome:

- Open source software is a niche market used only by a small group of hobbyists. In fact the opposite is true. There is wide adoption in the computer industry for open source software.
- There is no space on the lab computers to install this piece of open-source software. A decisions at the department level to use a certain open source software instead of a proprietary product helps in this case.
- Proprietary software is better and students learn more by using better software. Some open source projects are leaders in their market segment (see Section II-C) and many got great reviews from publications in the field (see [18]). So it can be argued that there is no significant difference in what can be taught using open source software or proprietary software.

We believe that both open source software and proprietary software have an important role to play in the computer industry of the future. While we do not advocate only using open source software for education we believe exposure to open source software is essential for student's success. As future work we plan to develop questionnaires that evaluate specific commercial software products and their open source counter-parts. The evaluation criteria will be how well each product helps in reaching the educational objective of the course.

ACKNOWLEDGMENT

This research was partially funded by the Welsh Institute of Visual Computing (WIVC).

REFERENCES

[1] The Apache Software Foundation. Online document. Accessed Aug. 31, 2009, http://www.apache.org.

[2] GNU Emacs. Online document. Accessed Aug. 31, 2009, http://www.gnu.org/software/emacs/.

[3] Evolution. Online document. Accessed Aug. 31, 2009, http://projects.gnome.org/evolution/.

[4] Free Software Foundation. The GNU Project and the Free Software Foundation. Online document. Accessed Aug. 31, 2009, http://www.gnu.org.

[5] Free BSD. Online document. Accessed Aug. 31, 2009, http://www.freebsd.org/.

[6] Halloween Document I - Open Source Software: A (New?) Development Methodology. Online document. Accessed Aug. 31, 2009, Published Nov. 1, 1998, http://www.catb.org/~esr/halloween/halloween1.html

[7] Halloween X: Follow The Money. Online document. Accessed Aug. 31, 2009, Published Mar. 3, 2004, http://www.catb.org/~esr/halloween/halloween10.html.

[8] Halloween Document II - Linux OS Competitive Analisys: The next Java VM. Online document. Accessed Aug. 31, 2009, Published Nov 3, 1998 http://catb.org/~esr/halloween/halloween2.html.

[9] Ragib Hasan. History of Linux. Online document. Accessed Aug. 31, 2009, Published Jul. 2002, https://netfiles.uiuc.edu/rhasan/linux/.

[10] IBM and Linux. Online document. Accessed Aug. 31, 2009, http://www.ibm.com/linux/.

[11] Open Source Software Market Accelerated by Economy and Increased Acceptance From Enterprise Buyers, IDC Finds. Onlinedocument. Accessed Sep. 03, 2009, Published Jul. 29, 2009, http://www.businesswire.com/portal/site/home/permalink/?ndmViewId=news view&newsId=20090729005107&newsLang=en.

[12] Open Source at Intel. Online document. Accessed Sep. 03, 2009. http://oss.intel.com/.

[13] Latex - A Document Preparation System. Online document. Accessed Aug. 31, 2009, http://www.latex-project.org/.

[14] LaTeX Prosper Class. Online document. Accessed Aug. 31, 2009, http://amath.colorado.edu/documentation/LaTeX/prosper/.

[15] Ralph Nader and James Love. RE: US v. Microsoft proposed final order. Online document. Accessed Sep. 22, 2009, Published Nov. 5, 2001, http://www.nader.org/releases/msfinalorder.html.

[16] NetBeans IDE. Online document. Accessed Aug. 31, 2009, http://www.netbeans.org/.

[17] Netcraft. Netcraft Web Server Survey. Online document. Accessed August 31 2009, http://www.netcraft.com/survey.

[18] OpenOffice.org Product Reviews. Online document. Accessed Sep. 2nd,2009, http://www.openoffice.org/product/reviews.html.

[19] Oracle's Technical Contributions to Linux. Online document. Accessed Aug. 31, 2009, http://www.oracle.com/us/technologies/linux/026042.htm.

[20] Open Source Initiative. Online document. Accessed August 31, 2009, http://www.opensource.org.

[21] Qt Development Frameworks a Nokia unit. Online Document. Accessed Aug. 31, 2009, http://qt.nokia.com.

[22] Eric S. Raymond. The Cathedral and the Bazaar. Online Document. Accessed Aug. 31, 2009, Published August 02, 2002 http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/.

[23] Richard Sallman. The GNU Project. Online document. Accessed Aug. 31, 2009, http://www.gnu.org/gnu/thegnuproject.html

[24] Initiative for Software Choice. Online document. Accessed Aug. 31, 2009, http://softwarechoice.org/.

[25] Richard Stallman. Why you shouldn't use the Lesser GPL for your next library. Online document. Accessed Aug. 31, 2009, http://www.gnu.org/licenses/why-not-lgpl.html.

[26] Unison File Synchronizer. Online document. Accessed Aug. 2009, http://www.cis.upenn.edu/~bcpierce/unison/.

[27] Winzip, A Corel Company. Online document. Accessed Aug. 31, 2009, http://www.winzip.com.

[28] Xfig Drawing Program for the X Windows System. Online document.Accessed Aug. 31, 2009, http://www.xfig.org/.

[29] X.org Foundation. Online document. Accessed August 31, 2009, http://www.x.org.

AUTHORS PROFILE

Dan Lipsa received a bachelor's degree in computer science, from Polytechnic University, Bucharest in 1994. In 1998, he received a master's degree in computer science from University of New Hamphire, Durham. He held positions of technical lead and senior software engineer at various companies in US and Europe and he was an Assistant Professor at Armstrong University, Savannah between 2003-2009. He has been a Research Assistant at Swansea University in the Department of Computer Science since 2010 where he is also working on his PhD. His research interests are in the areas of scientific visualization, computer graphics and software engineering. Lipsa has published six peer-reviewed, scientific papers and a book chapter.

Robert S. Laramee received a bachelor's degree in physics, cum laude, from the University of Massachusetts, Amherst in 1997. In 2000, he received a master's degree in computer science from the University of New Hampshire, Durham. He was awarded a PhD from the Vienna University of Technology, Austria at the Institute of Computer Graphics in 2005. He was a Senior Researcher at the VRVis Research Center and at the same time a software engineer at AVL (www.avl.com) in the department of Advanced Simulation Technologies from 2001-2006. He

has been a Lecturer in Visualization at Swansea University in the Department of Computer Science since 2006. His research interests are in the areas of scientific visualization, computer graphics, and human-computer interaction. Laramee has published over 50 peer-reviewed, scientific papers, including 20 journal papers, in visualization, human-computer interaction, software engineering, and simulation.Dr Laramee has served on over 20 International Programme Committees (IPCs) organized 4 international workshops, and serves as a reviewer for more than 25 journals, conferences, workshops, and institutions.