

ose Coupling through Open Interfaces: from Content to Function

Chao-Kuei Hung

CSIE Department, Shu-Te Univ., Taiwan

ckhung@mail.stu.edu.tw

Tel: +886-7-615-8000 x 4611

Fax: +886-7-615-8999

Abstract

In the field of consumer electronics and computer hardware/software, it is well known that loose coupling of components through open interfaces is technically preferable, encourages fair competition, and protects consumer rights. Among other things, consumers retain their rights to space-shift and time-shift the information and contents they produce, keeping it available even long after the technology that created it grows obsolete and becomes non-available.

In contrast, successful spread of tightly coupled information and communication technologies (ICT) employing proprietary interfaces inevitably results in or aggravates serious social problems such as monopoly, otherwise avoidable digital divide, and illegal software copying. We investigate the mechanism of such spread of proprietary interfaces through consumers' emphasis on features and technical merits over long term sustainability of the content and the system. Examples from other subfields of ICT are given to show the devastating effects of such spread through chain reactions. Eventually the society as a whole has to pay a very high price to escape the proprietary interfaces, and the price would be even higher if it decides to remain chained to the proprietary interfaces.

In the past few years, distance education has also witnessed the problem of monopoly. The rise of open standards such as SCORM aims to decouple contents from the system. It has received positive response from the dominating vendor in this field and has partially solved the problem of long term archiving and reuse of education contents.

What is less discussed in distance education is the decoupling of *functions* within the system, i.e., course management, screen broadcasting, sound broadcasting, online student responses, etc. We investigate the possibility of decoupling the functions of a distance learning system by employing certain less popular, low cost, and open source software pieces that interact through open interfaces. Some pieces such as moodle and gaim have obvious applications in distance education. Other pieces have interesting applications that are unexplored by both the free software community and the distance education community. For example, vnc is not only a tool to display remote desktop, but also can be used to broadcast screens, with a finer granularity and more flexible configuration than many existing commercial distance learning tools. Decoupling of functions also leads to the possibility of executing educational applications remotely. Mature technologies such X and rdesktop are available.

In summary we propose that the distance education community collect existing open source software and glue them to form a loosely coupled system. The aim of creating such a system is not to compete feature by feature with existing commercial systems, but rather to foster a healthy and cooperative environment for research and practice without a software dictator. Yet the challenges mostly come from the legal rather than the technical front. Attempts to discourage the loose coupling paradigm and to prevent competition include patents, DMCA, DRM, and TC. Researchers may not be able to remain neutral in this conflict.

Keywords: interface, coupling, scorm, monopoly, broadcast

Introduction: Loose Coupling and Consumer Rights

In the field of consumer electronics and computer hardware/software, it is well known that loose coupling of components through open interfaces is technically preferable, encourages fair competition, and protects consumer rights. Examples include:

1. Mobile phones of different models by different manufactures can talk to each other through layers of interfaces that are not owned by any particular manufacture.
2. Televisions, CD players, and stereos of various brands and models interact with each other through the simple 3-line AV connectors and well-defined signals whose specifications again are not owned by anyone.
3. In an IBM PC, extension cards of various brands can be attached to motherboards of yet some other various brands through a series of evolving interfaces such as ISA, EISA, PCI, and PCIe. [1]

In each case, there is less threat of monopoly when complicated electronic systems are composed of loosely coupled components that interact with each other through open interfaces not owned by any single vendor. The markets are relatively vibrant and competitive and the prices are reasonable. From a social perspective, the most important consequences involve consumer rights and social justice:

1. Consumers can replace one component with another of *different* brands.
2. Consumers can mix and match components of *different* brands in a working system.
3. Consumers buying components of *different* brands can exchange data or information with each other.
4. Consumers can transfer their own data or information from one component to another, made of *different* brands. This is a form of the *space shifting* property. [2]
5. Consumers can retain their own data or information across a long period of time, keeping it accessible even when the old technology that produced the data or information is long gone. This is a form of the *time shifting* property.

In the consumer electronics market, the loose coupling doctrine is very well accepted and frequently, if technically possible, vendors go as far as providing backward compatibility connectors or converters when a new interface has to be proposed to replace the old interface. With these converters, hardware components that communicate through the old interface may continue to be used in the new environment. The ps2-to-usb converter (for mouses and keyboards) and the SD-to-T-flash card converter are two recent examples.

In summary, it would be inconceivably outrageous if consumers who desire to replace one component (keyboard, network card, the amplifier, etc.) were demanded by the hardware vendor to replace the entire system instead.

Lessons from Other Subfields of ICT

The landscape of the software market, however, is drastically different. Here one readily finds a small number of, but very popular examples of successful spread of tightly coupled information and

communication technologies (ICT) employing proprietary interfaces. It inevitably results in or aggravates serious social problems such as monopoly, otherwise avoidable digital divide, and illegal software copying. [3] Distance learning as a subfield of ICT is no exception in this regard. In this section we investigate the mechanism and give some examples from other subfields of ICT as lessons to learn for the distance learning community.

But first let's review a little terminology. In the software setting, by *interface* we mean a definite, agreed-upon, set of orders/rules/ways of exchanging information between two pieces of software. This can be either a file format (FF), a communication protocol (CP), or an application programming interface (API). [4] By *open interface* we mean an interface whose defining set of rules is published for anyone to study and follow without legal restrictions. A common confusion is to mix the idea of open interface with that of "open source software". The latter is a political alias for the original name "free software", and is often further aliased as Free/Libre/Open Source Software (FLOSS) for political correctness. FLOSS typically supports open interface in the sense that every rule of communication with other pieces of software is publicly known since the source code itself is publicly available for anyone to examine, and since the license terms do not prevent anyone from writing her own program to communicate in the same way, whether the program is proprietary or FLOSS. Conversely, open interface does *not* demand that the participating software be FLOSS. A typical example is Microsoft Excel, a non-FLOSS software, supporting both the CSV (comma-separated-value) format, an open interface, and the XLS format, its own proprietary (that is, non-open) interface.

When presented with choices of competing technologies, it is typical that uninformed consumers place priority on software features and technical merits over the much less visible issues of long term data preservation and susceptibility to monopoly, let alone the even more intangible idea of tight vs loose coupling, and proprietary vs open interfaces. Yet the prices to pay for choosing a tightly coupled system built of components communicating through proprietary interfaces are often very high as they lose their rights to time-shift and space-shift their very own information, among other rights. As a result, customers once hooked to such a system are very unwilling to switch to any other system even if the alternatives provide better functionalities. Sometimes the harm done unto the society reaches far beyond merely those consumers exercising their own choices, especially if these customers have direct or indirect influences on the market. Specifically, governments, Universities, and schools are among those most susceptible to vendor targeting since they are very influential consumers. When files and/or services from these institutions are presented to citizens, teachers, and students through proprietary interfaces, the latter as receivers/clients are forced to use a limited set of tools dictated by the specific vendor owning the interfaces since proprietary interfaces by their very nature prohibits competitions from other vendors. The receivers/clients are likely to acquire the dictated tools, either from buying it or from legally/illegally copying it if it is a piece of software. This process then repeats itself, file receivers starting circulating files in the same format, forced purchase or illegal copying results in further such activities, and a chain reaction ensues. Sometimes the vendors choose to offer the tools free of charge. This is welcomed by some as a sign of the benign gesture from the vendor and as an excuse to accept the vendor dictation. Yet in the larger scheme of things it actually accelerates the propagation of the proprietary interface and help the vendor tightens its control over the market. The gratis reader or converter does not help other vendors at all. Competition is stifled even as consumers applaud the generosity of the software dictator. [5]

A specific case in point is Microsoft's strategic spread of non-standard web technologies through the popular web authoring tool FrontPage and other web-programming tools, while vigorously ensuring that all computer-using citizens use only its gratis Internet Explorer to browse the net. When faced with the anti-trust suit, Microsoft alleged that this gratis browser is inseparable from the operating

system, even though both the technical investigations and the prevalent technical practice recommended by good software engineering suggested otherwise. [6] [7] Eventually no other commercial web browsers were able to maintain more than a tiny fraction of the market and the consumers lose their choice of alternatives. These days it is very common to find web sites that refuse any browser other than the Internet Explorer, as if the most important objectives of these sites were preaching a specific piece of software for a specific software company rather than selling their own products, promoting their own schools, or serving the citizens. This situation is particularly anguishing for a country like Taiwan, which makes major investments in attracting international tourism while a large proportion of the tourism-related business unknowingly embrace the "we-welcome-only-IE" attitude towards potential visitors simply because they know of no other browsers in their day-to-day transactions with the Taiwanese clients and/or business partners.

A second example is the spread of the proprietary file formats created by the Microsoft Office suites. Again, a gratis reader is provided, though only for the Windows platform. Being provided with low cost deals of the Office suites, Universities, schools, and government offices unwittingly engage in helping force the public to adopt these file formats. Many such influential consumers insist on exchanging files using "the .doc format" even though in fact it is not one format but rather a set of sometimes mutually incompatible formats. Any organization seriously considering switching to any other office suites is highly discouraged by the needs of exchanging documents with its business partners without losing the precise formatting details. This leaves us with a market without competition, legacy documents in a myriad of proprietary formats whose readability 50 years from now is in serious question, and an e-society whose daily operation heavily depends upon the good will of one single company. In some way this is reminiscent of the history when the fate of an entire opium-using population is in the hand of one single company.

Interestingly, such strategy is mostly successful with software but unsuccessful with the hardware market. For example, at one time IBM was pushing their proprietary Micro Channel Architecture (MCA) for PC buses. This proprietary architecture battled with the open EISA architecture for a while, and eventually the market gravitated towards the newer and open PCI architecture.

Decoupling of Contents from the System through open FF

The prevailing technology for distance education employed by many schools and institutions across the globe has been the Blackboard system. Its recent acquisition of WebCT, its chief competitor and the second largest player, results in a higher-than-80% market share of the LMS market in higher education. This raises the concerns of the antitrust experts and bloggers. [8] [9]

This comes as no surprise from an interoperability point of view. Interoperability issue of distance education software has been raised since 2002 or earlier. [10] [11] Universities have expressed concerns over the possibility of vendor lock-ins through proprietary proprietary file formats. The need to migrate course data from one system to another inevitably arises in the long run. Yet as a direct (and quite possibly intended) result of proprietary file formats, Universities lose the content they themselves created to the entangled system. It is often only at this unfortunately late point that the importance of adopting open file formats over flashy functionalities and interfaces are recognized.

SCORM addresses part of the problem by proposing open file formats for different distance education systems to exchange information with each other. By separating the content from the system, schools

can recover and migrate their long time investment in the creation and continual maintenance of course materials whenever the need arises. It also opens the door for schools using systems from different vendors to share and exchange part of their course content. The realization of these benefits, however, depends on vendors' willingness to make their products support and conform to open interfaces, SCORM or otherwise. From a business strategic point of view, supporting an open interface works against the largest vendor in the market, since it encourages competition. Typically it takes persistent and vocal demands from the customers in order for the vendor to decide to support the open interfaces.

In fact the file format concerns exist not only at the level of course packages, but also at the more fundamental level of individual files. We would like to bring the attention of the distance education community to the open document formats (ODF), the emerging ISO standard for office documents, as a replacement for the prevalent proprietary Microsoft formats. [12] As for the audio and video formats, the open formats ogg vorbis and ogg theora can be used as substitutes for the proprietary formats such as wma. [13]

Decoupling of Functions within the system through open API and CP

The separation of the content from the system still leaves one major problem unresolved. Different distance education systems (or different software programs, for that matter) for the same application typically offer different functionalities or different levels of sophistication for one functionality, each excelling in its own distinctive way. What if the consumer wishes to use function x from system A, function y from system B, and function z from system C? That is, may the consumer enjoy the best combination of features from several competing software programs?

Again the answer would be a resounding yes if we were talking about a hardware system such as the home theater. The consumer is free to choose a TV of one brand, a DVD/VCD player of the second brand, and a speaker of the third brand, because each interacts with one another through open interfaces and hence each can be replaced by a compatible and competitive product. In the software arena, the situation is in stark contrast to the above. The general public are under the false impression that compatibility means requiring the use of the identical piece of software, or software from the same manufacture. Microsoft's insistence on the non-breakability of IE from Windows partly relies such public misconception. Such insistence would become patently absurd when applied to the home theater scenario or the PC hardware scenario.

In reality, or at least in reference to good software engineering practice, a large system is preferably broken into loosely coupled components interacting through CP's (such as http) and/or API's (such as OpenGL) with clearly defined and publicly available specifications, just like the home theater or the PC hardware scenario. Sometimes loose coupling is achieved through an even looser architecture such as plug-in, making it extremely easy to replace one component with any other functionally equivalent one. In short, true compatibility does not require identicalness. Requiring identicalness is exactly the sign of lack of compatibility with alternatives.

Interestingly it is easiest to find such loosely-coupled software systems from the world of FLOSS, though a small number of proprietary software vendors (most notably Sun) also promote this idea. To browse the web from a GNU/Linux system, one could use the all-in-one browser firefox, or one could use a

combination of lynx for reading the text, ee for viewing the graphics, mpg123 and playmidi for listening to the sound, and so on. Each of these components can be replaced by one of many other alternatives, for example ee by xli, mpg123 by mpg321, playmidi by timidity, and lynx itself by w3m or links. In fact even firefox relies heavily upon the plug-in and extension architectures where additional (and often competitive) components are developed outside of the major development team. Open source software that involve text editing typically provide simple mechanisms for the user to replace the built-in text editor with any alternative editor of her choice, allowing the user to boost up her productivity upon first use of this new software by employing the experience, built over many years, of using a powerful editor such as vim or emacs.

Can we do the same with distance education software? Are we allowed to combine the nice functionalities found in the Blackboard system with some others from the WebCT system? Can the students carry over not only their familiarity with the instant messengers *they have been using all along*, but also this very piece of software directly to the distance education system we impose on them? (By the way, there are strong social and behavioral arguments favoring such an arrangement over forcing the students to abandon their familiar instant messengers.) Most interestingly, can we use existing, non-networked education software, or education software that employs networking in its own distinctive way, in the distance education system? This last point seems less explored in the distance education literatures, and yet excluding this possibility means failing to tap on the rich repertoire of generations of education software before and of the network age.

We suggest that most pieces are already there, and the most prominent missing piece is the integration. As the author is most familiar with FLOSS, the following list consists of such software. In practice any software, proprietary or FLOSS, supporting open interface can be used instead.

1. Course management: moodle <http://moodle.org/> It is employed by many schools and Universities. The system itself is also composed of modules and plug-ins, so that new functionalities can be easily added and unwanted features can be removed.
2. Content editing/playing: reload <http://www.reload.ac.uk/> It makes the creation and management of SCORM-compliant content a practical possibility rather than a theoretical utopia. For example, documents exist that explain how to migrate a course content from WebCT system to a moodle system.
3. Instant messaging: gaim <http://gaim.sourceforge.net/> It supports open protocols such as jabber and irc, as well as proprietary protocols such as msn and icq.
4. Screen broadcasting: vnc <http://www.tightvnc.org/> It was originally designed to control one's own computer while away from the office for example. However, there can be multiple clients simultaneously accessing the same server, and the server can optionally provide a read-only mode, meaning that only certain clients are allowed to use the mouse and keyboard to control the desktop. In effect it can be used for broadcasting the screen. It does not take up the entirety of the students' desktop, and it offers different modes of compression for trade-offs between speed and visual comfort.
5. Sound broadcasting: esd (the enlightenment sound daemon) <http://www.tux.org/~ricdude/overview.html> According to the documents, it is possible to effect sound broadcasting using esd, although the author has yet to find a suitable environment to successfully test it.
6. Visual communication: Ekiga <http://www.gnomemeeting.org/> Formerly known as gnomemeeting, it provides VOIP and video conferencing through H.323 and SIP protocols.

The above is certainly not a complete list of all functionalities required by a fully functional distance

education system. Yet the point is not completeness but rather decomposition of functionalities and integration of the individual pieces. It will be more economical from both the developers' point of view and the users' point of view for us to integrate these existing pieces of software into one system. It also allows for easy migration and all the other benefits mentioned elsewhere in this paper.

Standing on the Shoulders of the Giants

But visual and auditory interaction between the teacher and the students is not all that there is to distance education, and displaying a PowerPoint presentation remotely is hardly the most powerful thing that a computerised environment can offer. We propose that the distance education community investigate distance education *through executing remote applications*, a concept that is not found or emphasized in existing distance education software systems.

Sure, many education software today runs over the web using java, ajax, or a multitude of new web technologies. Yet as we posed earlier, there is the question of cutting off our resources from the legacy of the huge repertoire of existing software that do not employ these networking technologies. If we in the field of information technology agree with the culture of science and other technologies at large, which encourages "standing on the shoulders of the giants" instead of "reinventing the wheel", it will only be logical for us to better explore this avenue. Undoubtedly there are many such examples for Microsoft Windows. The author would like to list mathematics examples for the GNU/Linux system since these are the subject and system he is most familiar with.

1. Dr. Geo <http://www.offset.org/drgeo/> It is a geometry teaching toy that physics teachers will also love. The author of this paper has created interactive Dr. Geo figures for illustrating the effect of convex and concave lenses, the effect of gravity on a projectile, and the addition rule of velocity under the Lorentz Transform according to relativity.
2. kalzium <http://edu.kde.org/kalzium/> This is a periodic table of elements with an optional equation solver.
3. Geomview <http://www.geomview.org/> 3-D mathematical objects are probably best learn not only through equations, but also through interactive examination. Geomview facilitates this need.
4. gnuplot <http://www.gnuplot.info/> For a less sophisticated rendering demand but much easier experimental control and full 2-D drawings, gnuplot is an ideal tool for learning mathematics.

These programs can be very instrumental in helping students learn mathematics, physics, chemistry, and perhaps some other subjects. Yet some of them run only over unix-like operating systems such as various distributions of Linux and BSD. Symmetrically, there are many useful education programs that run only over the Windows family of operating systems. Vnc can be helpful since both the server and the client run on any of the above operating systems, allowing a student running Microsoft Windows to see how the teacher running GNU/Linux interacts with these education software, and vice versa. In a realistic virtual classroom setting, however, vnc broadcasting most likely operates in the read-only mode for security reasons. Is it possible, for example, for the each student to interact with Dr. Geo individually to learn Pappus' Theorem without requiring every student to install GNU/Linux in his own computer?

In a physical classroom setting, this request is partly addressed by the Linux Terminal Server Project (LTSP) <http://www.ltsp.org/> or Diskless Remote Boot Linux (DRBL) <http://drbl.sourceforge.net/>. Each is a piece of software that requires only one Linux server to be installed, and that all other

computers in the classroom be bootable from the network, either using a PXE-aware network card or using a floppy disk or cdrom that runs the etherboot protocol. Both LTSP and DRBL leave alone the hard disks of the classroom computers and simply boot them from the network to run the Linux operating system supplied by the server. LTSP is very much a thin-client model, is more suitable for low-end classroom computers, and requires a more powerful server. DRBL on the other hand distributes the computing loads to the classroom computers, hence requiring both the server and the clients to be fast enough, but neither is required to be very powerful.

LTSP and DRBL may not be suitable for a distance education setting, but their use of the underlying technologies may serve as an inspiration. In LTSP, each classroom computer serves as an X server, an NFS client, and an NIS client. In DRBL, each classroom computer is mostly simply an NFS and NIS client, with all X applications running locally instead of across the network. This is why it works more smoothly and why it demands more computing power from the classroom computer. Both LTSP and DRBL choose to employ existing, mature, and open network protocols instead of devising their own protocols. Thus they can leverage the power of all existing X applications, which are too numerous for any one, or even any dozen of software companies to ever come up with. Both LTSP and DRBL are deployed by many primary and secondary schools around the world, some of which are located in rural areas, have tight budgets, and/or have low end computers. Given enough network speed, there is no reason why distance education in Universities should ask for less than the full set of X applications.

Access of X applications is not limited to Unix-like systems alone. XLiveCD, a technology assembled by the Indiana University from the components of cygwin, allows Microsoft Windows users to remotely execute X applications. Conversely, for a GNU/Linux user to execute a Windows application from afar, one can turn on the Windows Terminal Service, and start rdesktop on his GNU/Linux box. It employs Microsoft's proprietary Remote Desktop Protocol. As this paper emphasizes, it is important to choose open interfaces whenever possible. However, the large install base of Windows machines necessitates this compromise. Microsoft has as yet not threatened the rdesktop project with legal suits, which it did to the samba team with its proprietary CIFS [14] and to the vocal FLOSS community at slashdot with its own obfuscated version of the originally open kerberos protocol. [15]

Conclusion: Emerging Possibilities and Looming Threats

Distance education seemed to have focused on the system aspect of software and paid relatively little attention to the remote application aspect of software. It is possible that the nature of proprietary licensing terms discourage development in this technically exciting possibility. By extending the idea of decoupling of contents from system to that of decoupling of functions within the system, we naturally arrive at the question of "broadcasting" not only the images and sounds but also the applications. FLOSS-based solutions for remote execution of applications and for other more conventional needs of distance education provide scattered but inspiring answers to our questions.

Possible actions of further researches include:

1. Collect existing FLOSS components and glue them together to form a coherent but loosely coupled system using scripting languages such as ruby or python. Such a system can be compared to and criticized against the commercial systems.
2. Integrate the capability to execute remote applications into the system, a missing feature in existing commercial systems, and study its novel applications in distance education.
3. Recruit and/or sponsor FLOSS developers using research grants for doing the above.

There are multiple reasons for such suggestions. Such researches will be built on top of existing, publicly available knowledge, which not only is in better accordance with the spirit of science but also relieves researchers of technical and legal burdens such as reverse engineering and counting number of seats of licenses. Moreover, it is reasonable for publicly funded researches to share its fruit with the maximum audience possible. Finally, by producing a FLOSS implementation we as a community of consumers of distance education software will have a reference platform for better negotiating with the (only dominating) proprietary vendor. We may then request the existing dominant system be decoupled, allowing exchange of its component with those of our system and/or those of other lesser players in the commercial markets, thus keeping it honest and keeping the market competitive.

On the other hand one also sees looming threats on the horizon, whether we build our own loosely coupled system or not. The anti-circumvention provision in Digital Millennium Copyright Act (DMCA) discourages the loose coupling paradigm by making it illegal to study and interoperate with proprietary interfaces of any form, including FF, CP, and API. [16] It is not unlikely that some day fair use loses its legal ground and researchers fall victims to DMCA as did the fifteen-year old Norwegian boy decoding DeCSS [17], the Russian engineer decoding Adobe e-book [18], or the American Professor unwittingly accepting the SDMI challenge. [19]

The e-book along with its legal case represents just one precursor of the coming Digital Right Management (DRM) technologies. [20] Despite Sony BMG's failed attempt at rootkitting consumers' Windows operating system [21] and Microsoft's heavily criticized Windows Genuine Advantage program which spies on legal and illegal users alike without user consent, [22] one would be naive to believe that the big software and media companies stop here. These technologies, along with the DMCA and patent laws, are originally aimed mostly at preventing illegal copying. Yet their strong measures hurt a much broader range of legitimate activities including fair uses such as space-shifting and time-shifting. It works by promoting proprietary interfaces, while the loose coupling paradigm depends on open interfaces.

As the domain of DRM extends, and as the hardware and the new Microsoft operating system Vista collaborate to enforce Trusted Computing (TC) [23] it will only be a matter of time before the field of distance education has to face these legal threats from these companies. Considering the extent and nature of distance education technologies, scholars of our time will probably be held accountable for the success or failure of the defense of consumer freedoms against big companies. Any technology based on proprietary, non-interoperable interfaces we help foster will eventually likely become part of the force that tries to control peoples' freedoms, while the FLOSS technologies along with proprietary software based on open interfaces will be an important defensive force. The road ahead for us seems no less political than academic.

References

1. Computer bus. (2006). In Wikipedia, The Free Encyclopedia. Retrieved July 10, 2006, from http://en.wikipedia.org/wiki/Computer_bus
2. Space shifting. (2006). In Wikipedia, The Free Encyclopedia. Retrieved July 10, 2006, from http://en.wikipedia.org/wiki/Space_shifting
3. Hung, C. K. and Shen, Y. L. (2005). *Social Considerations in Choosing Digital Archiving Technologies*. International Conference on Digital Archive Technologies. (2005)
4. Dennis, J. (2000). *Protocols, APIs and File Format Libraries*. Linux Weekly News. Retrieved September 9, 2006 from: <http://old.lwn.net/2000/0504/backpage.phtml>
5. Smith, I. (2003). *The Rise of Proprietary Formats*. Retrieved September 9, 2006, from http://opensource.mimos.my/foscon2003cd/paper/slides/09_imran_william_smith.pdf
6. Paul E. Ceruzzi. (2002). *A War on Two Fronts: The U.S. Justice Department, Open Source, and Microsoft, 1995-2000*. Iterations. An Interdisciplinary Journal of Software History. Retrieved September 8, 2006, from <http://www.cbi.umn.edu/iterations/ceruzzi.html>
7. McDougall, Steven. (2005). *Is The Browser Part of the Operating System?* Retrieved September 8, 2006, from <http://world.std.com/~swmcd/steven/rants/browser.html>
8. O'Hara, T. (2005). *Blackboard's WebCT Deal Spurs Antitrust Questioning*. The Washington Post. Retrieved July 5, 2006 from <http://www.washingtonpost.com/wp-dyn/content/article/2005/11/25/AR2005112501193.html>
9. Leslie, S. (2005). *More on the new behemoth - Timing, Open Source and Interoperability*. EdTechPost. Retrieved July 5, 2006 from: <http://www.edtechpost.ca/mt/archive/000717.html>
10. Arnone, M. (2002). *Mixing and Matching Distance-Education Software*. The Chronicle of Higher Education. (The issue dated May 24, 2002) Retrieved July 5, 2006 from: <http://chronicle.com/free/v48/i37/37a03301.htm>
11. Kempfert, T. (2002). *Preparing for Change: UW and the Quest for a New e-Learning System*. Teaching with Technology Today 9(2). Retrieved July 5, 2006 from: <http://www.uwsa.edu/ttt/articles/taskforce.htm>
12. OpenDocument. (2006). In Wikipedia, The Free Encyclopedia. Retrieved September 10, 2006, from <http://en.wikipedia.org/wiki/OpenDocument>
13. Ogg Vorbis. (2006). In Wikipedia, The Free Encyclopedia. Retrieved September 10, 2006, from http://en.wikipedia.org/wiki/Ogg_vorbis
14. Shankland, S. (2002). *Microsoft steps on Samba's toes*. Retrieved September 11, 2006 from: http://news.zdnet.com/2100-3513_22-904089.html
15. Roblimo. (2000). *Microsoft Asks Slashdot To Remove Readers' Posts*. Retrieved September 10, 2006 from: <http://features.slashdot.org/article.pl?sid=00/05/11/0153247>
16. Electronic Frontier Foundation. (2006) *Unintended Consequences: Seven Years under the DMCA*. Retrieved September 10, 2006 from: http://www.eff.org/IP/DMCA/unintended_consequences.php
17. Touretzky, D. S. (2000). *Gallery of CSS Descramblers*. Retrieved September 10, 2006 from: <http://www.cs.cmu.edu/~dst/DeCSS/Gallery/>
18. Touretzky, D. S. (2006). *Gallery of Adobe Remedies*. Retrieved September 10, 2006 from: <http://www.cs.cmu.edu/~dst/DeCSS/Gallery/>
19. Edward Felten. (2006). In Wikipedia, The Free Encyclopedia. Retrieved September 10, 2006, from http://en.wikipedia.org/wiki/Edward_Felten

20. Digital Rights Management. (2006) *Digital Rights Management and Copy Protection Schemes*. Retrieved September 10, 2006 from: <http://www.eff.org/IP/DRM/>
21. 2005 Sony CD copy protection scandal. (2006). In Wikipedia, The Free Encyclopedia. Retrieved September 10, 2006, from http://en.wikipedia.org/wiki/2005_Sony_CD_copy_protection_controversy
22. Mondok, M. (2006). *Microsoft places Windows Genuine Advantage on cheaper calling plan*. Retrieved September 10, 2006, from <http://arstechnica.com/journals/microsoft.ars/2006/6/14/4333>
23. Trusted Computing. (2006) *Trusted Computing*. Retrieved September 10, 2006 from: http://www.eff.org/Infrastructure/trusted_computing/

