How long can you wait for CMMI® Compliance?

Manage your software and systems engineering projects in guaranteed compliance with the CMMI NOW!

**processMax** 2i and 3i include all the necessary policies, procedures, guidelines, criteria, templates, and forms in role-based, step-by-step instructions, ready for use—everything you need for compliance with the CMMI-DEV at Level 2 and Level 3 respectively. Integrated with robust document management, workflow, and automated measurement and reporting, **processMax** is the intranet web-based solution for effective and efficient management of your software and systems engineering projects. **processMax** enables you to start operating in compliance with the CMMI immediately! We guarantee that organizations using processMax will not fail a formal appraisal led by an independent SEI-licensed appraiser.

**pragma** SYSTEMS CORPORATION
www.pragmasytems.com
877.838.PMAX
info@pragmasytems.com

GSA Schedule Contract No. GS-35F-0559S, processMax is a registered trademark of **pragma** SYSTEMS CORPORATION. Although processMax makes use of portions of “CMMI for Development, Version 1.2,” CMU/SEI-2006-TR-008, copyright 2006 by Carnegie Mellon University, neither the Software Engineering Institute nor Carnegie Mellon University have reviewed or endorsed this product.

Copyright © 2008 **pragma** SYSTEMS CORPORATION.
Two trends in software acquisition are becoming of increasing prominence. One is increasing concerns with Intellectual Property (IP) rights in software and data. Another is Free/Libre/Open Source Software (FLOSS), in which software is distributed with a mechanism that allows others to freely use the IP embodied in the software. Both of these trends present challenges to managers of systems with significant software subsystems.

One challenge concerns how one can protect the intellectual property embodied in systems that one develops. Another challenge is to understand the legal rights and limitations in the software one acquires. What can one do with that software, and what is one prohibited from doing? A final challenge concerns the expectations of some customers that they will receive FLOSS. How can it make business sense to provide licenses that allow others to freely use your IP? This issue of Software Tech News addresses these challenges.

In the first article, “An Introduction to U.S. Intellectual Property Law,” Robert Vienneau and Milton Johns, both of ITT Corporation, overview how intellectual property in software may be protected under United States law. The authors describe trade secrets, trademarks, copyrights, and patents, which are all currently applicable to software.

In his article “Software and Intellectual Property Rights,” Robert Dewar, co-founder, President, and CEO of AdaCore, describes his company’s approach to the commercial distribution and use of their software products. AdaCore produces Ada development tools and environments. They distribute their products under a FLOSS license. Dr. Dewar explains why this makes business sense, including how it encourages the provision of quality products and services.

In her article, titled “Preparing for Open Source,” Emma McGrattan, Senior Vice President of Engineering at Ingres, provides a case study of the decision to transition Ingres, an enterprise relational Data Base Management System (DBMS), to open source. Ingres faced issues arising from closed source and competition-sensitive code included in Ingres, from choosing a FLOSS license, in encouraging the FLOSS community to contribute to Ingres, and in convincing some customers that providing a FLOSS license was a positive development. This article discusses how Ingres addressed these issues in an ultimately successful effort.

In the last article, titled “Software Freedom and Web Applications,” the Internet activist, Aaron Swartz, provides a brief history of Richard Stallman’s development of the idea of free software. He also discusses the history of the World Wide Web and the growing importance of Web applications, such as those used on social networking sites. He suggests these developments inhibit the realization of Stallman’s principles. The user of a Web application cannot typically modify server-side code and is more likely to be interested in data and networking effects. Mr. Swartz considers the implementation of social networking capabilities in peer-to-peer applications to provide software freedom, in Stallman’s sense.

About the Author
Robert L. Vienneau is a senior analyst at the DACS. He has published articles on formal methods, on financial analysis techniques in software cost modeling, and on a real-time Synthetic Aperture Radar (SAR) demonstration of High Performance Computing (HPC). Mr. Vienneau holds a BS in mathematics from Rensselaer Polytechnic Institute and a MS in Software Development and Management from the Rochester Institute of Technology.

Author Contact Information
Robert Vienneau: robert.vienneau@itt.com
Introduction to U.S. Intellectual Property (IP) Law

As software becomes ubiquitous in the Department of Defense, the understanding of intellectual property issues in software becomes more important.

By Robert L. Vienneau and Milton Johns, ITT Corporation

Under current United States law, intellectual property in software may be protected by trade secrets, trademarks, copyrights, and patents. The capability of the Federal government to provide copyright and patent protection is granted in Article 1, Section 8 of the U.S. Constitution, which states:

“The Congress shall have power … to 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.”

Trade secrets are defined in a slightly different way in each state, but the Uniform Trade Secrets Act, as amended in 1980 (hereafter “UTSA”), is a good point of departure for a general definition. Under the UTSA,

“ ‘Trade secret’ means information, including a formula, pattern, compilation, program, device, method, technique, or process, that (i) derives independent economic value, actual or potential, from not being generally known to, and not being readily ascertainable by proper means by, other persons who can obtain economic value from its disclosure or use, and (ii) is the subject of efforts that are reasonable under the circumstances to maintain its secrecy.”

Thus, as adopted in most states, there is no formal filing or designation by a government entity to establish that certain information is a trade secret. The definition is rather a guideline for a finder of fact in a court proceeding to determine whether any particular information is a trade secret. Each state then provides protection for the owners of trade secrets by providing civil and in some cases criminal liability for misappropriation of trade secrets. The Federal government recognized in 1996 that there was lack of uniformity in the enforcement to trade secret legislation from state to state, and enacted the Economic Espionage Act. The Economic Espionage Act provides for Federal criminal and civil penalties for theft of trade secrets by foreign organizations and entities, as well as any party in the U.S. stealing secrets related to a product “produced for or placed in interstate commerce or foreign commerce.” “Misappropriation” under the UTSA and many states is defined, in short, as the wrongful taking and disclosure of the trade secret of another by a person who has not independently discovered by invention, reverse engineering, or from public information sources the information at issue. The Economic Espionage Act provides similar definitions.

The key then to protecting trade secrets is protecting trade secrets! One of the key factual inquiries that must be made in any trade secret case is whether reasonable means were used to protect the information, such as advising employees of the existence of the trade secrets, of limiting access to the information or the physical location of the information. For instance, publishing a trade secret in a trade or scholarly journal will defeat any other protections put in place.

Trademarks (and service marks) protect intellectual property but are very different from trade secrets. A trademark is a word, phrase, symbol or design, or a combination thereof, that identifies and distinguishes the source of the goods of one party from those of others. A service mark is the same as a trademark, except that it identifies and distinguishes the source of a service rather than a product. In a way similar to trade secrets, no governmental registration is required to create and establish one’s intellectual property in a trade or service mark. However, unlike with trade secrets, there is a formal process for Federal registration of trade and service marks to provide added protection.

The symbols “TM” or “SM” have become ubiquitous in American commerce, but Federal registration is not necessary to apply such designations. As soon as one intends to make a claim to the public as to the exclusive right to use a mark, a TM or SM as appropriate may be added. However, the similarly
familiar registered trademark symbol ® cannot be added until it is fully and formally registered at the U.S. Patent and Trademark Office, the Federal agency chartered to examine and register trade and service marks. Federal registration, while not necessary to establish rights in a mark, does provide significant additional protections, including: “constructive” notice to the entire US marketplace of the claim of ownership of the mark; the ability to bring an action in Federal court to enforce the mark nationwide; and the ability to obtain registration of the mark in foreign countries to prevent foreign infringement of the mark.

Copyright protection is similar to trademark protection. Copyrights protect expressions of ideas, but not ideas themselves (for instance, ideas, concepts, and discoveries). Copyright ownership inures to the writer as soon as the work is created in a fixed (typically but not exclusively written) form. The work does not need to be published to create the copyright, nor does it need to be registered with the Library of Congress, the keeper of registered copyrights. The Library of Congress identifies the following works as being copyrightable: literary works; musical works, including any accompanying words; dramatic works, including any accompanying music; notated or recorded pantomimes and choreographic works; pictorial, graphic, and sculptural works; motion pictures and other audiovisual works; sound recordings; architectural works. These categories are viewed broadly and most computer programs are considered literary works.

The current length of copyright protection for new works is defined in the 1976 Copyright Act (for works created after 1978):

- For the life of the author plus 70 years
- If written by corporate authors, 120 years after creation or 95 years after publication, whichever is shorter.

U.S. law frowns on the copyrighting of functional works, in which the expression of an idea is inseparable from the idea itself. In principle, copyright does not preclude fair use of copyrighted material. The Digital Millennium Copyright Act (DMCA) protects copyrighted material in digital format by criminalizing technology that circumvents Digital Rights Management technology.

Patents are for novel, non-obvious, and useful inventions. Patents give the holder the right to prevent or charge others who use the invention during its 20-year life, whether they came up with the invention independently or not. A patent application discloses an invention, and must demonstrate the invention can be reduced to practice. Business processes are patentable. In principle, laws of nature and pure mathematics cannot be patented. Jonas Salk, the inventor of the polio vaccine, took this idea further than the law does: “There is no patent. Could you patent the sun?”

Lawyers use terms of art, and their understanding of the meanings of these terms is clarified by case law. In re Allappat is an important case in the history of software in which the U.S. Court of Appeals ruled, in July 1994, that a combination of software and a machine to run it on is patentable. By now, the United States Patent and Trademark Office (USPTO) has granted over 170,000 software patents (See http://www.brookings.edu/press/Books/2005/mathyoucantuse.aspx). Some recent court cases of interest include SCO vs. IBM and In re Bilski. SCO claims that IBM violated their Unix license in their development of some Linux code. But last year a court ruled, in a related case, that Novell was the rightful owner of the relevant Unix copyright. Bilski is appealing the USPTO decision not to grant a patent for “a method for managing the consumption risks of a commodity”. This case, since it deals with the patentability of a process, has the potential to change the patentability of software.

As software becomes ubiquitous in the Department of Defense, the understanding of intellectual property issues in software becomes more important. In support of addressing that need, this article has presented an introductory survey of the protection of intellectual property in software under current United States law.

About the Authors

Milt Johns is a Virginia attorney with 23 years experience in the defense industry. He is currently a department manager for ITT Corporation, Advanced Engineering and Sciences Division in Alexandria, Virginia. Mr. Johns hold a BA in political science and MA in security policy studies from George Washington University, and a JD with High Honors from George Mason University School of Law.

Robert L. Vienneau is a senior analyst at the DACS. He has published articles on formal methods, on financial analysis techniques in software cost modeling, and on a real-time Synthetic Aperture Radar (SAR) demonstration of High Performance Computing (HPC). Mr. Vienneau holds a BS in mathematics from Rensselaer Polytechnic Institute and a MS in Software Development and Management from the Rochester Institute of Technology.

Author Contact Information

Milt Johns: milton.johns@itt.com
Robert Vienneau: robert.vienneau@itt.com
Software quality and reliability are mission critical. The size, persuasiveness, and complexity of today’s software can push your delivery dates and budgets to the edge. That means rigorous quality discipline, and collaboration between development and QA teams has never been more important.

With offices worldwide and over 8,500 customers, Seapine Software is the leading provider of integrated quality-centric application lifecycle management (ALM) solutions. For over 13 years Seapine’s easy-to-use, award-winning tools have helped companies achieve success by streamlining communications, improving traceability, and making development and QA teams more productive.

Satisfy Your Quality Obsession.
www.seapine.com/gsa
In this article, we will examine the issue of commercial distribution and use of software. The first thing to be said is that the title of this article is confusing and inappropriate. We use the term “Intellectual Property Rights” because it is so familiar, but we don’t like this term, especially not if we are talking about software. Why not? Because the traditional tools for protecting these so-called rights—copyrights, patents, licenses, and trademarks—are means to an end, not an end in themselves.

Society’s interest is in promoting innovation and progress. In the U.S. this is enshrined in the constitution very explicitly: “Congress shall have the power to 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”. For producers and consumers of software, the issue is how to ensure that this innovation can be assured. For a company producing innovative software, a business model is needed which ensures that software production can generate sufficient revenue to pay top quality engineers and fund continuing development. Development of commercial software is an expensive proposition so finding ways to fund this is a critical task for society.

In some environments, the model of cooperative volunteer work can and has provided useful software, using an open source development model. But in practice, most software is produced in commercial environments by engineers who expect to make their living from their work. Even in the case of well known large scale open source projects, such as the Linux kernel, Eclipse, and the GNU project, most major work is done by companies, such as IBM, Red Hat, and my own company, AdaCore, that pay people to work on these projects.

Recently, most commercial software companies, including Microsoft, have come to rely on energetic enforcement of their Intellectual Property Rights (IPRs), with extremely restrictive licenses, and rigorous enforcement of copyrights and patents. Unfortunately this model is severely flawed when it comes to software. To see why, let’s consider some examples from other fields. If Prada designs an elegant pair of shoes, you can buy them in a shop, and once you have bought them, you can fully enjoy them without any restrictions. You can lend them, wear them, sell them, and even carve holes in them if that is your wish. The fact that Prada energetically protects its rights in an attempt to prevent unauthorized copying by other manufacturers does not in any way restrict your rights as a consumer. In fact you benefit from these efforts, since you know that something with the Prada name will have the provenance and quality that you expect when you plonk down your credit card to buy them.

Similarly, if you buy a fancy coffee-making machine, you can do anything you like with it. If you open it up and start fiddling around, you probably void the guarantee, but you don’t end up as a target in a lawsuit. You can even resell modified equipment; an example of this is the many distributors of region-free DVD players, who buy standard DVD players, and modify them to region zero and resell them. In these cases the equipment you buy is likely to be protected by patents, but the fact that the company from which you bought the equipment carefully protects these patents does not directly affect you as a consumer.

With software, the situation is entirely different. If you buy software, you often have to agree to a license whose whole purpose is to restrict your rights as a consumer to make full use of the software, for instance by restricting its use to a single machine, forbidding you to transfer it to another person, preventing you from modifying it, and even making simple use of the software inconvenient by the use of license keys or similar devices, or in the worst case when protection technologies fail, preventing you from using the software entirely. This happened to people with Microsoft’s Vista system, where systems designed to prevent unauthorized use malfunctioned and prevented legitimate use. Another example from the music industry, where similar considerations apply, is the infamous root kit from Sony, where buying and listening to a CD on your computer severely compromised the entire operation of the computer.

Now back to the software manufacturer: unlike the shoemaker, who is determined to provide the most beautiful...
shoes and provide them to a consumer, ensuring that the consumer can fully enjoy the product, the manufacturer of proprietary software is caught between two concerns. On the one hand, they want to provide the software in the most usable form. On the other hand they are concerned with protecting their investment, which leads in the direction of highly restrictive licenses. In most industries, the general maxim is to give the consumer what they want, and providing software with licenses that are inconvenient and intrusive hardly seems to meet this goal.

Is there another way? We believe that in many situations the answer is yes. Now remember we are looking at this issue from the point of view of a commercial company that needs to make money to fund continued development and innovation. We are not on some ideological campaign here to eliminate all notion of copyright. In fact we are pretty sure that, for example, distribution of modern computer games that cost tens of millions of dollars to produce requires some form of copyright control. We are not so sure that software patents ever contribute to innovation, but that’s a discussion for another day.

At AdaCore, we produce advanced development tools and environments centered around the Ada language that are used by major companies such as Boeing, Lockheed, and Airbus to build the next generation of civilian and military aircraft, critical space systems, air traffic control systems, and other large scale applications requiring absolute reliability. Developing such systems is a complex task, and we have spent somewhere between fifty and a hundred million dollars so far building this technology. Most likely if we had outside investors, they would be very concerned about protection of our intellectual property rights in our systems, and expect us to use restrictive licenses to vigorously protect these “rights”.

In fact we distribute all our software using open source/free software licenses that are the polar opposite from being restrictive. Are we crazy? Or perhaps ideological zealots intent on undermining the basis of commercial software? Not at all! We are running a business, where, like anyone else running a business, we are concerned with maximizing revenue so that we can continue to develop and innovate (not to mention paying ourselves good salaries to support our families.) So how can we afford to give our technology away? That’s the question that is often asked, but it is quite off target and confused. We don’t give our technology away; we sell it at competitive prices in a market where many of our competitors do use restrictive licenses. But we believe in giving the consumer what they want and need. Restrictive licenses are a big pain in the neck. For us it is a significant competitive advantage that our software avoids these licenses, and many of our customers regard this as a big plus when it comes to deciding what technology to choose in a highly competitive market.

Despite this, people are still puzzled by our approach. We use the GPL license for our main tools, and a modified version of the GPL for run-time libraries and other components that our customers must be able to redistribute without restrictions. Our customers are certainly not operating in the open source environment. Many of them are developing highly proprietary systems, and in some cases highly classified systems. We suspect that some of them would in fact benefit from the use of more liberal licenses, but it is not our job to tell customers what to do—it is our job to give customers what they want! So, if we are using licenses like this, people wonder, how can we possibly make money? The answer is simple, and it’s the same answer any software company would give in describing their key to success—we provide well-tested quality software, with excellent support and upgrades, along with clear licenses that make the legal situation apparent. It is true that versions of our software can be obtained cheaply or free (nearly every version of distributed GNU/Linux comes with some version of our technology), but our customers are willing to pay for the service we can offer. In particular, the support services we provide are of key importance. If you have a large team working on a critical project that gets held up because of some misunderstanding of the technology they are using, the cost just in lost productivity can be huge, never mind the costs of late delivery. In addition, it is very important for large companies to have a very clear legal idea of the licensing of the software they use, and a company to stand behind the license. Downloading miscellaneous software from the Internet can be risky to your legal well-being!

It’s worth saying a bit more about support. Why do we provide excellent support? Well part of the answer is that we are committed to our technology and proud of our achievements, and want to make sure that people who use it are successful. But more importantly, and much more convincingly from our customer point of view, is that we charge for our software on an annual subscription basis. We make money if people renew their support contracts. They don’t have to—they
could continue to use the technology without support. So we have a strong incentive to provide good support so that our customers will indeed renew their contracts. It’s always good when the financial interests of a company are aligned with customer needs in this way. But, you say, if you know about the GPL, what’s to stop people from freely redistributing your technology, as permitted by the license? Yes, this is theoretically possible, but no one can provide the support we do, and our customers benefit from our high pace of continued innovation. Furthermore, our customers are simply not in the business of redistributing our tools. Interestingly, there is nothing to stop another company competing directly with us to provide support and improvements. This hasn’t happened so far, but the possibility is always there. Our business model has a certain “innovate or die” aspect, which keeps us hard at work, and most certainly benefits our customers.

What about users of our software? Due partly to deliberate spreading of misinformation by companies committed to a highly proprietary model, there are still those who mistrust commercially distributed Free Software. They worry about “losing their IPR’s”, or being forced to distribute sources of their proprietary applications. These concerns are misplaced. Of course it is important to carefully read license agreements and make sure you adhere to them. Our licenses allow you to do anything you could do with a Microsoft End User License Agreement (EULA), and a lot more as well, but they don’t allow arbitrary use. If you copy a chunk of our compiler technology into your proprietary code, that’s a copyright violation, just as it would be if you similarly copied Microsoft code. Both situations should be avoided! What about the “requirement” in the GPL that in this situation you are forced to distribute your own sources? There is no such requirement in the GPL. In the situation where you have illicitly copied GPL’d code in violation of the copyright, the GPL does permit but not require you to cure the copyright violation by publishing the source code, but you are not going to take this option if you have proprietary code you don’t want to disclose. Instead you will adopt other strategies for cure, such as getting a different license for the offending code or removing it.

It’s never a bad thing to have more freedom in what you can do with the software you buy, but education is needed. If you have programmers who simply assume they can do absolutely anything they like with open source/free software, you need to disabuse them of this confusion, or you could run into trouble. Remember also that just because you have the right to do something it is not necessarily a good idea to exercise that right. In a recent conversation with U.S. Navy officials in charge of establishing navy policy on open source usage, one of them said to me “Our lawyers are warning us that we may acquire some undesirable legal liabilities if we redistribute GPL’ed software.” My reply was that their lawyers were quite right and that they should probably issue a policy forbidding this. After all, I told them, the U.S. Navy is not in the business of redistributing software tools, why should they change this policy? Still there is confusion. In one case a large company insisted that we write them a special more restrictive license, since their lawyers were suspicious of the very free licenses we provided. However, as time goes by, more and more large companies depend on widespread use of open source tools, and their lawyers get more comfortable with the idea of copyright being applied to protect user rights and not simply those of the vendor.

In conclusion, the notions of free/open source software licenses, and commercial for-profit software manufacture are not necessarily in conflict. Our technology at AdaCore is not always the least expensive, but in the view of our growing family of Ada customers, it is the best, and they are willing to write us the checks that have allowed us to build and maintain a growing business for fourteen years that now employs over fifty full time people world-wide.

About the Author
Dr. Robert B. K. Dewar is co-founder, President and CEO of AdaCore; he also has had a distinguished career as Professor of Computer Science at the Courant Institute of New York University. He has been involved with the Ada programming language since its inception in the early 1980s and, as co-director of both the Ada-Ed and the GNAT projects, he led the NYU team that developed the first validated Ada compiler. Dr. Dewar was one of the authors of the requirements document for the Ada 95 revision, and he served as a distinguished reviewer for both Ada 83 and Ada 95. He has co-authored compilers for SPITBOL (SNOBOL), Realia COBOL for the PC (now marketed by Computer Associates), and Alys Ada, and he is a principal architect of AdaCore’s GNAT Ada technology. He has also written several real-time operating systems, for Honeywell Inc. A talented public speaker, he is frequently invited to conferences to share his thoughts on computers and on open-source software, and he has delivered papers and presentations on a variety of topics dealing with programming language issues and safety certification.

Author Contact Information
dewar@adacore.com
Technical Training On Demand

Cost effective way for organizations to provide continuous learning opportunities for their employees

- Pay one annual subscription fee ($475)
- Take as many classes as you want

AFFORDABLE

Classes available 24/7/52
From:
- Home
- Office
- Travel

FLEXIBLE

- Fit course work into your schedule
- Self paced
- Refresh knowledge
- Study during lunch or after work

DACS On-Line Learning Center

450+ classes
12,000 topics
Programming + Web Development Course Catalog
Latest Technologies
Java  XML
Oracle  Server Technologies
.NET Framework

To view the catalog visit www.thedacs.com/training
For details call 1.800.214.7921

COMPREHENSIVE

ACCREDITATION CERTIFICATION

ORACLE

MICROSOFT Certification Suite
- MCAD
- MCDBA
- MCITP
- MCPD
- MCSD
- MCSE
- MCTS
- MCAS

- CEUs granted
- IACET Accreditation Supported

STN 11-2 August 2008: Intellectual Property and Software
Preparing for Open Source

INGRES HAS PROVEN ITSELF IN MISSION CRITICAL DEPLOYMENTS FOR OVER A DECADE, AND HAS A GLOBAL SUPPORT TEAM THAT UNDERSTANDS WHAT IT TAKES TO PROVIDE ROUND THE CLOCK SUPPORT TO THE ENTERPRISE WHERE A MINUTE OF DOWNTIME CAN EQUAL LOSSES INTO THE MILLIONS OF DOLLARS.

by Emma McGrattan, SVP Engineering, Ingres

Open source projects typically start life in one of two ways. They can be born of the open source development community and grow organically in open source. Examples of this include Apache Tomcat, PostgreSQL, and Linux. Or, they can start life as a closed source product that is contributed to the open source community at some point during its lifecycle, such as Ingres, Eclipse, and Apache Derby. In this article I will outline a brief history of Ingres, the process that was followed to prepare Ingres for its contribution to the open source community, and touch on some of the lessons learned along the way.

The original INGRES project, founded at UC Berkeley in the 70’s and funded in part by the Air Force Office of Scientific Research, the Army Research Office, and the Navy Electronic Systems Command, was the foundation on which a number of commercially successful relational database solutions – Ingres, Sybase and Microsoft SQL Server – to name a few, were built. Developed as an open source solution under the Berkeley license, the original University INGRES is still in existence today. The commercial branch of the University INGRES project, which was also called Ingres, was acquired by Computer Associates (CA) in 1994 and as well as having its own users, was also provided to CA customers as the embedded database within CA’s infrastructure management product lines including Unicenter, BrightStor and eTrust.

In mid-2004, following a period of significant investment in the development of the Ingres technology, CA decided to contribute Ingres to the open source community in an effort to speed up the pace of innovation and in parallel leverage the open source development model to respond to customer needs more rapidly and effectively by engaging the customer in the process from design through to delivery. At that time MySQL and PostgreSQL had proven that there was a demand for an open source, lower cost alternative to Oracle, Sybase, SQL Server etc, but neither product had the feature set required for mission critical, enterprise deployment, nor did they have the ability to provide 24x7 support. Ingres was different. It had proven itself in mission critical deployments for over a decade, and had a global support team that understood what it took to provide round the clock support to the enterprise where a minute of downtime can equal losses into the millions of dollars.

Before contributing a project to the open source community, one must first understand the Intellectual Property (IP) ownership and provenance of every line of code. In 2004 the notorious “SCO Suits” were raging, and it was important to us to be able to provide indemnification, an insurance against IP infringement issues, to our customers. The Ingres team used a combination of automated tools, as well as a visual inspection of the code to ensure that the origins of every one of the millions of lines of source code were clearly understood. An additional scan of the code was performed to ensure that no customer names were inadvertently divulged in the code, and that any obscenities or inappropriate comments were removed. Decisions had to be made about a piece of Ingres functionality, the code for which had been licensed from a third party and could not be contributed to open source. In addition, a GUI Database Administration tool set modeled after CA’s Unicenter Database Management Tools was of concern. Opening this source could provide Unicenter’s competitors with an edge. It was ultimately decided that both pieces of functionality would be provided on-demand to the end user in binary form, but the source for them would be excluded from the project.

Choosing the right License is critical to the success of an open source project. CA reviewed the complete inventory of open source licenses approved by the Open Source Initiative (OSI) at that time and found that there wasn’t one that met its specific needs which was to sustain and promote the collaborative open source development of the code base, while maximizing the code’s ability to be used and integrated with software licensed...
under other licenses, including many commercial licenses. It was also important to CA to be able to provide CIOs with an assurance that CA would guarantee the provenance of the source code and would provide indemnification. CA set about creating what is now the CA Trusted Open Source License. In hindsight this was a mistake. The open source community was suspicious of CA’s motives and was deeply skeptical of CA’s commitment to building an open source community around Ingres. Creating a new license increased their skepticism. In 2005, CA divested the Ingres product line, and one of the first product related decisions that the newly formed Ingres Corporation made was to release the product under the GNU General Public License (GPL). The GPL is one of the most widely adopted open source licenses and is well understood by the open source community. Adoption of the GPL also enabled a dual licensing strategy which means that Ingres partners and customers can also license Ingres under a business license if they choose. That is, acquiring Ingres under a business license permits the distribution of products incorporating Ingres source code to be distributed without such products being themselves open source.

CA had no real hands-on experience running an open source project, and made a number of mistakes with its first and only product contribution to open source – the license was one, the Ingres Million Dollar Challenge was another. To encourage participation in the Ingres open source project, CA established a competition for open source developers with a million dollar prize pool behind it. Contestants had to create a toolkit to enable the migration from a competing database technology to Ingres. In hindsight, believing that one could buy the affection of an open source community was a mistake and while there were about a dozen reasonable submissions received, the interest of the contestants waned when the contest ended. Focus and investment should instead have been put on the creation of a collaborative community development environment that encouraged participation.

In 2004, the typical Ingres customer was not familiar with open source, and many reacted negatively to the announcement that Ingres was to become an open source project. Customers feared that the integrity and the security of the product would be compromised by exposing the code. Many assumed that as an open source project there would be no ownership of stewardship of the code and that anyone could make changes on a whim. An educational road show was conducted to debunk open source security myths and to educate the user community on the open source development process. As a closed source product, Ingres was developed by engineers in over a dozen development centers, and the distributed development process that had proven successful was adopted for the open source project. All changes are subject to peer review and a rigorous set of acceptance tests must be passed before the change can be accepted into the project. Open source development is a meritocracy and only those contributors that have earned the trust of the project team are awarded committer rights with direct access to update the source tree.

Ingres provides the security features that one would expect to use in a mission critical development project including user, group and role based security, discretionary access control, encryption and fine grained auditing of all system and data access. The number of security vulnerabilities that have been uncovered in Ingres in the past three years is inconsequential when compared to that of its closed source competition, proving that most security vulnerabilities are found through reverse engineering, rather than by reading the code.

I believe that the Ingres open source project can be declared a success. It has met the goals that were set in 2004 when it was first contributed to open source and now provides the CIO with a real alternative to prohibitively expensive, closed source, database solutions. Ingres has enjoyed many successes as an open source project, winning new customers and partners, and building a vibrant open source development community. Community contributors provide input into roadmap planning, test new releases early in the development cycle, report bugs and work together with the engineering and support teams to deliver new features that benefit the Ingres community at large. Those customers that were wary of open source in 2004 embrace it today, and have expanded their use of open source beyond the Linux operating system and the Ingres database.
WANTED SOA DESIGNERS!!!


DACS is looking for someone with 8+ years of experience, familiar with or knowledgeable in the following areas:

- SOA
- Web Services
- XML
- Pub/Sub Architectures
- DoD Architectures
- TIA (Technical Implementation Architecture)
- Programming in JAVA, C++, C # (C SHARP)

Work Environment:
- R&D Community
- Rome, NY
- Working with the customer
- Leading a small R & D team

Interested candidates should send resume and cover letter to soa-jobs@dacs.dtic.mil

About the Author

Emma McGrattan is Senior Vice President of Engineering at Ingres. She is responsible for the development of the Ingres database and associated products and technologies. A leading authority in Data Base Management System (DBMS) technologies, Ms. McGrattan has been instrumental in the ongoing success of the Ingres product line. Her blog, http://blogs.ingres.com/emmamcgrattan/, is titled “The View From 25B”. Born in Ireland, Ms McGrattan earned a Bachelor of Electronic Engineering from Dublin City University. Recognized for having open source development at heart, she

is also a member of the board of directors for the Eclipse Foundation.

Author Contact Information
Emma.mcgrattan@ingres.com
Software Freedom and Web Applications

COMPUTER SOFTWARE WAS SUPPOSED TO BE A TOOL TO SERVE PEOPLE; THAT’S WHY STALLMAN AND HIS LABMATES SPENT THEIR TIME WRITING SOFTWARE.

by Aaron Swartz, watchdog.net

PART ONE: THE RISE OF FREE SOFTWARE

It all started with a paper jam. It was 1980 and the Artificial Intelligence Lab at MIT had received an elegant new printer from Xerox. The printer, however, had an unfortunate tendency to jam, causing print jobs to pile up and nothing to get printed until someone happened to notice and fix the jam.

For Richard Stallman, one of the programmers at the AI Lab, this wasn’t such a big deal. With their previous printer, Stallman had simply changed the printer driver to detect whether the printer was jammed and, if it was, to notify anyone who had sent it a print job.

“If you got that message, you couldn’t assume somebody else would fix it,” Stallman later recalled. “You had to go to the printer. A minute or two after the printer got in trouble, the two or three people who got messages arrive to fix the machine. Of those two or three people, one of them, at least, would usually know how to fix the problem.” (Free as in Freedom, ch. 1)

But the Xerox printer was different: Xerox hadn’t provided the lab with the source code to their printer drivers. There was no way for Stallman to add this new functionality to the driver. When Stallman asked Xerox for the code, they refused to provide it, insisting that it was an important trade secret for their business. And when Stallman found a student at Carnegie Mellon who had been given access to the software, that student also refused to provide a copy, saying he’d signed a contract with Xerox not to share it.

Stallman was outraged. Computer software was supposed to be a tool to serve people; that’s why he and his labmates spent their time writing software. And yet, through a combination of greed and legal restrictions, people were forced to suffer because they were prevented from improving these tools.

Stallman wanted to ensure no one else would be forced to suffer in this way; he wanted to build a computer system based around principles of freedom. In 1984 he quit his job and announced the GNU project.

Stallman later clarified that free software was software that guaranteed users four freedoms:

1. The freedom to run the program, for any purpose.
2. The freedom to study how the program works, and adapt it to your needs. (Source code is a requirement for this.)
3. The freedom to redistribute copies so you can help your neighbor.
4. The freedom to improve the program, and release your improvements to the public, so that the whole community benefits. (Again, source code is a requirement for this.)

“Stallman codified these freedoms in the GNU General Public License or GPL. If you modify a piece of software that is licensed under the GPL and redistribute it, the license requires that you also redistribute the source code at no extra charge and allow everyone who receives a copy to do likewise.

Since 1984, the GNU operating system (whose most popular flavor is GNU/Linux) has been built and released under the GPL. A 2007 study found that 13% of servers and 1% of desktops were sold running GNU/Linux. And anyone can download the entire operating system for free off the Internet.

The success of GNU/Linux has led to a larger free software movement as well as the “open source” movement, which releases software and its source code under copyright licenses that provide some of the software freedoms.

The Mozilla Firefox browser, for example, is open source and currently makes up around 15% of market. Large portions of the Mac OS X operating system are also open source, including WebKit, the core of Safari, the Mac OS X web browser.

The open source and free software movements have now
built free alternatives for just about every major type of computer application, from word processing to video games. And for a time it seemed like Stallman’s dream had come true: one could truly continue to use computers without having laws restrict one’s freedom -- it was possible “to get along without any software that is not free.”

Tim Berners-Lee, an Englishman living in France who worked at a physics lab in Switzerland, was frustrated with how difficult it was for physicists to share documents. And so, in 1989, he came up with the World Wide Web, developed the standards that made it work, and built the first web browser and web server.

The power of the browser was its flexibility (or, in law professor Jonathan Zittrain’s phrase, its “generative nature”). Just as a general-purpose computer allowed you to run any program, from a music player to a graphing calculator, the web browser let you view any kind of document. A book, a physics paper, or photos of cats with funny captions -- the web browser doesn’t care; it displays whatever the server provides it with.

This seems like a trivial point now, but it was a vast change from other networked software at the time. Email programs, for example, are designed simply to display email -- they have an enormously specialized interface for composing emails, finding emails, seeing who an email is from and to, and placing emails in different folders. The same was true for discussion software, chat software, and other pieces of software that communicated over the network.

The Web was different: it did not specialize in any particular type of content, but let you share whatever you like.

This lack of specialization in the Web browser allowed people to move this specialization to the Web server. The traditional Web server simply served up static documents that someone had previously written. But it was quickly clear that there was no reason the server had to be so constrained.

Instead of simply serving up previously-composed documents, the server could compose new documents “on-the-fly” as they were requested. Thus, instead of simply having a document which listed what restaurants have tables available, a web server could be instructed to query the different restaurants, learn their availability, and construct a page from the results.

And users, instead of passively requesting different pre-written documents, could submit requests to the server and actually begin to interact with it. Thus, they could ask the server to reserve one of the tables and send their name and phone number along with that request.

The result was that the humble web browser quickly began to overtake all the other “specialized” applications. Instead of having a special program just for reading email, people read their email over the Web. Similarly, discussion groups, chat rooms, and other forms of social interaction have moved inside the Web browser.

But software developers quickly discovered that, for social creatures like us humans, everything has a component of social interaction. For example, titling and categorizing the photos you take would seem like an obviously solitary activity. But sites like Flickr demonstrate that people love to discuss and categorize photos of their friends, or even strangers, and that people, all things considered, would prefer to organize their photos in a program that exposes them to other people.

The result is the recent “Web 2.0” phenomenon, in which just about every piece of computing is moved onto the Web and made social in some way. For photos and videos, there is Flickr and YouTube. For news, there are sites like Digg and Reddit where you can submit, edit, and vote on news stories. Calendars, todo lists, even music collections and word processors are all being made into dynamic social web applications.

Pundits now discuss a not-too-distant future of “dumb clients” and “cloud computing” where the other applications on the computer disappear and all that is left is the web browser. And for people who use kiosk computers or Internet cafes, that future is already here.

For some, this is an exciting prospect. But for those, like Stallman, concerned with issues of software freedom, it is frightening. Even in the dark days of the proprietary printer driver, Stallman still had control over the computer which drove the printer, even if he did not have the source code to modify it. But with a Web 2.0 application, you don’t have even that. The computer running your software is locked away in some distant server farm. You can only communicate with it through your web browser.

Now this does provide some flexibility. Web browsers can be programmed to block ads or extract content. Plugins like Zotero and Greasemonkey let users add new functionality to existing sites by intercepting and modifying documents as they come back from the Web server.
SOFTWARE FREEDOM AND WEB APPLICATIONS (CONT.)

But this is a rather pale notion of freedom, like saying that moviegoers have control over the films they watch because they can hold pictures up in front of the screen as they watch.

Another option, which some sites are exploring, is that of providing APIs -- Application Programming Interfaces which allow people to write software to do the same kind of things a user would normally do with an app. Thus, instead of having to manually click the “buy” button on an Amazon page to buy a new set of razors, with an Amazon API you can have a program automatically purchase the razors for you every month.

This is undoubtedly useful, but again, a rather pale notion of freedom compared to the four freedoms that free software provides. If Amazon was truly free, you wouldn’t just be able to write programs to automate your usage of the application, you’d be able to change how the application actually works.

The obvious solution to this challenge is simply to release the software on the Web server under the GPL or some other free software license. Then anyone could download a copy and modify it to their heart’s content. And a new version of the GPL has been released, AGPLv3, which requires that people who use its software in web applications make their software available to the application’s users under its free terms.

But only a completely asocial web application consists purely of software. The vast majority of them are interesting because they give you access to data contributed by other users as well. For example, the software that lets people edit web pages is just about the least interesting thing about Wikipedia. The reason the site is so popular is because so many people have put their accumulated knowledge into that software.

Wikipedia has addressed this by going one step further -- not only is the source code free, the data is too. Anyone can download a copy of the Wikipedia database (excluding users personal information) and start up their own copy of Wikipedia based on it. And then they can modify their copy of Wikipedia’s software to work however they please.

It’s beautiful in theory, but in practice, of course, nobody does this. Even if your version of Wikipedia was full of fantastic new features, it would still be nearly impossible to get anyone to use it. People use Wikipedia because that’s where all the other people are; it’s practically impossible to get everyone to switch.

For Wikipedia, the problem is somewhat ameliorated by having some psuedo-democratic control over the site. So Wikipedia is run by a board elected by (a tiny subset) of its users and the board has nominal control over the software and modifications that get made to it. But this is still a far cry from the freedom GNU/Linux users have in the non-networked world. Running for office, getting elected, then pushing your patches through a change-resistant bureaucracy is a lot more difficult than modifying some source code files on your computer and restarting.

And so, the hard-core partisans of software freedom propose that we will see the pendulum once again swing away from centralized server computing.
The DACS Gold Practice Initiative:

- Promotes effective selection/use of software acquisition & development practices
- Defines essential activities/benefits of each practice
- Considers the environment in which each practice is used
- Addresses the timeliness of practice benefits
- Recognizes interrelationships between practices that influence success or failure
- Contains quantitative and qualitative information
- Includes discussion forums, case studies and survey opportunities
- A continually evolving resource for the DoD, Government, Industry and Academia
- Free to use/free to join

Current Gold Practices:

- Acquisition Process Improvement
- Architecture-First Approach
- Assess Reuse Risks and Costs
- Binary Quality Gates at the Inch-Pebble Level
- Commercial Specifications and Standards/Open Systems
- Ensure Interoperability
- Formal Inspections
- Formal Risk Management
- Goal-Question-Metric Approach
- Integrated Product and Process Development
- Model-Based Testing
- Plan for Technology Insertion
- Requirements Management
- Requirements Trace-Off/Negotiations
- Statistical Process Control
- Track Earned Value

Learn More About the DACS Gold Practice Initiative:
http://www.goldpractices.com

Become a Registered Member!
http://www.goldpractices.com/registration/register_step1.php

Data & Analysis Center for Software
P.O.Box 1400
Rome, NY 13442-1400

http://iac.dtic.mil/dacs
and back to a world where we all run applications on our local machines. Only this time, instead of being applications that don’t use the network or only talk to a distant server, they will be peer-to-peer applications, seeking out other users and interacting with them directly.

Some great strides have been made in building peer-to-peer software, in no small part because of the vast amount of interest in using the technology to share music without getting caught by enforcers of the law. But, especially compared to Web 2.0 server technology, peer-to-peer is still in its infancy. Writing a social application so that it’s peer-to-peer is about a thousand times harder than writing the same program as a web app.

Still, peer-to-peer software, if we could make it work, would seem to give the best of both worlds: the freedom to modify how a program functions on our local computers as well as the ability to share and collaborate with others across the Internet. And so, for those who care about freedom (as well as those who care about sharing music), this seems like an important avenue for further research.

Reference

About the Author
Mr. Aaron Swartz is a writer, web developer, and activist. He is the co-author of the RSS 1.0 specification; co-founder of reddit.com, a social bookmarking system for news; and founder of watchdog.net, a site for political information and organizing.

Author Contact Information:
me@aaronsw.com
Are you getting All you can from your software investment?

The DACS ROI Dashboard

Impact on Quality (% defect reduction)

75th Percentile: 90

Oklahoma City Air Logistics Center, Test Program Set and Industrial Automation line observed the defect density decrease 91%, from 3.3 to 0.3 Defects per KSOLOC, while achieving CMM Level 4.

Technologies Covered:
- SEI CMM/CMMI
- SEI Team Software Process (TSP)
- SEI Personal Software Process (PSP)
- Inspections
- Reuse
- Cleanroom
And Many More!

Graphs Showing Impact Of Software Technologies On:
- ROI
- Productivity
- Quality

Summarizes Facts From Open Literature

Data & Analysis Center for Software
P.O. Box 1400
Rome, NY 13442-1400

http://iac.dtic.mil/dacs

Access the DACS ROI Dashboard!
http://www.thedacs.com/databases/roi/
The DACS website has been updated with new research information.

The DACS is a central distribution hub for software technology information sources. The DACS offers a wide-variety of Technical Services designed to support the development, testing, validation, and transitioning of Software Engineering technology.

---

LITERATURE AND REPORTS RELEVANT TO IP ISSUES


**Research on Innovation** ([http://www.researchoninnovation.org/](http://www.researchoninnovation.org/)): Research on Innovation is a non-profit organization created to conduct, sponsor and promote research on technological innovation and to disseminate the results of this research to a broad audience, both in academia and in industry. Provides a number of downloadable papers by James Bessen and others.

---

KEY SOURCES OF OPEN SOURCE CODE

**Apache Software Foundation** ([http://www.apache.org/](http://www.apache.org/)): The Apache Software Foundation provides support for the Apache community of open-source software projects. Apache projects are characterized by a collaborative, consensus based development process, an open and pragmatic software license, and a desire to create high quality software that leads the way in its field.

**Freshmeat** ([http://freshmeat.net/](http://freshmeat.net/)): A large index of Unix and cross-platform software, themes and related “eye-candy”, and Palm OS software. Thousands of applications, which are preferably released under an open source license, are cataloged in the freshmeat database, and links to new applications are added daily.

**Open Channel Foundation** ([http://www.openchannelfoundation.org/](http://www.openchannelfoundation.org/)): A mechanism to efficiently publish software from universities while allowing for the commercialization of the most promising programs. This publishing entity is a non-profit organization devoted to publication and community building. It is separate from the commercialization process, which is handled by Open Channel Software, the parent company.

**Source Forge** ([http://sourceforge.net/](http://sourceforge.net/)): Provides free hosting to Open Source software development projects with a centralized resource for managing projects, issues, communications, and code.

**Linux** ([http://www.linux.org/](http://www.linux.org/)): The free Unix-type operating system originally created by Linus Torvalds with the assistance of developers around the world.

---

Be sure to check it out at [https://www.thedacs.com/](https://www.thedacs.com/)
The first 50 people to send in a completed survey will receive a FREE DoD/IT Acronym CD from the DACS.

This valuable CD-ROM contains over 9,000 Department of Defense and Information Technology acronyms. There are hundreds of acronym lists available but none are as well done as this CD AND specifically targeted towards DoD and Information Technology. This unique-shaped CD-ROM plays in your computer’s regular, hub-mounted, CD drive. You’ll use this great resource over and over again. It’s FREE, just for filling out our brief survey on the next page!

Data & Analysis Center for Software (DACS)

http://iac.dtic.mil/dacs/

Data & Analysis Center for Software (DACS)
1. Which volume of the Software Tech News did you receive? ________________________________

2. When did you receive the newsletter? (month/year) ________________________________

3. How satisfied were you with the CONTENT of the newsletter? (Article Quality)
   - Very Satisfied
   - Satisfied
   - Neither Satisfied nor Dissatisfied
   - Dissatisfied
   - Very Dissatisfied

4. How satisfied were you with the APPEARANCE of the newsletter?
   - Very Satisfied
   - Satisfied
   - Neither Satisfied nor Dissatisfied
   - Dissatisfied
   - Very Dissatisfied

5. How satisfied were you with the OVERALL QUALITY of the newsletter?
   - Very Satisfied
   - Satisfied
   - Neither Satisfied nor Dissatisfied
   - Dissatisfied
   - Very Dissatisfied

6. How satisfied were you with the ACCURACY of the address on the newsletter?
   - Very Satisfied
   - Satisfied
   - Neither Satisfied nor Dissatisfied
   - Dissatisfied
   - Very Dissatisfied

7. Approximately how much of the newsletter do you read?
   - The entire issue
   - Most of the content
   - About half the content
   - Briefly Skimmed
   - Didn’t Read

8. Would you read this newsletter in an E-mail newsletter format?
   - Definitely
   - Probably
   - Not Sure
   - Probably Not
   - Definitely Not

9. How did you request the product or service?
   - Phone Call
   - E-mail
   - DACS Website
   - Subscription Form
   - Other __________________________

10. Would you recommend the Software Tech News to a colleague?
    - Definitely
    - Probably
    - Not Sure
    - Probably Not
    - Definitely Not

11. What topics would you like to see this newsletter devoted to? __________________________

    Comments (optional) _________________________________________________________________

    ________________________________________________________________________________

REGISTER FOR THE FIRST TIME  |  UPDATE CURRENT SUBSCRIPTION

Name: ________________________________  Position/Title: ________________________________

Organization: ________________________________  Office Symbol: ________________________________

Address: ________________________________  City: ________________________________

State: ___________________  Zip: _______________  Country: ________________________________

Telephone: _____ - _____ - ______ Fax: _____ - _____ - ______  Email: _______________________

Functional Role: ______________________________________________________________________

Organization Type:  □ Air Force  □ Army  □ Navy  □ Other DoD  □ Commercial  □ Non-Profit  □ Non-US
   □ US Government  □ FFR&D  □ Other ____________________

*Note: You must give us your address to receive the CD.
ABOUT THE SOFTWARE TECH NEWS

STN EDITORIAL BOARD

Ellen Walker
Managing Editor
ITT Corporation, DACS

Philip King
Production Editor
ITT Corporation, DACS

Paul Engelhart
DACS COTR
Air Force Research Lab

Morton A. Hirschberg
Editorial Board Chairman
Army Research Lab (retired)

Thomas McGibbon
DACS Director
ITT Corporation, DACS

Dr. Kenneth E. Nidiffer
Software Engineering Institute

Dennis Goldenson
Software Engineering Institute

John Scott
Mercury Federal Systems

ADVERTISEMENTS

The Software Tech News is now accepting advertisements for future newsletters. In addition to being seen by the thousands of people who subscribe to a paper copy, an electronic version is available at the DACS website, exposing your product, organization, or service to hundreds of thousands of additional eyes every month.

For rates and layout information contact: news-editor@dacs.dtic.mil

COVER DESIGN

Wendy Butcher
Graphic Designer
ITT Corporation, DACS

ARTICLE REPRODUCTION

Images and information presented in these articles may be reproduced as long as the following message is noted:

“This article was originally published in the Software Tech News, Vol.11, No.2 August 2008. Requests for copies of the referenced newsletter may be submitted to the following address:

Philip King, Production Editor
Data & Analysis Center for Software
P.O. Box 1400
Rome, NY 13442-1400

Phone: 800-214-7921
Fax: 315-838-7130
E-mail: news-editor@dacs.dtic.mil


In addition to this print message, we ask that you notify DACS regarding any document that references any article appearing in the Software Tech News.

ABOUT THIS PUBLICATION

The Software Tech News is published quarterly by the Data & Analysis Center for Software (DACS). The DACS is a DoD sponsored Information Analysis Center (IAC), administratively managed by the Defense Technical Information Center (DTIC).The DACS is technically managed by Air Force Research Laboratory, Rome, NY and operated by ITT, Advanced Engineering and Sciences Division.
<table>
<thead>
<tr>
<th>Article Title</th>
<th>Authors</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tech Views — Intellectual Property and Software</strong></td>
<td>by Robert L. Vienneau, ITT Corporation</td>
<td>3</td>
</tr>
<tr>
<td><strong>Software and Intellectual Property Rights</strong></td>
<td>by Robert B.K. Dewar, AdaCore</td>
<td>7-9</td>
</tr>
<tr>
<td><strong>Preparing for Open Source</strong></td>
<td>by Emma McGrattan, SVP Engineering, Ingres</td>
<td>11-13</td>
</tr>
<tr>
<td><strong>Software Freedom and Web Applications</strong></td>
<td>by Aaron Swartz</td>
<td>14-16,18</td>
</tr>
</tbody>
</table>