ical databases are about words, with far more structure—WordNet provides more than twelve lexical relations. The content of WordNet is organized around conceptual clusters of words, called “synsets,” with an extensive taxonomy that is rooted in a fairly small number of top-level terms. The goal, at least for WordNet, is to organize words into groups according to their meaning, and thus it was necessary to build a taxonomy of concepts that do not correspond to words, such as “Amount of Matter,” but rather to the kinds of things in the world that words refer to. As a result, lexical databases such as WordNet have a dual status. They have ontologies that include classes that correspond to things in the world and not words, however these ontologies are driven by words and their meanings, not by a careful analysis of the world itself. The difficulty here is that the part of WordNet that is more clearly an ontology is not discernable; one must simply know where the dictionary begins and the ontology ends.

**Catalogs**

A catalog is a specification of the kinds of inventory that some organization keeps—for example, the products that a company sells. In this sense, catalogs are quite clearly ontologies, albeit simple ones: they specify the “kinds of things that exist” in that domain, assign symbols to them, and often provide some meaning for those symbols. Catalogs are not traditionally considered ontologies because they are simple and require little analysis or specification—the things that exist are the things you sell. Modern online order systems, however, that provide functionality such as configuration, need to have information about compatibility, power consumption, and so forth, that clearly require more sophisticated analysis.

**Classes and Instances**

Ontologies describe the *kinds* of things in some domain, and normally in information systems *classes* are used to represent kinds of things. It is important in general to understand the difference between classes, such as “book,” and instances, such as *The Old Man and the Sea*. It is common to view information system ontologies as composed only of classes (and relations), and not their instances, however this is another incorrect association—the class/instance distinction is different. Consider the example of defining the class “Italian”—the definition requires Italy, an instance of the class “Country.”

Christopher A. Welty

See also Information Organization; Information Retrieval; Lexicon Building

**FURTHER READING**


**OPEN SOURCE SOFTWARE**

The term open source software was coined in 1998 to describe both a method of software distribution and a form of software licensing. An overlapping version of this idea, dating back to 1984, is called free software. In this article, both concepts will be discussed collectively as free and open source software (F/OSS).

**Fundamental Concepts**

The free and open source distribution of software means that users of the software have access to a form of the software that can be modified and is readable by humans. The distribution of software under an open source license means that the users of the software are free to modify and share the software with other users, possibly subject to a few limited
restrictions, depending on which open source license is used. Although these are the two main features of F/OSS—the distribution of human-readable code and the freedom to modify and redistribute the code—other aspects of F/OSS are noteworthy. Some of these aspects include (1) a new software development model, (2) a new software business model that threatens established software businesses, (3) a unique online community and culture, (4) a new model for work and employment, (5) the globalization of software development, (6) a model for open sharing of other intellectual content (for example, music, books, educational materials, and scientific knowledge), and (7) a challenge to conventional ideas about economic motivations, intellectual property, the digital divide, and government policy.

Open source software projects are self-organizing, with many developers from around the world volunteering to form software development teams. Often they do this as a hobby, but an increasing number of developers are working as paid consultants or as employees of firms that use the F/OSS product under development as part of their product or service offerings. Typically the software is distributed at no or little cost among the community of users in what is described as a “gift economy.” Prior to the introduction of the open source label, software that could be freely modified was called “free software.” Many still choose to call it free software, but they emphasize that “free” does not mean free of cost, but free to modify and redistribute. Richard M. Stallman, a recipient of the ACM Grace Hopper award and the first president of the Free Software Foundation, often whimsically clarifies this point with the quote: “Think of free speech, not free beer!”

For many programming languages, the human-readable source code is converted by a compiler into computer instructions comprised of 0’s and 1’s called the binary format of software. This binary form of software is optimized to run on computers, but because it is not easy for humans to read or modify, it cannot be called open source software unless the human-readable source code is also freely available. In the 1950s and 1960s, computer firms made their profits from selling computer hardware and often bundled their software in what we call the open source format today. But by the 1980s, computer firms, especially those primarily focusing on software, began to sell closed source software, often in copy-protected forms. This proprietary software came with end user licenses that restricted modification or redistribution to protect the authors’ ownership of the intellectual property embedded in the software. Violations of these license restrictions against the copying and sharing of closed source software was called software piracy, and software industry trade associations used aggressive enforcement against such software pirates.

Despite the prevalence of the closed-source copy-restricted software distribution business model, the development and distribution of free and open source software, which started in the 1980s, has grown in volume, popularity, and importance. Software distributed under free or open source licenses is dominant on the Internet. Examples include the Linux and BSD operating systems, the GNU development tools, the Apache Web server, Sendmail for mail distribution, BIND for Internet address translation, the Mozilla web browser, the MySQL and PostgreSQL relational databases, and many programming languages including Perl, PHP, TCL/Tk, Python, and GNU/GCC. Not coincidently, free and open source software grew in popularity and importance as the Internet and the WWW took off in popularity and importance, feeding each other’s growth. The Internet enabled the collaboration of widely dispersed F/OSS developers, many of whom were also programming using systems on the Internet. However, free and open source software is not without controversy, both inside and outside the F/OSS community.

**Early History**

The story of free and open source software largely begins in the late 1960s at two locations: the MIT Artificial Intelligence (AI) Laboratory and the AT&T Bell Labs. At the MIT AI Lab, a community of faculty, students, and programmers, working on computers running their locally developed ITS operating system (the Incompatible Timesharing System) developed a computer culture playfully called the hacker culture. This was long before the media co-opted the mean-
The term 'hacker' developed to describe people who illegally crack into computers for malicious purposes. This culture developed its own vocabulary, folklore, and traditions, which were characterized by an openness to sharing ideas and software, hard work, non-standard work hours, brilliant programming, and a playful irreverence toward authority (it was the 1960s). Stallman, an important actor in the free software movement, began his programming career at the MIT AI Lab in the early 1970s. At the AT&T Bell Labs, the UNIX operating system was developed by Ken Thompson and the programming language C was developed by his colleague Dennis Ritchie, both ACM Turing Award recipients.

These two innovations, UNIX and C, were designed to be portable, flexible, and extensible, making them attractive to many programmers at that time. Copies were shared with the computer science department at the University of California at Berkeley. Modifications and additions were added and collected into a Berkeley Software Distribution, referred to as a widely used operating system, improved and enhanced by many people since its development in 1969 by Bell Laboratories.

to other computer architectures, and shared with many other research sites. At these three sites, the MIT AI Lab, Bell Labs, and later UC Berkeley, the programmer culture, programming languages, software tools, and operating systems, evolved, spread, and became a significant influence on the free software movement which later evolved into the open source movement, although neither term existed at that time.

In the early 1980s two important events triggered the beginnings of the free software movement. First, the community around the MIT AI Lab collapsed because of the loss of many of the lab's programmers to AI company start-ups, and the replacement of lab computers by new computers with closed source operating systems and software. Second, after the breakup of the AT&T monopoly, which had been restricted from competing in the computer industry, the company commercialized the UNIX system. This led to a long, acrimonious legal battle between AT&T and UC Berkeley over the distribution rights to UNIX and copyright infringements by the BSD UNIX distributions.

In 1983, Stallman started an ambitious project to develop a replacement for UNIX called GNU (GNU is Not UNIX) and to distribute the GNU software freely (as in free speech, not free of cost). Proceeds from the sales of the GNU software are used to support the Free Software Foundation he later founded to promote the idea and development of free software. As a reflection of the strong ideology behind the move, and to protect the GNU software from being converted into closed source, all GNU software is distributed under the GNU General Public License (GPL). GPL permits the software to be freely distributed and freely modified. If the modified software is redistributed, the modifications must be included in the source code so that other users can inspect, benefit from, and further modify those modifications. GPL does not require a company that modifies a program for internal use to share the modifications; this is only required if the modified program is redistributed.

Authors of software licensed under GPL may sell their software for any price, as long as the source code is included or made available elsewhere. Also, a person who buys such software is free to resell it or share copies of it with others for no charge. Under GPL, embedded or modified and redistributed software must also be GPL; this is sometimes called the viral property of GPL. The Free Software Foundation uses the GNU General Public License (and variations) for the GNU software and documentation. Linux and many other FOSS programs are also distributed using GPL. Some other FOSS projects have developed alternative licenses for their software which are close in spirit to the GNU GPL. Examples include the Mozilla Public License, the Apache Software License, and the Berkeley Software License. Many of these other licenses are less restrictive on how derived works need to be licensed. For example, under the Berkeley Software License, software can be embedded in a commercial product and that product does not have to be distributed under the Berkeley license, as would be the case with GPL.
Linux
The term Linux is used to identify either (1) an operating system kernel, (2) a UNIX-like operating system, or (3) a complete server or desktop software distribution including end user applications. This F/OSS project was started in 1991 as a hobby project by a 21-year-old student in Finland, Linus Torvalds, who is currently the lead maintainer of the Linux kernel, the critical component at the core of the Linux operating system. For Linux to function as a complete operating system (similar to a Windows or Macintosh operating system), hundreds of other support programs must accompany it. Collections of software consisting of the Linux kernel, its support programs, and additional software are called a Linux distribution. Popular examples include the RedHat Linux, Debian Linux, and SuSE Linux. They typically include the GNU software tools and utilities, programming languages, desktop windowing environments, browsers, office productivity suites, and networked services (for example, Apache, PostgreSQL, Samba, and Sendmail), and include the added value of providing configuration and compatibility for all the included programs. The Linux distributions guarantee that everything works together and provide maintenance and security updates.

Participation on the Linux kernel project grew quickly after Torvalds released the first version on the Internet, with estimates of hundreds to thousands of contributors. The Linux operating system is considered comparable in quality, security, and performance with other commercially developed closed source operating systems (for instance, Windows). This surprised many software engineers, whose experience suggested that the development of a large complex piece of software like an operating system kernel was a major undertaking requiring a tightly managed project using strong centralized control of the development process. Even members of the free software community were surprised at the success of Linux.

The Cathedral and The Bazaar
Eric Raymond, a F/OSS developer and outspoken advocate of F/OSS, analyzed and documented software engineering principles that could be learned from the Linux development success. The title of his white paper and book, The Cathedral & The Bazaar, suggests two very different software engineering styles. The Cathedral-style tends to be top-down, closed, highly structured and ordered, and under strong central control. The Bazaar-style tends to be bottom-up, open, less structured, and under loose central control. Closed proprietary software and even some F/OSS projects tend to use the Cathedral-style approach, while Linux was his exemplar of the Bazaar-style approach. The paper presents software engineering lessons that could be learned from the Linux case. Two of these lessons, which are often quoted, are (1) Release early, Release Often, and Listen to your customers, and (2) Linus’s Law: Given enough eyeballs, all bugs are shallow.

Perhaps one of the most important consequences of Raymond’s white paper was that it influenced Netscape’s decision to release the company’s browser software as open source and to initiate the Mozilla project. This event gave the F/OSS movement credibility in the eyes of corporate information technology managers who were typically uncertain of its credibility.

The Open Source Initiative
After the Netscape decision to embrace the free software idea, several leaders in the community decided to deal with the problem that the word “free” in “free software” had multiple meanings, and that the obvious one, free of charge, was misleading, and probably hindered the adoption of free software by businesses. Under the GPL and other free software licenses, anybody is allowed to sell copies of free software, thus confronting new users with a confusing contradiction. (And “free” in the context of free software may also suggest cheap, shoddy and low quality.) To address this problem and to look for ways to better promote free software adoption by business users, several activists and thinkers including Eric Raymond, Sam Ockman, John Hall, Todd Anderson, Larry Augustine, and Christine Peterson met and Peterson suggested the new label “open source.” Linus Torvalds supported the idea, and Richard Stallman originally supported it (but reverted back to a preference for “free software”). In 1998 Bruce
perens (then project leader of the free debian linux distribution) and eric raymond co-founded the not-for-profit open source initiative (osi). perens adapted the debian free software guidelines into the open source definition (osd).

the osd contains ten points: (1) freedom to redistribute must be retained, (2) source code must be available, (3) modifications and derived works are permitted, (4) derived works may require name changes or other actions to protect the integrity of the original author's source code, (5) no discrimination is allowed against persons or groups, (6) no discrimination is permitted against fields of endeavor, (7) a license must follow the software when it is redistributed, (8) a license must not be specific to a product, (9) a license must not "contaminate" other software on the distribution medium, and (10) a license may not be predicated on any individual technology or style of interface. the precise details of the osd can be obtained online at http://www.opensource.org/. open source licenses that are in conformance with the osd and certified by osi as such can use the osi certification mark. by 2004, over fifty open source licenses were osi certified. the osd was intended to be inclusive and most of the early free software licenses are osi certified, including the gnu gpl, the apache license, and the bsd license. the open source concept gained momentum as major commercial software vendors (for example oracle) began to port their products to linux. one other event in 1999 generated publicity for f/oss—confidential internal microsoft documents acknowledged the quality of f/oss and the potential business threat f/oss posed for the company.

the business of free and open source software

even though the earlier gpl and the later osd permit charging a price for open source software, the requirement that the source be included or available, and that the buyer be able to copy and freely share the software seems to limit the opportunities for a profitable business built on f/oss. how can any business survive in the f/oss marketplace when the product is free of cost? despite the apparent anti-business feature of f/oss, many examples of highly successful business models exist. cygnus solutions, founded in 1989, built a successful business around contract development modifying and porting gnu software to new computing platforms, including embedded processors. in 1999 the company was acquired for over $600 million by red hat software, which had been founded four years earlier by robert young. the red hat initial public offering (ipo) saw the eighth-biggest first-day gain in wall street history with a capital value of nearly $3 billion at the end of the day. several weeks later, the va linux ipo was the largest single-day gain in wall street history.

of course the dot-com bubble has burst and both firms market capitalization is now much lower, but both are still in business and growing sales. these are examples of pure-play open source firms making profits by adding value to the free software. other examples include computer firms using f/oss as a commodity component in larger product offers. for example, ibm sells linux on their servers and built a large e-business offering, websphere, on top of the apache web server. apple computer built their new os x operating system on top of the open source darwin, a derivative of the open source bsd unix.

free and open source software

controversies and dilemmas

several of many controversies and dilemmas associated with f/oss revolve around the long-term viability of the f/oss model, its impact on the established software economy, and its impact on innovation and competitiveness in the software industry.

internal tensions

will internal tensions limit the long-term success of the f/oss model? within the f/oss community there is philosophical tension between the free software purists (like richard stallman) and the more pragmatic open source software engineers (like linus torvalds and eric raymond). while stallman
considers the principles and political issues of freedom in the use of software as primary, others like Torvolds write software for fun and are less concerned about how the software is used and distributed. Other tensions exist between the GNU project and the Linux project as evidenced over the question of what to name the resulting operating system. While Stallman and the Free Software Foundation came close to achieving the GNU project's goal of a replacement operating system for UNIX (only the kernel was missing), the independently developed Linux kernel project used the available GNU software and effectively produced a replacement for UNIX. Should the combination of GNU and the Linux kernel be called Linux, or GNU/Linux or LinGNUx?

The large fortunes generated by the IPOs of F/OSS firms like Red Hat and VA Software has created resentment among some developers in the free software community. Disagreements and sometimes acrimonious quarrels between F/OSS leaders on public discussion boards can be found. Although many projects have been spectacular successes, many more have failed through neglect or imploded because of disagreements between the developers. These events raise questions about the sustainability of the F/OSS model—is it a utopian experiment doomed to failure? Are there limits to the size of F/OSS projects and long-term survivability issues? On the other hand, the availability of the source minimizes the risk. Should a project fold, at a minimum, in-house programmers or contract programmers can be hired to support the software.

Motivations

Why do they program for free? Questions about why F/OSS developers seem to work for free, sharing their software at no cost, is perplexing to many, including potential corporate customers. Recent studies of this phenomenon provide some answers. For example, Alexander Hars and Shaosong Ou conducted a survey of the motivations of seventy-nine F/OSS developers and discovered that although some (16.5 percent) were motivated by altruism, 70.9 percent did so to improve their programming skills and 51.9 percent wished to build a professional network. The latter cases suggest that many are working on F/OSS with the expectation of future rewards. Of interest in the Hars and Ou study is that programmers paid to work on F/OSS comprise 16 percent of the sample, but make up 38 percent of the estimated cumulative working hours on F/OSS reported by the respondents. This suggests that F/OSS developers are in fact motivated by either current or future returns. It has been reported that as many as 80 to 90 percent of the developers on the Apache projects are paid for their work by their employers, which suggests that for-profit firms may be embracing open source as a commodity or common goods, similar to the standards and collaborative research seen in many industries.

Innovations

Does F/OSS enhance or diminish innovation? A broad look at typical F/OSS projects suggests that many are copycat software projects imitating previously developed commercial software. For example, GNU and Linux are re-implementations of UNIX. GIMP, a digital-image editing program, is very similar to Adobe's Photoshop. R is a F/OSS statistical package that mimics the commercial S. OpenOffice likewise mimics Microsoft Office. Businesses invest income from current sales into R & D on new products. If F/OSS copycat products cut into sales and reduce resources for new product R & D, will the long-term effect be a reduction in innovation? On the other hand, the F/OSS community can point to new classes of tools and products that they created—for example, powerful new types of editors like Emacs, or innovative programming languages like Perl and Python. Investigations into new product innovation suggest that many breakthrough paradigm-shift innovations come from small groups of researchers while large corporate research often does defensive R & D to protect current businesses.

Economic Competition and the Market Economy

How can for-profit firms compete against not-for-profit copycat software development? Is F/OSS too utopian or too similar to communism? Some firms (for instance, Microsoft) consider F/OSS a threat to their core business strategy. For example, ACM Turing Award recipient Jim Gray argued in March 2004 that the open source model endangers the software economy. Others argue that open source is pro-
viding an opening for non-United States software firms to compete in what had been a United States-dominated global software market. Meanwhile, some large commercial firms (for instance, IBM) have embraced the F/OSS model, are investing billions of R & D dollars on open source technologies. Their business model uses lower-level technologies, such as the operating system or Web server, as commodities and aims for competitive advantage through higher-level proprietary software built on top of the open source commoditized technology. Additional business opportunities are pursued through integration and consulting services around the open source technologies.

F/OSS activists will argue that open source is all about freedom of choice. Michael Tiemann, founder of Cygnus Solutions, argues that the philosophical treatises on free software written by Richard Stallman read like a business plan and motivated his business startup. Perhaps when there is change, established businesses will be threatened and adaptation will be needed. Change presents both threats and opportunities to all the actors.

Gregory Madey

See also Compilers; Programming Languages

FURTHER READING


OPTICAL CHARACTER RECOGNITION

Optical character recognition (OCR) lies at the core of the discipline of pattern recognition by computers. OCR is the process of converting the image of a text document into its digital text equivalent in order to interpret a sequence of characters in an alphabet. For general computer users, OCR also has the practical use of being able to scan a document and then manipulate the text in a word processor.

Characters of an alphabet are usually rich in shape but can be subject to many variations in terms of fonts and handwriting styles. Despite these variations, a basic abstraction of shapes identifies such