



# INTRODUCTION TO OPEN SOURCE

---

THE NEW DRIVING FORCE IN SOFTWARE

---

## ABSTRACT

---

This paper provides a very basic introduction to open-source software. It is intended primarily for non-technical readers. Its purpose is to help business people understand what open-source software is, how it is impact both the software industry, and how it might affect their own business.

# INTRODUCTION TO OPEN SOURCE

THE NEW DRIVING FORCE IN SOFTWARE

---

## WHAT IS OPEN-SOURCE?

---

*“The basic idea behind open source is very simple: When programmers can read, redistribute, and modify the source code for a piece of software, the software evolves. People improve it, people adapt it, people fix bugs. And this can happen at a speed that, if one is used to the slow pace of conventional software development, seems astonishing.” – The Open Source Initiative*

The prevalence and impact of open-source software has grown over the past decade to the point where it now impacts the business decisions of non-programmers. What is open-source? Is it software? Can I install open-source on my computer? Where do I buy it? In these short few pages, I hope to help the non-technologist answer such questions and gain a practical understanding of what open-source software actually is and is not.

In this paper, I will explain what makes software open-source. You will find that the distinction is very simple, but the impact is profound. We will review some of the impacts open-source software has already had on the software industry. We will then review some of the advantages open-source can provide to your business. We will conclude with some background information on particularly notable open-source software projects and where you can go to learn more. But first, let me describe what open-source software actually is.

Open-source software is not a particular software package the way, say, Microsoft Word or Oracle 9i is. Open-source software refers to a category of software. A particular software package is or is not open-source. For instance, the Linux operating system is open-source while the Microsoft XP operating system is not. Likewise, the PostgreSQL [sic] database is open-source but the Oracle 9i database is not.

How does one categorize open-source software? There are three basic characteristics that make open-source software what it is. First, the source code of open-source software is open and available. Don't let the term source code scare you. I will explain that in a minute. Second, the way in which open-source software is created is very different than other software. Third, the way in which open-source software is licensed is very different than other software. We will discuss each of these three characteristics.

## OPEN SOURCE CODE

The key to understanding what open-source software is lies in the meaning of the word *source*. This is a technical term used in the software industry, but with a little explanation it is simple enough to understand by anyone.

When it comes right down to it, the only thing a computer understands is ones and zeroes. Software, when it is in the memory of the computer and actually running, is nothing but a very, very

long string of ones and zeroes. This language of ones and zeroes is called *binary*. It is impossible, or at the very least exceedingly impractical, for programmers to write software as binary code. The human mind just doesn't work well that way. For instance, even a well-trained programmer would have trouble reading the following:

```
10010001000101100110010011001001111
```

It says HELLO.

In order to write software, programmers use a language that is easier to understand, one that uses letters and words and resembles English. Some of the more popular programming languages are C, C++, Visual Basic and Java. Instead of ones and zeroes, the programmer writes something like this:

```
x = 0
repeat
  print x
  x = x + 1
until x > 10
```

In order for the software to actually run, it must be translated from this human understandable format into the binary format that computers understand. This process of translating the software into ones and zeroes is called *compiling*. The original format that the programmers use and understand is called *source code*. After it is translated into ones and zeroes, it is called *compiled code*. Another programmer can read the source code, understand what it is supposed to do and even modify it to make it do something slightly different. Once the software is compiled, however, there is nothing a programmer can do with it.

When you buy software like Microsoft Word or Oracle 9i, you get the compiled code. You do not get access to the source code. Many companies consider the source code of their software a trade secret. Other programmers cannot change the compiled software or look and see how it works.

With open-source software, however, the source code is openly available to anyone. Hence, it is called open-source. Other programmers can look at how the software works and even change the way it works by changing the source code. Once they make these changes, they can compile the source code and have new software.

The open availability of the source code is a telltale characteristic of open-source software, but that is only part of what makes open-source what it is. The other two characteristics, the licensing and the developing, deal with the consequences of making the source code openly available. It is the consequential characteristics that impact the software industry and your business.

#### WAY OF DEVELOPING SOFTWARE

Probably the most important characteristic of open-source to the software industry is the way in which it is developed. When software's source code is available, programmers can understand how the software works and even modify it. People make software open-source because they want other programmers to understand and modify the software.

Usually, companies employ programmers to write software that the company will sell. Most of the time, the company is contracted to build custom software to the specific needs of a business customer. It is estimated that 90 percent of all software code in the world is written for custom applications. Other companies create software for general use by many customers. This kind of

software is sometimes called *shrink-wrapped software* because it often comes in a box encased with clear shrink-wrap plastic. Although shrink-wrapped software may only account for 10 percent of the code, it certainly demands the most attention in the market. When we think of the software industry, we usually think of companies such as Microsoft and Oracle that produce and sell shrink-wrapped software.

Software companies employ hundreds or even thousands of developers. There are large marketing and sales staffs that determine what features the software should have. The programmers spend months building and testing the new features. The company may release a new version of the software with these new features perhaps once a year. There is a period just before each new release called *beta testing* in which the company tries to find all the bugs in the software and fix them. It requires a large management staff to coordinate this whole process.

Open-source software is completely different. There is usually no company responsible for creating the software. Individual programmers from around the world write the software and contribute their changes to the source code for free. The open-source operating system Linux, for instance, has approximately 10 million lines of source code written by perhaps as many as 10,000 programmers, from Finland to Formosa. Instead of releasing new features once a year, open-source software projects make new releases almost constantly, anywhere from once a month to once a day. This changes the whole bug fix and beta testing process completely.

#### WAY OF LICENSING SOFTWARE

Probably the most important characteristic of open-source to your business is the way in which it is licensed. Earlier, we had to do some basic explanations of what source code is. Here we are going to have to discuss what a software license is. I believe you will find this pretty straightforward.

Most of us do not buy software; we license it. You can buy a car and you can buy a chair because what is and what is not the chair or car that you bought is pretty clear to all of us. With software however, it is not so very clear because software can be copied so easily. If I buy a chair, it is only going serve the purposes of one person at any given time. If I buy Microsoft Word and there are no restrictions on copying it, then there is no limit to how many people the software will serve.

When I buy a car or chair, I own it and can tear it into pieces and share those pieces with whomever I wish. The company that sold me the chair or car is not going to care. There are still just as many people out there without chairs or cars as there were before. In fact, through my own destructive stupidity, there is one less: me. The company has not loss any potential revenue. If I make ten thousand copies Microsoft Word, however, and give those copies to ten thousand of my closest friends and relatives, the situation is completely different. Now there are ten thousand fewer people in the world that do not need a word processor. Microsoft has arguably lost potential revenue.

Thus, software companies cannot give customers full outright ownership of software as property the way a car or chair manufacturer can. They instead give their customers a license to use the software within specific boundaries. Most notably, customers are not allowed to copy and distribute the software to other people.

Open-source is very different. The source code is freely available to anyone in order to encourage contribution by other programmers. It is impractical to restrict distribution of source code, so most open-source software projects do not. In fact, to be truly open-source, there can be no restrictions on any party selling or giving away the software. The price of open-source software is free.

The fact that open-source software is free does not preclude companies from paying for it. Red Hat made more than \$90 million in revenue in 2000, mostly from selling Linux, an open-source operating system. The reason is because software does not exist by itself in a vacuum. The software itself is only part of the solution. The software must be compiled properly. Some people pay Red Hat for that alone. The software must be installed, tested, integrated and, most critically, supported. Many companies choose to pay Red Hat to support their Linux installs. That way, if something isn't working as expected, the company has someone to call to get help. The price for support on Linux is still much less expensive than software and support for commercial proprietary operating system.

The open-source license has some other interesting characteristics besides free distribution. For instance, most open-source licenses allow anyone to own and sell any additions or changes they might make to the software. Most of the licenses make pains to prevent a bait-and-switch with the software, that is, suddenly demanding payment for software that use to be free. Further discussion of open-source licenses can be found in Appendix A.

---

## IMPACT OF OPEN-SOURCE

---

The Internet and the World Wide Web are arguably built on open-source software, not just the servers and browsers, but also the networking guts like domain lookups and mail routers. Reciprocally, it is the Internet that enables large open-source software communities to work together so effectively. Open-source grows the Internet and the Internet grows open-source.

Open-source software is a significant force within the software industry. According to a recent survey conducted by InfoWorld magazine,

“A vast majority of today’s corporate IT executives (+65%) are now using or plan to use open-source OSes and Web servers for their enterprise applications... About half of the CTOs we polled trust their business to open-source application development tools and application servers. Almost all respondents reported that open-source projects save their company time and not just a little money.”

It is not just the Internet. Open-source projects now provide relational databases and office productivity suites, and the list is growing constantly. The following bullet points provide some further highlights to the impact of open-source.

### WEB SERVERS

Apache has dominated the web server market since 1996. As of September 2001, 60% of the public domain web servers use Apache, with Microsoft at 28%, iPlanet (formally Netscape) at 4% and all others at only 8% [Netcraft <http://www.netcraft.com/survey/>].

IBM decided in 1999 that it would not be able to compete against Apache and chose instead to private label Apache as IBM WebSphere.

### OPERATING SYSTEMS

According to survey conducted by IDC in January 2001, Windows accounted for 41% of new server operating system sales in 2000, growing by 20% - but GNU/Linux accounted for 27% and grew even faster, by 24%.  
[[http://www.computer.org/computer/homepage/june/ind\\_trends/index.htm](http://www.computer.org/computer/homepage/june/ind_trends/index.htm)]

Linux accounts for 30% of all public domain web servers.  
[<http://www.netcraft.com/Survey/index-200109.html>].

In 2001, IBM has pledged to invest \$1 billion annually in Linux products and the Linux community. Linux generated more than \$90 million in revenue in 2000 for companies such as Red Hat and Suse.

#### **OTHERS**

The number of large open-source communities is growing.

- In 1998, Netscape made its web browser an open-source product under the name Mozilla. Now Netscape (and presumably AOL) licenses its browser from Mozilla.
- In 2000, Sun Microsystems made Star Office an open-source product under the name Open Office.
- In 2001, IBM made WebSphere Studio an open-source product under the name Eclipse. The intellectual property donated to the project was worth \$40 million.

---

#### **ADVANTAGES OF OPEN-SOURCE**

---

##### **RAPID DEVELOPMENT**

As a general rule, open-source software evolves much more rapidly than its commercial counterparts. That means key features are made available much faster. The amazing thing is that if a particular feature is critical to a company, they need not wait on the vendor to include it. They can hire developers to build it themselves.

##### **SECURITY**

Rapid advance is particularly important with security. The Apache and Linux projects have been known to release patches for security holes within hours after they are reported.

Open-source software is generally more secure than commercial counterparts due to the built process of peer review. The community as a whole reviews every line of code. There is no security in obscurity. Software security is only achieved through sound algorithms. The built-in peer review process that creates open-source software is very good at producing sound algorithms. Having dozens, sometimes hundreds of eyeballs on the source code leaves little room for poor code. As a result, there are much fewer cracks for the malicious hacker to exploit.

In August 2001, Gartner Group, a leading industry analyst, issued a strong recommendation that companies stop using Microsoft IIS, a commercial web server, because of its security holes. They recommended that companies instead use Apache, an open-source web server that had proven to far more secure. [<http://www.gartner.com/DisplayDocument?id=336339>]

##### **SUPERIOR CODE**

What goes for security goes for all the features of open-source software. The built-in peer review process is pitiless with poor code. It will be rejected or changed by more competent programmers. More importantly, the rapid evolution of the software allows even good code to become better. It is

common for programmers to discover the best way to solve a problem only after they have built a solution to it.

The communities that build open-source software are an exceedingly competent and self selected group of men and women. They have a passion for what they do. The developers of open-source software participate and submit code to the project to great extend because they seek the respect of their fellow programmers. It may be truly said that, in open-source, the pride of a job well done is payment enough.

#### **LOWER COST**

Open-source software saves money. According to a recent survey by InfoWorld magazine, “almost all CTOs using OSS, 93 percent, reported reduced cost of application development or acquisition as a chief benefit. Of those CTOs using OSS, the most – 32 percent – reported a cost savings of more than \$250,000.” [[http://www.infoworld.com/cto/t\\_opensource\\_survey.html](http://www.infoworld.com/cto/t_opensource_survey.html)]

Lower cost is not only because the software source code is available for free, although that is certainly a nice advantage. Open-source is very compelling in the total cost of ownership over the life cycle of the applications. Simple economics dictate that a larger supply drives prices down. If there are more developers, programmers and consultants that know the open-source software than the commercial equivalents, then the cost of hiring the talent needed to support open-source software should be less.

#### **LOWER RISK**

Open-source software lowers risk. Many large companies have start-up software companies place their source code in escrow in case the company goes bankrupt. Having access to the proprietary source code, however, is a poor assurance if there are no developers left around who understand the source code and can support it.

The survival of open-source software is not dependent upon the survival of any particular company. The software will continue to exist and grow as long as there is a compelling reason for it to exist, just as your company will continue using the software so long as there is a compelling reason to do so.

In a notorious internal memo now known as the Halloween Papers, Microsoft acknowledged that “fear, uncertainty and doubt” (FUD) would not work well against an open-source competitor because there is little doubt that the open-source software is going to go away.

There are other more subtle risks that open-source lowers. For instance, if there are more developers, programmers and consultants that know the open-source software, then your company has a greater chance of finding the skills it needs to support the software. If your company is dependent upon a specific feature in the software, there is a lower risk of that feature not being built or being discontinued. You can hire a contractor or employee to build and maintain the feature.

#### **GREATER FREEDOM**

Open-source lowers risks in part because it provides greater freedom to its users. Users are not dependent upon the solvency of another company. They are not dependent upon the development schedule of another company. They are not dependent upon what another company is willing to reveal about the software. Support is not dependent upon a select group of people with proprietary knowledge.

Open-source provides the freedom to modify and optimize the software to meet one's particular needs. It provides the freedom to choose who will support the software from a large and growing talent base.

---

## CONCLUSION

---

Open-source software is a category of software characterized most notably by the open availability of the software's source code. The true differentiator of open-source software, however, is the way in which it is created: by a community of developers stretched across the globe making and releasing additions to the software for free at an overwhelming pace. What makes open-source software possible is the Internet and open-source software licenses schemes. The one provides a physical means for cooperation of the community while the other provides the legal means for cooperation.

Open-source is no panacea. However, if properly applied, open source software can provide tremendous advantage to the information technology infrastructure of any enterprise. The most notable advantage is cost savings, quantified by organizations in the hundreds of thousands of dollars. Other advantages include rapid development, security, superior code, lower risk and greater freedom.

Open-source is a movement within the software industry that has proven it can produce high quality software for free. Although it can never completely replace all commercial software, open-source is a significant force within the industry and should be a serious addressed in any company's information technology strategy.

---

## APPENDIX A: DEFINITION OF OPEN-SOURCE

---

The Open Source Definition, version 1.9, from the Open Source Initiative, <http://www.opensource.org/docs/definition.html>. The indented, italicized sections below appear as annotations to the Open Source Definition (OSD) and are not a part of the OSD. Reprinted verbatim with permission.

### INTRODUCTION

Open source doesn't just mean access to the source code. The distribution terms of open-source software must comply with the following criteria:

#### 1. FREE REDISTRIBUTION

The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.

Rationale: By constraining the license to require free redistribution, we eliminate the temptation to throw away many long-term gains in order to make a few short-term sales dollars. If we didn't do this, there would be lots of pressure for cooperators to defect.

#### 2. SOURCE CODE

The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost—preferably, downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed.

Rationale: We require access to un-obfuscated source code because you can't evolve programs without modifying them. Since our purpose is to make evolution easy, we require that modification be made easy.

#### 3. DERIVED WORKS

The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

Rationale: The mere ability to read source isn't enough to support independent peer review and rapid evolutionary selection. For rapid evolution to happen, people need to be able to experiment with and redistribute modifications.

#### 4. INTEGRITY OF THE AUTHOR'S SOURCE CODE

The license may restrict source-code from being distributed in modified form only if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.

Rationale: Encouraging lots of improvement is a good thing, but users have a right to know who is responsible for the software they are using. Authors and maintainers have reciprocal right to know what they're being asked to support and protect their reputations.

Accordingly, an open-source license must guarantee that source be readily available, but may require that it be distributed as pristine base sources plus patches. In this way, "unofficial" changes can be made available but readily distinguished from the base source.

## **5. NO DISCRIMINATION AGAINST PERSONS OR GROUPS**

The license must not discriminate against any person or group of persons.

Rationale: In order to get the maximum benefit from the process, the maximum diversity of persons and groups should be equally eligible to contribute to open sources. Therefore we forbid any open-source license from locking anybody out of the process.

Some countries, including the United States, have export restrictions for certain types of software. An OSD-conformant license may warn licensees of applicable restrictions and remind them that they are obliged to obey the law; however, it may not incorporate such restrictions itself.

## **6. NO DISCRIMINATION AGAINST FIELDS OF ENDEAVOR**

The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

Rationale: The major intention of this clause is to prohibit license traps that prevent open source from being used commercially. We want commercial users to join our community, not feel excluded from it.

## **7. DISTRIBUTION OF LICENSE**

The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

Rationale: This clause is intended to forbid closing up software by indirect means such as requiring a non-disclosure agreement.

## **8. LICENSE MUST NOT BE SPECIFIC TO A PRODUCT**

The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.

Rationale: This clause forecloses yet another class of license traps.

## **9. THE LICENSE MUST NOT RESTRICT OTHER SOFTWARE**

The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.

Rationale: Distributors of open-source software have the right to make their own choices about their own software.

Yes, the GPL is conformant with this requirement. Software linked with GPLed libraries only inherits the GPL if it forms a single work, not any software with which they are merely distributed.

---

## APPENDIX B: RESOURCES

---

### LEADING OPEN-SOURCE PROJECTS

The following are some of the most notable open-source projects. This is by no means an exhaustive list. There are hundreds if not thousands of open-source projects. Some open-source projects even compete.

Project	Description	Commercial equivalents/ competitors	Website
Linux	Operating system	Microsoft NT, Sun Solaris	
Apache	Web server	Microsoft IIS, iPlanet	<a href="http://www.apache.org">www.apache.org</a>
PostgreSQL	Relational database	Microsoft SQLServer, IBM DB2, Oracle	
Open Office	Office productivity	Microsoft Office, WordPerfect	
Mozilla	Web browser	Microsoft Explorer, AOL	<a href="http://www.mozilla.org">www.mozilla.org</a>
Eclipse	Integrated development environment	Microsoft Visual, IBM VisualStudio, Sun Forte	<a href="http://www.eclipse.org">www.eclipse.org</a>

### ONLINE RESOURCES

Open Source Institute, <http://www.opensource.org>

Source Forge, <http://sourceforge.net>

### FURTHER READING

Frank Hecker, “Setting Up Shop: the Business of Open-source Software”.  
<http://www.hecker.org/writings/setting-up-shop.html>

David Wheeler, “Why Open Source Software? Look at the Numbers”.  
[http://www.dwheeler.com/oss\\_fs\\_why.html](http://www.dwheeler.com/oss_fs_why.html)

Each of the major services is currently developing wireless, self-forming, self-healing Internet-based networks for the battlespace. The objective of these emerging network infrastructures is to enable mass integration of people, processes and applications on the battlefield. Distributed Instruments’ expertise is specifically focused on enabling large organizations such as the US Department of Defense to leverage emerging battlespace Internet infrastructure for mass integration.

Distributed Instruments webservice capabilities includes:

- Open-source webservices development
- Uniform resource name (URN) management
- Asynchronous and instant webservices
- Legacy data translation for “bolt-on” webservices

Using these capabilities, Distributed Instruments technology can enable legacy defense C4ISR systems to immediately and cost effectively participate in the emerging network infrastructures.