Open source isn’t about saving money, it’s about doing more stuff, and getting incremental innovation with the finite budget you have.

- Jim Whitehurst, CEO, Red Hat
The concept of sharing source code freely was standard long before software was developed and packaged for profit. Original software applications were developed exclusively for use with a specific set of hardware and seen not as stand-alone products, but as a portion of the overall package. Early software developers and organizations embraced the concept of sharing source code for the sake of speedy feedback and collaboration. However, early data showed that the large corporations employing or retaining these developers were unresponsive to their suggestions and the needs of customers. As a result, many disgruntled employees left their respective companies and developed applications that rivaled their commercial counterparts, knowing that their system architecture and necessary features meant certain success for these rogue developers. Many current large-scale OSS projects were started through this process, including UNIX, Linux, Sendmail and Apache.

Figure No.5 shows a brief historical timeline of the significant open-source developments.
Figure No. 5: Brief Historical Timeline of Significant Open-Source Events

1969
UNIX development starts at AT&T Bell Labs for the PDP-7

1984
Richard Stallman starts GNU project

1989
GNU General Public License (V1) is published

DEC., 1993
The UNIC OS, FreeBSD 1.0 is released

JAN., 1995
Apache Web server is started by programmers from the National Center for Supercomputing Applications (NCSA).

JULY, 2003
Mozilla Foundation is formed.

MAY, 2007
Dell announces it preloads Linux on its computers

1969
UNIX V7 is released, the grandfather of all extant UNIX systems

1979
Richard Stallman creates Free Software Foundation

1985
AUG., 1991
Linus Torvalds begins developing an operating system kernel

1993
MAR., 1994
Linux V1.0 kernel is officially released

1995
The term “open-source” is coined during a conference in Palo Alto, California; the Open-source Initiative is founded

1998
FEB., 1998
Sun Microsystems licenses Solaris as open-source.
UNIX:

During the early days of computing, large-scale commercial computers were developed by IBM and AT&T Bell Labs and other companies. These commercial computers each had distinct operating systems that were written specifically for a unique hardware profile—the software could not be run on multiple platforms. To eliminate the hardware profiling of software, many programmers started developing a new operating system that would reach beyond a computer’s original hardware. In 1969, as Jesus Gonzalez-Barahona describes it, “Kenneth Thompson, Dennis Ritchie and others at AT&T Bell Labs began developing a small operating system on a little-used PDP-7. The operating system was soon christened UNIX, a pun on an earlier operating system project called MULTICS (Multiplexed Information and Computing Service)”. The UNIX project was the most successful of all the attempts to develop a cross-platform operating system. One of the biggest catalysts for success was the collaborative nature of the project. The UNIX source code was freely shared among the many talented programmers who contributed to the project. Carolyn Kenwood of The MITRE Corporation credits a majority of the collaboration to a computer network.

After years of development, the seventh edition (V7) of UNIX was released in 1979. This version is described as “the grandfather of all extant UNIX systems.” Although UNIX had rapid success, the project was not without problems. Throughout the software’s history, no entity had tried to claim property rights to the source code until AT&T did in the early 1980s. In response, Stallman started out to produce a free version of UNIX One of the important aspects of this new project was that any individual could contribute to source code. As Kenwood tells it:

“This project, called GNU, allowed individual programmers, regardless of individual or commercial interests, to contribute to the development effort. GNU stands for ‘Gnu’s not Unix’. In the end, users were not charged for the operating system”.
From this project, the GNU GPL was developed, dictated by the following restrictions as stated by Kenwood:

☞☞☞

“Software licensed under GNU General Public License can be copied and distributed under this same license.

Products obtained and distributed under this license may be sold.

Users may alter the source code, but if they distribute or publish the resulting work, they must make the software available under the same licensing terms.

Ancillary technology can be developed and as long as such products do not include code licensed under the GNU General Public License, they need not be licensed or made available under the terms of the GNU General Public License”.

Stallman’s intentions were to ensure that UNIX-compatible software would remain free and catalyze more collaborative programming and development. In *GNU Manifesto*, he states:

“I consider that the golden rule requires that if I like a program I must share it with other people who like it. Software sellers want to divide the users and conquer them, making each user agree not to share with others. I refuse to break solidarity with other users in this way. I cannot in good conscience sign a nondisclosure agreement or a software license agreement”.

On a parallel plane, the Computer Science Research Group (CSRG) at the University of California at Berkeley built upon the proprietary UNIX system. David Wheeler, an expert in computer security and high-risk software systems, recounts, “*The academic community … developed a variant called the Berkeley Software Distribution (BSD)*”. Like GNU, the UNIX variant BSD was developed by a worldwide network of programmers and UNIX hackers who “helped debug, maintain and improve the system”. Continuing in the footsteps of GNU, BSD was distributed under the BSD License making BSD another open-source alternative to UNIX. However, despite
BSD’s open-source nature, each user needed the proprietary AT&T license to run parts of the core operating system and utilities that made BSD a usable system.

**LINIX:**

Arguably the quintessential open-source project is the Linux operating system. Linux is regarded as the closest competitor to Microsoft’s Windows operating system and receives continuous contributions from programmers worldwide. It was conceived in 1991 by Linus Torvalds, a student at the University of Helsinki. Dissatisfied with his school’s choice of the MINIX operating system, Torvalds decided to create a free operating system based on UNIX. Kenwood writes:

>“Linus Torvalds…created the Linux operating system and gave hackers his code so they could contribute to the development. Many programmers analyzed his code and wrote improvements that Linus incorporated into Linux. Linux grew and expanded into an advanced and powerful, multi-use operating system”.

March 1994 marked the first official release of the Linux kernel, the foundation of modern Linux distributions. According to Michael Godfrey and Qiang Tu of the Computer Science department at the University of Waterloo, this release “contained 487 source code files comprising over 165,000 lines of code”. However, the most notable aspect of this release was the maintenance methodology from that point forward. Two directions were formed to help carve out future products: developmental releases and stable releases. Developmental releases contain mostly untested and experimental code, while stable releases contain updates and are relative to the previous stable release. This process has led to many new distribution releases and millions of people using the Linux operating system. In addition, companies have been formed that sell distribution copies of Linux as well as support contracts. The most notable organizations are SuSE, RedHat, Ubuntu and Caldera.

The Linux operating system is becoming a huge competitor to Microsoft Windows in both the server and desktop arenas.
**OTHER MAJOR PROJECTS:**

The UNIX and Linux operating systems occupy most of the history of OSS. However, many projects have followed in their footsteps including the Apache Web server, one of the most recent successful OSS projects. Started in 1995 by Brian Behlendorf, Apache’s story parallels that of UNIX, in which frustrated employees left in order to create better software. While working at the National Center for Supercomputer Applications (NCSA), Behlendorf and several other employees became frustrated “in getting the NCSA staff to respond to their suggestions”. In an effort to implement fixes, Behlendorf and “six other pioneering developers decided to establish a mailing list to collect and integrate the patches to the NCSA server software”. Apache 0.8 was released in August 1995 and named after the extensive use of “patches”. The server software continued to grow, mainly due to a lack of equivalent competition. A Netcraft survey done in November 2000 found that 59.7% of websites used the Apache Web server while Microsoft’s IIS only had 20.2% of the market.

Sendmail is another example of open-source innovation and success. The OSS project was “originally developed in the late 1970s by Eric Allman, a graduate student in computer science at the University of California at Berkeley”.

Faced with the incompatibility of the two networks on campus, BerkNet and Arpanet, “Allman developed… a program called ‘Delivermail’, which provided a way to greatly simplify the addressing problem”.

Two years later in 1981, the software was released as ‘Sendmail’ which “soon became the standard method of routing email on the Arpanet”. Sendmail continued a successful path and in 2000, “the program was estimated to handle about 75% of all Internet email traffic”.

A multitude of OSS projects have started, many becoming so successful that they compete in or even dominate a market filled with proprietary products and profit seeking corporations, such as the Apache Web server. Although the motivation and inspiration for programmers to participate in these projects vary greatly, one of the driving forces behind OSS development is dissatisfaction with current proprietary
software on account of service and or cost. This has led to the development of a comparable OSS project for most, if not all, commercial applications. OpenOffice.org, based on the StarOffice suite started in the 1980s, is an open-standard, XML-based office productivity suite that compares to Microsoft’s Office in terms of features and capabilities. In the operating system arena, Linux has been a direct competitor of Windows, both in desktops and servers. MySQL is another open-source project that is based on the Structured Query Language (SQL) and is a direct competitor of MS SQL.

Finally, Microsoft’s Internet Explorer is rivaled by its open-source competitor, Firefox, developed by Mozilla. Although there is constant debate as to whether Microsoft’s software or comparable OSS is better, many OSS projects have succeeded in developing large market share.

2.1.1 - Need of Open Source Library Software:

There is abundant availability of library softwares. Library Authority has to choose one of them which serve their service needs, in terms of economics and end users’ requirements. In this critical situation, open source library software fulfills, among other requirements, the shoestring budget scenarios of most, if not all libraries.

Problems Faced by Libraries:

☞ Severe Budget Cuts: Libraries are facing resource crunch and consequent funding problems which adversely affects the smooth functioning of libraries services and operations. Escalations in input cost of libraries viz. library materials, books, periodicals and services act a serial constraint on the already abridge budget allocations. In short development of libraries is a difficult proposition in the present circumstances and new noble projects imbibing new technologies and innovation have to be postpone or abandoned because such new projects absorb huge funds which libraries can ill afford.

☞ Increased Demand for Services: Libraries are changing their pattern of services and collection through the use of information and communication technology.
However, the implementation of state of the art technology needs huge funds. Since budget cuts and ceilings are the order of the day, year on year basis, development of libraries suffers and as a result the quality and quantity of services to end users are adversely affected.

☞ ☞ ☞

Lack of Adequate Staffing: Libraries are facing problems of inadequate staff once again budget cuts and ceiling have freeze levels of human resources which in terms of placed an embargo on recruitment. Of course, non-availability of professionally qualified library personal is an additional factor affecting quality recruitment where there are vacancies. However, the library budget is dominated by human resource cost which accounts for nearly 50% of the total library budget thus dissuading management and authorities from adding to human resources.

Given the above budget scene for libraries, libraries adopting open source software are better place because they are able to reduce the operational cost of various library services and operations. Once again Library automation and digitization requires a lot of funds and OSS can help through its cost effective characteristics. Open source softwares are available free for all and can be modified and tailor made to suit the varied needs of different libraries. Further, regular updates are possible and bug support mechanism helps in smooth functioning of open source softwares in libraries.

2.1.2 - Open Source Software Versus Proprietary Software:

The open source comes with the source code whereas proprietary software comes without the source code. The major difference between OSS and proprietary software is the commercial support. OSS is low cost, whereas proprietary software is costly. The proprietary software can be regularly and easily updated. The open source software involves erratic updates. The advantage of proprietary software is professional support and training available and the disadvantage is having closed standards that obstruct further development.
The advantage of OSS is no license fee and having open standards that facilitates integration with other systems. However the disadvantage is lack of professional support.

Table No. 1: Closed Source vs Open Source

<table>
<thead>
<tr>
<th>Source Code</th>
<th>CLOSED SOURCE</th>
<th>OPEN SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hidden from public</td>
<td>Open to public for modification</td>
</tr>
<tr>
<td>Product State</td>
<td>Compiled executable</td>
<td>Pre-compiled binaries along with source code</td>
</tr>
<tr>
<td>Product Price</td>
<td>License fee</td>
<td>Free</td>
</tr>
<tr>
<td>Subscription</td>
<td>Fee</td>
<td>Free</td>
</tr>
<tr>
<td>Documentation</td>
<td>Fragmented</td>
<td>Comprehensive relatively much easier to use</td>
</tr>
<tr>
<td>End User Support</td>
<td>One-stop shop</td>
<td>Foss Community, wikis, forums</td>
</tr>
<tr>
<td>OS Distribution</td>
<td>Windows ME, Windows XP,</td>
<td>Ubuntu, Fedora, RedHat</td>
</tr>
</tbody>
</table>

2.1.3 - Open Source Software vs Free Software:

Term ‘Open Source’ creates an ambiguity with the term ‘Free Software’. Open Source is also known as Free / Open Source Software (FOSS) or Free / Libre / Open Source Software (FLOSS). Open Source software is freely available to all and it may be used, copied, studied, modified and redistributed without any restriction whereas free software is free to download, install and use but cannot be modified. Further development and assistance is fee based in free softwares. Majority of free softwares is for advertisement only. In comparison, Open Source highlights the source code which is viewable and modified.
2.1.4 – Development and Support of Open Source Software:

There are three broad models for open source software development and support:

☞ **Volunteer Community**: A large proportion of open source software is developed by a community of skilled people who usually communicate online. In this model, there is no specific corporation managing the development process. Support is available through the members of the community, who have forums and other feedback mechanisms to receive requests from users. There is generally no service level agreement available from the community. Popular packages such as the Apache web server and the Linux operating system have been developed using this model.

☞ **Corporate - Backed Community**: Some commercial organisations provide support for open source software. The commercial organisation may choose to create its own community to develop the open source software or they may choose to leverage off an existing product created by a volunteer community. The commercial organisation usually provides support to a defined service level agreement. More than one organisation can provide support for a product, leading to competition based on the quality and price of the service. For example, Oracle’s and IBM’s web servers are both based on the community-developed Apache.

“*The word “free” in our name does not refer to price; it refers to freedom. First, the freedom to copy a program and redistribute it to your neighbors, so that they can use it as well as you. Second, the freedom to change a program, so that you can control it instead of it controlling you; for this, the source code must be made available to you.***

– Free Software Definition by Richard Stallman


☞ **Commercial Open Source:** Some open source software is developed or supported by a single corporation. Sun Microsystems (now owned by Oracle) provides the OpenSolaris operating system under this model.

## 2.1.5 - Benefits of Open Source Software:

Open source software has a number of potential benefits. These benefits are not applicable in every instance; however, they can be seen as general characteristics of open source software. Some of these benefits can be realised only when agencies contribute back to the community.

☞ **Usually has No Upfront Payment:** The lack of upfront payment may seem to benefit agencies financially; however, as with all software, agencies should consider the total cost of ownership, including all support services that will be required to operate the software over its lifespan.

☞ **Encourages a competitive market for support services:** Because the source code is available, it is possible for any software organisation to provide support for an open source product. In addition, customers are able to support the software themselves.

☞ **Encourages a Collaborative Approach:** Open source software encourages an open exchange of ideas, where any user of the software can contribute ideas to improve it. This tends to promote a collaborative approach that may foster innovation.

☞ **Places Fewer Restrictions on the Users of the Software:** Most open source software licences place fewer restrictions on the users of the software and emphasize respect for the privacy of the users. However, agencies should ensure that they understand the obligation for reciprocity that is included in many open source licences.

☞ **Provides the Opportunity for Users to Take Direct Control of the Maintenance and Support of the Software:** This may be a benefit to agencies that possess the appropriate skill base.
**Allows the Opportunity to Try the Software Before Committing to it:** This will enable agencies to test the viability of the software before fully committing to it.

**May Reduce Vendor Lock-In:** As the source code is publicly available, most licences will allow any individual or group to further develop the software without the obligation to support other users, even if the original community discontinues development. Commercial organisations may provide support for an open source package, if there are enough users willing to pay for that service.

**Allows Users to View and Modify the Source Code:** The ability of users to scrutinize and change the source code of open source software may lead to increased stability and security. It also allows agencies to tailor the software to their own needs.

**Allows Users to Take Advantage of the Improved Functionality of New Releases More Rapidly:** Many new open source software communities follow the maxim of release early, release often’, meaning that users can quickly gain extra functionality for the software.

**Increases Interoperability:** Many open source software packages use open standards, which tend to lower the costs of integration and improve interoperability.

**Usually is Modular:** Open source software packages are generally modular, which means that changes to one part of the source code is less likely to affect the rest of the software package.
2.3 - SWOT Analysis of Open Source Software:

**SWOT**

---

**Strengths**
- Less reliance on a single vendor.
- Fits into institutional and individual budgets.
- Lower costs.
- Development community size.
- Quick release rate.
- Freedom to modify given to all stakeholders.

**Weaknesses**
- Lack of technical support.
- Microsoft-acclimated users.
- Lack of open source applications.
- Hard to originate.
- Not a Revenue model.

**Opportunities**
- To reach society at large.
- Replace legacy software with open source.
- Reuse older legacy hardware.
- Ability to choose support vendor.
- Opportunity to develop specialized applications.

**Threats**
- Lack of training.
- Lack of patches, updates, and documentation.
- High life cycle costs.
- Lack of compatible applications.
- Piracy – lack of Intellectual Property Protection.
N.B.: The OSS assumes importance for many reasons. Among the important reasons were severe budget cuts in various libraries across the world, increased demand for services volume range, mix, speed and duration and lack of adequate trained staff in library. However, the SWOT analysis is in order because OSS is not a panacea for all ills.

2.3.1 – **Strengths:**

2.3.1.1 - **Less Reliance on a Single Platform Vendor:**

The open source community also provides non proprietary tools that allow agencies to move data from one form to another as open source vendors adhere to open standards that do not favor any particular platform.

2.3.1.2 - **Lower Costs:**

Open source software typically has lower acquisition costs than commercial software solutions. Agencies with limited available funds may be able to take advantage of the cost savings. Most open source software packages have low initial acquisition and update costs. Some opponents of open source will cite the example that replication and fault-tolerance functions for open source database packages cost money and require a certain level of technical skill. However, the majority of the surveyed agencies that use open source do not have the need for such advanced features and functions. Open source software packages often operate with less intensive hardware requirements, which allow agencies to leverage older systems that may otherwise have no daily operational use.

2.3.1.3 - **Development Community Size:**

Open source, as a whole, has a large community of developers. SourceForge has more than 130,000 registered open source products and more than 1.4 million registered users. This web site provides an open source software repository, open source development tools and hosts open source development projects.
2.3.1.4 - Quick Release Rate:

Within certain open source categories, releases and patches are made more frequently than in commercial software.

2.3.2 – Weaknesses:

2.3.2.1 - Lack of Technical Support:

Users want someone accountable for the software they use. It is important to them to be able to call the software vendor for help when needed. They do not want to have to rely on a community of developers to help resolve any issues.

2.3.2.2 - Microsoft Accustomed Users:

Computer users are less familiar with open source software than with the commercial – often Microsoft-based – equivalents. Often commands and pull-down menus are in different locations and use different terminology.

This means that agencies purchasing software are limited in their ability to find open source solutions that meet their requirements. In addition, some software is only open source for the base package. These packages require that additional funds be spent for advanced features.

2.3.2.3 - Hard to Originate:

Open source software offers the greatest power when it has an open source community supporting it. Until that time, the product is dependent upon the initial developers for enhancements/upgrades. It takes time to build the level of interest in an open source application and have a correspondingly large user community. While it is relatively easy to write code and promote it as open source, it is difficult to originate
and grow the open source application to the point of having a supporting open source community.

2.3.3 - Opportunities:

2.3.3.1 - Replace Legacy Software with an Open Source Solution:

Open source forums such as SourceForge have a large registered user base. This user population presents a large number of potential developers from which to find others with an interest in collaborating on a software project.

2.3.3.2 - Reuse Older Legacy Hardware:

Hardware that is no longer able to run current commercial software. This hardware can often be used to run an open source solution.

2.3.3.3 - Ability to Choose Support Vendor:

There are number of open source support vendors available to choose from. This provides open source adopters with choices based upon price, hours of support and product rather than being locked into a particular vendor.

2.3.3.4 - Opportunity to Develop Specialized Applications:

Chances of developing specialized applications are more in open source software.

2.3.4 – Threats:

2.3.4.1 - Lack of Training:

Open source software typically does not come with the same level of training available as a commercial product. Many commercial products come bundled with
user manuals. Often open source is supported through blogs (a user-generated website where entries are made in a journal style with questions and answers).

### 2.3.4.2 - Lack of Patches, Updates and Documentation:

Lack of patches, updates and documentation is similar to the lack of training. As well as having limited training options, some of the more specialized open source applications may also have limited patch, update and documentation support.

### 2.3.4.3 - High Life-Cycle Costs:

Specialized or customized open source solutions may not have a wide installation base. If an agency requires new or modified functionality, they may be required to implement it themselves. There is a cost in time and effort to the agency to do this that may be higher than that associated with a commercial application or more widely used open source solution.

### 2.3.4.4 - Lack of Compatible Applications:

If the organization’s standard operating environment for hardware and software differs from the one used for the new open source application, the new system will be incompatible with the agency’s environment and support infrastructure.