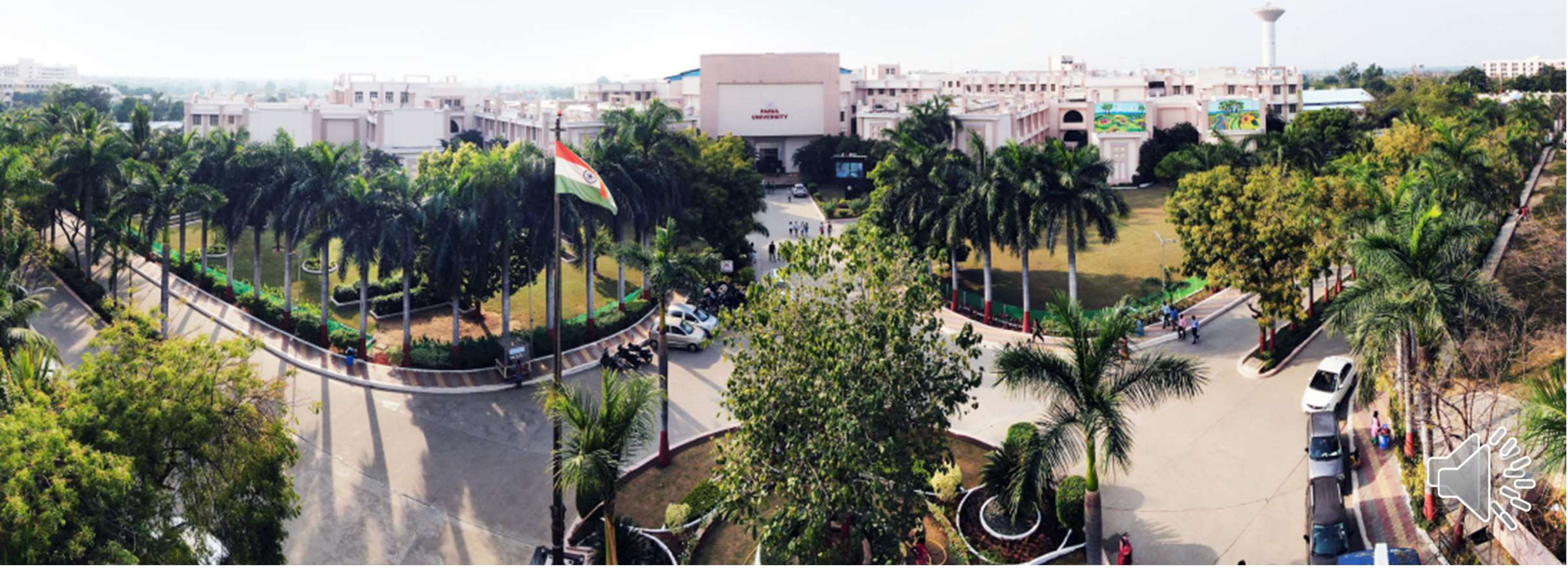
Open-Source Software

Prof. Asiya Durani, Assistant Professor
Computer Science and Engineering



CHAPTER-2

Open-Source Principles and Methodology

Open-Source History

- Open source software (OSS) started out as an assumption without a name or a distinct alternative in the thirty years between 1970 and 2000.

- It has developed into an advanced movement that has given rise to some of the most reliable and well-liked software programs ever created.

Open-Source History

- The open source software (OSS) movement has revolutionized software development since 1998. The revolution in this quickly evolving profession, however, truly has roots that go back at least thirty years.

- Many people are already familiar with the alternative software distribution mechanism known as OSS.

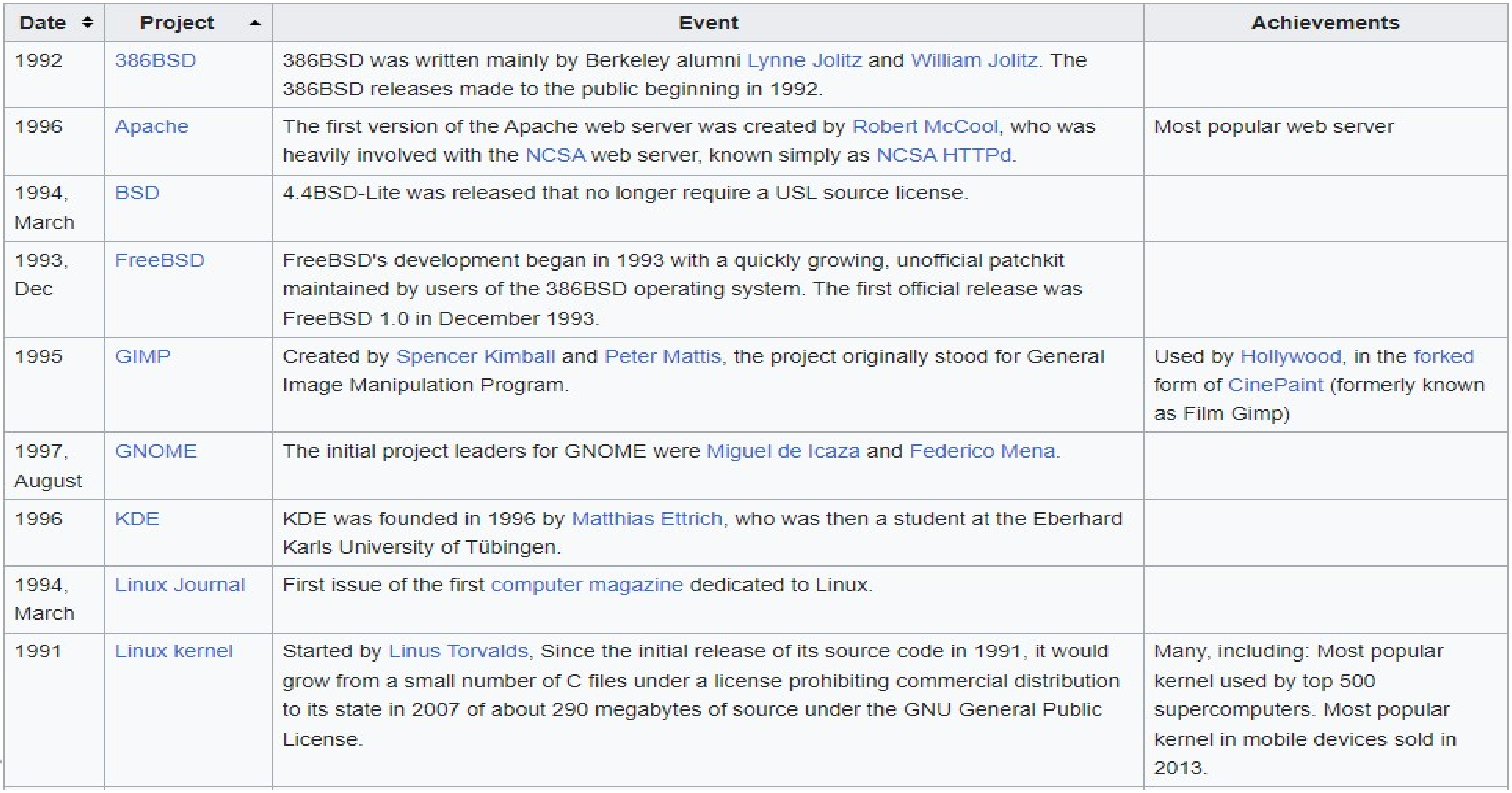
Who started open source software?

- The A-2 (Arithmetic Language v2 system) was the first piece of free open source software created in 1953 by Remington Rand's UNIVAC company, and it was made available to customers with the source code. Additionally, they were encouraged to send back their upgrades.
- The first example of free and open-source software is believed to be the A-2 system, developed at the UNIVAC division of Remington Rand in 1953, which was released to customers with its source code. They were invited to send their improvements back to UNIVAC.

Parul® University

DIGITAL LEARNING CONTENT

Open-Source History Table



Date	Project	Event	Achievements
1992	386BSD	386BSD was written mainly by Berkeley alumni Lynne Jolitz and William Jolitz. The 386BSD releases made to the public beginning in 1992.	
1996	Apache	The first version of the Apache web server was created by Robert McCool, who was heavily involved with the NCSA web server, known simply as NCSA HTTPd.	Most popular web server
1994, March	BSD	4.4BSD-Lite was released that no longer require a USL source license.	
1993, Dec	FreeBSD	FreeBSD's development began in 1993 with a quickly growing, unofficial patchkit maintained by users of the 386BSD operating system. The first official release was FreeBSD 1.0 in December 1993.	
1995	GIMP	Created by Spencer Kimball and Peter Mattis, the project originally stood for General Image Manipulation Program.	Used by Hollywood, in the forked form of CinePaint (formerly known as Film Gimp)
1997, August	GNOME	The initial project leaders for GNOME were Miguel de Icaza and Federico Mena.	
1996	KDE	KDE was founded in 1996 by Matthias Ettrich, who was then a student at the Eberhard Karls University of Tübingen.	
1994, March	Linux Journal	First issue of the first computer magazine dedicated to Linux.	
1991	Linux kernel	Started by Linus Torvalds, Since the initial release of its source code in 1991, it would grow from a small number of C files under a license prohibiting commercial distribution to its state in 2007 of about 290 megabytes of source under the GNU General Public License.	Many, including: Most popular kernel used by top 500 supercomputers. Most popular kernel in mobile devices sold in 2013.

Open Source Initiatives

- The modern free software movement and the Open Source Initiative share a common history with Unix, Internet free software, and hacker culture. However, their fundamental ideologies are different, with the free software movement placing more emphasis on the ethical considerations of software than their open source counterparts.
- According to founding member Michael Thiemann, the Open Source Initiative chose the phrase "open source" in order to "dump the moralizing and confrontational attitude that had been associated with 'free software'" and promote open source concepts on the basis of "pragmatic, business-case grounds."

Open Source Initiatives

- Open source software is defined by the Open Source Initiative as software that is distributed with source code that can be read by humans, giving users the freedom to run, examine, change, improve, and modify the code for any reason.

The following is its mission statement:

- The Open Source Initiative (OSI) is a non-profit organization with a global reach that was created to spread awareness of and support the advantages of open source software as well as to forge connections between various open source community groups.

Cont....

- Software development using open source harnesses the potential of distributed peer review and process openness. Better quality, more dependability, more flexibility, cheaper costs, and an end to exploitative vendor lock-in are the promises of open source.

- As a standards organization, one of our most significant tasks is to keep the Open Source Definition current for the community's benefit.

Cont....

- Developers, users, businesses, and governments can coordinate open source cooperation around the Open Source Initiative Approved License trademark and program.

- We can see the outstanding characteristics of OSI in the aforementioned statement: it is a community-based culture that encourages sharing and support, as well as inclusion of the original source and provider. Similar to this, social media supports the idealistic attitude of gift economy growth by fostering connections and sharing.

What is Open Standards Principles

- Open Standards may use license clauses to prevent standard subversion through embrace-and-extend strategies.
- The license could impose conditions on the creation and distribution of software that is compatible with the extensions as well as the publication of reference materials.
- It might not forbid the application of augmentations.
- A specification alone is not an Open Standard. The openness of the standard is a result of its guiding principles as well as the way it is offered and used.

List of Open Standards Principles

1. Availability : Open Standards are available for all to read and implement.

2. Maximize End-User Choice : Open Standards create a fair, competitive market for implementations of the standard. They do not lock the customer in to a particular vendor or group.

3. No Royalty : Open Standards are free for all to implement, with no royalty or fee. Certification of compliance by the standards organization may involve a fee.

4. Extension or Subset - Implementations of Open Standards may be extended, or offered in subset form. However, certification organizations may decline to certify subset implementations, and may place requirements upon extensions (see Predatory Practices).

List of Open Standards Principles

5. No Discrimination - Open Standards and the organizations that administer them do not favor one implementer over another for any reason other than the technical standards compliance of a vendor's implementation. Certification organizations must provide a path for low and zero-cost implementations to be validated, but may also provide enhanced certification services.

6. Predatory Practices - Open Standards may employ license terms that protect against subversion of the standard by embrace-and-extend tactics. The licenses attached to the standard may require the publication of reference information for extensions, and a license for all others to create, distribute, and sell software that is compatible with the extensions. An Open Standard may not othewise prohibit extensions.

List of Practice of Open Standards Principles

1. Availability: Open Standards are available for all to read and implement. Thus:

- The ideal scenario is for the reference implementation and the standards document to be freely downloadable over the Internet.
- Without undue effort, any software project should be able to finance a copy. The price shouldn't be significantly higher than that of a college textbook.
- Any party must not be prohibited by licenses affixed to the standard documentation from implementing the standard using any kind of software license.
- It is excellent practice for software reference platforms to have licenses that are compatible with all types of software licensing, including both proprietary and free software (open source). Regarding licensing constraints that might be suitable for a software reference platform, read Predatory Practices instead.

Parul® University

List of Practice of Open Standards Principles

2. Maximize End-User Choice

A fair, competitive market for standard implementations is created through open standards. Thus:

- They must permit numerous implementations by commercial enterprises, academic institutions, and government initiatives.
- They must support a variety of pricing options, from extremely expensive to free.

List of Practice of Open Standards Principles

3. No Royalty

There are no fees or royalties associated with the implementation of open standards. There can be a charge for the standards organization to certify compliance. Thus:

- Standards-related patents must be licensed without restrictions and without any additional fees.
- Self-certification should be inexpensive or free as part of certification programs, but they may also offer more expensive options with better branding.

List of Practice of Open Standards Principles

4. No Discrimination

Other than a vendor's implementation's adherence to technical standards, Open Standards and the organizations that administer them do not favor one implementer over another. The validation of low- and no-cost implementations must be made possible by certifying bodies, who may also offer improved certification services. Thus:

- The standards body could choose to use a license similar to the Sun Industry Standards Source License for the reference implementation that goes along with the standard documentation. For any standard extensions, the Sun Agreement mandates release of a reference implementation (not the actual commercial implementation). A standards group can now actively protect interoperability without restricting innovation thanks to this.

Parul® University

List of Practice of Open Standards Principles

5. Extension or Subset

Open Standards implementations can be expanded upon or made available in subset form. Although extensions may be subject to requirements, certification agencies may refuse to certify subset implementations (see Predatory Practices).

List of Practice of Open Standards Principles

6. Predatory Practices

Open Standards may use license clauses to prevent standard subversion through embrace-and-extend strategies. The license could impose conditions on the creation and distribution of software that is compatible with the extensions as well as the publication of reference materials. It might not forbid the application of augmentations.

Open Source Methodology

- There has always been methodology since when there has been a need to create solutions to problems.
- In the case of open source software development, methodology has been essential even though not very visible in creating successful development processes.
- The Linux and Apache projects are but a few of success Stories of open source development projects.

Open Source Methodology

- A collaborative approach to software development known as "open source software methodology" entails making a software project's source code available to everyone for free.
- This indicates that anybody can access the source code and use it, edit it, and distribute it as long as they follow the guidelines of the project's open source license.

- The following are some of the main concepts of open source software methodology:

Open Source Methodology

1. Open Access: Everyone has access to the source code, which enables users to comprehend how the program functions and alter it to suit their needs.
2. Redistribution: The software and any modifications made by users may be shared with others. This encourages a form of community-driven development.
3. Collaboration: Developers are urged to work together so that they can collaboratively improve the product by adding to the codebase.
4. Transparency: The development process is open and transparent, enabling users to see how the software evolves over time and how decisions are made.
5. Meritocracy: Typically, contributions are judged on their merits rather than the history of the author, promoting an inclusive environment for developers to engage.

Open Source Methodology

6. Community-Driven: Open source projects are often developed and maintained by a community of volunteers or organizations who share a common interest in the software.
7. Licensing: Open source projects are governed by licenses that define the terms and conditions under which the software can be used, modified, and distributed.

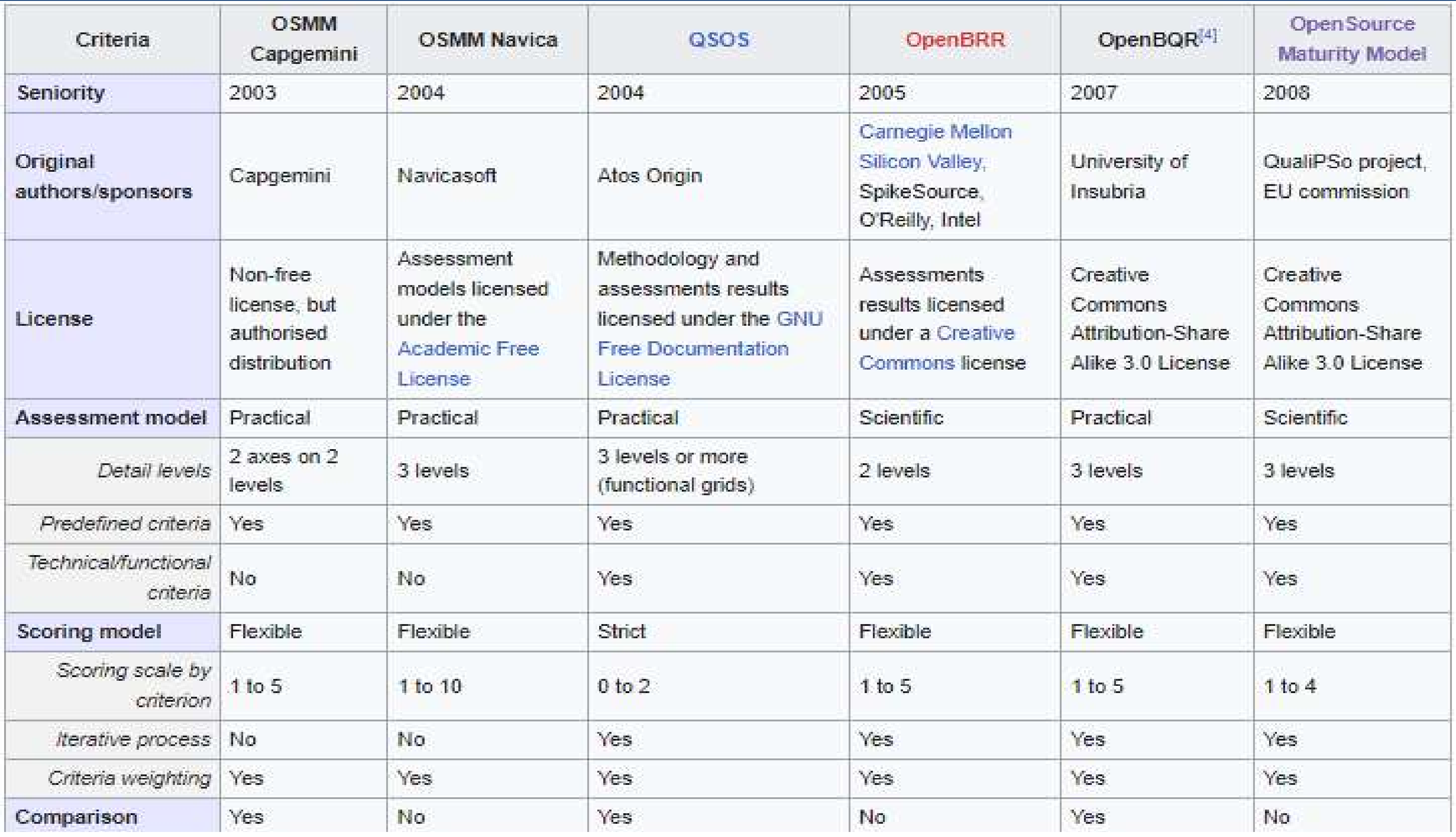
Numerous commonly used software products, including the Linux operating system, the Apache web server, the Mozilla Firefox web browser, and the Android mobile operating system, to mention a few, have been developed using open source methodologies. Open source development's collaborative approach has shown to be effective in promoting innovation and creating high-caliber software that benefits a wide user base.

List of Open Source Methodology

1. Open Source Maturity Model (OSMM) from Capgemini
2. Open Source Maturity Model (OSMM) from Navica[2]
3. Open Source Maturity Model (OSSMM) by Woods and Guliani[3]
4. Methodology of Qualification and Selection of Open Source software (QSOS)
5. Open Business Readiness Rating (OpenBRR)

Parul® University

Comparison Chart : The comparison process is defined by the methodology



Criteria	OSMM Capgemini	OSMM Navica	QSOS	OpenBRR	OpenBQR[4]	OpenSource Maturity Model
Seniority	2003	2004	2004	2005	2007	2008
Original authors/sponsors	Capgemini	Navicasoft	Atos Origin	Carnegie Mellon Silicon Valley, SpikeSource, O'Reilly, Intel	University of Insubria	QualiPSo project, EU commission
License	Non-free license, but authorised distribution	Assessment models licensed under the Academic Free License	Methodology and assessments results licensed under the GNU Free Documentation License	Assessments results licensed under a Creative Commons license	Creative Commons Attribution-Share Alike 3.0 License	Creative Commons Attribution-Share Alike 3.0 License
Assessment model	Practical	Practical	Practical	Scientific	Practical	Scientific
Detail levels	2 axes on 2 levels	3 levels	3 levels or more (functional grids)	2 levels	3 levels	3 levels
Predefined criteria	Yes	Yes	Yes	Yes	Yes	Yes
Technical/functional criteria	No	No	Yes	Yes	Yes	Yes
Scoring model	Flexible	Flexible	Strict	Flexible	Flexible	Flexible
Scoring scale by criterion	1 to 5	1 to 10	0 to 2	1 to 5	1 to 5	1 to 4
Iterative process	No	No	Yes	Yes	Yes	Yes
Criteria weighting	Yes	Yes	Yes	Yes	Yes	Yes
Comparison	Yes	No	Yes	No	Yes	No

Open source philosophy and software freedom

The ideals and objectives of the open source software movement are underpinned by two notions that are intimately tied to one another: software freedom. They place a strong emphasis on the concepts of openness, teamwork, and user rights within the software development community. Let's delve deeper into each of these ideas:

Open Source Philosophy

Open source philosophy is a set of guiding principles that promotes the idea of open access to software source code, encouraging collaboration and community-driven development. The core tenets of the open source philosophy include:

1. Transparency: Anyone can learn how the program is implemented and how it works by accessing the source code for open source projects. This openness guarantees that the software is free of dangerous code and hidden features.

2. Community Collaboration: The combined contributions of a varied community of developers, users, and enthusiasts enable open source development to flourish. Through the use of a collaborative process, a variety of viewpoints, suggestions, and contributions can be incorporated into the creation of the software.

Open Source Philosophy

3. Meritocracy: Open source projects frequently use a merit-based structure, where contributions are judged on their technical merits as opposed to their affiliations or hierarchical positions. This encourages the participation and contribution of developers from all backgrounds in an inclusive atmosphere.

4. Continuous Improvement: Open source projects have a broad and diverse community of contributors, which helps them develop quickly. New features are introduced, bugs are rapidly detected and corrected, and the software is flexible enough to meet evolving customer demands.

Software Freedom

In terms of using, modifying, and sharing software, users need to have a specific set of fundamental rights. This is known as "software freedom." These liberties, which are often protected by open source software agreements, consist of:

1. Freedom to Use: Users are free to use the software however they see fit.
2. Freedom to Study: Access to the source code allows users to research how the software functions.
3. Freedom to Modify: Users have the freedom to modify the source code to meet their demands or resolve problems.
4. Freedom to Distribute: Users are free to distribute the program and any customized versions they make.

Software Freedom

Open source software licenses guarantee that users have control over the program they use by offering certain freedoms.

Contrast this with proprietary software, which imposes limitations on users' rights and frequently prevents them from seeing the source code or making changes to it.

What is Open-Source Software Development?

- Software is developed through a collaborative process known as "open-source software development," which allows users to get it for free. The word "open-source" denotes that the software's source code is available for anyone to read, alter, or distribute. This architecture promotes openness, stimulates community involvement, and enables software engineers from all around the world to contribute and enhance it.

What is Open-Source Software Development?

- The process of planning, building, testing, and maintaining computer programs and applications is known as software development. Gathering requirements, planning, coding, testing, debugging, and deploying the software are some of the stages involved.
- Building desktop applications, web applications, mobile applications, system software, and other types of applications are only a few of the many uses for software development. Individual developers, small teams, or major corporations can all take on the task while using different approaches like Agile, Waterfall, Scrum, etc.

Key characteristics of open-source software development include:

1. Source Code Access
2. Collaboration and Community
3. Licensing
4. Continuous Improvement
5. Peer Review
6. Diverse Uses
7. Community Support

Software licenses

Software licenses are contracts that regulate how it may be used, distributed, and modified. They specify the guidelines under which users may use the software. Proprietary licenses and open-source licenses are two of the most common forms of software licenses.

- Proprietary Licenses: Redistribution, modification, and access to the source code are all prohibited by the terms of proprietary software licenses. The majority of the time, users must pay for the software, and its usage is constrained.
- Open-Source Licenses: Users are free to access, alter, and redistribute software under an open-source license. Different open-source license types exist, some with more liberal terms and others with stricter guidelines to protect the open-source status of derivative works.

Copyright vs. Copyleft

- A copyright is a legal privilege that gives the author of a unique work (including software) the sole authority to control how it is used and distributed.
- It offers defense against unauthorized duplication or dissemination of the work without the author's consent.
- Contrarily, the idea of Copyleft is frequently connected to open-source licenses. It is a tactic to guarantee that works derived from open-source software projects continue to be open-source.
- Copyleft licenses stipulate that in order to maintain the software's openness and encourage the community to share changes, if you modify it and distribute it, you must do so under the same Copyleft license.

Patents

- Patents are exclusive rights given to inventors for a particular procedure or innovation. Software patents can be contentious in the domain of software.
- Some contend that software shouldn't be patented because doing so could restrict innovation and result in a patent war in which big businesses gather software patents for either defensive or offensive objectives.
- On the other hand, proponents contend that software patents promote R&D spending and protect inventors' rights.

Zero Marginal Cost

- Zero marginal cost in the context of digital commodities, such as software, refers to the occurrence where the cost of manufacturing and dispersing new units of the product approaches zero.
- Once the original work is complete, it is inexpensive to make extra copies or distribute the software online.
- This quality, which enables developers to disseminate their work without incurring large production costs, is a major factor in the popularity of free and open-source software.

Income-Generation Opportunities

Despite the fact that open-source software can be downloaded for free, there are a number of ways for developers and organizations to make money such as:

- Selling Support and Services: Companies can provide support, maintenance, and customization services to users of open-source software for a fee.

- Dual Licensing: A copyleft license and a proprietary license are two of the different licenses that some open-source projects provide their software under. If businesses want to get around the copyleft requirements, they can pay to use the proprietary version.

Income-Generation Opportunities

- Crowdfunding and Donations: Open-source software creators frequently get help from the community in the form of direct donations or support from crowdfunding websites.

- Commercial Extensions and Add-ons: Businesses can create and market paid add-ons or extensions that improve the functionality of open-source software.

Internationalization

- The process of designing and developing software in a way that makes it easily adaptable to multiple languages, countries, and cultures is known as internationalization, which is frequently abbreviated as "i18n" (due to the 18 letters between the 'i' and 'n').

- Text strings, date formats, currency symbols, and other localizable components are separated from the code in this process, and they are then saved in resource files.

- Making software more internationalization-friendly makes it simpler to produce localized versions (translations) for diverse target markets, improving accessibility and usability for a worldwide audience.

What Is A License?

- The creation of free and open-source software (FOSS) is made easier by open source licenses. Laws governing intellectual property (IP) prevent the sharing and modification of creative works.
- Free and open-source software licenses take advantage of these already-existing legal frameworks to give freedoms that encourage sharing and cooperation. The rights to use the software, view the source code, alter it, and share the modifications are granted to the recipient.
- These licenses apply to software for which the creation of modifications may require the source code. They also encompass circumstances in which there is no distinction between the executable program sent to end users and the source code.
- Hardware, infrastructure, beverages, books, and music are all subject to open-source licenses.

What Is A License?

- Permissive and Copyleft licenses fall within the two major categories of open-source licensing. In academia, permissive licenses first appeared. With specific restrictions, they grant the right to change and distribute.
- These academic licenses typically provide for a disclaimer of warranties as well as acknowledgement to the original authors.
- The free software movement serves as the source of Copyleft licensing.
- The rights to change and distribute are also granted by Copyleft, which also mandates attribution and disclaims warranties.
- The distinction is that reciprocity is required by Copyleft.
- Any derivative works must be distributed under a Copyleft license and include the source code.

How to Create Own Licenses

Steps to create License:

1. Determine Your Objectives: Clearly outline the purpose and objectives of your license. Decide what rights you want to grant to users and what restrictions you want to impose.
2. Research Existing Licenses: Familiarize yourself with existing open-source licenses and proprietary licenses to understand how they are structured and what provisions they include.
3. Choose a License Model: Decide whether you want to create a permissive license (allowing broad use) or a copyleft license (requiring derivative works to be licensed under the same terms).

Cont....

4. Draft the License: Write the license terms in clear and unambiguous language. Cover aspects such as usage, distribution, modification, warranties, liability, and attribution.

5. Seek Legal Advice: Consult with a legal expert to review your license and ensure it complies with local laws and is suitable for your specific use case.

Important FOSS Licenses (Apache, BSD, GPL, LGPL)

1. Apache License 2.0: A permissive license that allows users to use, modify, distribute, and sublicense the software. It requires users to retain copyright notices and disclaimers and provides a clear patent grant.

The key features of the Apache License 2.0 are:

- Permissive: The license offers users a wide range of usage rights without imposing stringent copyleft limitations.
- Patent Grant: It comes with a clear patent grant, giving customers the assurance that they won't get into problems with the original authors about patent-related matters.
- Notice and Disclaimer: When redistributing the software, users must keep the copyright notices, licensing conditions, and disclaimers.

Important FOSS Licenses (Apache, BSD, GPL, LGPL)

2. BSD License: Another permissive open-source license used for different software projects is the BSD License, often known as the Berkeley Software Distribution License. Under specific restrictions, the BSD License permits users to freely use, modify, and distribute the software. The original 3-clause BSD License and the updated 2-clause BSD License are the two primary variations.

The key features are:

- Permissive: The BSD License is permissive, just like the Apache License, granting users extensive freedom to use the program without imposing strict copyleft obligations.
- Redistribution Clause: The three-clause BSD License has a clause requiring the redistribution of the license text and copyright notice when distributing the software.

Important FOSS Licenses (Apache, BSD, GPL, LGPL)

3. GNU General Public License (GPL): The strong copyleft GNU General Public License (GPL) strives to advance software freedom and the open-source nature of software. Any modifications or derivative works based on a software that is licensed under the GPL must also be distributed under the GPL.

The key features are:

- Copyleft: The GPL stipulates that any modifications and derivative works must be licensed in accordance with the same GPL rules.
- Availability of Source Code: GPL-licensed software must include the source code or make a documented promise to do so upon request.
- Concerns about compatibility: Projects that use GPL code in their codebase may also need to be distributed under the GPL.

Important FOSS Licenses (Apache, BSD, GPL, LGPL)

4. GNU Lesser General Public License (LGPL):

The GPL has been changed and is now known as the GNU Lesser General Public License (LGPL). In order to enable more flexible usage in both open-source and proprietary projects, it is primarily intended for libraries and shared software components.

The key features are:

- Lesser Copyleft: The LGPL enables developers to utilize LGPL-licensed libraries in proprietary software by allowing linking with non-GPL-compatible software.
- Source Code Availability: The LGPL demands, like the GPL, that the library's source code be distributed or that a written promise to do so be made.

Copyrights and copy lefts

- Copyright is a legal right that grants exclusive ownership to the creator of an original work, such as software, music, literature, etc. It provides the copyright holder the authority to control how the work is used, distributed, and modified.

- Copyleft is a concept associated with certain open-source licenses, notably the GPL. It ensures that derivative works based on copyleft-licensed software must also be distributed under the same copyleft license. In this way, copyleft licenses aim to preserve the open-source nature of the software and promote the sharing of improvements and modifications within the community.

Patents

- Patents are exclusive rights given to inventors or discoverers of new things, including creative software algorithms or methods. The topic of software patents can be divisive in the context of open-source software.

- Software patents, according to some, discourage innovation and result in court cases over patent infringement.

- However, patents can also safeguard innovators' legal rights and promote R&D.

- A patent holder cannot utilize their patents against users or developers of the open-source software, hence open-source software projects may choose to incorporate provisions in their licenses to address any potential patent difficulties. These provisions may include issuing patent grants or licensing agreements.

References

1. https://en.wikipedia.org/wiki/History_of_free_and_open-source_software
2. http://www.opensource.org/docs/osd
3. Free Software Foundation www.fsf.org/
4. https://www.youtube.com/watch?v=1ehpgbb3XD0
5. http://www.opensource.org/docs/definition_plain.html
6. https://en.wikipedia.org/wiki/Open-source_software
7. https://en.wikipedia.org/wiki/Free_software
8. https://opensource.org/

References

1. https://en.wikipedia.org/wiki/Software_development
2. https://en.wikipedia.org/wiki/Software_license
3. https://en.wikipedia.org/wiki/Software_patent
4. https://www.w3.org/International/questions/qa-i18n
5. Https://www.apache.org/licenses/LICENSE-2.0
6. https://www.gnu.org/licenses/lgpl.html
7. https://opensource.org/licenses/BSD-3-Clause
8. https://www.youtube.com/watch?v=kigC1ijYGyo

DIGITAL LEARNING CONTENT