Open Source Software Engineering

SE ZG587

BITS Pilani
Pilani Campus

Kumar Manish

22th July, 2023

BITS Pilani
Pilani Campus

Introduction to Open Source Software

Session 1 : Agenda

- About Course and Evaluation criteria
- Proprietary software and examples
- Open Source : what and why
- Open Source Initiatives
- Open Source software and examples
- Advantages and Disadvantages of Open source software

Proprietary Software

- Proprietary software, also known as **Protected** or **Restricted** software,
- Usually owned by an organization or a group of people;
- In order to use a proprietary software, end users must **accept** a License
- Proprietary software, also known as **Non Free software** or **Closed software** , is computer software for which the software's publisher or owner retains Intellectual property rights – usually copyrights of the software, and also sometimes patent rights.
- Non-free or Closed software :
  - It's not about cost of software; it's about source code unavailability

Propriety Software….

- Such Licenses restrict the right of the user, in any of the following manner :
  - Restricted right of redistribution
  - Restricted on reconstructions source code
  - Restricted ways in which the Product can be used or embedded within another Product
- Until recently, proprietary software was the only real model that was used by commercial software
- In proprietary software, the sources code is only available with the owner organization
- Some trusted partner may be given rights to use the same - under the judication of the non-disclosure agreement (NDA)

Open Source

- What is open source ?
  - Open
    - ➔ Collaboration is open
  - Source
    - ➔ Source is freely available
  - What is mean by freely ?
    - Share, Adapt, Modify and Collaborate

Open source is about philosophy and consumer's right, it’s about free collaborations by community

Why open source

- Access to the best software
- Greater Privacy and Control
- Extended Hardware Life
- Doing things your way
- Comprehensive Support
- Greater security
- Quick Bug and Security Fixes
- Instant Gratifications
- No cost

Open Source Initiatives (OSI)

- Open-source initiatives (OSI) is a public benefits corporation
- For Education, Advocacy, and stewardship for collaboratives Development
- Founded in 1998 – actively engaged in building open-source community.
- Opens Source Definitions
- Open Standard Requirements
- Open Source Licenses

Open Source Initiative (https://opensource.org/)

Open Source Software (OSS)

- Open Source software (OSS) is a type of software in which Source code is released
- Released under a **License** in which the **copyright** holder grants users the rights to **use**, **study**, **change**, and **distribute** the software and its source code to anyone and for any purpose.
- Publicly available for anyone to **use, edit, inspect, modify** and **distribute.**
- Open Source software is usually developed in a collaborative Public manner

Examples of Open source software

Categories of software	Open Source software
Programming Languages	PHP, Java, Python
Presentation Software	Libre Office's Impress, Apache Office's Impress
Communications Software	Free Switch, openPBX, Thunderbird
Content Management System	PHP-Nuke, WordPress, Joomla
Operating system	Linux, Ubuntu, Fedora, FreeBSD
Databases	MySQL, PostgreSQL
Web Design software	Adobe Dreamweaver, Brackets,
Web Browsers	Firefox, Mozilla, Arena, Chromium
Application Servers	JBoss, Tomcat, Glassfish, WebSphere
Source cord Management/ Version control	Git, GitHub, Subversion

Advantage of OSS

Low Cost : OSS usually does not require a licensing fee, hence free of cost

Flexible : Can be modified by anyone and tailored to suit specific business needs

Quality and Reliability : One needs to identify and select an OSS that suits business needs and is of good quality . Usually, a mature OSS is generally viewed to be of good quality and considered to be more reliable

Vendor Independence : in case of use of proprietary software , there is usually a contract with a specific vendor (involving high cost, restricted usage). This is overcome in OSS

Advantage of OSS…

Availability of external Technical Support Services by vendors : For example

- Red Hat (now owned by IBM) provides support for many OSS
- MySQL supports provided by the patent company MYSQL AB – now owned by Oracle

- Several open source products have active Online community supports

Disadvantage of OSS

Lack of Personalised support services :

- Unlike proprietary software, OSS product or packages do not come with personalised support facility over phone or email.
  - However for Some mature OSS, there might be some commercial service provides that may provide support services
- **Limited Choices of OSS products**
- **Continuous changes :** continuous changes are made to open source software – difficult to get a compatible , and bug free version at a particular time
- **No warranty :** No warranty support is provided along with OSS since it is not owned by a single company

References and further readings

Open Source Initiative (https://opensource.org/)

Open Source Resources (https://opensource.com/)

Open Source Guides (https://opensource.guide/)

Working with GitHub for Open Source Software Development (https://github.com/)

Open Source Software Engineering

SE ZG587

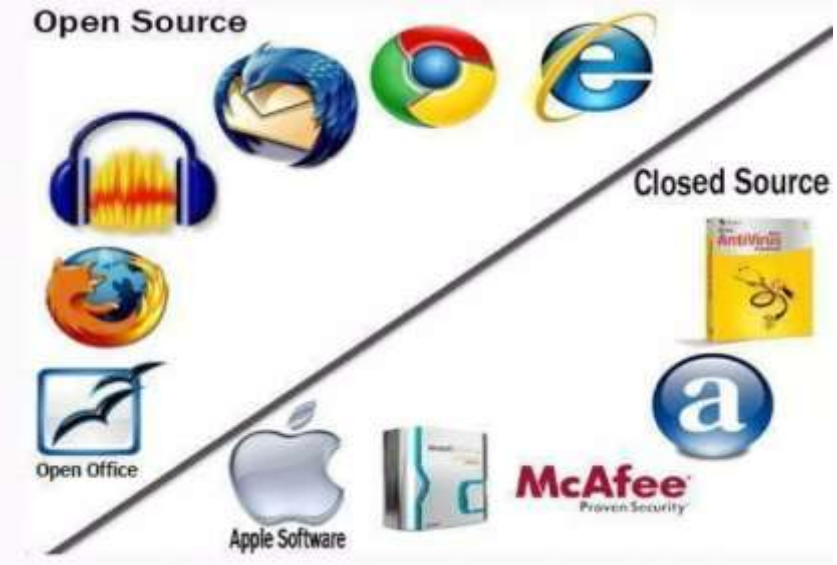
BITS Pilani
Pilani Campus

Kumar Manish

29 July, 2023

Recap : Session 1

- Open source : What and Why
- Open source Initiatives and examples
- Proprietary Software and examples
- Advantages and Disadvantages of Open source software



Session 2 : Agenda

- Understanding of Open Source Software
- Principle of Open Source software
- Cost of Open Source Software
- History of Open Source Software
- Other software :
  - Understanding of Free Software
  - Understanding of Freeware
  - Understanding of Public domain software

Understanding of Open Source Software

Definition : Open source software (OSS) is a software in which

- Source code is released under licenses, and
- The owner of the software (or copyright holder) permits the users the rights to use , modify and distribute the software to anyone and for any purpose

- Philosophy : Development methodology – open collaboration Model
- Rules : Governed by rules of Open source Initiative - open source Definitions (OSD) https://opensource.org/osd)
- Charge : Available free of charge, in most cases . But in principle , it need not necessarily be free of cost.
- Copyright : Yes
- Examples :

Principle of Open Source software

Openness :

Publishing the design and source code of a software to public - with intent that

- Openly fixed or contributed to
- Openly scrutinised / criticized
- Open feedback obtained
- Analysed for bugs or defects
- Analysed for quality

Principle of Open Source software..

Transparency :

- Ability of the community to see the current progress and future plans :
    - projects roadmap is made available to the community
    - A defects tracking system is put in place - for reporting and reviewing defects
    - Publish design documents
- To make more effective decision and understand how decision affects us.

Collaboration : everyone free to participate, enhance each others work in unanticipated ways to unlock new possibilities

Principle of Open Source software..

Release Early and Often : Rapid prototypes and iterative approach

- Any changes proposed or made by any one are made public immediately.
- Contributions are expected to occur early – resolve errors early in development Lifecyle
- Contributions are expected to occur Often - changes shared with others immediately / regularly

Expectations of Community : Participants in an open source projects have expectations of a community - to be formed – that works together - to contribute to the development of the projects

Open Source Software – Open collaborations

The Main Principle behind software development model are :

- Decentralised Software Development
- Open collaboration, and
- Peer Production

As per Wikipedia

- Open collaboration is “a System of innovation or Production or Development that relies on goal-oriented, yet loosely coordinated, participants who interact to create a product (or service) of economic value, and make it available to contributors and non contributors alike”.

Rules for Distributions of open source Software

- Free Redistributions
- Source code
- Derived works
- Preserving Integrity of Original software
- No discriminations based on Person or groups
- No discrimination based on Field of Endeavours
- Distribution of license
- License should not be product specific
- License Must not restrict other

Open Source Software – License

License must not restrict other software :

No restrictions should be placed on the license for the other software that would be distributed along with the licensed software .

For example : the license must not insist that all other programs distributed along with this software and through the same channel, should also be open-source

License must be Technology neutral :

License should not be grouped based on any specific technology or interface style

Open Source Software – Licenses

Open source Licenses examples :

- Apache Licenses 2.0 (Apache 2.0)
- 3 clause BSD licenses (BSD -3 clause)
- 2 clause BSD licenses (BSD 2-clasue)
- GNU General Public Licenses (GPL)
- GNU Lesser General public License (LGPL)
- MIT License (MIT)
- Mozilla Public License 2.0 (MPL 2.0)
- Eclipse Public License 2.0 (EPL 2.0)

Cost of Open source Software

Although OSS is free , there are some hidden cost :

1. Total cost of Ownership (TCO)
    1. Cost associated with adopting and managing the software
    2. Access to software updates, support services
2. Switching cost :
    1. Migrating data from older systems
    2. Training cost to Resources / users etc
3. Additional Cost :
    1. POC/ POV : Evaluation and Selection
    2. Integration cost
    3. Fixing critical bugs

History of Open Source Software

1960

Hardware Expensive but bundled Free Software

1970

- **Emergence of Operating Systems, Compilers**
- Separate selling

1980

Free Software Movement

- 1985 : Free Software Foundations (FSF) GNU project
- Berkeley (BSD Unix
- 1991 : Linus Torvalds – Launched GNU LINUX
- 1994 : Robert McCool - Apache HTTP server Open source Webserver

1990

Open Source Software Movement

1998 : Bruce Perens and Eric S. Raymond - Open Source Initiatives (OSI) as Organisations Open source definitions (OSD)

innovate achieve lead

BITS Pilani

Pilani Campus

Understanding of Free Software

Free Software

- Free software" means software that respects users' freedom and community. It refers to free use of software and not price
- **Definition** : Free software is the software that can be **used** , **modified**, **studied**, **copied**, **changed** and **redistribute** (with or without modifications) with no restrictions.
- **Philosophy** : Social Movement
- **Charge :** Free software is available free of charge , in the most cases , But , in principle, free software need not necessarily be free of cost;
  - Free software does not mean non commercial
  - One always has freedom to change or copy free software and then sell it.
  - You may even sell the original software
- **Copyright** : Yes
- Examples : Linux ubuntu MySQL Apache

Free Software – Social Movement

- The free software movement is a Social Movement
  - With the aim of gaining and assuring certain freedoms for software users
  - Movement was Founded by Richard Stallman , in 1983, by launching **GNU Project**
  - In 1985 established **The Free software Foundations**

"The Free Software Foundation (FSF) is a nonprofit with a worldwide mission to promote computer user freedom. We defend the rights of all software users."

- Four essential Freedoms of free software
  - Software which meets these freedom requirements is termed free software

Free software – Four essential freedoms

- **A program is “free software” if its users have the following** :
  - Freedom 0 : The freedom to run the program as you wish, for an purpose
  - Freedom 1 : The freedom to study how the program works , and change it as per your requirements
  - Freedom 2 : The freedom to redistribute copies so you can help others
  - Freedom 3 : The freedom to distribute copies of your modified versions to others
    - **Pre-requisite “ Access of the source code”**

Free Software - Legal considerations

- Legal considerations :
  - The owner of the free software does not have power to withdraw or invalidate the license , or add additional restrictions to its terms and conditions,
- The license terms should be permanent and irrevocable

Free Software - Licenses

A large number of licenses qualify as free software licenses and are fee compatible with GNU General Public licenses :

Examples :

- GNU General Public License (GPL) version 3
- GNU General Public Licenses (GPL) version 2
- GNU All – Permissive License
- Apache License version 2.0
- Clarified Artistic License

Refer a complete list of licenses at :

https: // www.gnu.org/licenses/license-list.html

GNU in a Nutshell

- GNU was launched by Richard Stallman, as an operating system which would be put together by people working together for the freedom of all software users to control their computing.
- The name of the system, GNU, is a recursive acronym meaning GNU's Not Unix—a way of paying tribute to the technical ideas of Unix, while at the same time saying that GNU is something different.
- Technically, GNU is like Unix. But unlike Unix, GNU gives its users freedom.

References and further readings

Open Source Initiative https://opensource.org/

Open Source Resources https://opensource.com/

Open Source Guides (https://opensource.guide/

Working with GitHub for Open Source Software Development (https://github.com/)

How To Pronounce GNU - GNU Project - Free Software Foundation

GNU in a Nutshell - GNU Project - Free Software Foundation

GNU Licenses https://www.gnu.org/licenses/license-list.html

What We Do - Creative Commons

Open Source Software Engineering

SE ZG587

BITS Pilani
Pilani Campus

Kumar Manish

5th August, 2023

BITS Pilani
Pilani Campus

Freeware

Freeware Software

- Free refer to price ,
- freedom to use is restricted by owner
- No source code Available
- Copyright law : yes
- Philosophy :
  - Marketing Goals – Intended to benefit the owner
  - Make profit end of day

Examples : WhatsApp, Skype, Adobe, Gmail etc

Public Domain Software

Public Domain Software

- Belong to Public ; Can be modified , distributed or sold even without any attribution by anyone
- Rules : Creative common organisation
    - https://creativecommons.org
- Charge : No cost
- License : Creative Common Licenses
- Copyright law : No
- Example : SQL Lite

Creative Common

• **Creative Commons** is a nonprofit organization that helps overcome legal obstacles to the sharing of knowledge and creativity to address the world's most pressing challenges.

• Provide Creative Commons licenses and public domain tools that give every person and organization in the world a free, simple, and standardized way to grant copyright permissions for creative and academic works; ensure proper attribution; and allow others to copy, distribute, and make use of those works

• Work closely with major institutions and governments to create, adopt and implement open licensing and ensure the correct use of CC licenses and CC-licensed content

Summary

	Free software	Open source Software	Freeware	Public domain software
Definition	“FREE” is a matter of liberty , not price	“OPEN” does not just mean access to the source code; more about collaboration	Free refers to price , while freedom of the use is restricted by creator	PUBLIC domain belongs to the public as a whole
Philosophy	Social movement	Development methodology	Marketing goals	Copyright disclamation
Rules	Four freedoms	Open source Initiatives		Creative common Org
Free of charge	Not necessary	Not Necessary	YES	YES
Copyright law	YES	YES	YES	NO
Examples	Linux, Ubuntu , MySQL, Apache	Linux, Ubuntu , MySQL, Apache	Skype, Adobe acrobat	SQLite

Recap : previous sessions

- Open Source : Why, What and Examples
- Open Source Initiatives (OSI) and Free software Foundations (FSF)
- Advantages and Disadvantages of Open source software
- Principle of Open source software
- History of Open Source Software
- Cost of Open Source Software
- Understanding of Free Software
- Understanding of Open Source Software
- Understanding of Freeware
- Understanding of Public domain software

innovate achieve lead

BITS Pilani
Pilani Campus

Understanding Intellectual property Rights and Software Licenses

Topic :

- Understanding Intellectual Property Rights
- Understanding Software Licenses
  - **Licensing Models in OSS:**
    - Copyright,
    - Copyleft,
    - Permissive,
    - Creative Commons

Intellectual Property Rights (IPR)

- The creator of an artefact is considered to be the owner of the artefact, This implies that the owner who writes the code owns it – unless there exists a written contract that states differently
- In House software development :
  - The entire ownership or the copy right of the software remains with the Organisation
- Outsource software development :
  - The issues related to Intellectual Property (IP) rights are correctly Managed - By written contracts
  - Rights to artefacts like business ideas , images , diagram, source code and documentations remain under the sole ownership of the client company .

Intellectual Property Rights (IPR)

Types of Intellectual Property Rights : relevant to the software industry

- **Patents** – used to protect functional features , like hardware configurations etc
- **Copyrights** – used to protect works of authorship , like source code , diagram etc
- **Trade secrets** – used to protect internal business secrets, like business and pricing models
- **Trademarks** - used to protect brand recognition, through logo etc
  - Patents, copyright and trade secrets are used to protect the technology itself
  - Trademarks do not protect technology, but help in distinguishing a product in the marketplace

IPR in Software Domain

Who can claim the ownership of the IPR ?

All the people involved in a development project, may claim for the ownership of the IPR for the various artefacts developed as a part of the project.

- **Employee Involved** – are the first who have the IP rights to all the artefacts developed during employment ; however they are usually restricted by the employment contracts.
- **Consultant or contractors (Organisation or individual)**
  - unless a written contract says otherwise , they also claim for the ownership of the projects artefacts
- **Vendor company developing the software**
  - unless documented , although the clients pay for the services provided by the vendor , they are not necessarily the owners

IPR in Software Domain

Category of Software source code in Projects :

- **Unique code**
- **Open source code**
- **Existing / Third- party code**

- **Unique code :** refers to the code specifically developed for a particular project
  - source code and related artefacts of the software may be limited use to the developer or vendor organisations
  - Vendor organisation may be ready to assign ownership of the same or grant exclusive license to the client company
- **Open Source code** – refers to use of open source code or technologies which are Publicly available
  - Neither the client, nor the vendor owns the IP rights to the Open sources code or technology ; nor does anyone maintain exclave control over it.
  - Violation of the Open source license could lead to significant risks of the entire projects e.g. as per GPL – one must publish source code of the complete projects
  - Ensure open source are used legally and all Compliance requirements are met.

IPR in Software domain

Existing / Third party code – Witten by the developing company for other projects – reuse the same for the current projects

- **Existing code :** Vendor is not willing to give the ownership of this type of code, since they would want to continue using it for other clients
- **Third party code or technology :** Pay a license for he code in a way that is compatible with client's projects
  - Other projects artefacts like image , audio and video files etc .. Could also be included

References and further readings

Open Source Initiative https://opensource.org/

Open Source Resources https://opensource.com/

Open Source Guides (https://opensource.guide/

Creative commons https://creativecommons.org

GNU https://www.gnu.org/

Copyleft https://copyleft.org/

Open Source Software Engineering

SE ZG587

Session 4

BITS Pilani
Pilani Campus

Kumar Manish

12th August, 2023

BITS Pilani
Pilani Campus

Understanding Software Licenses in OSS

Software Licenses

- A software License is a **Legal instrument** which describes the manner in which a software can be used or redistributed - in both source code or object code forms.
- A typical software license grants the **Licensee** (end-user) , the permission to the software in a manner where such a use would **otherwise potentially constitute copyright Infringement of the software** owner's exclusive rights.
- **Software Licensing Models :**
  - Copyrights
  - Copyleft Licenses - Protective
  - Permissive Licenses - More free
  - Creative Commons Licenses - Public copyrights licenses

Copyrights

- Copyright is an important intellectual property license that gives the owner an exclusive right on the work crated by him / her. The owner is allowed to create copies of the creative work
- The creative work may be of any type - including literary artistic, educational, or musical form
- As a general rule , for works created after January 1, 1978 , copyright protection lasts for the life of the author plus a additional 70 years.
- In case of a **corporate authors** , the protection is for the shorter of **95 years from publication or 120 years from creation**

Copyleft

- Copyleft is a licensing method,
  - used to make a work (or program) free to use, modify , adapt or extend
  - The freedom to carry out all activities , as aligned with the **four essential freedoms** (prerequisite – source code available)
  - Copy left licenses includes the GNU general public license (GPL) – which was originally written by Richard Stallman
  - the work or program should be **made available to the recipients - Mostly in the form of source code**
- No re-licensing allowed, No commercial usage allowed
- All modified and extended version of the work) or program) should be free as well - hence called **protective licenses**
- Rules for copyleft –
  - all derivative works should be attributed to the creator, open sourced and copyleft
- Copyleft is a generic concept , and can't be used directly ; one needs to use a specific implementations of the concepts

Variants of Copyleft

- **Strong and Weak copyleft** : The strength of the copyleft license is decided based on the extent its provision are imposed on the derived works

  Examples :

  - Strong Copyleft :
    - GNU general public License
  - Weak copyleft
    - GNU Lesser General Public License –
    - Mozilla public license

Weak copyleft licensing :

- Most commonly, weak copyleft licenses are used to create **Software Libraries**
- Help software to link to the library and redistributed without requirement for the linking software to also be copyleft –licensed

Copyleft – share alike condition

Share-alike :

- Imposes the requirement that any freedom that is granted regarding the original work must be granted on exactly the same or compatible terms in all derived work
- However, in some cases the author may be willing to share only a certain part of the work, **but the share – alike agreement require that the whole body of the work is shared**
- The positive side – for an author of source code – any modification to the code will not only benefit the original creator, but that the author will be attributed and recognised and hold equal claim over the changed code.

Permissive Licenses

- Free software licenses with only minimal restrictions on how the software can be used , modified and redistributed (also called Berkeley software distribution : BSD-like or style licenses)
- Rules for usage - what ever user wants , but with few restrictions – derived works must be attributed to the creator
- Source code need not be open or made available in the public domain
- Re-licensing allowed - derivative works can be release under any other licenses or used as proprietary products
- Allows commercial usage
- Examples :
  - GNU All permissive License
  - MIT License
  - BSD licenses
  - Apple Public source licenses
  - Apache license

Comparing Permissive and Copyleft

Copyleft (Protective Licenses)	Permissive Licenses
Publication of software code of all modified versions or derived works under the original copyleft licenses - protective licenses	Provides No guarantee that derived works of the software will remain free and publicly available ; generally requiring only that the original copyrights notice be remained
	- Hence derived works , or future versions , of permissively – licenses software can be released as proprietary software - Permissive licenses offer highly extensive license compatibility as compared to copyleft licenses - Wider adaptability in the open source community

Creative Common Licenses (CCL)

- A public copyright licenses that **enable free distribution of copyright work** – allowing the users the right to share, use , and build upon a work that – someone else (the author) has created
- **Rules for usage** - whatever user wants without any restrictions - derived works must be attributed to the creator
- Availability of the source code : No specific terms about the distribution of source code
- **Re-licensing allowed**
- **Commercial usage allowed** , however author can decide to allow uses of given work as non commercial
- CCL applied all works : books , plays, movies, photograph , music, articles, blogs and websites
- CCL were initially released in 2002 by creative commons , Non profit Org founded in 2001
- So far Five version of licenses , latest in 2013 - version 4

Creative Commons Licenses

Compatibility with software :

- **Not recommended for use for software** - since they **do not contain specific terms about the distributions of source code**, which is important in order to ensure the free reuse and modifiability of software
- Most of **the CC licenses are not compatible with the major software licenses , so it would be difficult to integrate CC licenses work with other free software**
- However, they may be used for software documentations , or other artistic elements embedded within documentation.
- **Free Software Foundations (FSF) :** In 2011 added “CC 0 version”
- **Open Source Initiative (OSI)** : Not approved under OSI because of its clause which **excluded** from the scope of the licenses any relevant patents held by the copyrights holder.

How to choose an open source License ?

- Open source licenses help in protecting works contributed in the open source domain
- It also help protect contributors and users from any copyright infringements
- Interested developers or contributors or business will not touch a project without a license protection.

Which license should I choose :

- If I wish to work with a community
- If I wish to keep the licenses simple and permissive
- If I wish to share the improvements made
- If I wish to work without license

Recap

- Intellectual Property Rights and Software Licenses
- Licensing Models in OSS:
  - Copyright,
  - Copyleft,
  - Permissive,
  - Creative Commons

Understanding and Choosing Open Source Licensing Models

Choosing Open Source Licensing Models

- Option 1: Work with a community
- Option 2: Keep it simple and permissive
- Option 3: Need to share improvements
- Option 4: Work without a license

How to choose an open source License ?

Why to choose ?

- Open source licenses help in protecting works contributed in the open source domain
- It also help protect contributors and users from any copyright infringements
- Interested developers or contributors or business will not touch a project without a license protection.

Which License should I choose :

- If I wish to work with a community
- If I wish to keep the license simple and permissive
- If I wish to share the improvements made
- If I wish to work without a license

Work with a community

- To contribute to or modify or adapt an existing projects,
  - Continue contributions using original license available with the project
- Moreover, using the same license might also be a requirement
  - as per stated in the original license terms of the project
- Look at LICENSE or COPYING or README file for more info
- Certain Open source software projects require contributors to sign the Contributors License Agreement (CLA)
  - Simple CLAs vs Detailed CLAs
  - Individual CLAs vs Corporate CLAs
- No centralised body of knowledge engaged in standardizations of CLAs
  - Different organisations write their own versions
  - Larger project need formal CLAs
  - Generally baked by one or more corporations

    Ex – Apache, Django, Eclipse Foundations etc

Work with a community

- Normally, a CLA require the contributors to make certain classifications (In terms of legal proofs) , which may comprise of one or more of the following :
  - the contributors is the author of the contributions;
  - the contribution is an original work;
  - the contribution is not subject to third – party license , claims etc;
  - the contribution has the legal right to grant the copyright license;
  - the contributors does not have an employer that can claim rights in the copyright;
- Benefits of CLA : protection against copyright infringement
- Disadvantages : discourage contributions

Work with a community

How to manage the ownership and licensing of copyrights for individual software contributors to the project ?

Option 1 : The author (who is also, by default, the copyright holder) of the software contributions retains the software ‘s copyright and ownership and contributes it under the same open source license as used by the project.

Option 2 : The contributors assigns the ownership of the software copyright to The “Project Maintainer”. Who then releases the software under the projects’ Open Source License.

Option 3 : the last option is not to define any ownership policy

Keep it simple and permissive

- Choose the MIT License :
  - It is a short and simple permissive license with condition only requiring preservation of copyright and license notices.

- Licensed works, modification and larger works may be distributed under different terms and without source code.

- The MIT license allows users the permission to reuse code for any purpose , or embed it as a part of proprietary software or make any changes or modifications to the code to suit their needs.

- Restrictions - users are required to include the original copy of the MIT licenses

Keep it simple and permissive

- Permissions :
  - Commercial Use : the licenses materials and derivatives may be used for commercial purposes
  - Distribution : The licensed materials may be distributed
  - Modification : The licensed materials may be modified
  - Private use : The licensed materials may be used and modified in private
- Conditions :
  - A copy of the license and copyright notice MUST be included with the license materials.
- Limitations :
  - Liability : The license includes a limitation of liability
  - Warranty : The license explicitly states that it does NOT provide any warranty

Need to share improvements

- Choose the GNU general Public Licenses v3.0 – copyleft license - protective license
- This strong copyleft license provisions on making available complete source code of licensed works and modifications, including larger works under the same license.
- Copyright and license notice Must be preserved.
- Contributors provides an express grant of patent rights.
- GPL assures that patent can not be used to render the program non free.

Need to share improvements

- Permissions :
  - Commercial use : The licensed materials and derivates may be used for commercial purposes.
  - Distributions : the licensed materials may be distributed.
  - Modification : The licensed materials may be modified
  - Private use: The license materials may be used and modified in private
  - Patent use : the license provides an express grants of patent rights from contributors
- Limitation :
  - Liability : The license includes a limitation of liability
  - Warranty : The license explicitly states that it does NOT provide any warranty

Need to share improvements

Conditions :

- Disclose source : Source code MUST be made available when the licensed materials is distributed.
- License and copyright notice : A copy of the license and copyright notice Must be include with the license material.
- Same license : Modifications Must be released under the same license when distributing the licensed materials . In some cases a similar or released license may be used.
- State Changes : Changes made to the licensed materials Must be documented

Work without a license

- By default, all works are under exclusive copyrights of the creator – be it creative work or code
- Unless a license is associated with the copyrighted works, that states otherwise , no one can copy , distribute, or modify the work without being at risk of copyright infringement
- However , in the absence of a license file , one may still grant some rights – the holds in cases where one publishes the source code on a site that requires accepting terms of service
- For example , if the source code is published in a public repository on GitHub, and you have accepted their terms of service , by this you allow others to view and fork your repository

Work without a license

- You do not have to do anything to work without a license
  - However, in the case , one may wish to add a copyright notice and statement - indicating that no license has been included - in a prominent place – e.g. the project READADME
- Moreover, disallowing the use of your code might not be what you actually intent by “ no license”
  - An open source license allows reuse of your code while retaining copyright.
  - Add a contributors license agreement (CLA) to your non-licensed project, so that you maintain copyright permission from contributors, even though you are not granting the same.

Work without a license

To use or contribute to project that has no license associated with it

- Do not use the software : Find or create an alternative that is under an open source license
- Request the Project Maintainers to add a license :
  - Unless the software included strong indications to the contrary, lack of a license is probably an oversight.
    - If the software is hosted on a site like GitHub, open an issue requesting a license
    - If you are sure what license is most appropriate, open a pull request to add a license - with a suggestive license

References and further readings

- **Choose a license** https://choosealicense.com/
- **Open Source Initiative https://opensource.org/**
- **Open Source Resources https://opensource.com/**
- **Open Source Guides (https://opensource.guide/**
- **Creative commons https://creativecommons.org**
- **GNU https://www.gnu.org/**
- **Copyleft https://copyleft.org/**

Open Source Software Engineering

SE ZG587

BITS Pilani
Pilani Campus

Kumar Manish

19th Aug, 2023

BITS Pilani
Pilani Campus

Session 5 : Open Source Business Model

Recap Previous sessions

- **Intellectual Property Rights and Software Licenses**
- **Licensing Models in OSS: Copyright,**
  - **Copyleft – Strong and week, Permissive, Creative Commons**
- **Choosing Open Source Licensing Models**
- Option 1: Work with a community
- Option 2: Keep it simple and permissive
- Option 3: Need to share improvements
- Option 4: Work without a license

Agenda session 5

- Open Source Business Model
- By Intellectual Property :
  - Dual Licensing Model
  - Open Core Model
- By Services and other Business model
  - Open source SaaS model
  - Selling Users advertisement, Services and Merchandise
  - Donation, Crowd Funding and Crowd Sourcing
  - Freemium Business Model

Open Source Business Model

- In traditional commercial business model for companies :
  - Sale of Software Products - Product Companies
  - Sale of Software development - Services companies
  - Sale of Support services
- Companies that create and promote Open source software referred to as :
  - **Commercial Open source Software companies (COSS , or**
  - **Professional Open source Software companies (POSS)**
- **Why ? To solve the challenge of how to make money providing software that is by definition licensed free of charge**.

Open source Business model

- **Business strategies rests on the premise that users of open-source technologies are willing to purchase additional software features under proprietary licenses, or purchase other services or elements of value that complement the open-source software that is core to the business.**
- This **additional value** can be, but not limited to, enterprise-grade features and up-time guarantees to satisfy business or compliance requirements, performance and efficiency gains by features not yet available in the open source version, legal protection (e.g., indemnification from copyright or patent infringement), or professional support/training/consulting that are typical of proprietary software applications.

Open source Business Model

For Financial viable and successful, POSS companies Business Model :

1. Selling Intellectual property - the code itself

- **Dual Licensing model**
- **Open – core model**

2. Open Software as a Service (SaaS)

3. Selling Services

- Selling Support and Consultancy service – professional services

4. Others :

- Advertising
- Partnership with funding organisations
- Selling branded merchandise
- Freemium

1. Selling Intellectual property : By Dual License Model

Dual license products are generally sold as :

- Community version – under GPL
- Enterprise version – under commercial license

Examples : Oracle's MySQL database – dual licensed

- Under GPL : MySQL Classic edition , and My SQL community edition
- Under commercial license :
  - MySQL Enterprise Edition
  - MySQL Cluster CGE, an
  - MYSQL Standard Edition

Dual Licensing Model

In this model, the OSS licensed by the POSS company under two licenses :

- **An open source license - Using GPL licenses, and**
- **A commercial license**

- The company generates revenue by selling the OSS under commercial license.

Why would a consumer of OSS buy a license from a company by paying a heavy amount for a software which is already available freely and at a no cost ?

This is necessary when the consumer wants to link his own proprietary software to the OSS, but does not want this to cause its proprietary software to become open source ; as it would under the GPL license

Dual Licensing Model

- According to the GPL license, if someone uses the GPL licensed source code and links it with some other code (dynamic linking or static linking), the linked software also becomes open source.
  - Thus the only way proprietary software vendor can link GPL software without causing their own software to become GPL is by buying a commercial license from the company
- The POSS company provides the same software to the proprietary software company under a commercial license **which saves the proprietary software from becoming open source**

Open- Core Model

- In this model , a core portion of the software, which provides the basic functionality - is made available as open source
- The outer or the other portion or extensions are designed to provide extended or additional functionality and are built over the core.
- The extended portion is licensed under commercial license and sold as a propriety software.
- Also called **Split open source** software

Examples : Open-Core Model

Example : Used by - Apache and Mozilla

- GitLab CE (Community Edition) is under a MIT style open source license , while GIT Lab EE is under commercial license
- Initially, Elastic core , which includes Elastic search, Kibana , Logstash and Beats , is under an Apache 2.0 license , while additional plugins are distributed under Elastic’s own proprietary license

2. Open Software as a Service (Open SaaS)

Another way to commercialise an open source project is by using the Open SaaS business model

- **WordPress** and **Sharetribe** are popular examples products that make successful business through the open SaaS model
- software **stored in cloud** and accessible using a wen browser
- Purchase via **subscriptions**, offering varying levels if services
- No vendor locking
- **OpenSaaS**' : Coined in 2011 by **Dris Buytaert**, creator of **Drupal**.

3. Selling users : Professional services

- Sell services – such as consulting service , technical support service, or training services.
- Make available only the source code of OSS application or product :
  - Sell executable binaries to the buyers or customers on demand
- Offers the commercial services of compiling and packaging the software
- Example : companies like RedHat, IBM that have successfully used this model
- Additionally , selling good like physical installation media (e.g. DVDs) is another frequently used business model

Selling Users : Advertising – supported software

- Another POSS Business model :
  - For e.g. Google and Mozilla
- Example : Google pay Open source application for allowing whitelisted acceptable Ads display.
  - This is carried out by bypassing browser Ad remover.
- SourceForge, have a revenue model of advertising banner sales on their model

Selling users : Branded Merchandise

- Some POSS companies like Wikimedia foundation and Mozilla foundation, attract customers by selling branded merchandise articles like T-shirt and coffee mugs.

Freemium Business Model

- Feature limited - basic Features vs. Extended features
- Time limited – fixed duration after that paid versions
- Example : widely used in Gaming Industry, Anti-Virus, LinkedIn

References and Recommended Reading:

- How Developers Can Make money with Open Source Projects

https://rubygarage.org/blog/how-make-money-with-open-source-projects

- What Motivates a Developer to Contribute to Open-Source Software?

https://clearcode.cc/blog/why-developers-contribute-open-source-software/

- How do Open Source Programmers make money

https://www.thewindowsclub.com/open-source-companies-programmers-make-money

- **https://en.wikipedia.org/wiki/Business_models_for_open-source_software**

References and further readings

- Choose a license https://choosealicense.com/
- Open Source Initiative https://opensource.org/
- Open Source Resources https://opensource.com/
- Open Source Guides (https://opensource.guide/
- Creative commons https://creativecommons.org
- GNU https://www.gnu.org/
- Copyleft https://copyleft.org/

Open Source Software Engineering

SE ZG587

BITS Pilani
Pilani Campus

Kumar Manish

26 Aug, 2023

Session 6 : Open Source Business Model

- Donations, funding and Crowd - sourcing

Recap Previous sessions

- How individual / company Can Make money with Open Source Projects ?
- How do Open Source Companies and Programmers make money?

1. **Selling Intellectual property - the code itself**
   - **Dual Licensing model**
   - **Open – core model**
2. **Open Software as a Service (OpenSaaS)**
3. **Selling Services**
   - Selling Support and Consultancy service – professional services
4. **Others Business Model :**
   - Advertising, Selling branded merchandise
   - Partnership with funding organisations
   - Freemium

Dual License Model

Dual license products are generally sold as :

- Community version – under GPL
- Enterprise version – under commercial license
- According to the GPL license, if someone uses the GPL licensed source code and links it with some other code (dynamic linking or static linking), the linked software also becomes open source.
- Thus the only way proprietary software vendor can link GPL software without causing their own software to become GPL is by buying a commercial license from the company

Examples : Oracle's MySQL database – dual licensed

- Under GPL : MySQL Classic edition , and My SQL community edition
- Under commercial license : MySQL Enterprise Edition , MySQL Cluster CGE, and MYSQL Standard Edition

Open - Core Model

- In this model , a core portion of the software, which provides the basic functionality - is made available as open source
- The outer or the other portion or extensions are designed to provide extended or additional functionality and are built over the core.
- The extended portion is licensed under commercial license and sold as a propriety software.
- Also called Split open source software

Example : Used by - Apache and Mozilla

- GitLab CE (Community Edition) is under a MIT style open source license , while GIT Lab EE is under commercial license
- Initially, Elastic core , which includes Elastic search, Kibana , Logstash and Beats , is under an Apache 2.0 license , while additional plugins are distributed under Elastic's own proprietary license

Open Software as a Service (Open SaaS)

Another way to commercialise an open source project is by using the Open SaaS business model

- **WordPress** and **Sharetribe** are popular examples products that make successful business through the open SaaS model
- software stored in cloud and accessible using a wen browser
- Purchase via subscriptions, offering varying levels if services
- No vendor locking
- OpenSaaS' : Coined in 2011 by Dris Buytaert, creator of Drupal.

Other Business Models ...

- **Sell Services** – such as consulting service , technical support service, or training services. Example : companies like RedHat, IBM that have successfully used this model
- **Selling Users : Advertising –** supported software, Example : Google pay Open source application for allowing whitelisted acceptable Ads display. This is carried out by bypassing browser Ad remover. SourceForge, have a revenue model of advertising banner sales on their model
- **Selling Branded Merchandise :** like Wikimedia foundation and Mozilla foundation, attract customers by selling branded merchandise articles like T-shirt and coffee mugs.

Agenda : session 6

Open Source Business Model …….:

- **Freemium Business Model**
- **Re-licensing under a proprietary license**
- **Donations and funding**
  - **Crowd Funding for OSS**
  - **Crowd Sourcing for OSS**

Freemium Business Model

- **Freemium is combination of two words – free and premium**
- Feature limited - basic Features vs. Extended features
- Time limited – fixed duration after that paid versions
  - In this pricing strategy , the basic version of the software product or service is made available for free, but a premium amount is changed for additional or enhanced features or services that enhance or expand the functional of the free versions
- Source code not available
- Model extensively used in gaming industry

In OSS :

- Companies raises money by selling additional , but optional features , modules , extensions , or plugins to an open source software product

Re-licensing under a proprietary license

- In this model, a software company combines some portion of its proprietary software product, with an existing Open source software (under a permissive free software license),
- And relicenses the resulting software Product under proprietary licenses and sells it without the source code or associated freedoms.
- For example : Apple Inc, uses BSD Unix operating system kernel (Under BSD licenses) in the Apple's mac PCs, and sells it as proprietary product.

Revenue model followed by the version control system

- Different version control system provide different revenue models :
- SourceForge : making money Through advertising banner sales on their website
- **Making money by GitHub:**
  - Get funding for your GitHub Repository - interested company might pay you – they would may be using your code or need to promote themselves using your repository
  - Solving open issues in a GitHub repository
  - Finding bugs in GitHub

Donations and funding

- Voluntary Donations and Crowd – funding - other way of getting money for developers

Crowd Funding is the practice of raising funds for a project or event or a venture in small or large amounts - typically, from a large number of people - mostly via the internet.

Actors Involved :

- **Initiator** - Responsible for proposing idea of the project/ venture or event to be funded
- **Individual or group who support the idea**
- And **Moderating Organisation** that provides necessary infrastructure to launch and execute the idea
- Ex : for film making, eduction, medical expenses, NGOs, travel etc.

Crowd funding for OSS Projects

Open source projects seek support for financial assistance through crowd funding; platform examples

- BountySource https://bountysource.com/
  - Bountysource is the funding platform for open-source software. Users can improve the open-source projects they love by creating/collecting bounties and pledging to fundraisers.
- Snowdrift https://snowdrift.coop/
  - Snowdrift.coop is a nonprofit cooperative run by an international team driven by a common goal: To dramatically improve the ability of ordinary people to fund public goods – things like software, music, journalism, and research – that everyone can use and share without limitations.
- Gunio https://gun.io/

Crowd - Sourcing

- Crowd sourcing is a participative and online activity
- In crowd sourcing , an individual or an institution reached out to an individual or group of induvial , requesting them to voluntarily undertake the proposed task through a flexible open call.
- Expectations :
  - The crowd is expected to participate by bringing their **work, money , knowledge and or experiences**
  - **Mutual benefit**
- **Example for crowdsourcing are Linux, Google Android, Wikipedia**

What Motivates a Developer to Contribute to Open-Source Software?

- Improve Coding Skills
- Gain Early Experience
- Increase Community and Peer Recognition
- Greater Job Prospects
- Improve Software on a User and Business Level

How do Open Source Programmers make money

- Companies Pay Open Source Programmers
- Earning By Creating Special Plugins, Etc.
- Earning by Customization of Code
- Earning By Providing Support

References and Recommended Reading:

- https://en.wikipedia.org/wiki/Business_models_for_open-source_software

- **How Developers Can Make money with Open Source Projects**

https://rubygarage.org/blog/how-make-money-with-open-source-projects

- **What Motivates a Developer to Contribute to Open-Source Software?**

https://clearcode.cc/blog/why-developers-contribute-open-source-software/

- **How do Open Source Programmers make money**

https://www.thewindowsclub.com/open-source-companies-programmers-make-money

References and further readings

- Choose a license https://choosealicense.com/
- Open Source Initiative https://opensource.org/
- Open Source Resources https://opensource.com/
- Open Source Guides (https://opensource.guide/
- Creative commons https://creativecommons.org
- GNU https://www.gnu.org/
- Copyleft https://copyleft.org/

Open Source Software Engineering

SE ZG587

BITS Pilani
Pilani Campus

Kumar Manish

2 Sept, 2023

Session 7 : Consumption of Open source in Enterprise

Recap : Open Source Business Model

In summary :

How individual / company Can Make money with Open Source Projects ?
How do Open Source Companies and Programmers make money?

- **Selling Intellectual property - the code itself**
  - **Dual Licensing model (community and enterprise version)**
  - **Open – core model**
- **Open Software as a Service (OpenSaaS)**
- **Selling Services :** Selling Support and Consultancy service – professional services
- **Others Business Model :** Advertising, Selling branded merchandise ; Partnership with funding organisations; Freemium

Open source in Enterprise :

- How to consume open source software in enterprise?
- Is there a proper way to consume open source software in an enterprise ?
- WHAT ARE BEST PRACTICES FOR OPEN SOURCE SOFTWARE CONSUMPTION IN ENTERPRISE ?

COMMUNITY OPEN SOURCE

Key Characteristics:

- Fast moving with new features released regularly.
- Security and bug/fixes are applied to the most recent stable version.
- Bug fixes or features may be out of sync with your needs if community does not prioritize them.
- Mostly community supported.
- Requires some DIY to bridge functional gaps.

COMMUNITY OPEN SOURCE : LIMITING FACTOR

The most limiting factors in the wider adoption of community open source system from two reasons:

- Mostly supported by the community and thus require a good amount of DIY, patching and fixing capabilities within enterprise teams.
- Community decides which versions to patch and bug fix, which is mostly the latest version.
- This requires a **forced upgrade to latest version** in most cases and may incur significant technical debt in order to upgrade the enterprise software built using it. Sometimes this debt is not significant and easy to manage.
- But at scale it can present quite a challenge to review, patch, test and release thousands of binaries without a significant downtime or impact.

COMMUNITY OPEN SOURCE : FOR ADOPTION IN ENTERPRISE

DOs	DONT's
Experiment and evaluate new capabilities and use cases using community open source. For example, evaluate MySQL or Postgres as possible replacement of expensive proprietary databases.	Expose to production data in production environment without having a bullet-proof rollback and recovery strategy.
Build initial capability and momentum for a new business case. For example, use API frameworks to evaluate, build and provision REST APIs for partner or B2B integration.	Deploy at scale if you cannot upgrade at community's pace.
Release as part of a pilot or controlled testing. Might be okay for low-visibility or low-impact production which does not increase security or data corruption risk.	Deploy to production if you don't have capabilities to (a) detect security vulnerabilities in the new OSS and (b) patch it in-house.
	Deploy to business critical environments, where service, support and recovery guarantees require vendor support.

ENTERPRISE OPEN SOURCE OR OPEN CORE

Key characteristics:

- Supported by a vendor.
- Stable with long supported versions.
- One off bug fixes and feature enhancements are usually supported.
- Feature rich and complete package provided by vendor, to implement and run in production.
- Performance tuned to typical production workloads.

ENTERPRISE OPEN SOURCE OR OPEN CORE

DOs	DONT's
Deploy to production after thorough testing, performance benchmarking and integrating with support and operational procedures.	Select a vendor without understanding what they will and will not support. For example, if the vendor does not provide at least three years of support, be very deliberate about it.
Make sure all Dev, Test, QA and UAT environments have the same enterprise versions running as production.	Select a vendor that does not significantly contribute of the community open source equivalent. This is important for various reasons. Notably, vendor's ability to support issues on time and be able to drive the product's direction and provide innovation.
Adopt an enterprise roadmap to map out where the open source version might be able to displace proprietary stacks and still match or exceed feature + performance metrics. This could result in significant cost savings.	

BITS Pilani
Pilani Campus

Session 7 : Life cycle and methodologies in Open source software

1. Open Collaboration Model

- Open collaboration is “a system of innovation or production or development that relies on goal oriented, yet loosely coordinated, participants who interact to create a product (or service) of economic value, and make it available to contributors and non contributors alike.”
- Additionally, Riehle et at defined open collaboration as collaboration that is based on the following three principles of :
  - Egalitarianism or equalitarianism
  - Meritocracy, and
  - Self –organization

Open Collaboration Model

Egalitarianism

- Egalitarianism (from French égal 'equal') or equalitarianism is a school of thoughts within political philosophy that builds from the concepts of social equality, prioritizing it for all people.
- In the open collaboration model as applied to open source software domain , egalitarianism is used to specify that everyone can contribute.
- Open source software projects are made available over the internet and accessible to the entire human race, equally.

Open Collaboration Model

Meritocracy

- Meritocracy advocates a political system in which political power and value of economic goods or individual are measured on the basis of effort, talent, and achievement, rather than the social class to which one belongs to, or the amount of wealth possesses by him/her
- In the Open Collaboration Model as applied to open source software domain, meritocracy is used to **judge contributions**, in a transparent manner, based on the merit or the quality of the contribution.
- Additionally, all decisions are also made publicly available for evaluation of the judgement taken.

Open Collaboration Model

Self–organization

- Self – organization, in social science , is a process in which some system of order or arrangement emerges through interactions between individuals or parts of an initially disordered system.
- This process may be impulsive and usually not controlled by external agents
- The resulting system of order is
  - Decentralised
  - With all the components of the system distributed , yet connected
  - Robust and self repairing
- In the Open collaboration Model as applied to open source software domain , project communities organise themselves without any external control or influence from a person or process.

2. Community driven development Model

- Open source software are developed by a large number of developers around the globe some of these developers are independent while some are supported by companies
- These group of people are often termed as “Community“
- A community can also be defined as a group of people
  - Which are diverse in nature
  - And engage in sharing ideas , work and experiences
  - Through a common platform for a common cause
- In the software industry, the word community driven development is broadly used to describe an initiatives in which a group of contributors or developers align their activities together and engage toward the development of open source software

Community – driven software development

- Mostly of the open source software development communities structure themselves into groups based on job roles.
- This is similar to the manner in which professional software organisation organise their employee
- Structures OSS community comprises of various jobs , roles and multiple sub teams.
  - Developer's group
  - Builders group
  - Testers group
  - Release management group

Community – driven software development

- Self driven OSS development communities , often provide flexibility to learn members to shuffle between job roles.
- These communities are open to customers or end users , inviting them to join the community and contribute to the project
- Agile mindset, self motivated, self organised

Process

Community Development process

Community Development process

- **Assert Requirements Design**
  This phase is all about the need for the software that is being proposed to the community for development and also the requirements during the development process are kept in mind in this phase. Along with it the design or the approach to be followed during the development is also a part of this phase.
- **Develop OSS Code**
  The requirements are fulfilled; the need has been understood and the design is ready. Now, it's time for the contributors to start developing the backbone or the skeleton of the software by the means of Coding.

Community Development process

- **Manage Configuration**
  Once the basic OSS Code is developed it is necessary to provide an initial configuration too is that its integration works perfectly fine along with all the features provided with that particular version.
- **Download and Install**
  Once the initial version is ready and configured properly it is ready to go in the markets for general use.
- **End-Use**
  The users who require the software to fall into this phase and use the OSS daily to provide experience and feedback to the developers or for personal benefits.

Community Development process

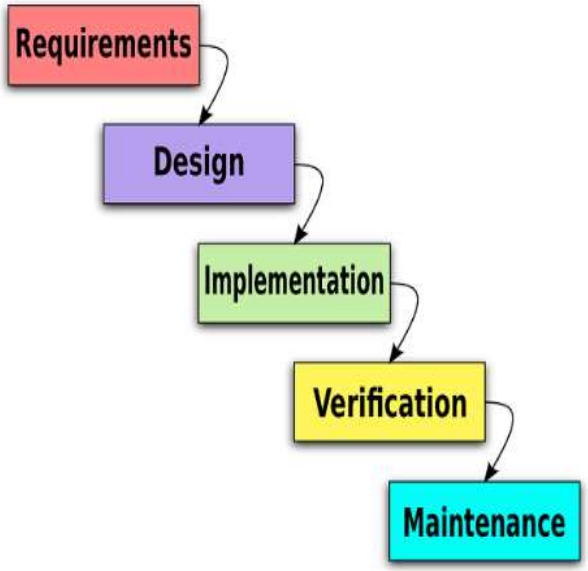
- **Communicate Experience**
  Once it is deployed in the market, the users share their experience with the OSS and give feedback, suggestions, and reviews on the functions that are good to be intact and also on the features that could be enhanced or added in the later versions of the software.
- **Read, Analyze and Redesign**
  Once the feedbacks are registered, it all comes back to the developers to work on the feedbacks, keep updating their Software and also track control of version.

Open Source Development Model process

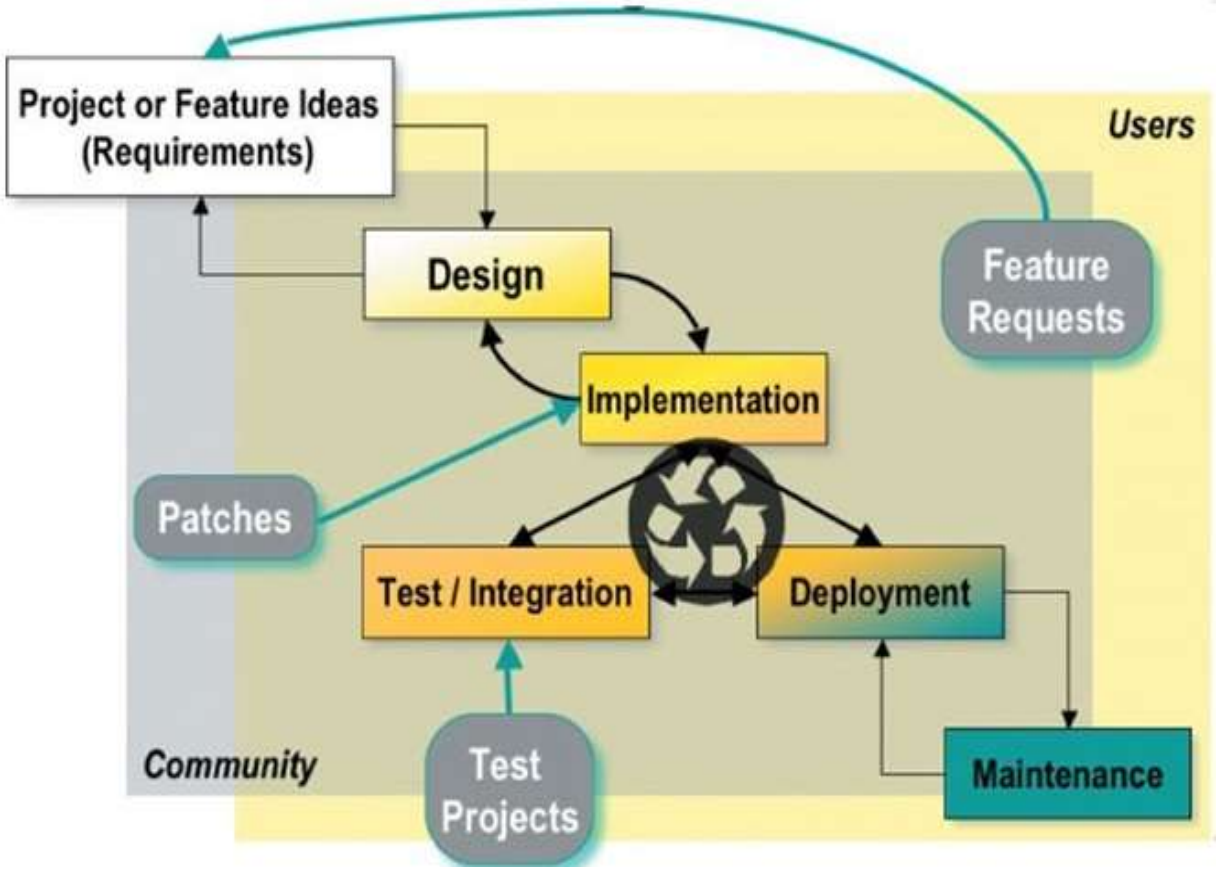
- **Open Source Development Model** Involves an interconnected OSS Community Development Process in which each stage or phase plays a vital role in building the ethics of the community, keeping contributions of each developer in mind, working with the latest technologies, keeping a track on the version control system and fixing the bugs in the software.

Open source software development : Process Model

- Waterfall Model / Linear development model
- OSSD : idea for a new project , or new features → design for proposed solutions – POC → implementations
- The moment the software runs (partially , mostly), it is released as development release (even though it may contain known and unknown bugs)
- In alignment of Philosophy : **“Release early, release often “**

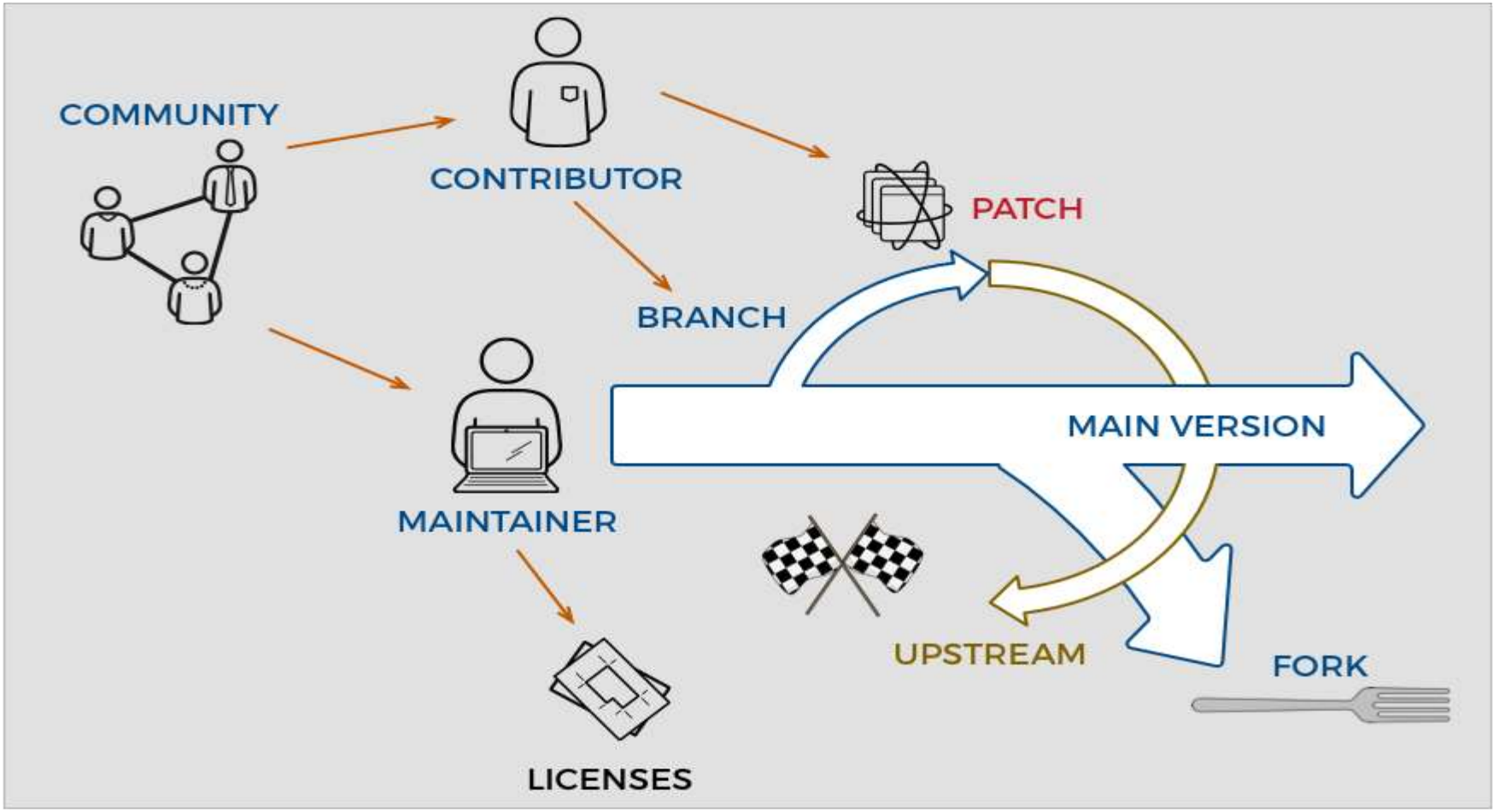


Open source software development : Process Model



- This model is followed when a request is made to the developers to build a product to best suit their needs to which the developers keep in mind certain factors and phases to deliver a fully functional and a product with no compromises to the client.

Open source software



Developers' group in OSSE

- Software Design – define the architecture and behaviour of the product
- Coding – implement the design
- A community driven software development project typically starts with the aim to address a typical or selected problem
- These problem (or pain points) specially the requirements of the software
- The designers group is responsible for creating software design which should be effective and appropriate
- The design should be self explanatory, easy to understand and created using standard notations like UML

Developers' Group in OSSE

- The next steps is to choose an appropriate language for working the source code
- Like in many other software development projects , in a community driven development also , the transition between design phase and coding phase typically tends to overlap with each other
- Developers usually begins implementing sub parts of a module or system, while other modules are still in their deign phase
- While this approach of running parallel phases (both design and coding) saves time, it might result in duplication of efforts in the event of a change in design.

Builders' group in OSSE

- The builders' group is responsible for taking care of the software build process which comprises of the following two tasks :
  - Compiling all source code in to object modules
  - Linking the object module code in to one whole
- Software building is continuous process , since developers continuously modify , add or delete software artefacts , which are placed in a common projects repository
- As a result , the build process also takes up a continuous form – since it is necessary to reflect these changes in the final build.
- The latest code needs to be recompiled and linked regularly

Builders' group in OSSE

- In order to speed –up this process , the underlying build process is usually designed to recompile only those source code files which were modified since the previous build was released
- For the remaining unchanged artifacts , the older compiled versions are picked and released.
- A large number of continuous integration and continuous deployment (CI/CD) tools exits that help to speed up to the build and release process .
  - Jenkins, TeamCity, Gitlab, CicleCI, TravisCI

Testers' group in OSSE

- Testing in the OSSE model is carried out by the testers' group of the community

May support :

- Mailing list, Discussion boards, Bug repots
- The testers group carries out testing , generate bugs reports which are shared with the developers / community – fixes are provided
- This cyclic process continues until the project community feels that the implementations is stable enough - an then it is released - development still continues

Refrences

https://www.stackinnovator.com/blog/what-is-opensource-anyway/
HOW TO CONSUME OPENSOURCE SOFTWARE IN ENTERPRISE?

Open Source Software Engineering

SE ZG587

BITS Pilani
Pilani Campus

Kumar Manish

9th Sept, 2023

Open source projects

Java open-source projects

Jenkins
Spring Framework
Elasticsearch
Bazel
Apache Tomcat

C++ open-source projects

Microsoft Cognitive Toolkit
IncludeOS
Kodi
SerenityOS
Monero

Python open-source projects

TensorFlow
Django
Flask
OpenCV
Ansible

https://opensource.google/
https://opensource.fb.com/projects/
https://www.gimp.org/ : Alternate of Photoshop
https://www.openoffice.org/ alternate of Microsoft office
https://www.alfresco.com/ alternate of Content management

Agenda : Recap

Lifecycle and methodologies in Open Source Software

- Open Collaboration Model
- Community Driven Development Model

Open Collaboration Model

- **Open collaboration** is "a system of innovation or production or development that relies on **goal oriented**,
- yet loosely coordinated,
- participants who interact to create a product (or service) of economic value,
- and make it available to contributors and non contributors alike."

Open Collaboration Model : principles

- **Equalitarianism :**
  - To specify that everyone can contribute.
  - social equality, prioritizing it for all people.
- **Meritocracy :**
  - To judge contributions, in a transparent manner, based on the merit or the quality of the contribution.
- **Self –organization :**
  - communities organise themselves without any external control or influence from a person or process.

Community Driven Development Model

- A group of contributors or developers align their activities together and engage toward the development of open source software
- These group of people are often termed as “Community“
- A community can also be defined as a group of people
  - Which are diverse in nature
  - And engage in sharing ideas , work and experiences
  - Through a common platform for a common cause
- Community comprises of various jobs , roles and multiple sub teams : **Developer’s group, Builders group , Testers group Release management group**

Community Driven Development Model

- **Self driven** OSS development communities , often provide flexibility to learn members to shuffle between job roles.
- These communities are **open to customers or end users , inviting them to join the community** and contribute to the project
- **Agile mindset, self motivated, self organised**

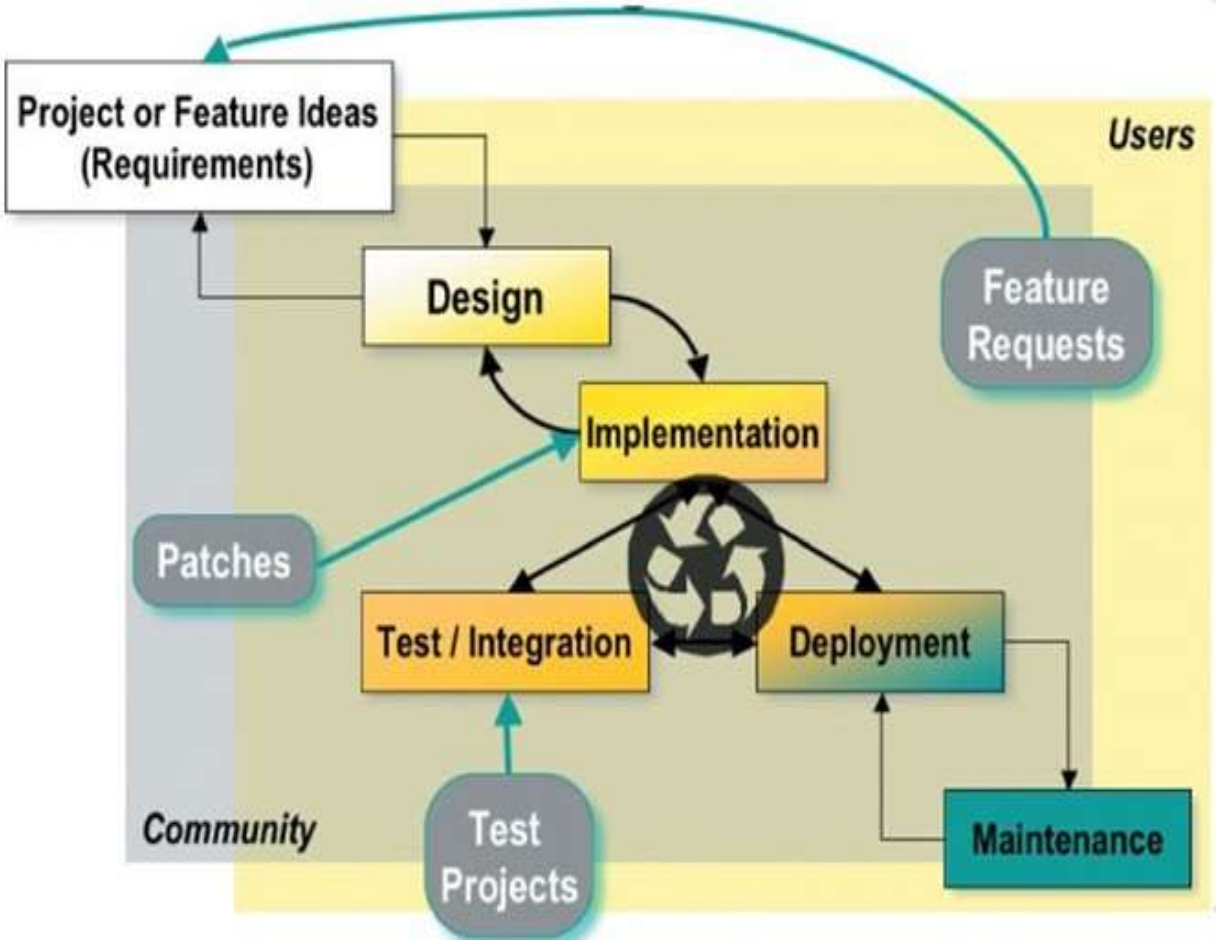
Community Driven Development Process

- Requirements & Design
- Develop Code
- Manage Configuration
- Download and Install
- End-Use
- Communicate Experience
- Read, Analyze and Redesign

Open Source Software Development Model

- **OSSD Model is different from the traditional waterfall or cascading model of software development.**
- **Open Source Development Model** Involves an interconnected OSS Community Development Process in which each stage or phase plays a vital role in building the ethics of the community, keeping contributions of each developer in mind, working with the latest technologies, keeping a track on the version control system and fixing the bugs in the software.
- The moment the software runs (partially , mostly), it is released as development release (even though it may contain known and unknown bugs) .
- In alignment of Philosophy : **“Release early, release often “**

Open Source Software Development Model



Release early, release often

- Idea for a new project or new feature or functionality
- Design for proposed solution
- Implementation by running (partially or fully)
- Development release (even bug)

Quick check

- Can open source software be used for commercial purposes ?
- Can I restrict how people use an open source licenses programs ?
- Can I call my programs : open source “ even if I don't use an approved licenses ?

Recap :

- **Open source :** Collaboration open and Source freely available
  - Freely : Share, Adapt, Modify and Collaborate
- **OSS :** Source code Released under a **License** in which the **copyright** holder grants users the rights to **use**, **study**, **change**, and **distribute** the software and its source code to anyone and for any purpose.
- **Key Principle of OSS :** Openness, Transparency, Collaboration, Expectations of Community
  - Release Early and Often : Rapid prototypes and iterative approach
- **Free software**" means software that respects users' freedom and community. It refers to free use of software and not price
- **OSI :** Definitions, Standard Requirements, Licenses
- **FSF :** To promote computer user freedom and defend the rights of all software users
  - Richard Stallman in 1983, by launching **GNU Project**

Recap :

Free software Four essential freedoms

- **A program is “free software” if its users have the following** :
  - Freedom 0 : The freedom to run the program as you wish, for an purpose
  - Freedom 1 : The freedom to study how the program works , and change it as per your requirements
  - Freedom 2 : The freedom to redistribute copies so you can help others
  - Freedom 3 : The freedom to distribute copies of your modified versions to others
    - **Pre-requisite “ Access of the source code”**
- **Legal considerations :**
  - The owner of the free software does not have power to withdraw or invalidate the license , or add additional restrictions to its terms and conditions,
- **License terms should be permanent and irrevocable**
- **License must not restrict other software**
- **License must be technology neutral**

Recap :

	Free software	Open source Software	Freeware	Public domain software
Definition	“FREE” is a matter of liberty , not price	“OPEN” does not just mean access to the source code; more about collaboration	Free refers to price , while freedom of the use is restricted by creator	PUBLIC domain belongs to the public as a whole
Philosophy	Social movement	Development methodology	Marketing goals	Copyright disclamation
Rules	Four freedoms	Open source Initiatives		Creative common Org
Free of charge	Not necessary	Not Necessary	YES	YES
Copyright law	YES	YES	YES	NO
Examples	Linux, Ubuntu , MySQL, Apache	Linux, Ubuntu , MySQL, Apache	Skype, Adobe acrobat	SQLite

Recap :

- Understanding Intellectual property Rights (**IPR**) to the software industry
  - **Patents** – used to protect functional features , like hardware configurations etc
  - **Copyrights** – used to protect works of authorship, like source code , diagram etc
  - **Trade secrets** – used to protect internal business secrets, like business and pricing models
  - **Trademarks** - used to protect brand recognition, through logo etc
- All the people involved in a development project, may claim for the ownership of the IPR for the various artefacts developed as a part of the project.

- **Understanding Software Licenses**
  - **Licensing Models in OSS:**
    - Copyright, Copyleft, Permissive, Creative Commons

Recap :

Copyleft or Protective Licensing Model :

- To make a work (or program) free to use, modify , adapt or extend.
- Aligned with the **four essential freedoms**
- No re-licensing allowed, No commercial usage allowed
- All modified and extended version of the work) or program) should be free as well - hence called **protective licenses**
- All derivative works should be attributed to the creator, open sourced and copyleft

Variants of Copyleft : **Strong and Weak copyleft :**

- The strength of the copyleft license is decided based on the extent its provision are imposed on the derived works
- Most commonly, weak copyleft licenses are used to create **Software libraries**
- Help software to link to the library and redistributed without requirement for the linking software to also be copyleft –licensed

Examples :

Strong Copyleft :

GNU general public License

Weak copyleft

GNU lesser General Public License –

Mozilla public license

Recap

Permissive Licenses

- Free software licenses with only minimal restrictions on how the software can be used , modified and redistributed (also called Berkeley software distribution : BSD - like or style licenses)
- Rules for usage - what ever user wants, but with few restrictions – derived works must be attributed to the creator
- Source code need not be open or made available in the public domain
- Re-licensing allowed - derivative works can be release under any other licenses or used as proprietary products
- Allows commercial usage
- Examples : GNU All permissive License, MIT License, BSD licenses, Apple Public source licenses. Apache license

Recap

Copy left (Protective Licenses)	Permissive Licenses
Publication of software code of all modified versions or derived works under the original copyleft licenses / protective licenses	Provides No guarantee that derived works of the software will remain free and publicly available ; generally requiring only that the original copyrights notice be remained
	- Hence derived works , or future versions , of permissively – licenses software can be released as proprietary software - Permissive licenses offer highly extensive license compatibility as compared to copyleft licenses - Wider adaptability in the open source community

Recap

Choosing Open Source Licensing Models : Based on Permissions, Distribution, Modification, Use.

- Option 1: Work with a community –
  - Contributors License Agreement (CLA)
    - Simple CLAs vs Detailed CLAs
    - Individual CLAs vs Corporate CLAs
- Option 2: Keep it simple and permissive - Choose the MIT License ,
- Option 3: Need to share improvements
- Option 4: Work without a license

Why to choose Open Source Licensing Models?

- To protect works contributed in the open source domain
- To protect contributors and users from any copyright infringements
- To Interested developers or contributors or business will not touch a project without a license protection.

Recap

Open Source Business Model

1. **By Intellectual Property :**

- Dual Licensing Model –
  - Community version – under GPL
  - Enterprise version – under commercial license
- Open Core Model : Core portion of the software, which provides the basic functionality is made available as open source

2. **By Services and other Business model**

- **Open source SaaS model** – example WordPress, Sharetribe...
- **Selling services** : such as consulting service , technical support service, or training
- **Donation, Users advertisement, and Merchandise**
- **Crowd Funding and Crowd Sourcing**
- **Freemium Business Model** - Feature limited - basic Features vs. Extended features Time limited – fixed duration after that paid versions

References and further readings

- Free software foundations https://www.fsf.org/
- GNU https://www.gnu.org/
- Open Source Initiative https://opensource.org/
- Open Source Resources https://opensource.com/
- Open Source Guides https://opensource.guide/
- Creative commons https://creativecommons.org
- Choose a license https://choosealicense.com/
- Copyleft https://copyleft.org/

References and Recommended Reading:

- How Developers Can Make money with Open Source Projects

https://rubygarage.org/blog/how-make-money-with-open-source-projects

- What Motivates a Developer to Contribute to Open-Source Software?

https://clearcode.cc/blog/why-developers-contribute-open-source-software/

- How do Open Source Programmers make money

https://www.thewindowsclub.com/open-source-companies-programmers-make-money

- **https://en.wikipedia.org/wiki/Business_models_for_open-source_software**

• Bounty Source https://bountysource.com/ for Cloud Funding
•Gunio https://gun.io/