Open Source Software Engineering

SE ZG587

BITS Pilani
Pilani Campus

Kumar Manish
2 Sept, 2023

innovate achieve lead

BITS Pilani
Pilani Campus

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.	

innovate achieve lead

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

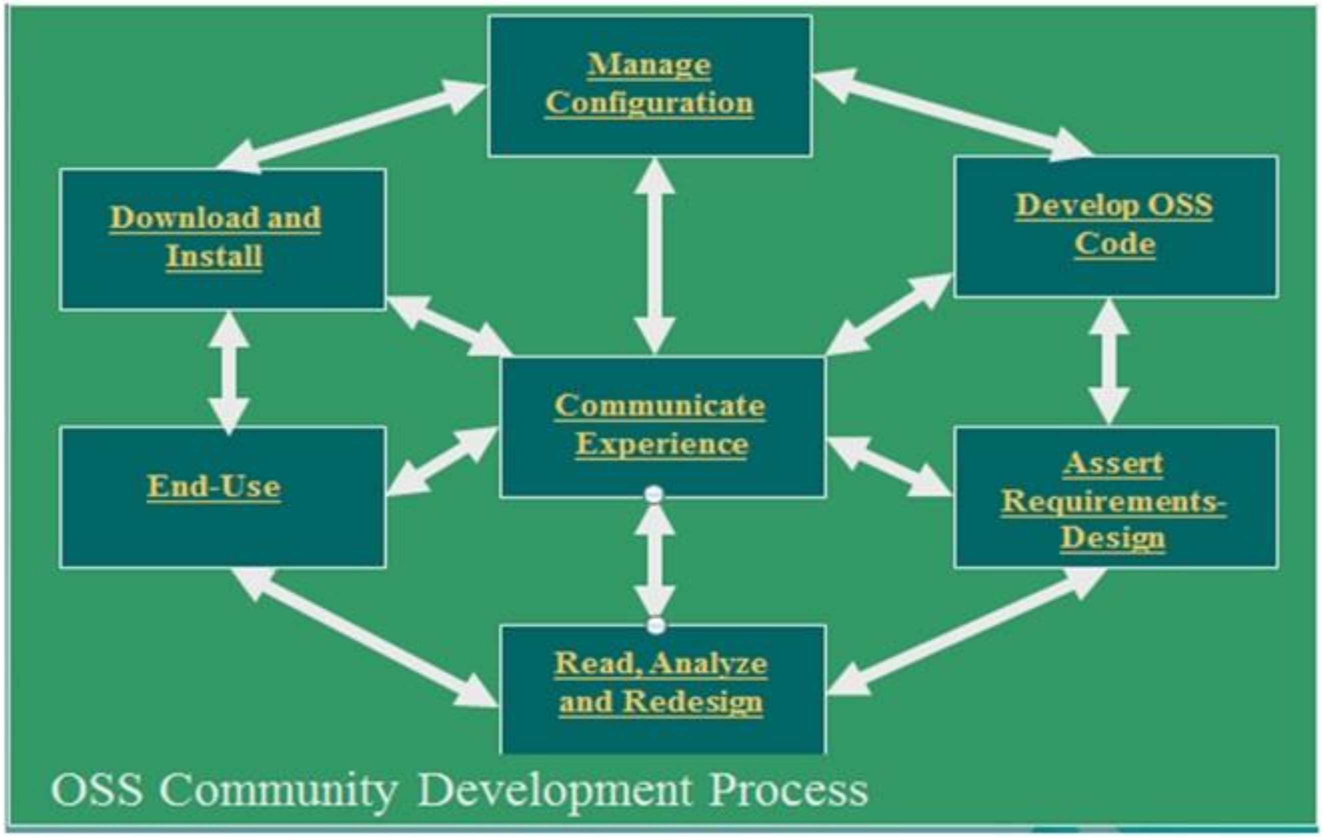
- Mostly of the open source software development communities structure themselves into groups based on job roles.
- This is similar to the manner in which professional software organisation organise their employee
- Structures OSS community comprises of various jobs , roles and multiple sub teams.
 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



OSS 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

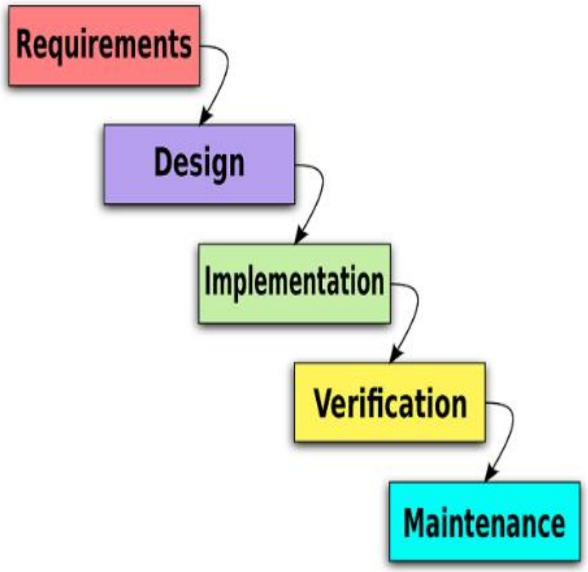
- **Communicate Experience**
  Once it is deployed in the market, the users share their experience with the OSS and give feedback, suggestions, and reviews on the functions that are good to be intact and also on the features that could be enhanced or added in the later versions of the software.
- **Read, 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

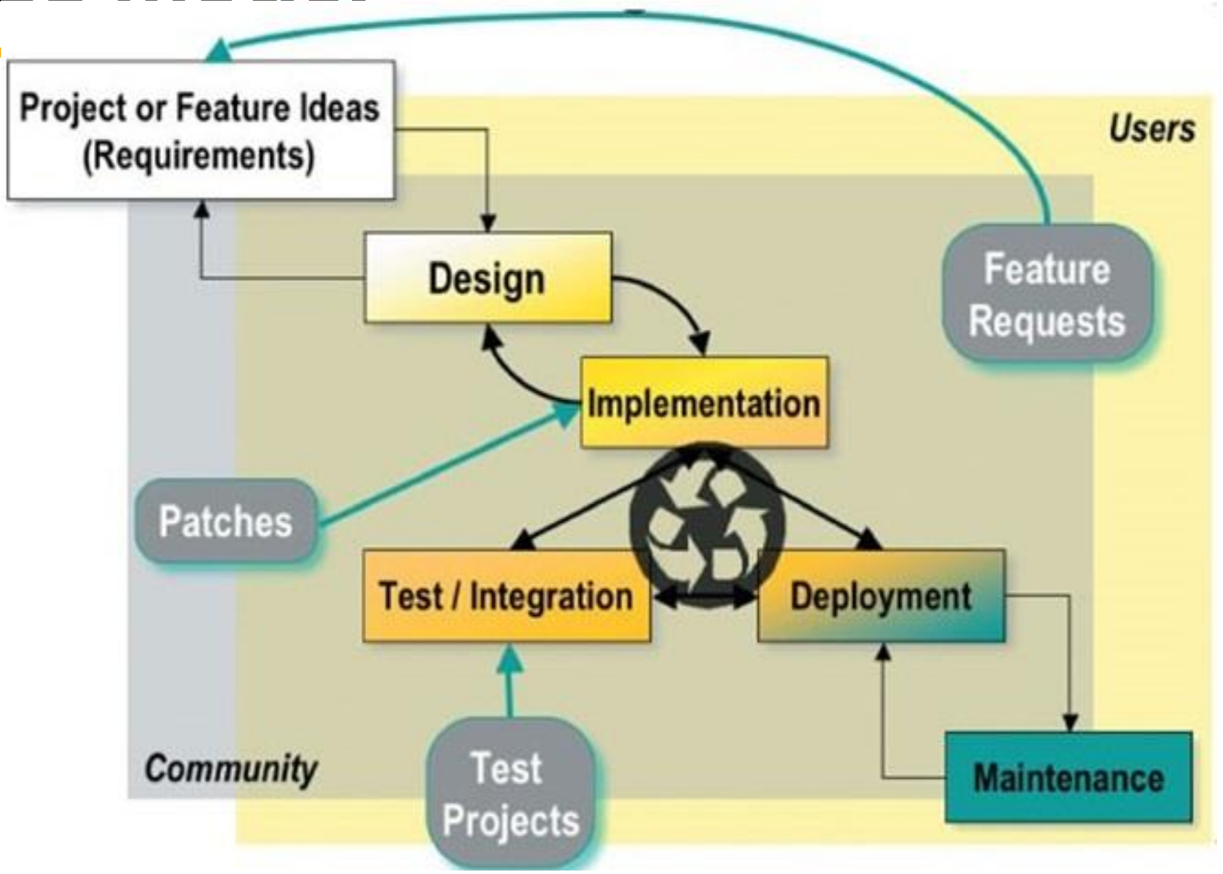
- **Open Source Development Model** Involves an interconnected OSS Community Development Process in which each stage or phase plays a vital role in building the ethics of the community, keeping contributions of each developer in mind, working with the latest technologies, keeping a track on the version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

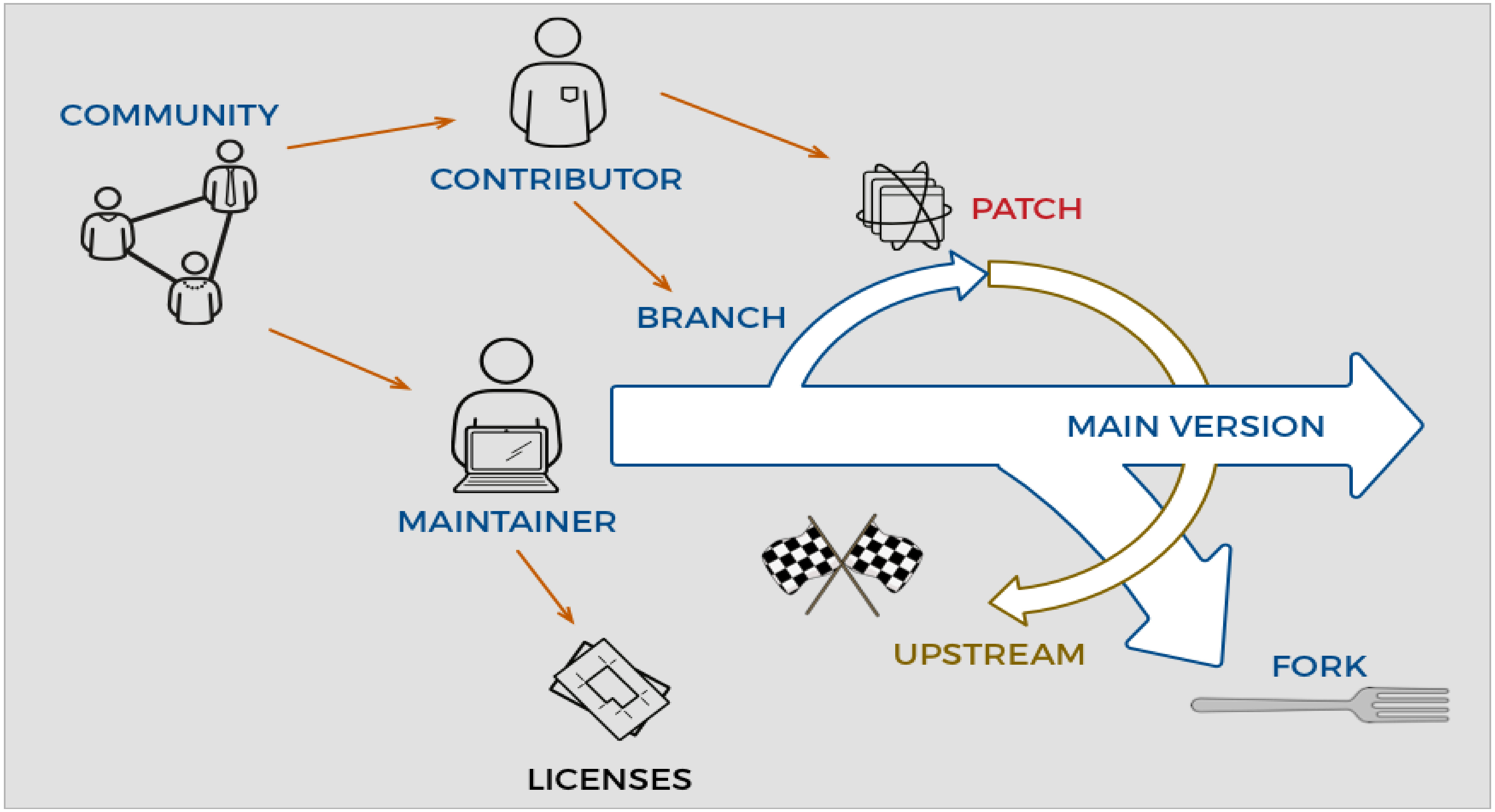


Open source software development : Process Model



- 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?