TABLE OF CONTENTS

Open Source Software Engineering

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Welcome & Introduction

- **About the Instructor**
  - Kallol Pal
  - Contact No: 98451 73500
  - E-Mail:
    - kallolpal@wilp.bits-pilani.ac.in

What to Expect From the Course?

What are your expectations from the course?

- Understanding the basic and advanced concepts of Open Source Software Engineering
- Awareness of the Open Source Movement, it's philosophy and history
- Understanding of various licensing models and issues with Open Source Software
- Learn about the Open Source processes, development methods and associated tools

Outline of the Course

- Introduction to Open Source Software
- Understanding Open Source Licensing Models
- Understanding Open Source Business Models
- Life Cycle Methodologies in Open Source Software
- Contributing to Open Source Software Projects
- Building and Scaling Open Source Ecosystems
- Consuming Open Source Projects
- Risks of Open Source Software
- Open Standards
- Open Access to Knowledge and Creativity

Courseware

- Course Handout
  - Plan of Self Study
- Text Books
  - Open-Source Projects - Beyond Code A blueprint for scalable and sustainable Open-Source projects John Mertic - 2023
  - Producing Open-Source Software: How to Run a Successful Free Software Project, 2nd edition, Karl Fogel
  - Software Transparency: Supply Chain Security in an Era of a Software-Driven Society, By Chris Hughes, Tony Turner · 2023

Evaluation Scheme

No	Name	Type	Duration	Weight
EC-1	Quiz – I	Online		5%
	Quiz – II	Online		5%
	Assignment	Online		20%
EC-2	Mid Semester Exam	Closed Book		30%
EC-3	Comprehensive Exam	Open Book		40%

Open Source Software Engineering
(Introduction to Open Source Software)

What is Open Source Software?

- *Open Source Software (OSS) is developed and maintained through Open Collaboration*
- *It allows anyone to use, examine, alter and redistribute the software freely*
- *Unlike proprietary software, OSS encourages transparency and community participation*

Characteristics of Open Source Software

- *Source Code Availability*
  - *Freely accessible code*
  - *Transparency Fosters Collaboration & Innovation*
- *Collaborative Development*
  - *OSS projects involve a global community of programmers*
  - *Developers collaborate to work in these projects*
- *Example of Open Source Software*
  - *Operating Systems: Android, Linux*
  - *Browsers: Firefox, Chromium*
  - *IDEs: Visual Studio Code, Android Studio, PyCharm*

Open Source Community and Contributions

- *Self governing , worldwide community of OSS Developers*
- *Decisions are collaborative*
- *The dynamics of the Open Source Community makes open source projects sustainable*
- *Contributions play a key role – Documentation, UI/UX – Goes Beyond Code*
- *Meetups – Enhances collaboration*

Open Source Software and License

From Wikipedia

- *Open Source Software is a type of computer software in which the source code is released under a license.*
- *In which the copyright holder grants users the right to use, study, change and distribute the software to anyone and for any purpose*
- *Open Source Software is usually developed in a collaborative public manner*
- *Open Source Software is a prominent example of open collaboration*

Benefits of Open Source Contributions

- *Skill Enhancement – Real World Coding Experience & Learning new programming skills*
- *Mentorship Programs – Guidance for first time contributors & Mentorship*
- *Immediate feedback*
- *Global Connections & Networks*
- *Financial Renumeration*
- *Enhances Career Opportunities*

Origin & Evolution of Open Source Software

Need for and the Significance of 'Openness'

Contrast with Proprietary

Unix and its contribution to the Open Source Movement

Apache & Open Source Initiative

Seeds of Collaboration (1950s to 1980s)

1950s to 1960s

- *Early Collaboration among universities and research institutions*
- *Sharing of source code as a norm for customization and bug fixing*

1970s

- *Unix & C*

1980s

- *GNU – G Not Unix*
- *GNU Public License (GPL)*
- *Richard Stallman – Software Programmer at MIT AI Labs - Vision of free software and user freedom – Emphasis on ethical and philosophical aspects of software freedom – Launched the GNU Project - Started the Free Software Foundation – Created GCC*

Open Source takes Centrestage

1991

- *Linux (Linus Torvalds)*

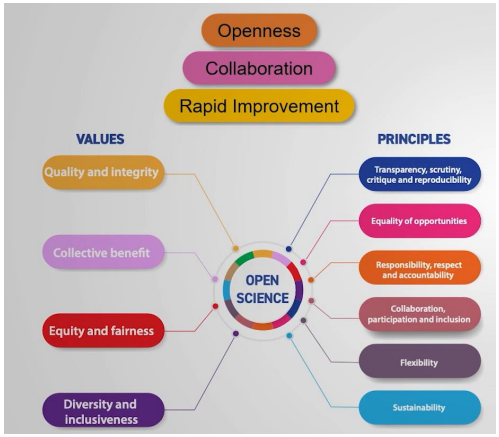
1998

- *OSI (Open Source Initiative) – Coined the term "Open Source" in place of then prevalent "Free Software Movement"*
- *Databases (MySQL), Web servers (Apache), Programming Languages (Python)*

The future of Open Source

- *Continued Growth and Impact across various industries*
- *Rise of Collaborative Development Platforms like GitHub*
- *Will continue to drive significant developments in areas like Cloud Computing, AI, ML, etc.*

Core Principles of Open Source



Core Principles of Open Source

- ***Openness***
  - *The foundation of open source lies in the accessibility of source code. Unlike proprietary software where the code is hidden, open source projects make their code freely available for anyone to inspect, modify and distribute.*
- ***Transparency***
  - *Ability of the community to see the current progress and future plans. Project roadmap is made available, a defect tracking system is put in place and all project artifacts are published.*
- ***Collaboration***
  - *Open Source thrives on the power of community. Developers from all over the world can contribute to a project, share their expertise, and work together to improve the software.*
- ***Rapid Improvement***
  - *With a global community constantly tinkering and innovating, open source projects benefit from a faster development cycle. Bugs are identified and fixed quicker, and new features are continuously being added*

Core Principles of Open Source

- ***Cost Effectiveness***
  - *Open Source is often free to use and modify, making it a cost effective solution for individuals and businesses alike. This can be a game changer for startups and organizations with limited budgets.*
- ***Security***
  - *With many eyes scrutinizing the code, open source projects tend to have a higher level of security. Vulnerabilities are identified and addressed quickly by the community.*
- ***Innovation***
  - *The open source model fosters a culture of innovation. Developers are free to experiment and build upon existing code, leading to the creation of new features and functionalities.*
- ***Flexibility***
  - *Open Source Software provides users with the freedom to modify the code to suit their specific needs. This level of customization is not possible with proprietary software.*

Core Principles of Open Source

- ***Early & Often***
  - *Any changes proposed or made by any one are made public immediately*
  - *Contributions are expected to occur early – resolving errors early in development lifecycle*
  - *Contributions are expected to occur often*
- ***Expectation of Community***
  - *Participants in an Open Source project have expectations of a community to be formed that works together to contribute to the development of the project*

Open Source Beyond Software

- ***Hardware:***
  - *Schematics are designs are freely available.*
- ***Open Science:***
  - *Researchers are increasingly embracing open source practices, sharding their data and methodology to accelerate scientific discovery*
- ***Open Education:***
  - *Open Source Educational resources are providing free and accessible learning materials to students around the world*

Examples

- *Linux*
- *Mozilla Firefox*
- *Python*
- *Java*
- *Apache HTTP Server*
- *GIT*
- *Open Office*

Definition of Open Source Licensing

- *Open source licensing refers to licenses that allow users to freely use, modify and distribute software's source code*
- *The origins of open source licensing can be traced back to the 1960s and 1970s*

Richard Stallman and the GNU Project

- *Richard Stallman launched the GNU project in 1983, advocating for free software and the right to modify and distribute it*
- *General Public License (GPL)*
  - *Was created in 1989, a landmark open source licensing*

Berkeley Software Distribution (BSD)

- *The Berkley Software Distribution (BSD) License originated from the University of California, Berkeley, in the late 1970s.*
- *Permissive licensing model*
  - *Allowed for reuse of open source software in Commercial Products*

Open Source Initiative (OSI)

- *Open Source Initiative (OSI) founded in 1998 to promote and protect open-source software*
- *OSI created a set of criteria defining open source software, ensuring it remains freely available for use, modification and distribution*

Types of Open Source Licensing Models

- *Free*
  - *Grants the user the freedom to run, copy, distribute, study, change and improve the software*
  - *GNU, Apache, MIT*
- *Open Source*
  - *Software with source code that anyone can inspect, modify and enhance*
  - *Transparency, collaboration and community driven*
  - *Linux, Firefox*
- *Freeware*
  - *Software that is available for use at no monetary cost*
  - *Users are not allowed to modify or redistribute the software*
  - *Skype, Acrobat Reader*
- *Public Domain*
  - *Software that is not protected by copyright and can be freely used, modified and distributed by anyone*
  - *No restrictions, No attributions required*
  - *SQLite*

Intellectual Property Rights (IP)

- *What is Intellectual Property Rights?*
  - *Refers to the legal right that protect creations of the mind, such as inventions, literary and artistic works, designs and symbols.*
- *Any software involves IP rights*
  - *Ideas*
  - *Designs*
  - *Solutions*
- *Types*
  - *Patents*
  - *Copyrights*
  - *Trademarks*

Software Licenses

A legal agreement that governs the terms and conditions of the software usage, distribution and modification

- *Purpose of software licenses is regulating the use of software*

Types

- *Proprietary*
  - *Restrictive - EULA*
- *Open Source*
  - *Not restrictive, but with conditions*
- *Free (Not Cost, But License!)*
  - *Non restrictive, very similar to Open Source*
  - *Every free software is open source, every open-source software is not free*

Why licenses in Open Source?

- ***With Freedom comes responsibility***
  - *Provides clear guidelines on how the software can be used modified and distributed*
- ***Open Source Initiative (OSI)***
  - *Defines 10 criteria*
  - *One of the key criteria is the presence of an OSI Approved license*
    - *Use the software for any purpose, including commercial use*
    - *Access the source code and modify it*
    - *Redistribute the software, with or without modifications*
    - *Distribute derivative work based on the software*

Categories of Open Source Licenses

- ***Permissive Licenses***
  - *Grant the most freedom*
  - *Allow the user to modify and redistribute the software under any license, including proprietary licenses*
  - *MIT, BSD, Apache*
- ***Copy Left Licenses***
  - *Enforces a form of reciprocity*
  - *Require the derivative works based on the software to also be open source and use the same or a similar license*
  - *GNU General Public License (GPL) & GNU Lesser General Public License (LGPL)*

Open Source Software Engineering
(Open Source Licensing Models)

Free Software

- ***Freedom to run the program for any purpose***
- ***Freedom to study and modify the program to make it do what you want***
- ***Freedom to redistribute copies***
- ***Freedom to improve the software and release your improvements to the public***

- ***Advantages***
  - *Cost Effective (No Fees)*
  - *Transparency (Access to Source Code)*
  - *Security*
  - *Customization*
  - *Innovation*

Open Source Software

- ***Access to source code***
- ***Freely use, modify and distribute the software***
- ***Fosters collaboration and innovation***
- ***The terms of use are clearly stated on the open source license***

- ***Advantages***
  - *Cost Effective (No Fees)*
  - *Flexibility*
  - *Security*
  - *Innovation*

Open Source Licenses

- ***Permissive***
  - *Users can freely use, modify and distribute the software even for commercial purposes*
  - *No obligation to share modifications*
- ***Copy Left***
  - *Users can freely use, modify and distribute the software*
  - *Modifications must be shared under the same or compatible license*

Freeware

- ***Type of open source license that grants users freedom to use the software for free, but with limitations compared to other open source models***
- ***Software licensed for free use, typically for non-commercial purposes***
- ***Users can download install and run the software without paying a fee***
- ***Source code is not available***
  - *Limited Support*
  - *No Modification*
  - *Limited Support*
  - *Bundled Software*
- ***Advantages***
  - *Cost Effective (No Fees)*
  - *Variety*
  - *Try Before you Buy*
  - *Optional, Community Support*

Public Domain Software

- ***There are no copy right restrictions***
- ***Anyone can freely use, modify and distribute the software without permission or paying royalty***
- ***No one takes responsibility for the software***
- ***Support could be lacking***
- ***Advantages***
  - *Unrestricted Use*
  - *Free to Modify*
  - *Flexibility*
  - *Collaborate and Innovate*
  - *Cost Effective*

Open Source Software Engineering
(IP Rights & Software Licenses)

Patents

- ***Novel & Not Obvious***
- ***Patentable: Algorithms, Data Structures, User Interfaces, Business Methods***
- ***Not Patentable: Ideas, Mathematical Formula, Abstract Principles***
- ***Benefits***
  - *Exclusive Rights*
  - *Competitive Advantage*
  - *Revenue Stream*
- ***Challenges***
  - *Enforcing*
  - *Rapid evolution on the industry*
  - *Patent Trolls*
  - *Anti open source?*

Copyright

- ***Purpose is to protect a software***
- ***Legal right that protects original work of authorship***
- ***It grants exclusive rights to reproduce the work, create derivative woks, distributing copies, etc.***
- ***Source Code, User Interface, Documentation cannot be copied***
- ***Public domain material and knowledge cannot be copyrighted***
- ***Fair Use: Copyright law includes a doctrine called fair use, which allows limited use of copyrighted materials without permission from the copyright holder***
  - ***Purpose***
  - ***Nature***
  - ***Amount***
  - ***Effect***

Trademarks

- ***Protects a Brand***
  - *User Interfaces, Logos, Key identifiers for the software*
  - *Trademarks Protect these identifiers*
- ***A distinctive mark (symbol) that identifies / distinguishes one party's goods and services from that of others***
  - *Logos*
- ***Trademarks are vital for brand recognition and building consumer trust***
- ***Trademarks need to be registered***

Trade Secrets

- ***Trade Secrets offer a powerful tool to protect confidential information giving you a competitive edge***
- ***Information that is commercially valuable, Not generally known or readily ascertainable by legitimate means***
- ***May include Specific algorithms or formulas, Source Code, Customer Lists, etc.***
- ***Enforced via Non Disclosure Agreements (NDA), Limiting Access (Need to Know), Secure Storage, Activity Monitoring, etc.***
- ***Help protect innovation, provide competitive advantage***

Open Source Software Engineering
(Licensing Models in OSS)

Copyright

- ***Copyright grants the creator of an original work exclusive rights***
  - *Reproduction*
  - *Distribution*
  - *Creation of Derivative Works*
- ***In open source software the copyright holder grants permission to use, modify and distribute software under specific terms***
  - *Terms are listed in the OSS License*
- ***Common OSS Licenses include***
  - *GNU General Public License (GPL) – Requires derived works to be open-source as well*
  - *MIT License – Permissive License allowing for both commercial and non-commercial use*
  - *Apache License – Similar to MIT but with additional terms regarding patent contribution*

Copyleft

- ***Modifications and Derivative works created from the original code must also be distributed under the same copyleft license.***
  - *This ensures that the software remains open source!*
- ***Advantages***
  - *Promotes continuous development*
  - *Ensures long term open source*
  - *Breeds innovation*
- ***Popular Copy Left Licenses***
  - *GNU General Public License (GPL) – Requires derived works to be open-source as well*
  - *Lesser GPL (LGPL) – The open source has to be used as a separate 'entity', all modifications made to the opensource must remain open source*
  - *Mozilla Public License (MPL) – The Core MPL code must always remain open source*

Permissive Licenses

- ***No obligation to share modifications made on open source code back to the open source community***
- ***Advantages***
  - *Freedom of use – unrestricted use including in commercial applications*
  - *Modification Liberty*
  - *Unrestricted Distribution*
  - *Seamless integration of Open Source with Commercial applications*
- ***Popular Permissive Licenses***
  - *MIT License – Ideal for small projects and code snippets*
  - *BSD3-Clause New or Revised License – Requires users to retain copyright and license notices and to not endorse the licensor*
  - *Apache License 2.0 – Business Friendly permissive license*
  - *Eclipse Public License – Designed for collaborative development, allowing for linking proprietary code but requiring code disclosure under certain conditions.*

Creative Commons Licenses

- ***Intended to bridge the gap between Copyright and Open Source (Copy Left)***
  - *Objective is to uphold open source principles*
  - *Attribute the creators*
- ***Key Aspects of Creative Commons Licenses***
  - *Attribution – Credit the Creator*
  - *NC (Non Commercial) – Prohibits commercial use*
  - *Share Alike – Derived work must use the same licenses*
  - *Non Derivative Works (ND) – Derivative work not allowed – Use As Is*
- ***Build your own license***
  - *BY: Most Permissive – Allows use by attribution*
  - *BY-NC: Non Commercial Content*
  - *BY-SA: Encourages Derivative work while requiring the same license*
  - *BY-ND: Restricts modifications but allows sharing with attribution*

Open Source Software Engineering
(Open Source Business Models)

Why Open Source Business Models?

- *Building a sustainable business is key to the success of an open source project!*
  - *Costs need to be recovered*
  - *Core team members need to derive monetary value*
  - *Key Contributors need to be rewarded for sustainability*

Open Source Revenue Streams

- *Support & Maintenance*
- *Training & Certification*
- *Subscription Services*
- *Custom Development*
- *Donations & Grants*

What is dual licensing?

- *It refers to making software under two different licenses*
  - *A combination of an Open Source and a Proprietary License*
- *Provides a choice for users based on their needs*

Advantages of Dual Licensing

- *Wider Adoption*
- *Increased Visibility*
- *Monetization Potential*
- *Improved Quality*

Considerations for Dual Licensing

- *License Compatibility*
- *License Proliferation, especially when multiple open source Licenses are involved*
- *Community Management*

- *Dual Licensing Offers a Strategic approach to open source business models*
- *Provides Balance Between open source principles and commercial opportunities*

Open Core Business Model

- *It is a popular approach for Monetizing open source software*
- *It offers a foundational set of features as free, open source software*
- *Additional features and functionalities are offered as proprietary, commercial products*
- *The open source core allows for community development and adoption*
- *Commercial offerings provide revenue generation and support for ongoing development*

Advantages & Disadvantages of Open Core

- *Advantages*
  - *Faster Development*
  - *Increased Adoption*
  - *Improved Quality*
  - *Reduced Development Cost*
- *Disadvantages*
  - *Complexity of maintaining two code bases*
  - *Potential for community conflict*
  - *Free-riding*
  - *Vendor lock-in for commercial features*
- *Examples*
  - *GitLab*
  - *MongoDB*

Open Source Software Engineering
(Selling Users, Services & Merchandise)

Monetizing Open Source

- *Businesses can generate revenue beyond just selling the software itself*
  - *Selling to Users*
  - *Offering Services*
  - *Create and Sell Merchandise*

Consulting & Customization Services

- *Freemium Model*
  - *Offer the basic version for free with limited features, encouraging the users to upgrade for premium functionalities*
- *Subscription Model*
  - *Provide access to advanced features, ongoing support and exclusive content through monthly or yearly subscriptions*
- *In-App Purchases*
  - *Integrate optional functionalities or cosmetic upgrades within the open source software for users to purchase within the application*
- *Consulting*
  - *Offer expert advise on deploying, integrating and customizing open-source software to meet specific user needs*
- *Trainings & Workshops*
  - *Provide educational services to help users understand and leverage open source software effectively*
- *Support & Maintenance*
  - *Offer ongoing support, bug fixes and security updates to ensure smooth operation of the open source software*

Tailoring the Open Source Experience

- *Custom Development*
- *Integration Services*
- *Theming & Branding*

Support & Maintenance Services

- *Support & Maintenance Services is a powerful strategy, offering valuable assistance to users and driving sustainable business growth for open source providers*
  - *Technical Support – Addressing user issues, bug fixes and trouble shooting*
  - *Performance Optimization – Tailoring the software to peak performance specific to user needs*
  - *Security Updates – Ensuring ongoing protection against vulnerabilities and threats*
  - *Customization and Integration – Adapting the open source software to user workflows and other systems*
  - *Training and Documentation – Providing educational resources to users to maximize software utilization*
- *Why choose support and maintenance services?*
  - *Increased ROI – Ensures smooth operation, reducing downtime and maximizing user productivity*
  - *Enhanced Security – Provides ongoing security updates, protecting users from vulnerabilities*
  - *Reduced Risk – Mitigates the risk of technical issues impacting business operations*
  - *Scalability and Flexibility – Tailored support adapts to user growth and evolving needs*
  - *Access to Expertise – Leverages the open source provider's deep knowledge*

Building a Support Model

- *Define Service Tiers*
- *Service Level Agreements (SLAs)*
- *Invest in a Skilled Support team*
- *Establish Effective Communication Channels*
- *Continuous Improvement*

Merchandise and Swag

- *Building Brand Loyalty – The power of Merchandise*
  - *Build brand loyalty, generate revenue and foster a string community*
- *How do Merchandise help?*
  - *Brand Awareness*
  - *Community Building – Develops sense of belonging*
  - *Revenue Generation*
- *Merchandising Ideas*
  - *T-Shirts & Sweatshirts*
  - *Stickers & Buttons*
  - *Mugs & Tumblers*
  - *Hats & Bags*
  - *Limited Edition Items*
- *Attributes*
  - *Simple & Eye-Catching, Project Specific, High Quality, Comfortable & Functional*

Marketplace for Add-ons

- *A platform where developers can publish and sell add-ons that extend the functionalities of core open source software*
- *Users can discover, purchase and integrate these seamlessly with their existing open source platforms*
- *Can be hosted by the open source project itself or by a third party vendor*
- *Benefits for Developers*
  - *Monetization Opportunity*
  - *Wider Reach*
  - *Community & Collaboration*
- *Benefits for Users*
  - *Customization & Flexibility*
  - *Increased Functionality*
  - *Reduced Development Time*

Donations

- *Users get an opportunity to show their support and contribute to the project's continued development*
  - *Directly contribute funds*
  - *Done by users who appreciate and benefit from the project*
- *Effective donation campaigns have the following options*
  - *Transparency*
  - *Multiple Donation Options*
  - *Recognitions with permission*
  - *Story Telling – Outcomes of donations made*
- *Attributes*
  - *Sustainability – Donations Fluctuate*
  - *Community Management*
  - *Consider Alternatives*

Corporate Sponsorship

- *Nature of Corporate Sponsorship*
  - *Corporate sponsorship offers a win-win situation for both companies and open source projects*
  - *Corporate sponsorships provide a reliable source of income to support development and maintenance*
  - *Increased Visibility for Projects*
  - *May also sponsor Expertise and Resources*
- *Why Corporate Sponsorship?*
  - *Brand Recognition*
  - *Access to Talent*
  - *Influence Development*
- *Types of Corporate Sponsorship*
  - *Financial*
  - *In-Kind (Hardware, Software, etc.)*
  - *Technical*
  - *Marketing*

Building a Corporate Sponsorship

- *Identify Your Needs*
- *Target the Right Sponsor*
- *Develop Sponsorship Packages*
- *Build Relationships*
- *Showcase the Value*

Sustainability is key to a successful Corporate Sponsorship

Crowdfunding Campaigns

- *Allows developers to raise money directly from the community who benefits from their work*
- *Benefits*
  - *Direct Funding to the project*
  - *Community Building*
  - *Provides validation for the open source project*
  - *Flexibility*
- *Crowdfunding platforms for open source*
  - *Kickstarter*
  - *Indiegogo*
  - *GitHub Sponsors*
  - *Open Collective*

Crafting Crowdfunding Campaigns

- *Clear and Compelling Description*
- *Engaging Rewards*
- *Community Building*
- *Transparency*

Grant Programs

- *Can be a powerful tool for businesses to leverage the open source community and its development models*
- *Types of Grants*
  - *Community Development Grants – Support the growth and engagement of open source community*
  - *Project Grants – Fund specific development efforts for open source projects*
  - *Infrastructure Grants – Provide resources for building and maintaining open source infrastructure*
- *Benefits*
  - *Access to funding*
  - *Community Growth*
  - *Increased Visibility*
  - *Enhanced Credibility*
- *Choosing a Grant Program*
  - *Align Project Goals, Funding Amount Required, Eligibility Requirements, Reporting Requirements*

Open Source Software Engineering
(Other Business Models)

Software as a Service (SaaS)

- *SaaS enables Open Source on the Cloud*
- *It is Cost Effective & Scalable – Enables easy globalization*
- *Fosters Increased trust and transparency*
- *Monetization Strategies*
  - *Subscription Model – Tiered Subscription*
  - *Support Model*
  - *Training and Onboarding – Global in nature*
  - *Consulting Services*

Freemium Model

- *Offers a core product or service for free*
- *Generates revenue through premium features or services*
- *Benefits*
  - *Increased user adoption*
  - *Community Building*
  - *Market Validation*
  - *Upsell Opportunities & Revenue Generation*
  - *Improved Brand Awareness*
- *Considerations*
  - *Clear Value Differentiation*
  - *Frictionless Upgrade Plan*
  - *Ongoing Development in all areas*
  - *Community Management*
  - *Metric based optimization*
- *Examples*
  - *WordPress, GitHub Public vs Private Repositories, MySQL Community vs. Enterprise*

Affiliate Marketing

- *What is Affiliate Marketing?*
  - *It is a performance based marketing strategy*
  - *Businesses partner with affiliates*
  - *Affiliates promote the business's products and services*
  - *Affiliates earn a commission for each successful sale or conversion generated*
- *Open Source and Affiliate Marketing*
  - *Increased Brand Awareness & User Acquisition*
  - *Targeted promotion to relevant audience*
  - *Cost Effective Marketing of Open Source*
  - *Potential Revenue Generation for the Open Source & Affiliate*
  - *Build community of passionate advocates*

Data Monetization

- *Allows open source businesses to leverage the valuable information they collect from their users*
  - *Legal compliance and data privacy issues need to be considered carefully*
- *Types of Data Collected*
  - *User Data – Demographics & Preferences*
  - *Usage Data – How users interact with the software*
  - *Contribution data - Code contributions, bug reports*
  - *Metadata – Data about data – Provides insights into user behaviour, software usage, etc.*
- *Monetization Strategies*
  - *Freemium Data Insights*
  - *DIaaS – Data Insight as a Service – Leverage AI*
  - *Support Subscriptions – Advanced and Customized Data Analysis Services*
  - *Targeted Advertising*
- *Concerns*
  - *Transparency – Be upfront with users on how their data is collected and used*
  - *User Choice – Provide options for user to opt-out of data collection*
  - *Alignment with project goals*

Open Source Software Engineering
(Lifecycle and Methodologies)

Open Source Software Lifecycle

- *The Open Source Software Lifecycle is a dynamic process*
- *Characterized by continuous iterations*
- *Typically consists of the following stages*
  - *Ideation*
  - *Planning*
  - *Development*
  - *Testing & Deployment*
  - *Maintenance and Evolution*

Open Collaboration Model

- *Open Collaboration Models are the driving force behind Open Source Development*
  - *Open Source*
  - *Community Driven*
  - *Collaboration*
- *Benefits of Open Collaboration*
  - *Faster Development*
  - *Improved Quality*
  - *Reduced Costs*
  - *Increased Innovation*
- *Challenges of Open Collaboration*
  - *Project Management*
  - *Code Quality Control*
  - *Sustainability*
- *Key characteristics: Transparency, Collaboration & Community*

Attributes of Open Collaboration

- *Open Collaboration is a system where people with diverse backgrounds and skill sets come together virtually on a shared goal*
- *Open Collaboration teams are loosely coordinated relying on self-organizing and shared purpose*
- *Participants can contribute in various ways to match their expertise*
- *Key Principles*
  - *Egalitarianism: Everyone's ideas are valued*
  - *Meritocracy: Contributions are judged on their quality*
  - *Self Organization: Teams set their goals, define roles and manage workflows collaboratively*

Key Challenges in Open Source Projects

- *Funding*
  - *Often rely on volunteer contributions*
  - *Limited funding can hinder development and maintenance*
  - *Creative solutions are required for funding*
- *Legal*
  - *Licensing complexity*
  - *Handling copy right infringements*
  - *Having a clear and well defined licenses*
- *Sustainability*
  - *Maintaining project momentum*
  - *Volunteer burnout and churn*
  - *A good process can aid in maintaining sustainability*

Community Driven Development

- *A collaborative approach to software development that harnesses the power of open source communities*
  - *Open Collaboration*
  - *Collective Ownership*
  - *Diverse Contributions*
- *Benefits*
  - *Faster Development*
  - *Higher Quality*
  - *Reduced Cost*
  - *Increased Innovation*

How does Community Driven Development Work?

- *Ideate: Any one can propose new features or improvements*
- *Discuss: The community discusses the proposal, its feasibility and potential impact*
- *Develop: If approved developers contribute code to implement the feature*
- *Testing: Code is thought tested via community participation*
- *Release: New features and improvements are integrated into the main project*

Development Process Model

- *Focuses on the key stages of requirements gathering, design, implementation and release in an iterative fashion*
- *Requirement Gathering*
  - *Collaborative Process, Based on user feedback, Developer discussions and proposals, Roadmaps*
- *Design*
  - *Defining architecture and functionality, Wireframes prototypes & mock ups, Documentation for developers, Discussion and Feedback*
- *Implementation*
  - *Developers write code based on design, Version control and change management, Unit testing, CI / CD Pipelines*
- *Release*
  - *Features and Bug fixes, Release Notes, User Testing & Feedback*

Testing & Quality Assurance

- *Ensuring quality is a Open Source project is key to the overall success of the project*
- *Community Driven Testing*
  - *Defect identification by a wider audience and varied use cases*
  - *Increased test coverage*
  - *Faster defect discovery and resolution*
- *Tools & Methods*
  - *CI Tools – Automate builds and testing – detect regression issue early*
  - *Automate Test Frameworks*
  - *Test Days & Defect Triage – Dedicated times for focused testing & prioritizing identified issues*
- *Developers and Testers Working as One*
- *Building High Quality Software Together*
  - *Widespread testing by a diverse community leads to a more robust product*
  - *Openness and collaboration fosters a culture of continuous improvement*
  - *Transparency in the process build trust with users*

Documentation & User Support

- *Good Documentation is key for the successful adoption of an Open Source Projects*
  - *Reduces time for onboarding*
  - *Empowers users*
  - *Facilitates Contribution*
- *Crafting Open Source Documentation*
  - *Focus on User Needs*
  - *Prioritize clarity and Conciseness*
  - *Use multiple formats – Tutorials, code samples, FAQs*
- *Establish Open Communication*
  - *Forums, mailing lists, chats*
  - *Encourage users to contribute to the knowledge base*
  - *Recognize and appreciate contributions*
- *Good documentation reinforces good user support*

Tools & Platforms

- *Version Control Systems*
  - *Is the core of any open source software development process*
  - *Tracks changes to code over time*
  - *Enables multiple developers to work on the same code base simultaneously*
  - *Example: GIT*
- *Issue Tracking Systems*
  - *Mainly to track and manage defects and Communication*
  - *GitHub Issues, Bugzilla, Jira*
  - *Feature Request, Task Management*
- *Code Review and Collaboration*
  - *Improves code quality and identifies potential issues in code*
  - *Ensures compliance to coding standards*
  - *Gerrit, Pull and Merge Requests in Git Hub*
- *Project Management*
  - *Planning, task management, documentation (GitHub, GitLab, Apache Software Foundation Platform)*

Collaborative Ecosystem

- *Code Repositories*
  - *Git & GitHub*
  - *Branches and Forks*
  - *GitLab, Bitbucket, SourceForge*
- *Communication*
  - *For real time discussions and task coordination*
  - *Slack, Discord*
- *Documentation*
  - *Wiki Platforms, Google Docs*
- *CI / CD*
  - *Jenkins*

Maintaining Community Engagement

- *Engagement*
  - *Drives project growth and innovation*
  - *Ensures long term sustainability*
  - *Provides user feedback*
- *How to foster engagement?*
  - *Be inclusive and respectful*
  - *Clear communication*
  - *Celebrate contributions, milestones & achievements*
  - *Keep the conversation flowing*
  - *Encourage Mentorship*
  - *Host regular events*
  - *Utilize communication platforms*

Strengths & Weakness of OSS

- *Leverage skills of a Global Developer community*
  - *How to build and maintain the community? How to ensure continuity?*
- *Collaboration improves innovation*
  - *How to achieve consensus and agreement? How to maintain a inclusive collaborative environment?*
- *Shared code base allows developers to learn from each other*
  - *How to prevent misuse of code?*
- *Transparency, Trust & Lesser security issues*
  - *How to keep away free loaders and entities with malicious intent?*
- *Rapid iterations and flexibility*
  - *How to ensure consistent quality? How to ensure that the pace is sustained over time?*

Choosing the right approach

- *Understand your project*
  - *Project Scope*
  - *Complexity*
  - *Timelines*
  - *Budget*
- *Understand your team*
  - *Team Size*
  - *Skill Sets*
  - *Communication Style*
- *Approaches*
  - *Waterfall – Not to be used other than for small and simple projects not exceeding 2 to 3 man months of effort.*
  - *Agile – The approach of choice!*
  - *Hybrid – Not recommended unless it is a very thought out and evaluated decision in the context of the project to be executed.*

Measuring Community Health

- *Success of an Open Source Project does not just hinge on the code but also on the health of the community that surrounds it.*
- *Why measure?*
  - *Informed decision making*
  - *Project Longevity*
  - *Contribution Opportunities*
- *Key metrics*
  - *Activity: Code Commits, Bug Fixes, etc.*
  - *Growth: Contributors, users and adoption*
  - *Engagement: Community interaction*
  - *Sustainability: Funding, Roadmap, developer participation*
- *Tools*
  - *GitHub Insights*

Comparing Development Methodologies

- *Agile or Waterfall or Hybrid*
- *Waterfall*
  - *Predictive*
  - *Rigid*
- *Agile*
  - *Adaptive*
  - *Iterative & Adaptive*
- *Rational Unified Process (RUP) - Iterative*
  - *Combines Elements of Agile and Waterfall*

Open Source Software Engineering
(Contributing to Open Source Software Projects)

Contribution Models

- *Open Source thrives on diverse contributions not just writing code*
  - *There are multiple ways to get involved, each offering unique benefits*
- *Code Contributions*
  - *Writing code, Fixing Defects, Adding features, Code Improvements*
  - *Deep understanding of the project, Honing coding skills, Code Mentorship*
- *Documentation Contributions*
  - *Improving existing documents or creating new ones*
  - *Tutorials, user guides, API references, translate of other languages*
  - *Requires strong communication and writing skills*
  - *Enhances project clarity, improves user experience, strengthens writing skills*
- *Testing Contributions*
  - *Writing and executing test cases, unit, integration & system testing*
  - *Requires knowledge of testing tools and methodologies*
  - *Improves project quality, strengthens testing skills, contributes to robust test case*

Community Contributions

- *Involves promoting the project, answering questions, and fostering a welcoming community*
- *Includes participating in forums, social media engagement and user support*
- *Requires strong communication and interpersonal skills*
- *Benefits*
  - *Expands project reach*
  - *Builds relationships*
  - *Strengthens communication skills*

Contribution Process

- *Fork – Pull Model:*
  - *Make a local copy (fork, branch) of the project code.*
  - *Modify it locally in the fork / branch*
  - *Submit a pull request for the maintainers to review and merge*
- *Collaborative Model*
  - *Work directly on the project's code base*
  - *Suggest changes and discuss them with the community*

Other ways to contribute

- *Bug Bounty Programs*
  - *Identity and report bugs in the software*
  - *Some projects reward for finding critical vulnerabilities*
- *Design Contributions*
  - *Create icons, logos or user interfaces*

Benefits

- *From Contribution to Open Source Projects*
  - *Enhance skills through practical experience*
  - *Boost your visibility*
  - *Build your network*
  - *Recognition*
- *To Open Source Projects*
  - *Faster Bug Fixes*
  - *Enhanced code quality*
  - *Wider Adoption*
  - *Improved Documentation*

Finding Open Source Projects for Contribution

- *Search in GitHub, Bitbucket, etc.*
- *Open Source Initiative*
- *Issue tracking platforms*
- *Your organization's sponsorships / participation in Open Source Projects*

Open Source Software Engineering
(Setting up an Open Source Project)

Project Ideation & Planning

- *Project Ideation*
  - *Recognize pain points in the field*
  - *Look for gaps in existing solutions*
  - *Identify areas with potential for improvements*
- *Set SMART Goals*
  - *Specific – Clearly define the problem your project will address*
  - *Measurable – Establish metrics to track your project progress*
  - *Attainable – Seat realistic and achievable goals for development milestone*
  - *Relevant – Ensure your project aligns with a genuine need in the open source community*
  - *Time Bound – Create a timeline with deadlines for development stages*
- *Guidelines*
  - *Conduct thorough research*
  - *Define the scope*
  - *Create a development roadmap*
  - *Secure Resources*

Building a community around an OSS Project

- *Why build a community?*
  - *Amplify Development Effort*
  - *Enhance Project Quality*
  - *Sustainability*
- *Strategies*
  - *Open Communication Channels*
  - *Lower the barrier to entry*
  - *Recognize Contributions*
- *Fostering a Welcoming Environment*
  - *Establish a code of conduct*
  - *Encourage Collaboration*
  - *Celebrate Diversity*

Setting up the Repository

- *Repository*
  - *A central storage location for all the project files*
  - *Tracks changes to code, documents and all other artifacts*
  - *Allows collaboration in the community*
- *Git*
  - *A popular distributed version control system*
  - *Runs locally on a machine and tracks changes and maintains versions*
  - *GitHub is built on Git and is the most preferred platform for publishing open source projects*

Maintainer Responsibilities

- *They are the guardians of the project, ensuring its quality, direction and smooth operations*
  - *Reviewing code submissions for functionality, efficiency and adherence to coding standards*
  - *Providing constructive feedback, to contributors, suggesting improvements and guiding them through the revision process*
  - *Merging high quality contributions into the projects codebase ensuring smooth integration*
- *Issue Management*
  - *Categorizing Issues*
  - *Assigning Developers*
  - *Tracking*
  - *Closing issues*
  - *Encourage Mentorship*
  - *Host regular events*
  - *Utilize communication platforms*
- *Fostering Collaboration*

Effective Communication

- *Strive for clear messages, structured communication, and respectful interactions*
- *Open source is about collaboration and collaboration depends on communication*
- *Communication Channels*
  - *Chat*
  - *Discussion Forums*
  - *Issue Tracking System*
  - *Code Review Tools*
  - *Emails*
- *Key Points*
  - *Clarity*
  - *Structure*
  - *Respect*

Talent Attraction and Retention

- *Why open sources projects attract talent?*
  - *Desire for contributing to Impactful Projects*
  - *Building a strong portfolio*
  - *Learning from experienced Developers*
  - *Increased visibility and recognition*
- *Benefits for a Company*
  - *Attract Talent*
  - *Company's commitment to innovation*
  - *Pipeline for good developers*
  - *Culture of innovation and learning*

Open Source Governance

- *To ensure that a project runs smoothly and achieves its goals, a governance model is needed.*
  - *The model defines how decisions are made, who has authority, and how contributors can participate*
- *Centralized Model*
  - *A single entity (individual or organization) holds ultimate decision making power*
  - *Benevolent Dictator or a Core Team*
  - *Clear Leadership and efficient decision making*
- *Decentralized Model*
  - *Decision Making authority is distributed among the contributor community*
  - *Emphasis on consensus and collaboration*
  - *Depends on Collaboration & Communication*
  - *Increased participation, more innovation, adaptable to changes, fosters community collaboration*
- *Hybrid*
  - *Combines elements of both the centralized and decentralized models*
  - *A centralized entity usually takes the final call*
  - *Specially good for large teams*

Choosing Between Governance Models

- *How decisions are made?*
- *Who makes the decisions?*
- *How does the project evolve?*
- *Is the Project Large and Complex?*
- *Is rapid decision making required?*
- *Do the Contributors have adequate skills and experience?*
- *Are the Project goals and long term visions clear?*

Testing & Continuous Integration

- *Why Testing?*
  - *Catches defects early*
  - *Improves code quality and stability*
  - *Ensures features work as intended*
  - *Builds user confidence in the project*
- *Types of testing*
  - *Unit Testing*
  - *Integration Testing*
  - *System Testing*
  - *Acceptance Testing*
- *What is CI?*
  - *Automates build, test and integration processes*
  - *Integrates and builds code changes frequently*
  - *Enables faster defect detection and fixes*

Brand Reputation & Trust

- *Participation in a good and reputed open source project boosts the brand image of a company*
  - *Transparency is the hallmark of Trust*
  - *Establishes Credibility through Expertise*

Using Open Source Libraries & Frameworks

- *Advantages*
  - *Reduce development time*
  - *Improve code quality*
  - *Increase Maintainability*
  - *Reduce Costs*

- *Key Considerations*
  - *Licensing Compatibility*
  - *Active community support*
  - *Documentation Quality*
  - *Security Considerations*
  - *Versioning Policy*
  - *Evaluate Dependencies & Dependency Management*
  - *Test the libraries / framework for suitability*

Cost Savings & Efficiency

- *No licensing fees*
- *Faster Time to Market*
- *Reduced maintenance costs*
- *Access to pre-written code*
- *Increased developer productivity*
- *Collaborative Problem Solving*

Package Managers & Dependency Management

- *Package Managers*
  - *It is a software that automates the process of finding, installing, configuring and updating software packages.*
  - *No need to reinvent the wheel. Find and install pre-built libraries and frameworks*
  - *Reduced errors*
  - *Version Control and Updates are Simplified*

- *Dependency Management*
  - *Refers to the practice of handling dependencies between different software components*

Examples of Package Managers

- *npm: (Node JS Package Manager) – Java Script Packages*
- *pip: Manages Python software packages*
- *Maven: Java Packages*
- *NuGet: .NET Framework*
- *Cargo: Rust*

- *Package managers provide command that allow the user to find and install / upgrade packages*
- *One of the easiest ways and the most widely accepted approaches to distribute Open Source Software / Components / Libraries*

Open Source Software Engineering
(Building & Scaling Open Source Ecosystems)

Open Source Project Metrics

- *Effective Metrics act as a compass, guiding you towards informed decisions to ensure your project's health and growth*
- *Categories of metrics*
  - *Community Metrics: Forks, Number of Contributions, Issue Resolution Time*
  - *Activity Metrics: Commit Frequency, Pull Request Volume, Download Counts, Release Velocity*
  - *Code Quality Metrics: Bug Reporting Rate, Code Coverage*
- *Decision Making for selection of an open source project*
  - *Open Source project metrics provide valuable insights for informed decision making*
  - *By analysing community, activity and code quality metrics, you can identify project strengths and weaknesses*

Scaling Infrastructure using Open Source

- *Kubernetes & Docker*
- *Other Cloud Infrastructure – VMs and Serverless Computing*
- *Data Storage Solutions*
- *Leveraging Open Source Libraries and Frameworks*
  - *Identify libraries and frameworks that align with your project's specific requirements*
  - *Evaluate factors like community support, documentation quality and licensing terms*
  - *Look for seamless integration with your project architecture*
  - *Update the code base frequently*

Community Management

- *The community is the heartbeat of an open source project*
  - *It plays a crucial role in fostering project health, growth and ultimately success*
  - *Should be welcoming and inclusive*
  - *Foster open communication*
  - *Recognition & Appreciation*
- *Cultivating Collaboration*
  - *Conduct Regular events: Hackathons, Seminars, Meetings, etc.*
  - *Encourage Content Creation*
  - *Create Challenges & Gamification*
  - *Utilize tools*

Scalable Documentation

- *Well structured and Adaptable Documentation is the lifeblood of a thriving open source ecosystem*
- *It Empowers developers to Contribute effectively, Fosters project adoption & Ensures long term sustainability*
- *Scalable documents: Focus on Clarity, Conciseness and Completeness*
- *Types of Documents: User Guides, API References, Contribution Guidelines*
- *Delivery Mechanism: Modular Structure, Version Control, Multi-Lingual*

Conflict Resolution in Open Source Communities

- *Open Source projects rely on a diverse group of contributors, and disagreements are inevitable.*
- *Proactive Measures for Conflict Resolution*
  - *Establish clear guidelines: Code of Conduct*
  - *Encourage Open Communication*
  - *Promote Empathy and Respect*
- *De-escalation Techniques*
  - *Listen Actively*
  - *Acknowledge Emotions*
  - *Focus on the issues, not personalities*
- *Collaborative Conflict Resolution*
  - *Seek Common Ground*
  - *Brainstorm Solutions*
  - *Seek Mediation*
- *Learn from Conflicts & look for improvements*

Code of Conduct

- *A strong code of conduct fosters an inclusive environment vital for attracting and retaining valuable contributors*
- *Attributes of a good Code of Conduct*
  - *Clear & Concise*
  - *Define Acceptable and Unacceptable behaviour*
  - *Include clear reporting procedures and violations*
  - *Outline the consequences for code of conduct breaches*
  - *Consider having a code of conduct committee especially for large open source communities*
- *Open communication is the key*
  - *Encourage open communication and active participation*
  - *Respond to concerns and reports promptly and professionally*
  - *Address violations transparently*

Managing Diverse Perspectives

- *Managing diverse perspectives in an open source project fosters a rich environment for collaboration and innovation leading to stronger open source projects.*
  - *Wider range of experiences and ideas*
  - *Improves problem solving and decision making*
  - *More inclusive and welcoming communities*
  - *Attracts a broader talent pool*
- *How to manage diverse perspectives?*
  - *Understand our biases*
  - *Create a safe space for discussions*
  - *Actively address discrimination and harassment*
  - *Embrace collaboration*

Contributor onboarding and retention

- *Contributor onboarding and retention is very critical to the success of an open source project*
- *Onboarding Process*
  - *Should be well defined to reduce friction and to set up contributors for success*
  - *Establishes clear expectations and lays the groundwork for long term engagement*
  - *Contributes to a more positive and welcoming community atmosphere*
  - *Consists of the following: Introduction and overview, setup instructions, guidelines for contributions, guide for first contribution, details of a code review process, how to get support*
- *Retention Strategies*
  - *Recognize and appreciate contributions*
  - *Provide opportunities for growth and development*
  - *Foster a sense of community*
  - *Hold regular events*
  - *Be responsive to feedback*

Commercialization of Open Source

- *To sustain open source projects incorporation of monetization and business models into open source projects is essential.*
  - *Support & Service*
  - *Enterprise version (Freemium)*
  - *Training and certification*
  - *Custom Development*
- *Models*
  - *Freemium*
  - *Dual Licensing*
  - *Subscription*
  - *Indirect Revenue (Derivative Usages)*

Open Source Talent Ecosystem

- *A corporate can create a Open Source Talent pool for scouting talent for their own projects*
  - *Start by engaging the open source community: Participate, Contribute, Attend, Mentor*
  - *Identify Top Performers: Continue to benefit from them in the open source community or attract them to your organization*
- *Corporates and leverage open source communities as incubators for talent*
  - *Try out new technologies and build skills*
  - *Train by collaboration with experienced developers*
- *As an individual showcase your expertise in an open source project to attract recruiter's attention*

Advocacy & Outreach

- *Amplify your open source project – Create Awareness*
  - *Leverage public relations and developer advocacy to amplify your projects reach and impact*
  - *Project the benefits of adopting and using the artifacts of the open source project*
- *Public Relations & Outreach*
  - *Press Releases*
  - *Publications, Blogs, Media Coverage*
  - *Influencers*
  - *Conferences and other forums*
- *Developer Advocacy – Outreach for the Developer Community*
  - *Developer adoption and usage is essential for the success of any software project*

Financial Metrics

- *Financial Metrics provide valuable insights to guide decision making and ensure long term success of the open source project*
- *Can be used to demonstrate the project's value to potential contributors and funders*
- *Core Metrics*
  - *Donations Received*
  - *Grants Received*
  - *Sponsorships Received*
  - *Earned Revenue*
- *KPIs*
  - *Funding Growth*
  - *Burn Rate*
  - *Donor Retention*
  - *Volunteer Hour Growth*
- *Financial Metrics can be used for Resource Allocation, Prioritization of Features, Growth Strategies, Computing ROI*

Transitioning Leadership

- *The purpose is to ensure a smooth transition when the founders, core contributors or existing maintainers move out and need to be replaced by new members of the community.*
- *Succession Planning*
  - *Identify potential successors early*
  - *Cultivate leadership skills in identified successors*
  - *Document knowledge and processes*
  - *Delegate Effectively*
  - *Provide Support and Guidance*
  - *Maintain Oversight*

Sunsetting an Open Source Project

- *Open Source Projects may need a Sunsetting plan to ensure a smooth exit when an Open Source Project has lost its 'value'*
  - *Declining use and development*
  - *Architectural issues*
  - *Obsolete*
  - *Project Forking*
- *Planning for sun set*
  - *Engage the community*
  - *Announce a date clearly with sufficient lead time*
  - *Provide a migration plan for existing users*
  - *Archive the codebase*
  - *Close Communication Channels*

Future of Open Source

- *Open source continues to revolutionize software development*
- *It continues to grow and has become a main stream movement in software development*
- *Every impacted stakeholder has no choice but to embrace open source in some form or the other*
- *The future of software development is in open collaboration and open innovation*

Open Source Software Engineering
(Consuming Open Source Projects)

Selecting Open Source Software

- *With so many options available choosing the right open source software can be daunting*
- *Understand your needs*
  - *Define project goals and functionality*
  - *Identify technical requirements*
  - *Consider non-functional requirements*
- *Researching & Evaluating Open Source Options*
  - *Leverage popular open source repositories like Github*
  - *Read documentation, reviews and community forums*
  - *Assess project activity. Maturity and license compatibility*
- *Integrate Open Source Software*
  - *Follow installation and configuration guidelines*
  - *Understand dependencies and potential conflicts*
  - *Leverage community resources like tutorials*
- *Contribute back to the Open Source community*

Integration and Dependency Management

- *Dependencies are libraries or any other form of reusable code including other applications on which the functionality of your application depends.*
- *Typically dependencies are managed using tools like Gradle, npm, Maven, pip, etc. via an IDE*
- *Things to consider*
  - *Compatibility*
  - *Versions*
  - *Upgrades*
  - *Low Coupling*

Contributing to Existing Projects

- *Contributing as a developer*
  - *As a contributor you get a chance to shape the future of the open source project*
  - *Get an opportunity to learn from and exchange ideas with experienced developers*
- *How to contribute?*
  - *Identify your interests and skills*
  - *Search for one or more matching projects*
  - *Read the documentation and contribution guidelines*
  - *Find the best match*
  - *Try out the project and continue if you like, else find a new project*
  - *Test what you contribute*
  - *Communicate effectively*
  - *Follow community guidelines*
- *Other contribution types*
  - *Documentation*
  - *Testing & QA – Good way to learn about the project*

License Compliance and Obligations

- *Open Source Licenses defines the terms of use of an Open Source Software*
- *Non-Compliance can have serious consequences*
- *Types of Licenses*
  - *Permissive (MIT, Apache)*
  - *Copy Left (GPL, LGPL)*
- *Usage Considerations*
  - *Review the License before using an open source project*
  - *Consider the impact of the license on your project*
  - *Attribution, Redistribution, Maintaining Copyright Notices*

Open Source Policies & Governance

- *Not all open sources are created equal!*
- *Understanding the Open Source Licenses is very important to avoid legal troubles*
  - *Copy right infringement lawsuits*
  - *Injunctions preventing use of software*
  - *Damage to reputation*
- *Build an Open Source Policy*
  - *Process for selecting and approving Open Source Software*
  - *Maintain an inventory of Open Source Software in use*
  - *Contributor agreement process*
  - *Communication and training on the Open Source Policy*

Training and Awareness Programs

- *It is essential to have a strong foundation through education and training in an organization to ensure smooth and compliant integration of Open Source Solutions.*
  - *Reduce Legal risks*
  - *Improves license compliance*
  - *Enhances Security practices*
- *Elements of a training program*
  - *Open Source Fundamentals*
  - *License Identification & Compliance*
  - *Security Considerations*
  - *Optionally, Contribution Best Practices*

Licensing Strategies for Commercial Use

- *Understand licensing implications and establish clear policies*
  - *Take professional legal counsel if required*
- *Establish a process for identifying Open Source Usage within your projects*
- *Implement a system to track and inventory all open source licenses used*
- *Conduct regular compliance assessments to identify potential issues*
- *Implement an ongoing monitoring to ensure compliance with licensing terms*

License Compliance

- *Starts with understanding the Open Source Licenses and its implication on the project where we intend to use the open source*
- *Track and Manage dependencies throughout the development lifecycle*
- *Continuously monitor and look for improvement opportunities*
- *Track changes and updates to Open Source Licensing for products used*
- *Use automated tools to help with License Compliance*
  - *SPDX (Software Package Data Exchange)*
  - *FOSSA*
  - *Black Duck*

Version Control Systems

- *Version Control Systems is a crucial tool for developers working on open source projects*
  - *Track Changes*
  - *Collaborate Effectively – Allows multiple developers to work on the same code base simultaneously*
  - *Maintain History*
  - *Rollback changes*
  - *Transparency*
- *It is a software tool that tracks in files over time*
- *Core Concepts*
  - *Repository*
  - *Working Directory*
  - *Staging Area*
  - *Commit*
  - *Branch*
  - *Merge*

Build and Continuous Integration Tools

- *Improves over efficiency & productivity, reduces manual errors*
- *Build Tools*
  - *Automate Compilation, Linking and Packaging of Software*
  - *Ex: Make, Maven, Gradle*
  - *Can be integrated with version control systems*
- *Continuous Integration*
  - *Automate build, test and deployment*
  - *Ex: Jenkins, GitLab CI/CD*
- *Software Bill of Materials (SBOM)*
  - *A detailed list of all software components, libraries and tools used to create, build and deploy a software application*
  - *Includes information on versions. Licenses and dependencies*
  - *Helps improve security, license compliance, audits*
  - *Many CI / CD tools help create and maintain SBOM*

Dependency Management

- *Dependencies are any external software component that your project relies on to function*
- *They can be libraries, frameworks or other software applications*
- *Dependencies in turn can have their own dependencies creating a dependency tree*
- *Not managing dependencies can lead to*
  - *Conflict of versions*
  - *Security Vulnerabilities*
  - *License Conflicts*
  - *Build Errors*
- *Package Managers*
  - *Automate Dependency Management*
  - *Track dependencies, resolve conflicts and install / update them*
  - *npm, Maven, PIP*

Build & Packaging

- *Build & Package Managers simplify the build, distribution, installation and un-installation process for a software*
- *Build is the process of transforming code into an executable program*
  - *Compilation is an important step*
- *Packaging is the process of creating a distributable archive containing the compiled program and dependencies*
- *There are many open source Tools that you may use for managing your Build and Packaging process*
  - *Make, CMake for Build*
  - *apt, yum, npm*

Container Security – Open Source in Cloud

- *Containers offer rapid application development and deployment*
  - *Docker & Kubernetes are widely adopted*
- *Cloud platforms work extensively with containers*
  - *Interconnectedness of containers can lead to security vulnerabilities*
- *Use Tools and Techniques for enhancing security*
  - *Container image scanning tools*
  - *Runtime security tools*
  - *Enforce security policies*
  - *Establish Security Operations Centre (SOC)*

Serverless and FaaS

- *Serverless computing offers a pay-per-use model where developers write code and deploy them without managing servers*
  - *Function as a Service (FaaS)*
- *FaaS platforms handle provisioning, scaling and security of the underlying infrastructure*
- *Security is a shared responsibility between cloud providers and customers*
  - *Cloud Providers secures the underlying infrastructure*
  - *Customers secure applications and data*
- *Cloud Native Computation Foundation (CNCF)*
  - *Provides Vendor Neutral standards for Cloud Computing*
  - *Knative is a CNCF project defining open standards for serverless computing*
  - *Knative promotes portability and interoperability across cloud providers*

Supply Chain Risk Management Practices

- *Introducing Open Source Software in your solution has the potential to introduce security risks*
  - *Dependency chain can be a reason for concern*
- *The onus is on you to ensure that there are no security risks introduced in your solution due to using Open Source Software*
- *Issues to consider*
  - *Vulnerabilities, License Compliance, Dependency Management, Update Management*
- *National Institute of Standards and Technology – NIST SP 800 – 161 for Secure OSS Consumption*
  - *Software Bill of Materials (SBOM)*
  - *Vulnerability Assessment*
  - *Secure Development Lifecycle*
  - *Inventory of approved Open Source Software Components*

Software Identification Tags

- *Software Identification Tags as the name suggests helps in identifying a software, especially provide and indicator for the software's compliance to standards*
- *They can enhance security and provide standardization*
- *An identification tag includes*
  - *Name, version, license and know security vulnerabilities, if any*
- *ISO/IEC 19770-2 – Standard for capturing external dependency information*
  - *Defines a common format for software identification tag*
  - *Ensure interoperability between different tools and platforms*

Open Source Software Engineering
(Open Standards)

Open Standards

- *Open standards are rules that allow users to create and use products, services, and processes that are consistent and compatible with each other.*
- *They are developed through a collaborative process, are available to the public, and are free*
  - *Occasionally there may be a cost associated*
- *The primary objective of open standards is to ensure interoperability without sacrificing independence*
  - *It fosters collaboration*
- *Examples*
  - *Open PGP (Pretty Good Privacy) - An open and free version of the Pretty Good Privacy (PGP) standard that defines encryption formats for private messaging.*
  - *PDF - A common open standard that is used by everyone, even if the PDF reader software isn't open source.*

Characteristics of Open Standards

- ***Collaborative development**: Open standards are developed through a collaborative process that involves all interested parties.*
- ***Transparent decision-making**: The decision-making process for open standards is transparent and published, and is reviewed by subject matter experts.*
- ***Publicly available**: Open standards are available to the public to access, use, or share.*
- ***Free or low cost**: Open standards are free or low cost to use.*
- ***Interoperability:** Open standards enable interoperability, which means that different products and services can exchange data.*
- ***Neutral foundation**: Open standards provide a neutral foundation for collaboration among contributors.*

Maintaining Open Standards

- *Most Open Standards are owned and maintained by a governing body*
  - *Qualified contributors develop the standards through a collaborative process*
- *Examples*
  - *XML, HTML* - World Wide Web Consortium (W3C)
  - *OpenPGP* - The Internet Engineering Task Force (IETF)
  - *4G/ 5G* - The International Telecommunication Union (ITU), the 3rd Generation Partnership Project (3GPP), and the Internet Engineering Task Force (IETF)

Open Standards vs. Open Source

- Open Standards and Open Source is not the same though there are similarities in the manner in which they are developed, maintained and distributed.
- Open Standards
  - Specifications that define how information is exchanged between systems.
  - Open standards ensure that products, services, and systems perform in an interoperable way
  - Open standards are developed through a consensus-driven process and are neutral towards software licensing models.
- Open Source
  - A software development philosophy that encourages open collaboration, rapid prototyping, and open governance.
  - Open Source Software is publicly available and free to use, modify, and share subject to the terms and conditions imposed by open source licenses
  - The licensing model of an Open Source Software may allow for free redistribution, modifications, and derived works.

Similarities: Open Standards & Open Source

- *Rely on open collaboration*
- *Promote Transparency*
- *Promote Voluntary Contributions from global community*
- *Freely Available*

Open Access

- *Also referred to as open data, open educational resources, and open science*
- *Open access (OA) is the practice of making electronic resources available to the public without copyright or licensing restrictions.*
- *It can include articles, books, journals, conference proceedings, theses, videos, music, software, audio, and multi-media.*
  - *Predominantly used to refer to open access to knowledge and information of any kind in the academic space*

Risks of Using Open Source Software

- *Licencing Consideration*
- *Security Considerations*
- *Availability of Support*
- *Stability of the Project*
- *Longevity of the Project*
- *Version Management*
- *Evolves as per expectations*
- *Long Term Availability of Skills*

Open Source Security Scores (OpenSSF)

- *OpenSSF (Open Source Security Foundation) launched Scorecard in November 2020*
- *Auto-generating a "security score" for Open Source Projects to help users as they decide the trust, risk, and security posture for their use case*
- *OpenSSF Scorecard assesses Open Source Projects for security risks through a series of automated checklists*
  - *It was created by OSS developers to help improve the health of critical projects that the community depends on.*
- *The OpenSSF Scorecard can be used for*
  - *Proactively assess and make informed decisions about accepting security risks within your codebase*
  - *Work with maintainers to improve codebases you might want to integrate*

OpenSSF Scorecard

- *It is an automated tool that assesses a number of important heuristics ("checks") associated with software security and assigns each check a score of 0-10.*
  - *The scores can be used to understand specific areas to improve in order to strengthen the security posture of a project*
- *Few Checks Performed*
  - *Binary-Artifacts (Is the project free of checked-in binaries?)*
  - *Code-Review (Does the project practice code review before code is merged?)*
  - *Dangerous-Workflow (Does the project avoid dangerous coding patterns in GitHub Action workflows?)*
  - *Dependency-Update-Tool (Does the project use tools to help update its dependencies?)*
  - *License (Does the project declare a license?)*
  - *Security-Policy (Does the project contain a security policy?)*
  - *Signed-Releases (Does the project cryptographically sign releases?)*
  - *Vulnerabilities (Does the project have unfixed vulnerabilities?)*
  - *Webhooks (Does the webhook defined in the repository have a token configured to authenticate the origins of requests?)*

Exploring OpenSSF & Security Score Card

- https://openssf.org/projects/scorecard/
- https://github.com/ossf/scorecard

Open Source Software Engineering
(Understanding Open Standards)

Understanding Open Standards

- *Open standards are technical specifications that are*
  - *Publicly Available*
  - *Royalty Free*
  - *Developed Through a Collaborative Process*
  - *Not Controlled by a single Vendor or Entity*
- *Benefits*
  - *Promotes interoperability*
  - *Promotes Innovation*
  - *Promotes Competition – No Vendor Lock-in*
  - *Reduced Costs*
  - *Increased Work Efficiency*
- *Examples*
  - *HTTP, HTML, GSM, LTE, USB, PDF, Python, Java*

Proprietary Standards / Open Standards Orgs.

- *Proprietary Standards*
  - *Controlled by a single vendor*
  - *Licensing fees may apply*
  - *Limited access and control*
  - *May hinder innovation and collaboration*
- *Open Standards Organizations*
  - *Develop and maintain open standards*
  - *Facilitate collaboration among stakeholders*
  - *Ensure interoperability and compliance*
  - *Promote adoption of open standards*
  - *Examples: ISO, IETF, W3C*

Open Software Standards

- *If an open source application or library needs to interoperate the de facto choice is Open Standards*
- *The likelihood of an open source application that uses a proprietary or a custom standard succeeding is very limited*
  - *It is also contrary to the spirit of open source software development*

Standards and Performance of Open Source

- *Open Standards provide a common base to build upon that help bring together diverse solutions that are able to 'work' together*
- *Given the multitude of Open Source solutions, the only way to ensure that they all fit together in a single larger application is via the adoption of open standards.*
- *Leads to better performance of the Open Source ecosystem of products as a whole*

Architecture Standards of Open Source

- *While building the architecture of an Open Source Software an architect should leverage open standards wherever applicable.*
- *It has the following benefits*
  - *Greater Interoperability*
  - *Compliance to accepted standards improves brand image*
  - *Greater acceptability of the Open Source application*
  - *Wider Adoption*
  - *Easier availability of skills*
- *Open Source and Open Standards leverage each other*
  - *Open Source drives Open Standards*
  - *Open Standards fuel better Open Source*

Open Internet Standards

- *IETF – Internet Engineering Task Force*
  - *The IEFT develops and promotes internet standards, particularly standards that comprise the internet protocol suite (TCP/IP)*
  - *Protocols like HTTP, SMTP, DNS*
- *IRTF – Internet Research Task Force*
  - *IRTF focuses on long term research related to internet protocols, applications, architecture and technology*
  - *Complements IEFT by addressing exploratory and long term challenges*
  - *Areas of current concern include Cryptography, Decentralized internet infrastructure, Network Management*
- *IAB – Internet Architecture Board*
  - *IAB provides oversight to the IEFT and IRTF offering strategic direction and ensuring internet's architecture remains sound*
  - *Architectural oversight of internet protocols and procedures*

Open Source Software Engineering
(License Compliance & Audit)

Identifying Licence Requirements

- *Check Project Documentation, License Information Databases*
- *Search Online Repositories, Discussion Forums*
- *Use License identification tools (SPDX)*
- *License Compatibility*
  - *Permissive Licenses (MIT / Apache) can mostly be used anywhere*
  - *Mixing GPL with a proprietary license in most cases will not work*
- *Tools and Resources*
  - *Open Source Initiative*
  - *SPDX (Software Package Data Exchange)*
  - *Free Software Foundation*
  - *Any many other online resources like "License How To", etc.*

Maintaining License Compliance

- Why Compliance Matters
  - Avoid Legal Troubles
  - Maintain a good Reputation
  - Secure access to future innovations (due to license termination)
- Creating a Compliance Foundation
  - Inventory your open source usage
  - Understand license terms
  - Implement Automated tools (Track license usage)
  - Training and Awareness
- Navigating Common Issues
  - Incompatible licenses
  - Attribution Requirements
  - Modification Restrictions
  - Legal Uncertainties

Open Source Audit

- *Why conduct an audit?*
  - *Identify and manage open-source components in your project*
  - *Assess security vulnerabilities and license compliance*
  - *Mitigate risks associated with Open Source dependencies*
  - *Maintain transparency and trust with users*
- *Stages of Open Source Audit*
  - *Planning and Scoping*
  - *Discover the inventory*
  - *Risk Assessment*
  - *Reporting and Communication*
  - *Remediation & Follow Up*
- *Helps assure compliance to Licenses and Standards*

Vulnerabilities in Open Source Projects

- *Vulnerable Components*
  - *Regularly update components, use automated tools to identify vulnerabilities*
- *Unmaintained Projects*
  - *Choose actively maintained projects*
- *Malicious Code*
  - *Conduct reviews of dependencies, use automated tools to scan for malicious code*
  - *Open access to code can invite malicious actors*
- *License Risks*
  - *Use a license compliance management process, use tools to look for license issues*
- *Poor Quality*
  - *Explicitly evaluate and benchmark a component before use, use reputed components*
- *Dependency Management*
  - *Ensure that dependent components do not bring in vulnerabilities inadvertently*

Open Source Software Engineering
(SBOM & License Management Demo)

Exploring Software Bill of Materials

- https://spdx.dev/
- Sample SBOM File: SPDX_SBOM_Example.json

Open Source Software Engineering
(Securing Open Source Applications)

Malware in OSS Repositories

- *Never access Open Source code or components from non-trusted sources, always download from the official repositories*
  - *Confusing and similar sounding names to popular Open Source components*
- *Supply Chain Attack*
  - *Attacks on your projects via dependent components that you have not used directly but are required by components that you have used.*
- *Unaddressed defects that appear to be innocuous*
  - *An attacker may find a way to exploit such defects*
- *Outdated and Unpatched Libraries*
  - *Automated updates via package and dependency management tools*

Securing Open Source Applications

- *Start with a Software Bill of Materials (SBOM)*
- *Regularly update Dependencies*
- *Secure Coding Practices – Secure Software Engineering*
- *Leverage Security Testing tools and Professional Services*
- *Opt for well tested reliable open source components*
- *DevSecOps*
  - *Integrate security throughout the development life cycle*
  - *Identify and fix vulnerabilities early*
  - *Proactive threat detection*

Mitigating impacts of Security Breaches

- *Impacts of Security Breaches*
  - *Security breaches are becoming increasingly common and expensive*
  - *Average cost of data breaches has risen significantly*
  - *Breaches have a devastating impact on businesses*
- *Proactive Security Measures*
  - *Multi Factor Authentication*
  - *Least privilege access control*
  - *Regular system updates and patches*
  - *Security awareness*
- *Data Loss Prevention*
  - *Data Encryption*
  - *Content Filtering*
  - *Data activity monitoring*

Incident Response Planning

- *When breaches do occur what is the response mechanism?*
- *Incident response plan should be in place comprising of:*
  - *Breach detection*
  - *Containment Strategy*
  - *Eradication Plan*
  - *Recovery Mechanism*
  - *Reporting*

OpenSSF

- *Security standards provide a baseline for secure software development*
- *Popular Security Standards*
  - *OWASP – Application Security Verification Standard*
  - *PCI DSS – Payment card industry data security standard*
  - *SOC 2 – Service Organization Controls*
- *Open Source Software Security Foundation (OpenSSF)*
  - *OpenSSF promotes a set of security best practices for OSS projects*
  - *Scorecard – An automated tool to help access vulnerabilities due to the used of open source components in a project*

OpenSSF Scorecard

- *A collection of security health metrics for Open Source projects*
- *Automates checks to assess security risks*
- *Generates scores that can be used to evaluate the security posture*
- *Key Score Card Metrics*
  - *Badges to represent different security focus areas and assigned a criticality*
  - *Each badge is assigned a score of 0 to 10*
  - *Overall project score is a weighted average of the badge scores*

Interpreting Scorecard Results

- *Security Scorecards are typically interpreted based on security standards and benchmarks*
  - *Select a standard that you want to comply with*
- *Benchmark standards provide a way to compare your organization's security performance against industry best practices of similar organizations*
- *Interpreting a scorecard*
  - *Compliance with standards*
  - *Comparison with benchmarks*
  - *Risk Assessment in the context of the organization*
  - *Security requirement in the context of the Business*

Security Evaluation Tools

- *Security evaluation tools are the instruments used to assess an organization's security posture against security standards and benchmarks*
- *These tools can identify vulnerabilities, detect threats, and measure the effectiveness of existing security controls*
- *Automated security evaluation tools offer speed, efficiency and scalability to the security assessment process*
- *They can continuously monitor systems for vulnerabilities and detect suspicious activity*

Open Source Software Engineering
(Creative Commons Licenses)

Creative Commons Licenses (CC)

- *Open Access (OA)*
  - *Sharing knowledge and creativity openly without financial, legal or technical barriers*
  - *Increases access to research, educational materials, and creative works*
- *Benefits of OA*
  - *Fosters Collaboration & Innovation*
  - *Accelerates Scientific Discovery*
  - *Promotes Education and Learning*
- *Creative Commons*
  - *Free, standardized legal tools for copyright holders*
  - *Specify how others can share adapt and build upon your work*
- *Building Blocks of Creative Commons Licenses*
  - *Attribution (BY): Requires credit to the creator, Most Permissive*
  - *Share Alike (SA): Derivative work must be shared under the same license*
  - *Non-Commercial (NC): Allows non-commercial use only*
  - *No Derivatives (ND): Prohibits derivative works*

Popular CC Licenses

- *BY: Attribution*
- *BY-SA: Attribution – Share Alike*
- *BY-NC: Attribution - Non-commercial*
- *BY-ND: Attribution – No Derivatives*
- *BY-NC-SA: Attribution – Non-commercial – Share Alike*
- *BY-NC-ND: Attribution – Non-commercial – No Derivatives*

Choosing the right CC License

- *Finding the right balance*
  - *Controlling how your work is used*
  - *Encouraging reuse and adaption*
- *Examples*
  - *Music: CC BY (Attribution)*
  - *Education: CC BY-NC (Attribution - Non Commercial)*
  - *Software Development: CC BY-SA (Attribution - Share Alike)*
- *Consider your Goals*
- *Select the license that aligns*

Using CC Licenses

- Using CC Licenses typically requires giving attribution and all of them have the "BY" prefix
- Finding CC Licensed Work
  - Search Engines
  - Sites for different types of Media Content
- What is attribution?
  - Clearly specifying the following
    - Creator's Name
    - Title of the Work
    - Link to the original work if available
    - Name of the License

Open Source Software Engineering
(Future of Open Source)

Future of Open Standards

- *Discussion Point: Do open standards stifle creativity?*

Future of Open Access

- *Two main routes of OA has evolved*
  - *Gold OA: Publish in OA journals*
  - *Green OA: Self deposit in repositories*
- *Discussion Point: Is OA the way forward?*
  - *Funding?*
  - *Monetization?*

Future trends in Security

- *AI Powered Security*
  - *AI Powered security systems are increasingly being used to automate tasks like threat detection and incident response.*
- *Impact of IoT on Security*
  - *IoT devices increase the surface of attack for cyber criminals*
  - *These devices can have vulnerabilities granting easy access to malicious actors*
- *Impact of Blockchain on Security*
  - *Holds promise for the future to store and share data securely in a decentralized manner*
- *Human Element in Security*
  - *Finally it is the awareness about security in humans, both developing a software system and using it, that will enable to build secure systems*

AI and the Future of Copyright in AI

- *The rise of AI in content creation raises new questions about copyright ownership and usage.*
  - *Does content created by an AI entity based on existing content require attribution?*
  - *Does the usage of content created by AI require attribution?*
- *Openness will definitely go a long way in fostering and enabling development of AI*
  - *CC licensing could be a key driver in this direction*

Open Art Work

- *Why discuss open art work?*
- *Open Artwork*
  - *Not licensed and can be freely used*
  - *Protected via CC Licenses*
- *Copyright*
  - *Protected via a regular licensing and requires payment of royalty for usage*
  - *Are often watermarked (Visible / Invisible)*
- *Typical Open Art Work*
  - *Images, Vector Graphics, Icons & Symbols*
- *Popular Platforms*
  - *Creative Commons*
  - *Pixabay*
  - *Unsplash*
  - *Pexels*

Open Source Software Engineering
(Exploring OpenSource Project Metrics & Projects)

Exploring Open Source Project Metrics

http://Open-Source.guide/metrics/

Exploring a few OSS Projects

- *Apache HTTP Server*
  - *Revolutionized the process of software development and was one of the main precursors for the success of OSS*
  - *Resulted in the set up of the Apache Group*
  - *Today it is still the most widely used Web Server*
  - *https://github.com/apache/httpd*
- *Linux*
  - *Linus Torvalds, a Finnish computer scientist*
  - *One of the all time best success stories of the OSS community*
  - *Resulted in multiple other derivative products like Ubuntu, Debian, Fedora, etc.*
  - *https://github.com/torvalds/linux*
- *Eclipse*
  - *Established by IBM the Eclipse Foundation is funded by at least 22 corporates*
  - *It is an Open source IDE, predominantly for Java Development but eventually supports multiple programming languages*
  - *Users and plugin functionality*
  - *https://github.com/eclipse*
- *Moodle (Modular Object Oriented Dynamic Learning Environment)*
  - *An Open Source LMS*
  - *BITS eLearn Portal used Moodle*
  - *https://github.com/moodle/moodle*

Open Source Software Engineering
(Course Summary & Recap)

Open Source Software

- *Open Source Software is a type of computer software in which the source code is released under a license.*
- *In which the copyright holder grants users the right to use, study, change and distribute the software to anyone and for any purpose*
- *Open Source Software is usually developed in a collaborative public manner*
- *Open Source Software is a prominent example of open collaboration*

Characteristics of Open Source Software

- *Source Code Availability*
  - *Freely accessible code*
  - *Transparency Fosters Collaboration & Innovation*
- *Collaborative Development*
  - *OSS projects involve a global community of programmers*
  - *Developers collaborate to work in these projects*
- *Usage Governed by Open Source Licensing*
  - *Permissive*
  - *Copy Left*

Open Source Community and Contributions

- *Self governing , worldwide community of OSS Developers*
- *Decisions are collaborative*
- *The dynamics of the Open Source Community makes open source projects sustainable*
- *Contributions play a key role – Documentation, UI/UX – Goes Beyond Code*
- *Meetups – Enhances collaboration*

Core Principles of Open Source

- ***Openness***
- ***Transparency***
- ***Collaboration***
- ***Rapid Improvement***
- ***Cost Effectiveness***
- ***Security***
- ***Innovation***
- ***Flexibility***
- ***Early & Often***
- ***Expectation of Community***

Types of Licensing Models

- *Free*
- *Open Source*
- *Freeware*
- *Public Domain*

Every free software is open source, every open-source software is not free

Categories of Open Source Licenses

- ***Permissive Licenses***
  - *Grant the most freedom*
  - *Allow the user to modify and redistribute the software under any license, including proprietary licenses*
  - *MIT, BSD, Apache*
- ***Copy Left Licenses***
  - *Enforces a form of reciprocity*
  - *Require the derivative works based on the software to also be open source and use the same or a similar license*
  - *GNU General Public License (GPL) & GNU Lesser General Public License (LGPL)*

Intellectual Property Rights (IP)

- *What is Intellectual Property Rights?*
  - *Refers to the legal right that protect creations of the mind, such as inventions, literary and artistic works, designs and symbols.*
- *Any software involves IP rights*
  - *Ideas*
  - *Designs*
  - *Solutions*
- *Types*
  - *Patents*
  - *Copyrights*
  - *Trademarks*
  - *Trade Secrets*

Why licenses in Open Source?

- ***With Freedom comes responsibility***
  - *Provides clear guidelines on how the software can be used modified and distributed*
  - *Establishes copy right on the source code to prevent theft and malicious use*
- ***Open Source Initiative (OSI)***
  - *Defines 10 criteria*
  - *One of the key criteria is the presence of an OSI Approved license*
    - *Use the software for any purpose, including commercial use*
    - *Access the source code and modify it*
    - *Redistribute the software, with or without modifications*
    - *Distribute derivative work based on the software*

Why Open Source Business Models?

- *Building a sustainable business is key to the success of an open source project!*
  - *Costs need to be recovered*
  - *Core team members need to derive monetary value*
  - *Key Contributors need to be rewarded for sustainability*

Open Source Monetization & Business Models

- *Dual Licensing*
- *Open Core*
- *Freemium*
- *Subscription Model*
- *In-App Purchases*
- *Consulting*
- *Training & Workshops*
- *Support & Maintenance*
- *Custom Development*
- *Integration Services*
- *Theming & Branding*
- *Merchandise*
- *Marketplace for Add-ons*
- *SaaS*
- *Affiliate Marketing*
- *Data Monetization (DIaaS)*

Open Source Funding Models

- *Donations*
- *Corporate Sponsorship*
- *Crowdfunding*
- *Grants*

Attributes of Open Collaboration

- *Open Collaboration is a system where people with diverse backgrounds and skill sets come together virtually on a shared goal*
- *Open Collaboration teams are loosely coordinated relying on self-organizing and shared purpose*
- *Participants can contribute in various ways to match their expertise*
- *Key Principles*
  - *Egalitarianism: Everyone's ideas are valued*
  - *Meritocracy: Contributions are judged on their quality*
  - *Self Organization: Teams set their goals, define roles and manage workflows collaboratively*

Open Source Software Lifecycle

- *The Open Source Software Lifecycle is a dynamic process*
- *Characterized by continuous iterations*
- *Typically consists of the following stages*
  - *Ideation*
  - *Planning*
  - *Development*
  - *Testing & Deployment*
  - *Maintenance and Evolution*

Tools & Platforms

- *Version Control Systems*
  - *Is the core of any open source software development process*
  - *Tracks changes to code over time*
  - *Enables multiple developers to work on the same code base simultaneously*
  - *Example: GIT*
- *Issue Tracking Systems*
  - *Mainly to track and manage defects and Communication*
  - *GitHub Issues, Bugzilla, Jira*
  - *Feature Request, Task Management*
- *Code Review and Collaboration*
  - *Improves code quality and identifies potential issues in code*
  - *Ensures compliance to coding standards*
  - *Gerrit, Pull and Merge Requests in Git Hub*
- *Project Management*
  - *Planning, task management, documentation (GitHub, GitLab, Apache Software Foundation Platform)*

Key Challenges for Open Source Projects

- *Funding*
  - *Often rely on volunteer contributions*
  - *Limited funding can hinder development and maintenance*
  - *Creative solutions are required for funding*
- *Legal*
  - *Licensing complexity*
  - *Handling copy right infringements*
  - *Having a clear and well defined licenses*
- *Sustainability*
  - *Maintaining project momentum*
  - *Volunteer burnout and churn*
  - *A good process can aid in maintaining sustainability*

Strengths & Weakness of OSS

- *Leverage skills of a Global Developer community*
  - *How to build and maintain the community? How to ensure continuity?*
- *Collaboration improves innovation*
  - *How to achieve consensus and agreement? How to maintain a inclusive collaborative environment?*
- *Shared code base allows developers to learn from each other*
  - *How to prevent misuse of code?*
- *Transparency, Trust & Lesser security issues*
  - *How to keep away free loaders and entities with malicious intent?*
- *Rapid iterations and flexibility*
  - *How to ensure consistent quality? How to ensure that the pace is sustained over time?*

Measuring Community Health

- *Success of an Open Source Project does not just hinge on the code but also on the health of the community that surrounds it.*
- *Why measure?*
  - *Informed decision making*
  - *Project Longevity*
  - *Contribution Opportunities*
- *Key metrics*
  - *Activity: Code Commits, Bug Fixes, etc.*
  - *Growth: Contributors, users and adoption*
  - *Engagement: Community interaction*
  - *Sustainability: Funding, Roadmap, developer participation*
- *Tools*
  - *GitHub Insights*

Contributing to Open Source Projects

- *Open Source thrives on diverse contributions not just writing code*
  - *There are multiple ways to get involved, each offering unique benefits*
- *Code Contributions*
  - *Writing code, Fixing Defects, Adding features, Code Improvements*
  - *Deep understanding of the project, Honing coding skills, Code Mentorship*
- *Documentation Contributions*
  - *Improving existing documents or creating new ones*
  - *Tutorials, user guides, API references, translate of other languages*
  - *Requires strong communication and writing skills*
  - *Enhances project clarity, improves user experience, strengthens writing skills*
- *Testing Contributions*
  - *Writing and executing test cases, unit, integration & system testing*
  - *Requires knowledge of testing tools and methodologies*
  - *Improves project quality, strengthens testing skills, contributes to robust test case*

Benefits of Contributions

- *From Contribution to Open Source Projects*
  - *Enhance skills through practical experience*
  - *Boost your visibility*
  - *Build your network*
  - *Recognition*
- *To Open Source Projects*
  - *Faster Bug Fixes*
  - *Enhanced code quality*
  - *Wider Adoption*
  - *Improved Documentation*

Setting up an Open Source Project

- *Project Ideation*
  - *Recognize pain points in the field*
  - *Look for gaps in existing solutions*
  - *Identify areas with potential for improvements*
- *Set SMART Goals*
  - *Specific – Clearly define the problem your project will address*
  - *Measurable – Establish metrics to track your project progress*
  - *Attainable – Seat realistic and achievable goals for development milestone*
  - *Relevant – Ensure your project aligns with a genuine need in the open source community*
  - *Time Bound – Create a timeline with deadlines for development stages*
- *Guidelines*
  - *Conduct thorough research*
  - *Define the scope*
  - *Create a development roadmap*
  - *Secure Resources*

Maintainer Responsibilities

- *They are the guardians of the project, ensuring its quality, direction and smooth operations*
  - *Reviewing code submissions for functionality, efficiency and adherence to coding standards*
  - *Providing constructive feedback, to contributors, suggesting improvements and guiding them through the revision process*
  - *Merging high quality contributions into the projects codebase ensuring smooth integration*
- *Issue Management*
  - *Categorizing Issues*
  - *Assigning Developers*
  - *Tracking*
  - *Closing issues*
  - *Encourage Mentorship*
  - *Host regular events*
  - *Utilize communication platforms*
- *Fostering Collaboration*

Open Source Governance

- *To ensure that a project runs smoothly and achieves its goals, a governance model is needed.*
  - *The model defines how decisions are made, who has authority, and how contributors can participate*
- *Centralized Model*
  - *A single entity (individual or organization) holds ultimate decision making power*
  - *Benevolent Dictator or a Core Team*
  - *Clear Leadership and efficient decision making*
- *Decentralized Model*
  - *Decision Making authority is distributed among the contributor community*
  - *Emphasis on consensus and collaboration*
  - *Depends on Collaboration & Communication*
  - *Increased participation, more innovation, adaptable to changes, fosters community collaboration*
- *Hybrid*
  - *Combines elements of both the centralized and decentralized models*
  - *A centralized entity usually takes the final call*
  - *Specially good for large teams*

Using Open Source Libraries & Frameworks

- *Advantages*
  - *Reduce development time*
  - *Improve code quality*
  - *Increase Maintainability*
  - *Reduce Costs*
- *Key Considerations*
  - *Licensing Compatibility*
  - *Active community support*
  - *Documentation Quality*
  - *Security Considerations*
  - *Versioning Policy*
  - *Evaluate Dependencies & Dependency Management*
  - *Test the libraries / framework for suitability*

Cost Savings & Efficiency

- *No licensing fees*
- *Faster Time to Market*
- *Reduced maintenance costs*
- *Access to pre-written code*
- *Increased developer productivity*
- *Collaborative Problem Solving*

Package Managers & Dependency Management

- *Package Managers*
  - *It is a software that automates the process of finding, installing, configuring and updating software packages.*
  - *No need to reinvent the wheel. Find and install pre-built libraries and frameworks*
  - *Reduced errors*
  - *Version Control and Updates are Simplified*
- *Dependency Management*
  - *Refers to the practice of handling dependencies between different software components*

Open Source Project Metrics

- *Effective Metrics act as a compass, guiding you towards informed decisions to ensure your project's health and growth*
- *Categories of metrics*
  - *Community Metrics: Forks, Number of Contributions, Issue Resolution Time*
  - *Activity Metrics: Commit Frequency, Pull Request Volume, Download Counts, Release Velocity*
  - *Code Quality Metrics: Bug Reporting Rate, Code Coverage*
- *Decision Making for selection of an open source project*
  - *Open Source project metrics provide valuable insights for informed decision making*
  - *By analysing community, activity and code quality metrics, you can identify project strengths and weaknesses*

Financial Metrics

- *Financial Metrics provide valuable insights to guide decision making and ensure long term success of the open source project*
- *Can be used to demonstrate the project's value to potential contributors and funders*
- *Core Metrics*
  - *Donations Received*
  - *Grants Received*
  - *Sponsorships Received*
  - *Earned Revenue*
- *KPIs*
  - *Funding Growth*
  - *Burn Rate*
  - *Donor Retention*
  - *Volunteer Hour Growth*
- *Financial Metrics can be used for Resource Allocation, Prioritization of Features, Growth Strategies, Computing ROI*

Code of Conduct

- *A strong code of conduct fosters an inclusive environment vital for attracting and retaining valuable contributors*
- *Attributes of a good Code of Conduct*
  - *Clear & Concise*
  - *Define Acceptable and Unacceptable behaviour*
  - *Include clear reporting procedures and violations*
  - *Outline the consequences for code of conduct breaches*
  - *Consider having a code of conduct committee especially for large open source communities*
- *Open communication is the key*
  - *Encourage open communication and active participation*
  - *Respond to concerns and reports promptly and professionally*
  - *Address violations transparently*

Transitioning Leadership

- *The purpose is to ensure a smooth transition when the founders, core contributors or existing maintainers move out and need to be replaced by new members of the community.*
- *Succession Planning*
  - *Identify potential successors early*
  - *Cultivate leadership skills in identified successors*
  - *Document knowledge and processes*
  - *Delegate Effectively*
  - *Provide Support and Guidance*
  - *Maintain Oversight*

Sunsetting an Open Source Project

- *Open Source Projects may need a Sunsetting plan to ensure a smooth exit when an Open Source Project has lost its 'value'*
  - *Declining use and development*
  - *Architectural issues*
  - *Obsolete*
  - *Project Forking*
- *Planning for sun set*
  - *Engage the community*
  - *Announce a date clearly with sufficient lead time*
  - *Provide a migration plan for existing users*
  - *Archive the codebase*
  - *Close Communication Channels*

Selecting Open Source Software

- *With so many options available choosing the right open source software can be daunting*
- *Understand your needs*
  - *Define project goals and functionality*
  - *Identify technical requirements*
  - *Consider non-functional requirements*
- *Researching & Evaluating Open Source Options*
  - *Leverage popular open source repositories like Github*
  - *Read documentation, reviews and community forums*
  - *Assess project activity. Maturity and license compatibility*
- *Integrate Open Source Software*
  - *Follow installation and configuration guidelines*
  - *Understand dependencies and potential conflicts*
  - *Leverage community resources like tutorials*
- *Contribute back to the Open Source community*

Open Source Policies & Governance

- *Not all open sources are created equal!*
- *Understanding the Open Source Licenses is very important to avoid legal troubles*
  - *Copy right infringement lawsuits*
  - *Injunctions preventing use of software*
  - *Damage to reputation*
- *Build an Open Source Policy*
  - *Process for selecting and approving Open Source Software*
  - *Maintain an inventory of Open Source Software in use*
  - *Contributor agreement process*
  - *Communication and training on the Open Source Policy*

Training and Awareness Programs

- *It is essential to have a strong foundation through education and training in an organization to ensure smooth and compliant integration of Open Source Solutions.*
  - *Reduce Legal risks*
  - *Improves license compliance*
  - *Enhances Security practices*
- *Elements of a training program*
  - *Open Source Fundamentals*
  - *License Identification & Compliance*
  - *Security Considerations*
  - *Optionally, Contribution Best Practices*

Risks of Using Open Source Software

- *Licencing Consideration*
- *Security Considerations*
- *Availability of Support*
- *Stability of the Project*
- *Longevity of the Project*
- *Version Management*
- *Evolves as per expectations*
- *Long Term Availability of Skills*

Software Bill of Materials (SBOM)

- *A detailed list of all software components, libraries and tools used to create, build and deploy a software application*
- *Includes information on versions. Licenses and dependencies*
- *Helps improve security, license compliance, audits*
- *Many CI / CD tools help create and maintain SBOM*

Dependency Management

- *Dependencies are any external software component that your project relies on to function*
- *They can be libraries, frameworks or other software applications*
- *Dependencies in turn can have their own dependencies creating a dependency tree*
- *Not managing dependencies can lead to*
  - *Conflict of versions*
  - *Security Vulnerabilities*
  - *License Conflicts*
  - *Build Errors*
- *Package Managers*
  - *Automate Dependency Management*
  - *Track dependencies, resolve conflicts and install / update them*
  - *npm, Maven, PIP*

Supply Chain Risk Management Practices

- *Introducing Open Source Software in your solution has the potential to introduce security risks*
  - *Dependency chain can be a reason for concern*
- *The onus is on you to ensure that there are no security risks introduced in your solution due to using Open Source Software*
- *Issues to consider*
  - *Vulnerabilities, License Compliance, Dependency Management, Update Management*
- *National Institute of Standards and Technology – NIST SP 800 – 161 for Secure OSS Consumption*
  - *Software Bill of Materials (SBOM)*
  - *Vulnerability Assessment*
  - *Secure Development Lifecycle*
  - *Inventory of approved Open Source Software Components*

Open Standards

- *Open standards are rules that allow users to create and use products, services, and processes that are consistent and compatible with each other.*
- *They are developed through a collaborative process, are available to the public, and are free*
  - *Occasionally there may be a cost associated*
- *The primary objective of open standards is to ensure interoperability without sacrificing independence*
  - *It fosters collaboration*
- *Examples*
  - *Open PGP (Pretty Good Privacy) - An open and free version of the Pretty Good Privacy (PGP) standard that defines encryption formats for private messaging.*
  - *PDF - A common open standard that is used by everyone, even if the PDF reader software isn't open source.*

Characteristics of Open Standards

- ***Collaborative development**: Open standards are developed through a collaborative process that involves all interested parties.*
- ***Transparent decision-making**: The decision-making process for open standards is transparent and published, and is reviewed by subject matter experts.*
- ***Publicly available**: Open standards are available to the public to access, use, or share.*
- ***Free or low cost**: Open standards are free or low cost to use.*
- ***Interoperability:** Open standards enable interoperability, which means that different products and services can exchange data.*
- ***Neutral foundation**: Open standards provide a neutral foundation for collaboration among contributors.*

Maintaining Open Standards

- *Most Open Standards are owned and maintained by a governing body*
  - *Qualified contributors develop the standards through a collaborative process*
- *Examples*
  - *XML, HTML* - World Wide Web Consortium (W3C)
  - *OpenPGP* - The Internet Engineering Task Force (IETF)
  - *4G/ 5G* - The International Telecommunication Union (ITU), the 3rd Generation Partnership Project (3GPP), and the Internet Engineering Task Force (IETF)

Proprietary Standards / Open Standards Orgs.

- *Proprietary Standards*
  - *Controlled by a single vendor*
  - *Licensing fees may apply*
  - *Limited access and control*
  - *May hinder innovation and collaboration*
- *Open Standards Organizations*
  - *Develop and maintain open standards*
  - *Facilitate collaboration among stakeholders*
  - *Ensure interoperability and compliance*
  - *Promote adoption of open standards*
  - *Examples: ISO, IETF, W3C*

Standards and Performance of Open Source

- *Open Standards provide a common base to build upon that help bring together diverse solutions that are able to 'work' together*
- *Given the multitude of Open Source solutions, the only way to ensure that they all fit together in a single larger application is via the adoption of open standards.*
- *Leads to better performance of the Open Source ecosystem of products as a whole*

Open Internet Standards

- *IETF – Internet Engineering Task Force*
  - *The IEFT develops and promotes internet standards, particularly standards that comprise the internet protocol suite (TCP/IP)*
  - *Protocols like HTTP, SMTP, DNS*
- *IRTF – Internet Research Task Force*
  - *IRTF focuses on long term research related to internet protocols, applications, architecture and technology*
  - *Complements IEFT by addressing exploratory and long term challenges*
  - *Areas of current concern include Cryptography, Decentralized internet infrastructure, Network Management*
- *IAB – Internet Architecture Board*
  - *IAB provides oversight to the IEFT and IRTF offering strategic direction and ensuring internet's architecture remains sound*
  - *Architectural oversight of internet protocols and procedures*

Open Access

- *Also referred to as open data, open educational resources, and open science*
- *Open access (OA) is the practice of making electronic resources available to the public without copyright or licensing restrictions.*
- *It can include articles, books, journals, conference proceedings, theses, videos, music, software, audio, and multi-media.*
  - *Predominantly used to refer to open access to knowledge and information of any kind in the academic space*

Open Source Security Scores (OpenSSF)

- *OpenSSF (Open Source Security Foundation) launched Scorecard in November 2020*
- *Auto-generating a "security score" for Open Source Projects to help users as they decide the trust, risk, and security posture for their use case*
- *OpenSSF Scorecard assesses Open Source Projects for security risks through a series of automated checklists*
  - *It was created by OSS developers to help improve the health of critical projects that the community depends on.*
- *The OpenSSF Scorecard can be used for*
  - *Proactively assess and make informed decisions about accepting security risks within your codebase*
  - *Work with maintainers to improve codebases you might want to integrate*

OpenSSF Scorecard

- *It is an automated tool that assesses a number of important heuristics ("checks") associated with software security and assigns each check a score of 0-10.*
  - *The scores can be used to understand specific areas to improve in order to strengthen the security posture of a project*
- *Few Checks Performed*
  - *Binary-Artifacts (Is the project free of checked-in binaries?)*
  - *Code-Review (Does the project practice code review before code is merged?)*
  - *Dangerous-Workflow (Does the project avoid dangerous coding patterns in GitHub Action workflows?)*
  - *Dependency-Update-Tool (Does the project use tools to help update its dependencies?)*
  - *License (Does the project declare a license?)*
  - *Security-Policy (Does the project contain a security policy?)*
  - *Signed-Releases (Does the project cryptographically sign releases?)*
  - *Vulnerabilities (Does the project have unfixed vulnerabilities?)*
  - *Webhooks (Does the webhook defined in the repository have a token configured to authenticate the origins of requests?)*

Interpreting Scorecard Results

- *Security Scorecards are typically interpreted based on security standards and benchmarks*
  - *Select a standard that you want to comply with*
- *Benchmark standards provide a way to compare your organization's security performance against industry best practices of similar organizations*
- *Interpreting a scorecard*
  - *Compliance with standards*
  - *Comparison with benchmarks*
  - *Risk Assessment in the context of the organization*
  - *Security requirement in the context of the Business*

Securing Open Source Applications

- *Start with a Software Bill of Materials (SBOM)*
- *Regularly update Dependencies*
- *Secure Coding Practices – Secure Software Engineering*
- *Leverage Security Testing tools and Professional Services*
- *Opt for well tested reliable open source components*
- *DevSecOps*
  - *Integrate security throughout the development life cycle*
  - *Identify and fix vulnerabilities early*
  - *Proactive threat detection*

Malware in OSS Repositories

- *Never access Open Source code or components from non-trusted sources, always download from the official repositories*
  - *Confusing and similar sounding names to popular Open Source components*
- *Supply Chain Attack*
  - *Attacks on your projects via dependent components that you have not used directly but are required by components that you have used.*
- *Unaddressed defects that appear to be innocuous*
  - *An attacker may find a way to exploit such defects*
- *Outdated and Unpatched Libraries*
  - *Automated updates via package and dependency management tools*

Mitigating impacts of Security Breaches

- *Impacts of Security Breaches*
  - *Security breaches are becoming increasingly common and expensive*
  - *Average cost of data breaches has risen significantly*
  - *Breaches have a devastating impact on businesses*
- *Proactive Security Measures*
  - *Multi Factor Authentication*
  - *Least privilege access control*
  - *Regular system updates and patches*
  - *Security awareness*
- *Data Loss Prevention*
  - *Data Encryption*
  - *Content Filtering*
  - *Data activity monitoring*

Incident Response Planning

- *When breaches do occur what is the response mechanism?*
- *Incident response plan should be in place comprising of:*
  - *Breach detection*
  - *Containment Strategy*
  - *Eradication Plan*
  - *Recovery Mechanism*
  - *Reporting*

Creative Commons Licenses (CC)

- *Open Access (OA)*
  - *Sharing knowledge and creativity openly without financial, legal or technical barriers*
  - *Increases access to research, educational materials, and creative works*
- *Benefits of OA*
  - *Fosters Collaboration & Innovation*
  - *Accelerates Scientific Discovery*
  - *Promotes Education and Learning*
- *Creative Commons*
  - *Free, standardized legal tools for copyright holders*
  - *Specify how others can share adapt and build upon your work*
- *Building Blocks of Creative Commons Licenses*
  - *Attribution (BY): Requires credit to the creator, Most Permissive*
  - *Share Alike (SA): Derivative work must be shared under the same license*
  - *Non-Commercial (NC): Allows non-commercial use only*
  - *No Derivatives (ND): Prohibits derivative works*

Popular CC Licenses

- *BY: Attribution*
- *BY-SA: Attribution – Share Alike*
- *BY-NC: Attribution - Non-commercial*
- *BY-ND: Attribution – No Derivatives*
- *BY-NC-SA: Attribution – Non-commercial – Share Alike*
- *BY-NC-ND: Attribution – Non-commercial – No Derivatives*

Choosing the right CC License

- *Finding the right balance*
  - *Controlling how your work is used*
  - *Encouraging reuse and adaption*
- *Examples*
  - *Music: CC BY (Attribution)*
  - *Education: CC BY-NC (Attribution - Non Commercial)*
  - *Software Development: CC BY-SA (Attribution - Share Alike)*
- *Consider your Goals*
- *Select the license that aligns*

Exploring a few OSS Projects

- *Apache HTTP Server*
  - *Revolutionized the process of software development and was one of the main precursors for the success of OSS*
  - *Resulted in the set up of the Apache Group*
  - *Today it is still the most widely used Web Server*
  - *https://github.com/apache/httpd*
- *Linux*
  - *Linus Torvalds, a Finnish computer scientist*
  - *One of the all time best success stories of the OSS community*
  - *Resulted in multiple other derivative products like Ubuntu, Debian, Fedora, etc.*
  - *https://github.com/torvalds/linux*
- *Eclipse*
  - *Established by IBM the Eclipse Foundation is funded by at least 22 corporates*
  - *It is an Open source IDE, predominantly for Java Development but eventually supports multiple programming languages*
  - *Users and plugin functionality*
  - *https://github.com/eclipse*
- *Moodle (Modular Object Oriented Dynamic Learning Environment)*
  - *An Open Source LMS*
  - *BITS eLearn Portal used Moodle*
  - *https://github.com/moodle/moodle*