Module-5

Lifecycle and methodologies in Open Source Software

Lifecycle and methodologies in Open Source Software

- Open Collaboration Model
- Community Driven Development
- Open Source Software Development Process Model
- Comparing OSS development methodologies with traditional methodologies

Open Collaboration Model

- is a system of innovation/production/development
- Participants interacting to create a product/service
  - Self motivated
  - Goal oriented
  - Loosely coordinated
  - Having some economic value
  - Making it available to contributors and non-contributors.

3 principles of open collaboration model

- **Egalitarianism:** [trend/school of thought in political philosophy]
  - equality of some sort
  - In OSS everyone can contribute
  - Accessible to all

- **Meritocracy**
  - Power and value is measured based on
    - Effort, talent, achievement
    - Quality of contribution
    - Not based on social class or wealth
    - Open and fair valuation
      - Decisions are made publicly available for evaluation of the judgement taken

Note: Based on Researchers lead by **D Riehle [research paper available]**

- **Self organization**
  - Process of arriving at some order/arrangement
    - Through interactions between individuals
  - Such that the resulting structure is:
    - Decentralized
    - Distributed yet connected
    - Robust and self repairing
  - Project groups organize themselves
    - Without any external control/influence

Community Driven Development model

- Community[contributors group]
  - Independent people
  - Supported by company
  - Align their activities to develop OSS

Community driven Software development

- Self driven, self motivated and self organized
  - Shuffling can also happen
- Structured based on job roles [similar to a private software company]
  - May have multiple sub-teams
- Can comprise of:
  - Developers Group[coders]
  - Builders/Integration group [joiners]
  - Testers group [testing and reporting]
  - Release management group[final packing and documentations]
- Communities are open to end users to join and contribute to the project.

Developers Group

- Designing and implementation
  - System design/architecture[blue print]
  - Coding
- Start with
  - Initial study[anything exists already?[part/full]]
  - Once the problem is defined
    - Specify the requirements of the software
    - Flexible design[adaptable, easy to modify/upgrade]
      - Follow the standards
      - Documentation

    - Choosing the appropriate language.
      - Considering the constraints

Developers Group : Continued

- Designing and coding can overlap.
  - Saves time.
  - Some duplication of effort can happen
  - Incremental design, iterative design

Builders Group in OSSD

- Compiling all source code into object modules
- Linking object modules into one whole[deployable]
- Building is a continuous process.[due to modification/changes/additions/deletions]
- Regular builds
  - Speedup the process: how?

Tools for continuous integration and deployment

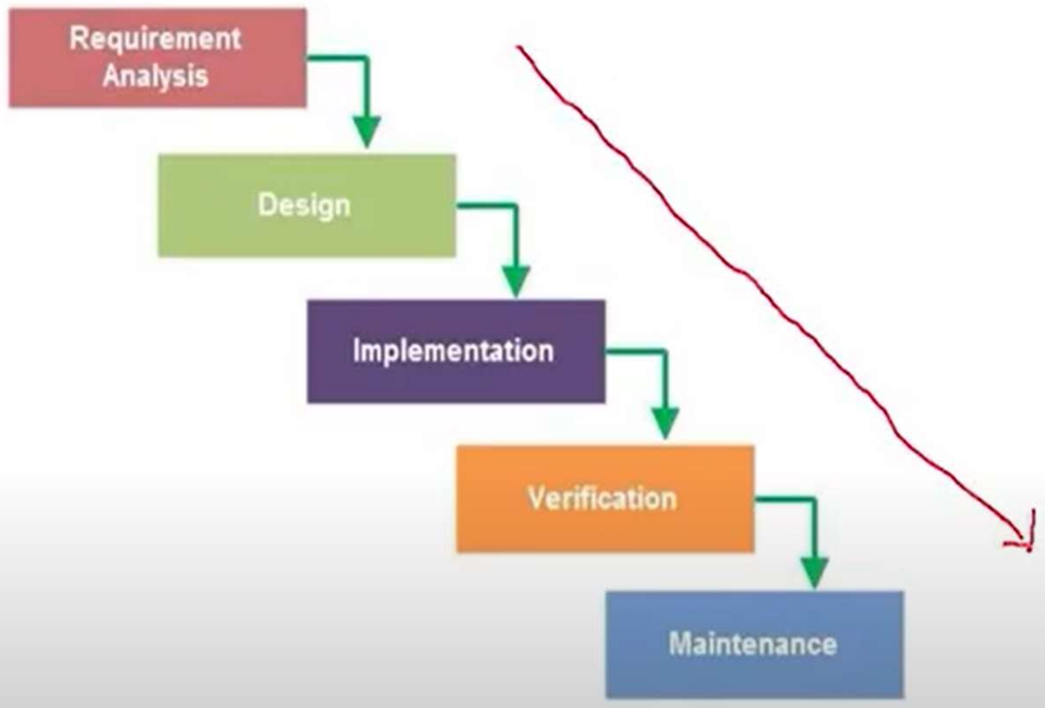
- Jenkins
- TeamCity
- GitLab CI
- Circle CI
- Travis CI

Testing Group in OSSD

- Make use of tools
  - Mailing lists
  - Discussion boards
  - Bug reports
- Test projects created
  - Testing
  - Generating bug reports
  - Test results
- Cyclic process
  - Till stable version is achieved

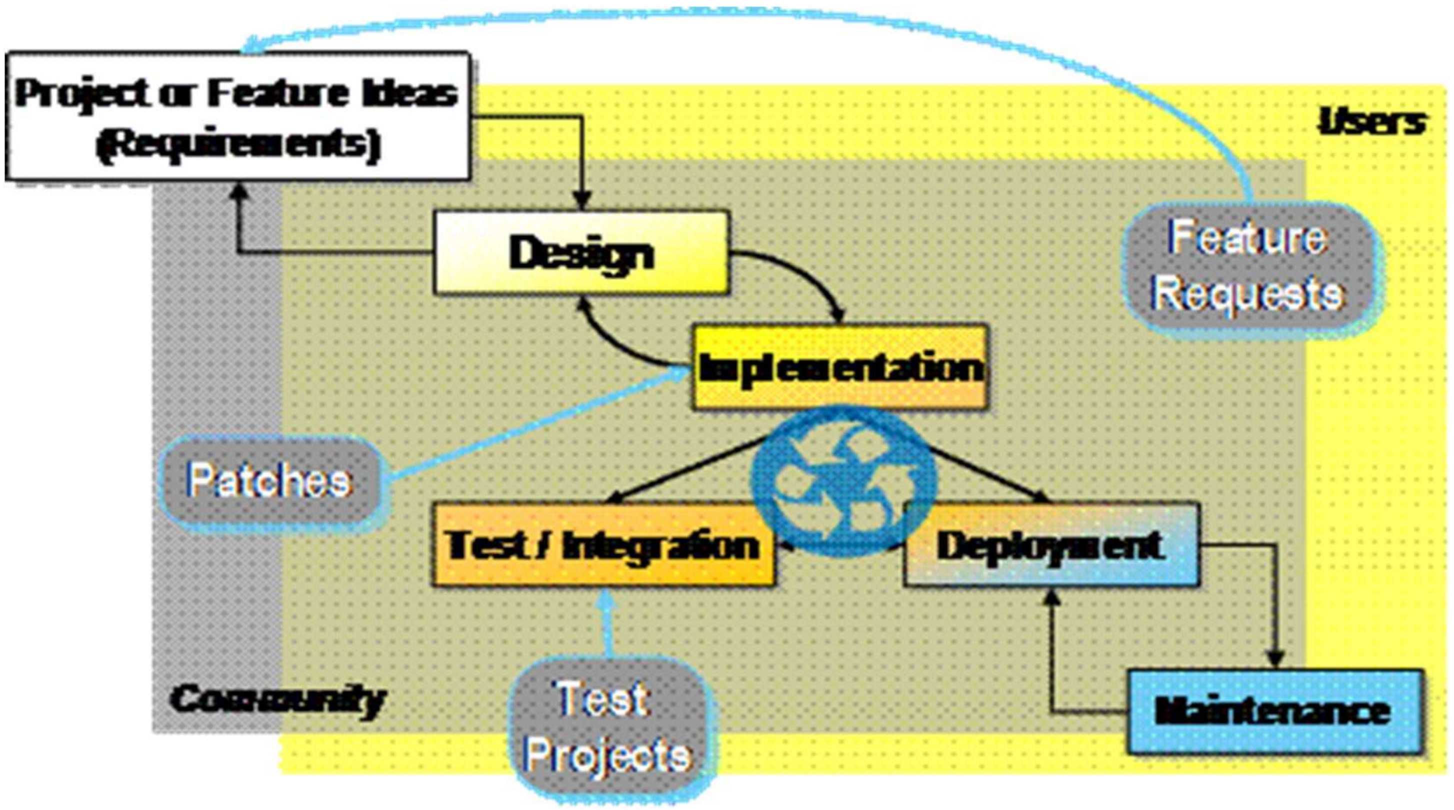
Open Source Software Development Process Model

- Different from traditional model[waterfall].
- Activities:
    - Collecting and analysing the requirements
    - Designing a solution
    - Developing the code
    - Testing
    - Deployment
    - Maintanance



Waterfall Model: Linear development model

OSSD Model


Project or Feature Ideas (Requirements)
Users
Design
Implementation
Feature Requests
Patches
Test / Integration
Deployment
Community
Test Projects
Maintenance

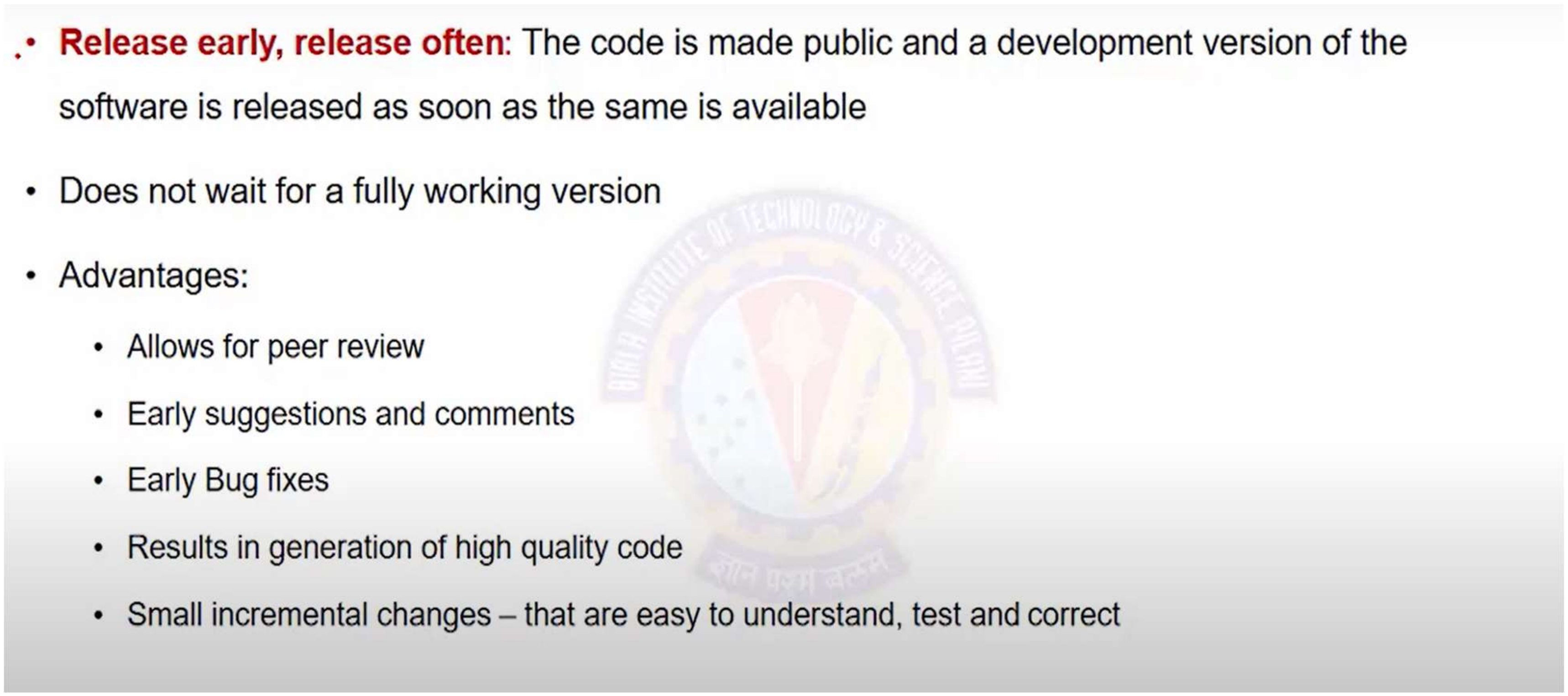
OSSD model is a Non linear model

- A new idea/project or a feature request for an existing project
- Design[iterative]
- Prototype/POC[for feasibility]
- Implementation[iterative]
- Development release[may contain bugs]
  - Release early, release often
- Testing
- Deployment [continuous integration and continuous deployment cycle]
  - May use DevOps

Patches

- Related to code changes.
    - Go through the review process
    - After confirmation incorporated as part of the product code.

Unique characteristics of OSS Development model



- **Release early, release often**: The code is made public and a development version of the software is released as soon as the same is available
- Does not wait for a fully working version
- Advantages:
  - Allows for peer review
  - Early suggestions and comments
  - Early Bug fixes
  - Results in generation of high quality code
  - Small incremental changes – that are easy to understand, test and correct

- **Peer Review:** Exhaustive review of code by community – more eyes looking at it
  - Faster comment and feedback
  - Early identification of bugs
  - Improved quality

- **Small, Incremental changes:** For a larger period of the development cycle, the changes are in the form of small code patches which are easy to understand, test and resolve
  - Beneficial since they help in focusing more on testing phase, which is cyclic and is continuously executed with every increment of the software
  - Small changes are less likely to introduce cascading or unintended consequences.

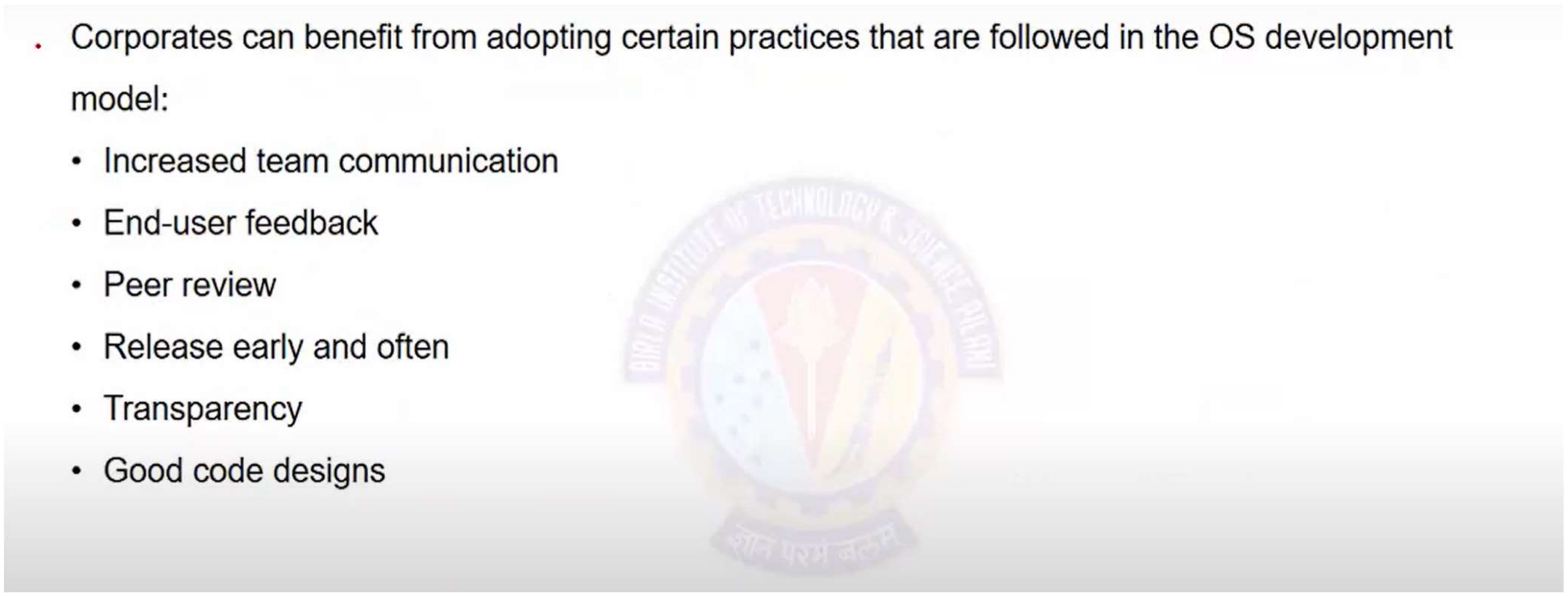
- **Highly secure code:** OS community considers security as a very important aspect and any code that raises security concerns is flagged and not included in the project.

- **Continuous quality improvement:** Exhaustive peer review and early identification and fixing of bugs leads to improved code quality.

- **Test projects:** Large open source projects also create separate test projects consisting of all large numbers of test suites and make use of automated testing tools.

- **End user involvement**: The end user are continuously involved in all phases of the development model.

Recommended Practises



- Corporates can benefit from adopting certain practices that are followed in the OS development model:
  - Increased team communication
  - End-user feedback
  - Peer review
  - Release early and often
  - Transparency
  - Good code designs

Agile Methodology

Highlights

- Small design steps
- Incremental development
- Frequent customer interactions
- Joint interactions[developer and customer] related to the next incremental releases.
- Iterations are of few weeks only.
- Release->customer feedback→fixing and updates.
- Many similarities between agile and OSSD model.
- But there are some major differences too.

Agile vs OSSD Methodologies

- Agile: Highest priority is to satisfy the customer.
  - Through continuous delivery of valuable software.
- OSSD: Customer concept do not exist.
  - Contributors and users of the product.
  - User community acts as customers.
    - Direct involvement of end users in decision making[features] is very limited.

- Agile: Changing requirements at a later stage also allowed.

- OSSD: as they are not customer driven,
  - Later stage changes in requirements are often resisted.
  - Possibility of branching out and working on exists.
    - Done based on the community feedback
      - Contributors community and end user community

- Agile: Delivery cycles: couple of weeks to couple of months
- OSSD: much shorter release cycle, usually daily builds.
  - Release early, release often.

- Agile: Business stake holders/people and developers must work together.
- OSSD: There is no concept of “business people” as such.
  - End users might pitch in into this role to some extent.

- Agile: Build projects/products with motivated individuals
    - By giving them the
        - Environment
        - Support
        - Trusting them to get the job done.

- OSSD: Participation is voluntary
        - Self driven
        - Self motivated
        - Individual interest

- Agile: Within the Dev team
    - Face to face conversations

- OSSD: More of written communication
    - Projects may be widely distributed.
    - All docs and communication is made available on a common/open platform.

- Agile: Maximizing the amount of work done is important.

- OSSD: As they are voluntary
    - No contractual commitments
    - Amount of work done depends on the individual developers.

Summary

- Many similarities and some differences.

- Both of the methodologies involve:
  - Continuous interaction with users/customers
  - Value good design and documentation
  - Engage self organizing and self motivating individuals
  - Strive to develop software targeted to a class of customers/users/community