Lecture Notes for Lecture 4 of CS 5500 (Foundations of Software Engineering) for the Spring 2021 session at the Northeastern University San Francisco Bay Area Campuses.

Focus on Open-Source Methodologies

Philip Gust,
Clinical Instructor
Department of Computer Science

http://www.ccis.northeastern.edu/home/pgust/classes/cs5500/2021/Spring/index.html

Focus on Open-Source Methodologies

Review of Lecture 3

- In lecture 3, we studied the *Agile* software engineering approach, and compared it to earlier evolutionary methodologies.
- We learned that with an agile approach, requirements and solutions evolve through collaboration with self-organizing, cross-functional teams and their customers.
- The focus is on customer satisfaction, active and continuous communication, incremental delivery, informal methods, minimal work products and flexible response to change.
- With Agile software development, software engineers, managers, customers, and stakeholders work closely together as an agile team whose members' interests are aligned.

Focus on Open-Source Methodologies

Review of Lecture 3

- We learned that Agile is an umbrella that includes several processes and methodologies. We looked at two of them in more detail.
- *Scrum* (1995) is a framework that can be adapted to create a process for a project. Increments work are performed in short *sprints* with fixed goals, daily meetings, and a demonstration.
- *Extreme Programming* (1996) takes traditional practices to extreme levels, with pair programming, continuous integration, “spike” prototypes, and shorter work increments.
- Scrum focuses on self-organization and workflow, while XP emphasizes engineering practices and process constraints.

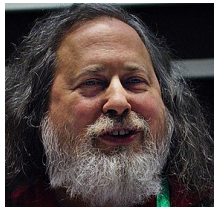
Focus on Open-Source Methodologies

Introduction

- A different type of software development methodology began to take shape with the advent of the open-source software (OSS) movement.
- The term *open source* refers to a computer software whose source code is available to the general public for use or modification to its original design.
- Open-source software development is meant to be a collaborative effort, where programmers improve upon the source code and share the changes within the community.

Focus on Open-Source Methodologies

Introduction

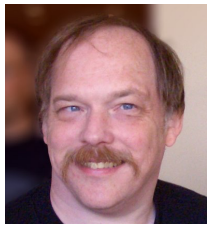


- In 1983, Richard Stallman began a movement for freely modifiable and redistributable software.
- His Free Software Foundation (FSF) released the GNU Public License (GPL) in 1985. It required programs using free software, as well as any source code modifications, be distributed as source code under the GPL license.

- In 1992, Linus Torvalds released an early version of Linux under the GPL, and developers world-wide wanted to add features and contribute to releases.
- The rapid growth of this worldwide community called for new light-weight methodologies for distributed, asynchronous software development projects.

Focus on Open-Source Methodologies

Introduction



- In 1997, Eric S. Raymond published a seminal essay: "The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary".
- It contrasts two different free software development models:
  - **Cathedral model:** source code is available with each software release, but code developed between releases is restricted to an exclusive group of software developers (examples: GNU Emacs, GCC).
  - **Bazaar model:** code is developed over the Internet in view of the public. Raymond credits Linus Torvalds, leader of the Linux kernel project, as the inventor of this process.

    The Linux community is like a babbling bazaar, with diverse agenda and approaches, out of which a comprehensible and stable system seemingly emerges only by a succession of miracles. (Raymond, 1999)

Focus on Open-Source Methodologies

Introduction

- Raymond’s central thesis is the proposition that, "given enough eyeballs, all bugs are shallow" (“Linus's law”):
  - *The more widely available the source code is for public testing, scrutiny, and experimentation, the more rapidly all forms of bugs will be discovered.*
- In contrast, an inordinate amount of time and energy must be spent hunting for bugs in the Cathedral model, since the working version of the code is available to a few developers.

Focus on Open-Source Methodologies

Introduction

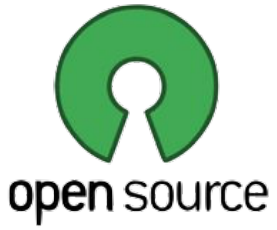
- The term ”open source" was first proposed by Christine Peterson at a strategy session that included Raymond, held on Feb. 3, 1998 in Palo Alto, California.
- Netscape about to release the source of its browser, inspired by Raymond’s essay, and these developers wanted to counter FSF’s perceived anti-commercial attitude on free software.
- Linus Torvalds endorsed the term the next day, and the *Open Source Initiative (OSI)* formed in a late February, shortly after Netscape first used the term ”open source” in a press release.

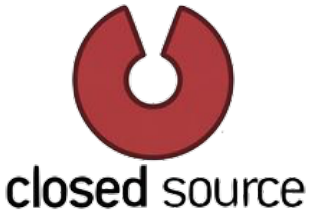
- Publisher Tim O'Reilly also renamed the upcoming “Freeware Summit” as the “Open Source Summit” and published Raymond’s essay as a book in 1999.

Focus on Open-Source Methodologies

Open-Source Model

- The open-source model is a decentralized software development model that encourages open collaboration among a community of developers and end-users.
  - Not all members of the community can contribute to the creation or development of the software, but they may contribute knowledge of the problem domain or the process.
  - Compare this to a closed software development model model where the developers and other participants do not necessarily represent the larger community of users.



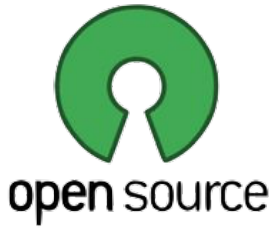


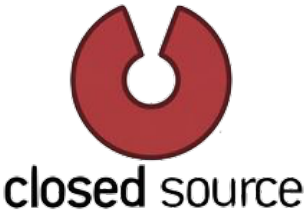
Logos: Open Source Initiative

Focus on Open-Source Methodologies

Open-Source Model

- Source code and all work products are freely available within a community. Developers can improve upon the source code and freely share changes within the community.
  - An open-source software development process allows examining code, prototyping features and discussing the merits of proposed changes based on experience and working code.
  - Compare this to a closed software development process where those outside the process have no ability to provide input or contribute working software to the project.



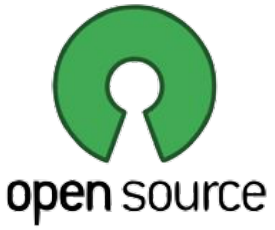


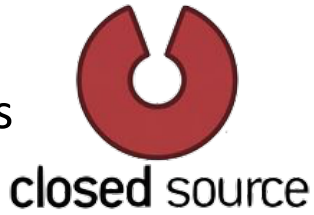
Logos: Open Source Initiative

Focus on Open-Source Methodologies

Open-Source Model

- The development process and all discussions about the project and its development are transparent and visible to the entire community.
  - Directly interacting during the development process enables community members to provide more timely and informed input into proposed new features and improvements.
  - Compare this to a closed software development model where the community has no visibility into the process, and limited ability to suggest features or improvements to developers.





Logos: Open Source Initiative

Focus on Open-Source Methodologies

Open-Source Model

- The open-source model is a fluid development framework, characterized by increased intra-team collaboration, continuous integration and testing and greater end-user involvement.
- This model describes a process and characteristics that would apply to most open-source projects, but it also states that every open-source project could have its own development process.

Focus on Open-Source Methodologies

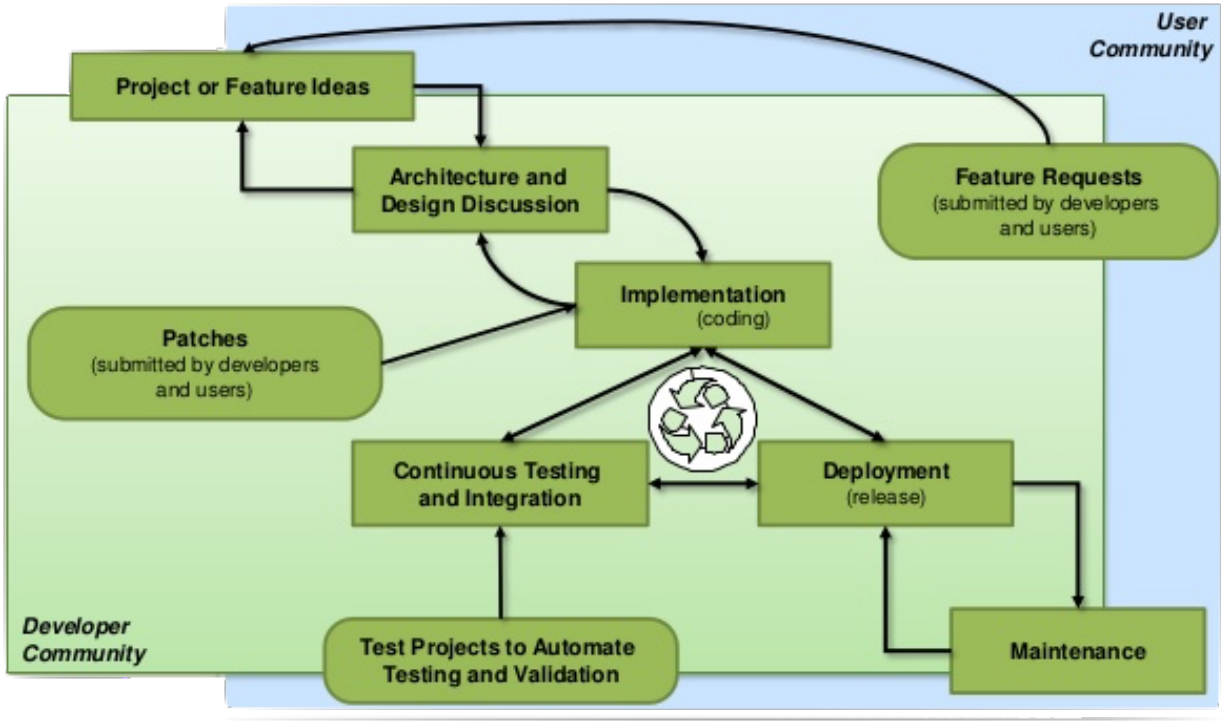
Open-Source Model

- The open-source model presumes development is apportioned among many teams, working around the world in a mobile structure that is not affected by new arrivals or departures.
- This model was originally developed by Ibrahim Haddad for the Linux foundation. It runs under the Feature Development lifecycle, and consists of the following framework activities:
    1. Project or feature ideas
    2. Architecture and design discussion
    3. Collaboration on implementation
    4. Source code submission and evaluation
    5. Continuous testing and Integration
    6. Source code release and deployment

Focus on Open-Source Methodologies

Open-Source Model

- Under an open-source model the end-user and developer communities work together collaboratively.



Samsung Open Source Group

Focus on Open-Source Methodologies

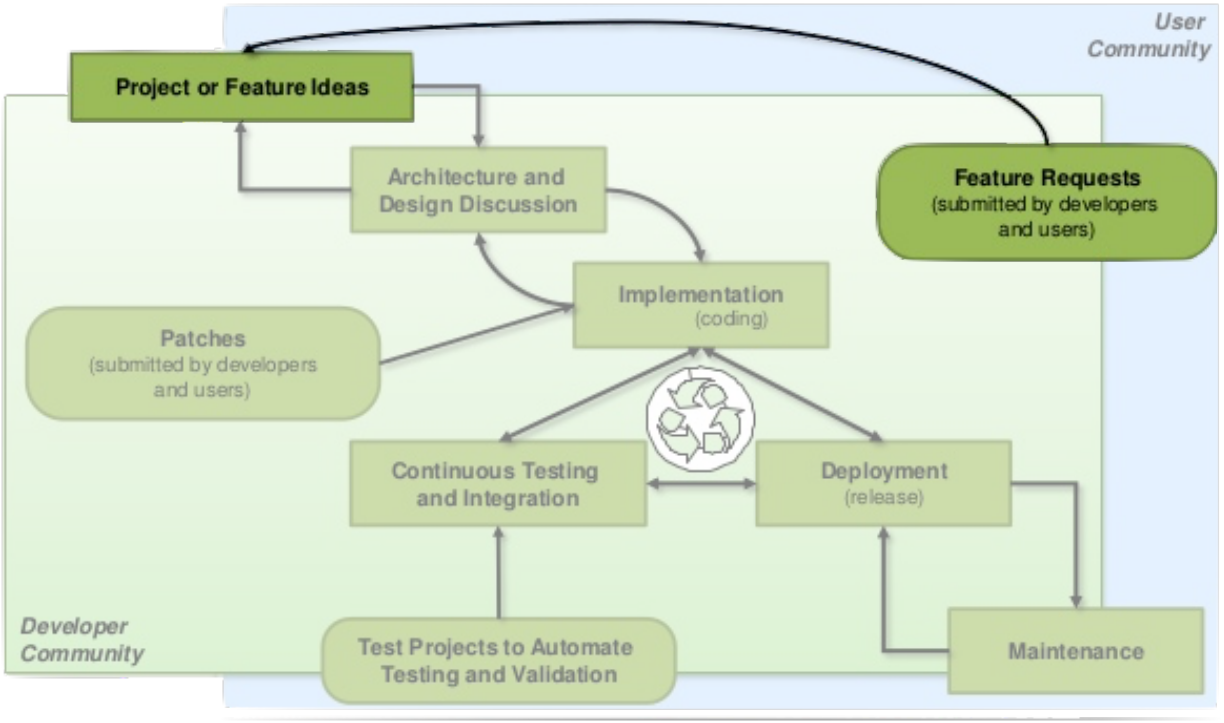
Open-Source Model

- Project or feature ideas
  - When a new feature is needed, a feature request is created to ensure a common understanding within the project team.
  - The request helps the team understand what feature has been requested, its priority, development status, associated bugs, blockers and scheduled release.
  - The project team contributors evaluate the request in terms of its need, strength and if it is fit for a future release.
  - This feature request must be communicated transparently to all team members, whether new or old.

Focus on Open-Source Methodologies

Open-Source Model

- A feature request ensures a common understanding within the project team.



Samsung Open Source Group

Focus on Open-Source Methodologies

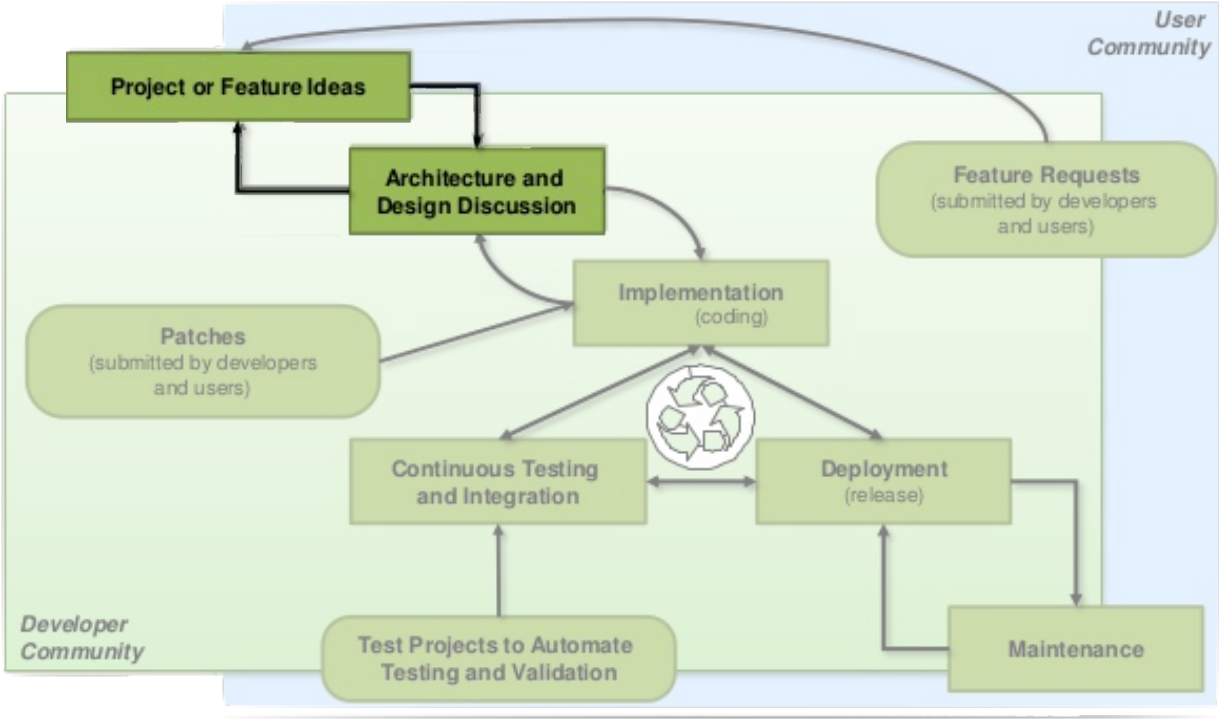
Open-Source Model

- Architecture design and discussion
  - The importance of communication comes under the Architecture and Design Discussion phase.
  - Because team members are often separated by both place and time, asynchronous communication is the most common form of communication.
  - Synchronous communications can be used when direct communication is necessary, such as during design meetings.
  - Communication challenges include:
    - time zones
    - languages
    - cultures

Focus on Open-Source Methodologies

Open-Source Model

- During the Architecture and Design Discussion phase, both asynchronous and synchronous communications may occur



Samsung Open Source Group

Focus on Open-Source Methodologies

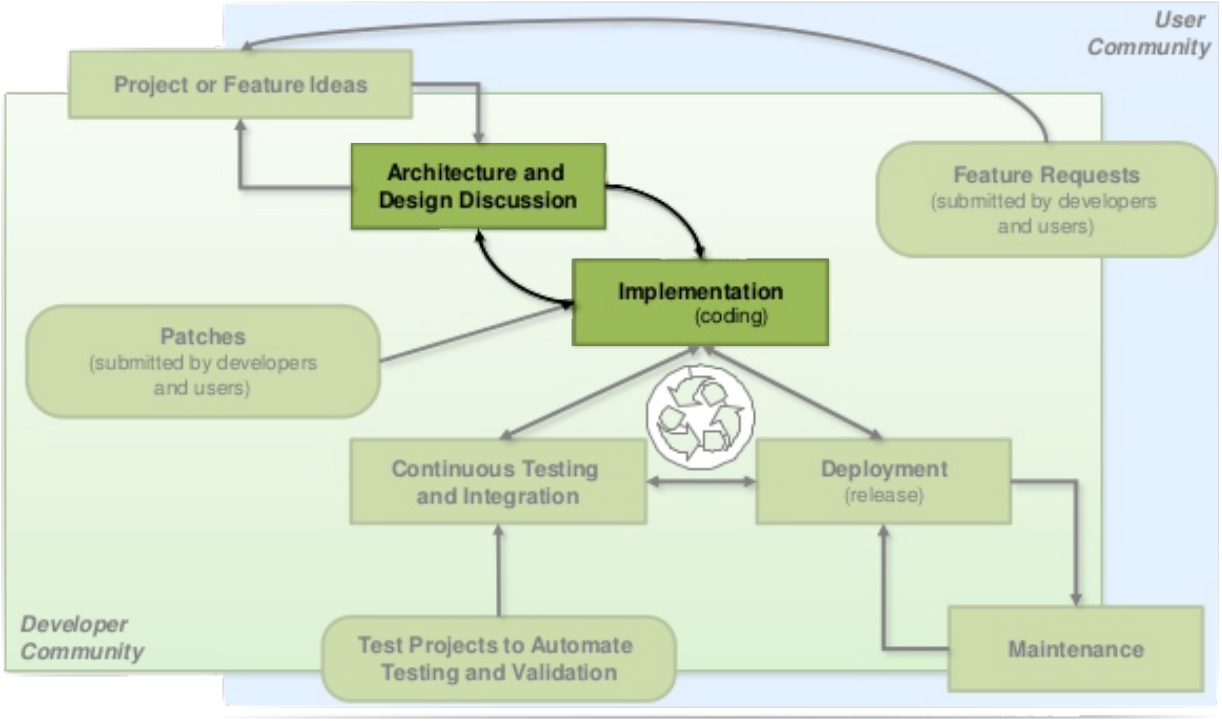
Open-Source Model

- Collaboration on implementation
  - After consensus is reached through architectural and design discussions, a collaborative implementation phase can ensue.
  - Keeping in mind that team members can be from around the world, project tools that support effective code contribution must be used, for example the open-source *git* repository.
  - For source code to be accepted, it has a life-cycle of its own which is often iterative in nature.
  - The process begins with collaborative development among the subset of developers who have taken the responsibility for developing the feature.
  - During implementation, there is very likely to be additional architectural and design discussions.

Focus on Open-Source Methodologies

Open-Source Model

- After consensus is reached through architectural and design discussion, a collaborative implementation phase can ensue



Samsung Open Source Group

Focus on Open-Source Methodologies

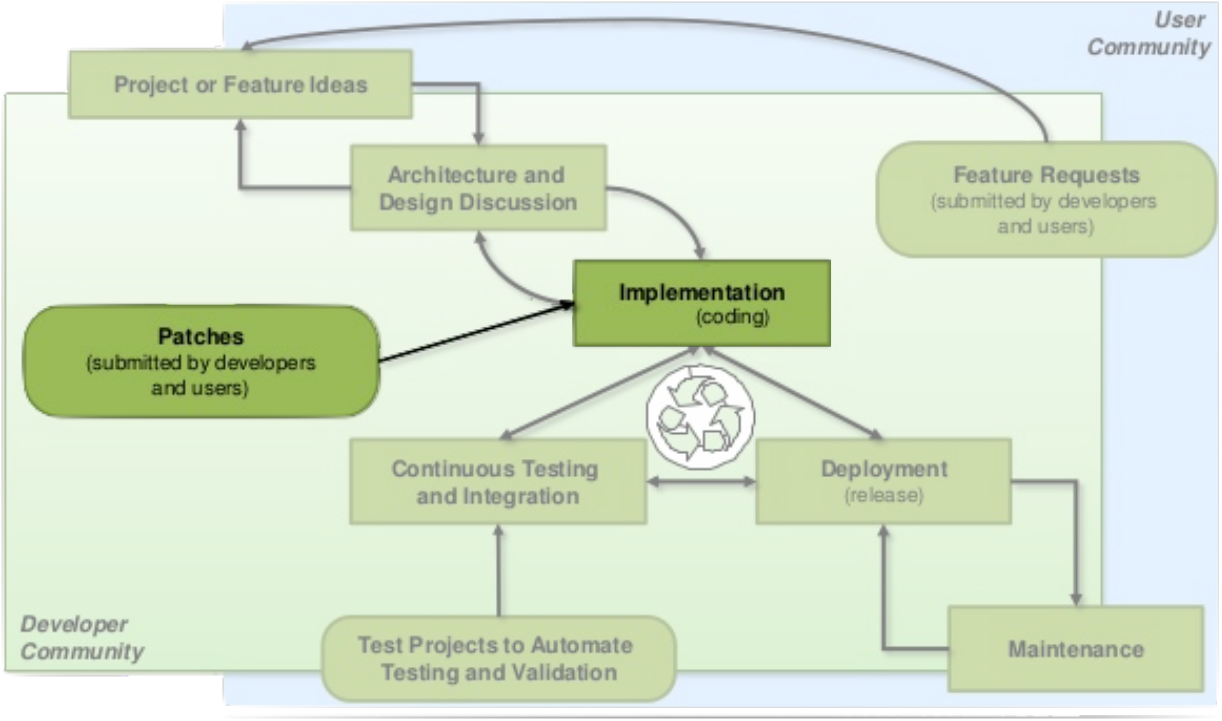
Open-Source Model

- Patches
  - Another source of input to an implementation is a patch, which is submitted by an end user or a developer to accompany a feature request or a defect report.
  - A patch is a file containing a series of commands that make changes to the current version of the source code to create a new version with the new feature or a fix for the defect.
  - Patches are subject to the same review and testing procedures as newly developed features.

Focus on Open-Source Methodologies

Open-Source Model

- Patches resolve defects reported by end-users or discovered by developers when reviewing the code for other purposes.



Samsung Open Source Group

Focus on Open-Source Methodologies

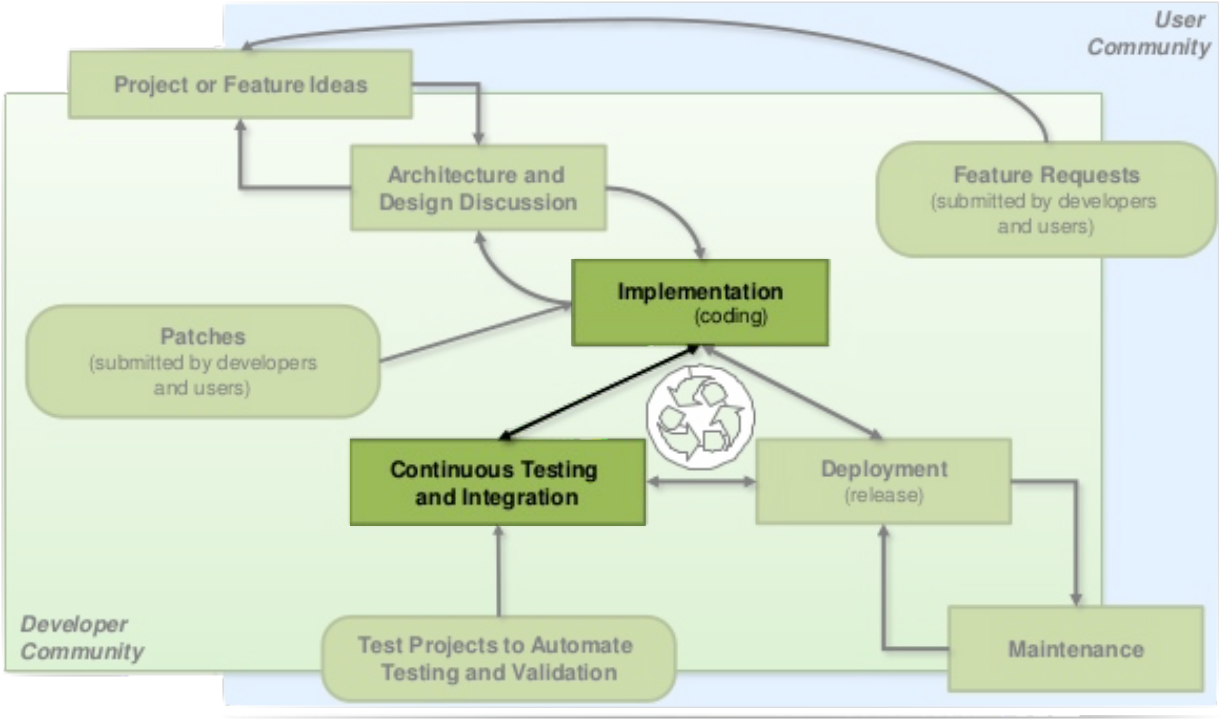
Open-Source Model

- Source code submission and feedback
  - When the code is functional and passed its unit tests, it is submitted to a *project advocate* for review and evaluation.
  - The advocator, together with other project participants, will review the submitted code and provide feedback.
  - If the review is positive, the code can be integrated into the source codebase for integration and testing.
  - The smaller the feature or patch that is submitted at an iteration, the easier it is to integrate, find bugs, and fix unintended consequences in the source code.

Focus on Open-Source Methodologies

Open-Source Model

- After review and feedback, code can be integrated into the main codebase for more extensive testing.



Samsung Open Source Group

Focus on Open-Source Methodologies

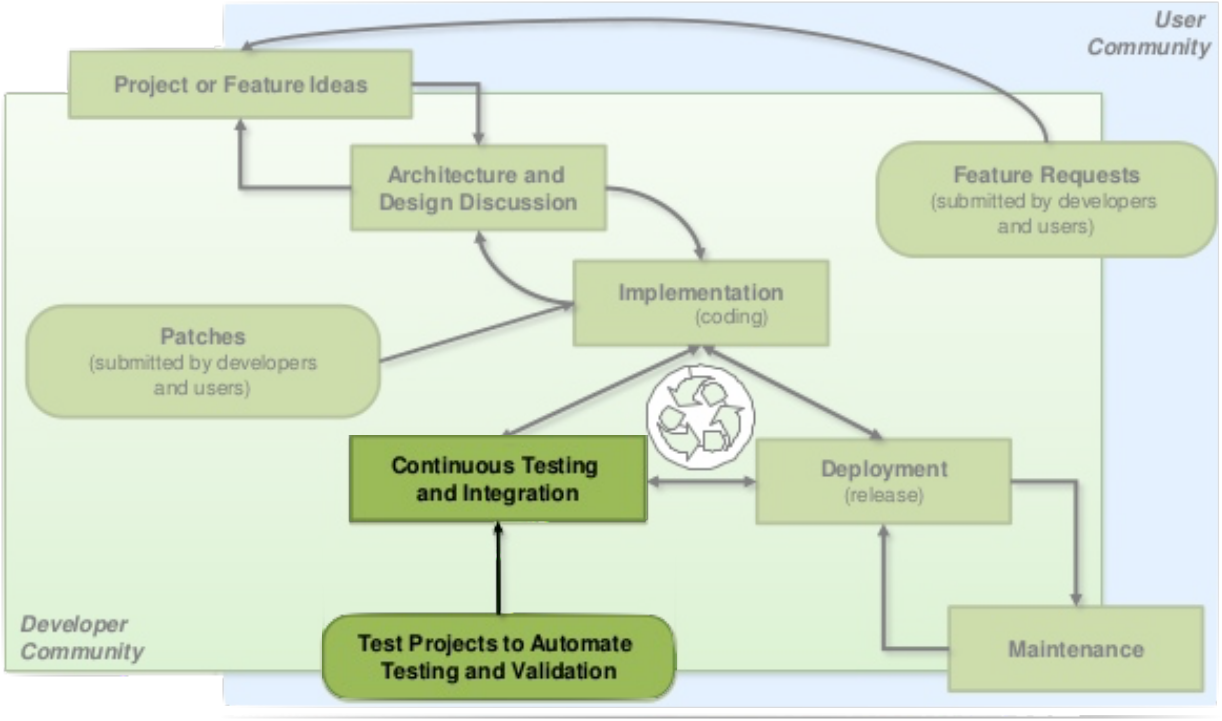
Open-Source Model

- Automated Test and Validation
  - Most open-source project have automated build tools and test suites.
  - These automations evaluate new code immediately after integrations to identify functional issues during active development.
  - Small incremental submissions are preferable to ensure that the quality of the source code is assured on all levels.

Focus on Open-Source Methodologies

Open-Source Model

- Automations evaluate new code after integrations to identify functional issues during active development.



Samsung Open Source Group

Focus on Open-Source Methodologies

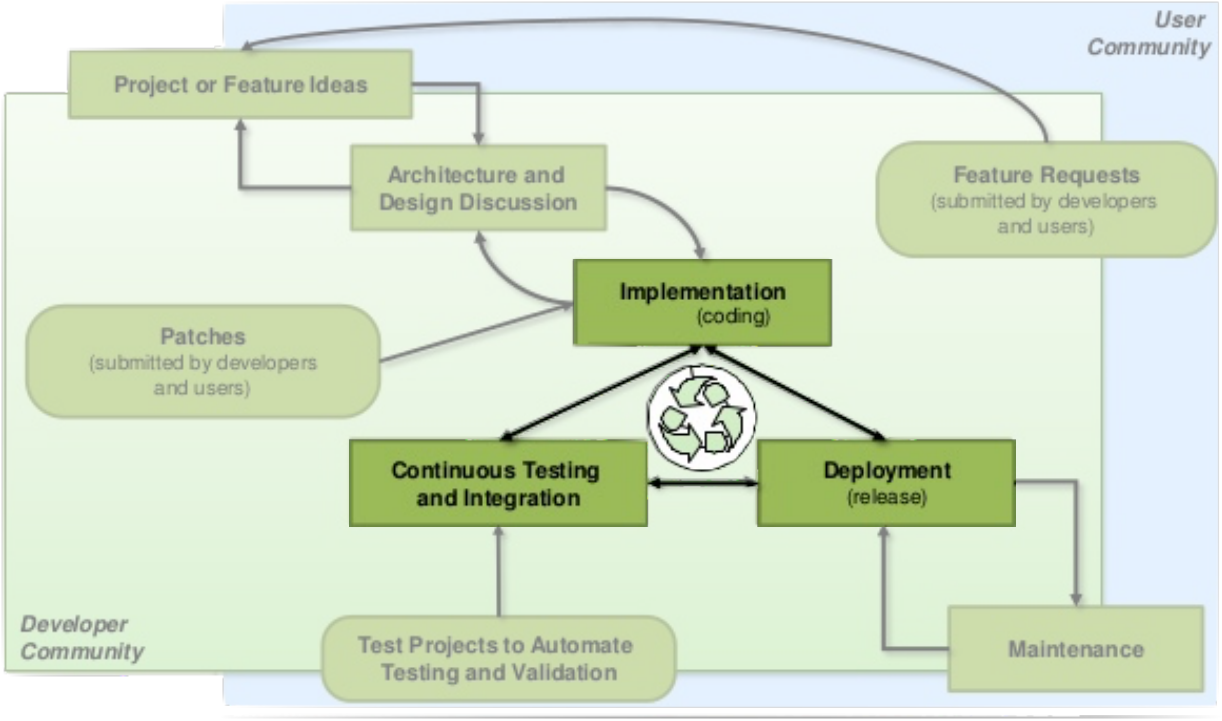
Open-Source Model

- Continuous build and release management
  - For all the feature and defect correction codes to be well-integrated, an automated continuous build and release management system is required.
  - The system pulls build targets, including the *current release*, *latest stable pre-release*, and *daily pre-release* from the source code system, and releases them to developers and end-users.
  - If errors are discovered, the system can also raise and log issues for someone in the community to evaluate and fix.

Focus on Open-Source Methodologies

Open-Source Model

- For code to be well integrated, a continuous build and release management system is required.



Samsung Open Source Group

Focus on Open-Source Methodologies

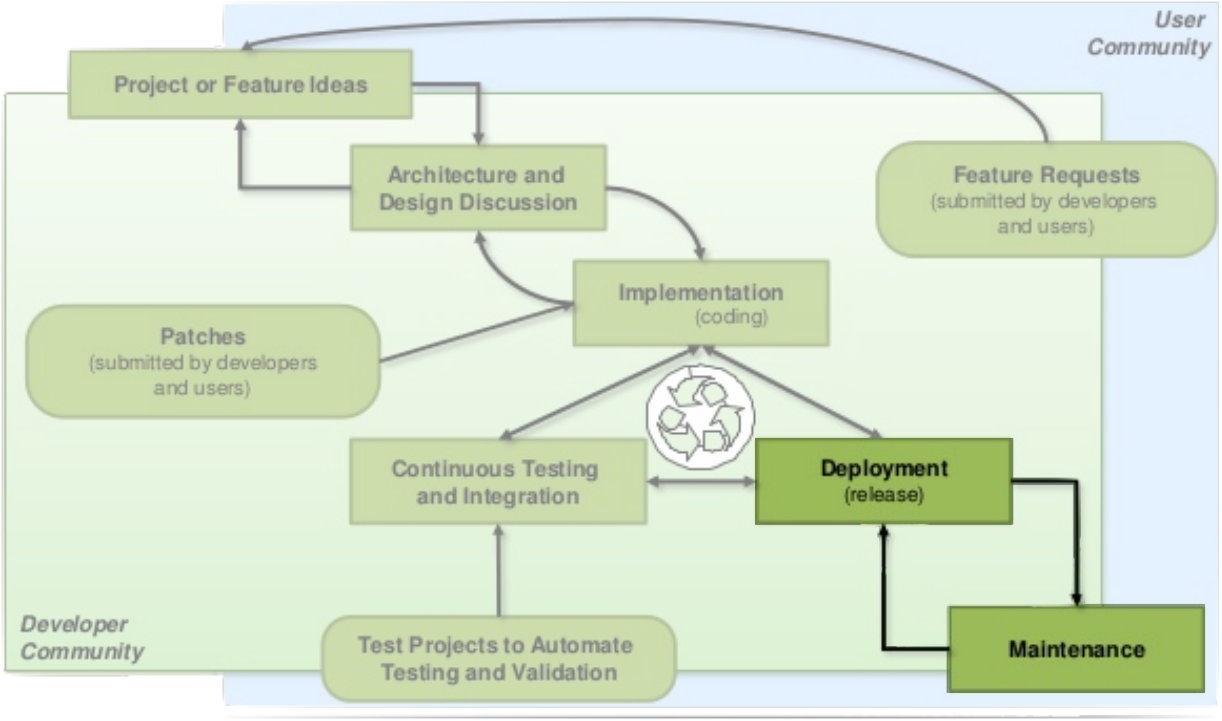
Open-Source Model

- Maintenance
  - Once a build target has been deployed, it enters a maintenance phase where new incremental releases are distributed to end-users and developers as they become available.
  - Defect reports are also captured and submitted against these targets, to be resolved later by end user or development communities.

Focus on Open-Source Methodologies

Open-Source Model

- As code enters maintenance phase, new incremental releases must be distributed to end-users and developers.



Samsung Open Source Group

Focus on Open-Source Methodologies

Open-Source Roles

- Some structuring of roles is needed in any online community that enables users to contribute freely.
- For example, online discussion groups have moderators who ensure everyone has an equal opportunity to participate, and the guidelines are being followed for everyone's benefit.
- Similarly, online knowledgebases like Wikipedia have levels of review for articles by new authors, as well as reviewers for specific knowledge areas such as computer science.

Focus on Open-Source Methodologies

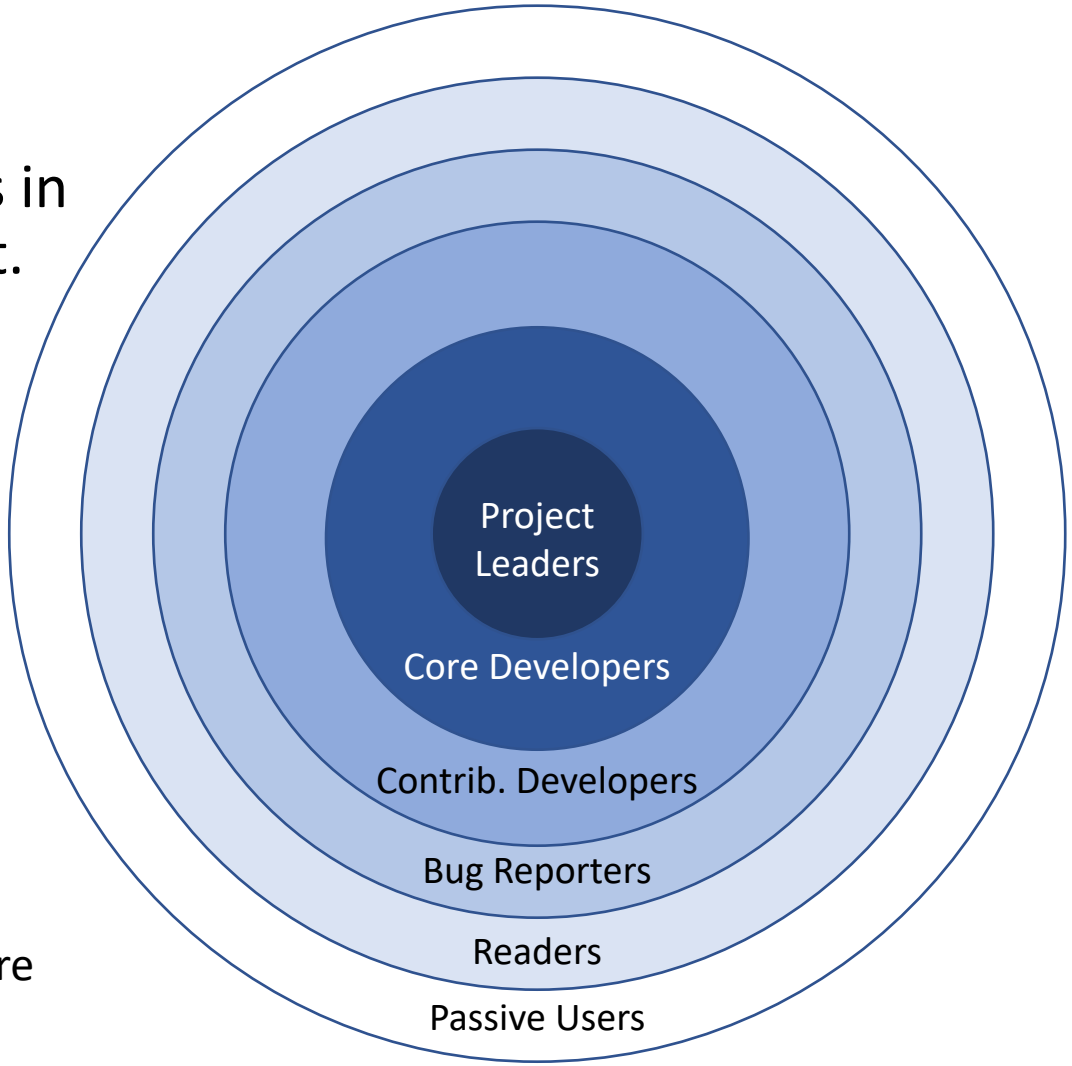
Open-Source Roles

- The same holds true within open-source communities.
- The most senior members of the community help ensure the integrity of the software by reviewing design and code changes that impact critical portions of the software.
- Generally, all submissions are peer-reviewed by other members of the community to ensure correctness.
- Members with a track record and are trusted can become “committers” and commit code directly to the codebase.
- Submissions by newer members of the community can only be committed by a “committer” after a full review.

Focus on Open-Source Methodologies

Open-Source Roles

- Here are sample roles in a open-source project.
  - **Project leaders** review architectural changes, evolution
  - **Core developers** work on core code
  - **Contributing developers** work on specific features
  - **Bug reporters** regularly review and test code
  - **Readers** evaluate code
  - **Passive users** use software



Focus on Open-Source Methodologies

Open-Source Licensing

- The earliest license for free software was written by Richard Stallman of the Free Software Foundation (FSF). It was known as the GNU Public License (GPL)
- Stallman believed free software should stay free, so the GPL required those who redistribute the software or a derivative work to use the GPL license and make the source available.
- This type of license where derivative works must be licensed under the same terms as the original work is known as *copyleft*. GPL was the first copyleft license to see wide use.

Focus on Open-Source Methodologies

Open-Source Licensing

- The group of developers who met in 1998 and coined “open source” believed that the GPL overly restricted uses of free software, especially in the commercial realm.
- They were especially critical of Stallman’s interpretation of a “derivative work”.
- For example, linking a GPL-licensed library into a program requires the source of the program be available under GPL, even if the shared library is not distributed with the program.
- As a more extreme example, shipping a utility program linked to a GPL licensed library with a larger system also requires releasing the source of the entire system under GPL.

Focus on Open-Source Methodologies

Open-Source Licensing

- When this group formed the Open Source Initiative (OSI), they realized that no one license met the needs of every situation.
- To ensure that their open software movement succeeded, they took a different approach to open-source licensing, one that also made it attractive for commercial adoption.
- Rather than providing a fixed license, they instead came up with a definition of “open source” and recognized any open-source license that is consistent with the OSI definition.

Focus on Open-Source Methodologies

Open-Source Licensing

- Here are the key elements of OSI's definition of open source.

OSI Open Source Definition	
Free Redistribution	must not restrict selling or giving licensed software away
Source Code	must include source code of licensed source code
Derived Works	must allow modifications and derived works under same license
Integrity of Source	may only restrict modified redistribution if patch file available
No Discrimination	must not restrict anyone from using licensed program
License Distribution	must apply to all recipients without separate license
No Specific Product	must not depend being part of specific distribution
No Restriction	must not restrict other software distributed with this software
Technology-neutral	must not be predicated on any individual technology

Focus on Open-Source Methodologies

Open Source Licensing

- The following OSI-approved licenses are popular, widely used, or have strong communities:
  - Apache License 2.0
  - BSD 3-Clause "New" or "Revised" license
  - BSD 2-Clause "Simplified" or "FreeBSD" license
  - GNU General Public License (GPL)
  - GNU Library or "Lesser" General Public License (LGPL)
  - MIT license
  - Mozilla Public License 2.0
  - Common Development and Distribution License
  - Eclipse Public License version 2.0

Focus on Open-Source Methodologies

Open-Source Licensing

- Can OSI licensed software be used for commercial purposes?
  - All open-source software can be used for commercial purpose; the Open Source Definition guarantees this. You can even sell open-source software.
  - Note that commercial is not the same as proprietary. If you receive software under an open-source license, you can always use that software for commercial purposes.
  - However, that does not always mean you can place further restrictions on people who receive the software from you.
  - In particular, copyleft-style open-source licenses require that, in at least some cases, when you distribute the software, you must do so with the same license under which you received it.

Focus on Open-Source Methodologies

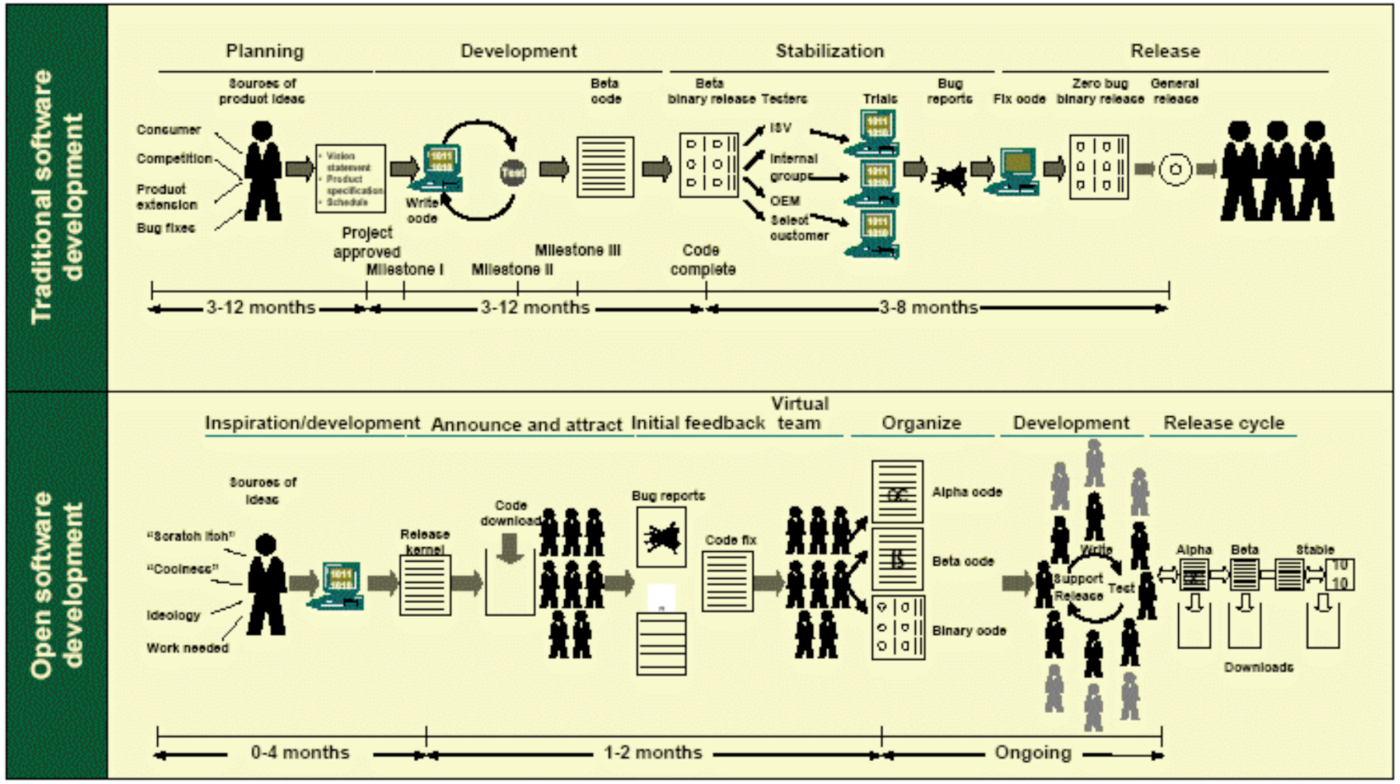
Comparing Open Source to Other Models and Methodologies

- Development of open-source software has some unique aspects because of different underlying conditions and assumptions.
- It is useful to compare open-source models with those of others we have studied so far to look for similarities and differences.
- We will compare it to a traditional “heavyweight” model like waterfall. The following slide shows both models along with a common timeline for comparison purposes.

Focus on Open-Source Methodologies

Comparing Open Source to Other Models and Methodologies

- Traditional vs. open-source development timeline



Kajsa Ahlgren and Elin Dahlberg (2008)

Focus on Open-Source Methodologies

Comparing Open Source to Other Models and Methodologies

- From this illustration, it is clear the open source is a lightweight rather than a heavy-weight model.
  - The time frame for the open-source model is considerably shorter than for a heavy-weight process like waterfall.
  - The amount time spent creating non-credited development artifacts are significantly less for open source.
  - The resourcing for the open-source model is much more efficient in terms of time to recruit and organize a community of end-users and developers.

Focus on Open-Source Methodologies

Comparing Open Source to Agile Models and Methodologies

- How does open source compare to the agile model of software development?
  - Agile and open source, at first glance, seem radically different. Agile methodologies are about small, collocated teams, and open source are about large, distributed ones.
  - Agile methodologies arose, largely, from the ranks of paid consultants, whereas open source seems like a “hippie” phenomenon.
  - The difference have more to do with conversations, diversity of participants, and transparency of the process to the outside world than the philosophy of design and development.
  - The two approaches share many principles and values.

Focus on Open-Source Methodologies

Comparing Open Source to Agile Models and Methodologies

- Most principles from the Agile Manifesto also apply to OSS.
  1. *Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.*

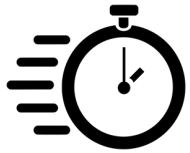
     - Open source does not discuss the customer, but open-source projects do nightly builds, and frequent releases for in-situ testing.
  2. *Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.*

     - Open-source projects resist major changes as time goes on, but there is always the possibility of forking if changes seem to be worthwhile.

Focus on Open-Source Methodologies

Comparing Open Source to Agile Models and Methodologies

- Most principles from the Agile Manifesto also apply to OSS.
  3. *Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter time scale.*



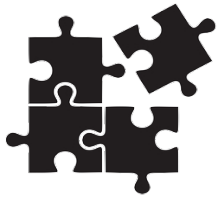
     - Open source usually delivers working code every night, and an open-source motto is "release early, release often."
  4. *Businesspeople and developers must work together daily throughout the project.*

     - Open-source projects do not have a concept of a businessperson with whom they work, user participants in the project serve the same role.

Focus on Open-Source Methodologies

Comparing Open Source to Agile Models and Methodologies

- Most principles from the Agile Manifesto also apply to OSS.
  5. *Make projects on motivated individuals. Provide the environment and support they need, and trust them to get the job done.*



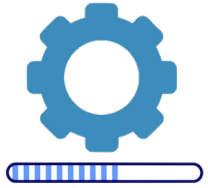
  - All open-source projects do this, almost by definition. Open-source projects are purely voluntary, so motivation is guaranteed.
  - Open-source projects use a set of agreed-on tools for version control, compilation, debugging, bug and issue tracking, and discussion.

Focus on Open-Source Methodologies

Comparing Open Source to Agile Models and Methodologies

- Most principles from the Agile Manifesto also apply to OSS.
  6. *The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.*

    - Open source differs most from agile methodologies here. Open-source projects value written communication over face-to-face communication.
    - On the other hand, open-source projects can be widely distributed, and don't require collocation.
  7. *Working software is the primary measure of progress.*
    - This is in perfect agreement with open source.



Focus on Open-Source Methodologies

Comparing Open Source to Agile Models and Methodologies

- Most principles from the Agile Manifesto also apply to OSS.
  8. *Agile processes promote sustainable development. Users, Developers, and Sponsors, should be able to maintain a constant pace indefinitely.*
     - Although this uses vocabulary that open-source developers would not use, the spirit of the principle is embraced by open source.

  9. *Continuous attention to technical excellence and good design enhances agility.*
     - Open source is predicated on technical excellence and good design.

Focus on Open-Source Methodologies

Comparing Open Source to Agile Models and Methodologies

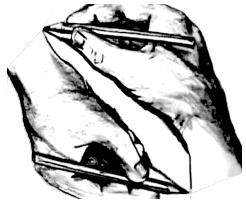
- Most principles from the Agile Manifesto also apply to open-source software.
  10. Simplicity - the art of maximizing the amount of work not done - is essential.
      - Open-source developers agree: simplicity is essential, but open-source projects also do not worry as much about scarcity of resources as agile projects do.
      - There are rarely contractually-committed people on open-source projects, certainly not the purely voluntary ones, so the amount of work to be done depends on the beliefs of the individual developers.

Focus on Open-Source Methodologies

Comparing Open Source to Agile Development Models

- Most principles from the Agile Manifesto also apply to OSS.
  11. *The best architectures, requirements, and designs emerge from self-organizing teams.*
      - Possibly open-source developers would not state things this way, but the nature of open-source projects depends on this being true.

  12. At regular intervals, the team reflects on how to become more effective, and then tunes and adjusts its behavior accordingly.
      - Not done much in open-source projects, but as open-source projects mature, they tend to develop a richer set of governance mechanisms.
      - For example, Apache started with a simple governance structure like that of Linux and now there is the Apache Software Foundation.



Focus on Open-Source Methodologies

Comparing Open Source to Agile Models and Methodologies

- In short, both the agile and open-source methodologies embrace many of the same principles and values:
  - trying to build software suited especially to a class of users
  - interacting with those users during the design and implementation phases
  - blending design and implementation
  - working in groups
  - respecting technical excellence
  - doing the job with motivated people
  - generally engaging in continuous (re)design.

Focus on Open-Source Methodologies

Getting Involved with Open-Source Projects

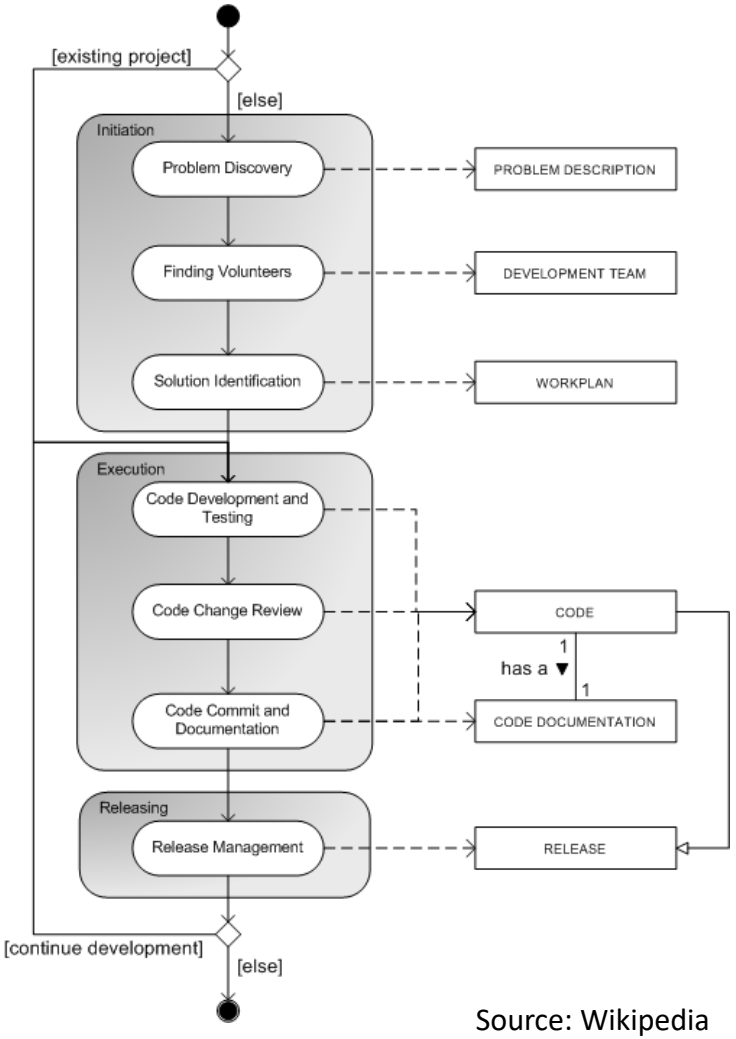
- Getting involved in open source for the first time can seem scary and a little overwhelming.
- Open source means being in a community, learning the rules, and becoming a good citizen of the community
- Open-source communities have a governance structure that determines how developers and others can join and contribute.

- If you decide to start your own open-source project, over time you will also need to develop a governance structure as others join the community.

Focus on Open-Source Methodologies

Joining vs. Starting a Project

- Your first decision is whether to join an existing project or start one of your own.
- It is important to investigate what is already there. It is often better to join an existing project than start your own.
- For a new project, the process starts at the Initiation phase.
- For an existing project, the process goes directly to the Execution phase.



Source: Wikipedia

Focus on Open-Source Methodologies

Joining an Existing Open-Source Project

- Why join an existing open-source project?
  - Improve software you rely on
    Many contributors start as users of software they contribute to.
  - Improve existing skills
    If you're looking for practice, there is a task for you.
  - Meet people interested in similar things
    Many people form lifelong friendships through their participation.
  - Find mentors and teach others
    The acts of learning and teaching can be a fulfilling
  - Build public artifacts to help grow your reputation
    You get free examples to demonstration what you can do.
  - Learn people skills
    Practice leadership and management skills

Focus on Open-Source Methodologies

Joining an Existing Open-Source Project

- Examples of open-source organizations with open-source projects you can join:
  - Android Open-Source Project
  - Apache Software Foundation
  - Document Foundation
  - Eclipse Foundation
  - Free Software Foundation
  - Linux Foundation
  - Mozilla Foundation
  - Python Software Foundation

Focus on Open-Source Methodologies

Joining an Existing Open-Source Project

- For more open-source organizations with projects you can join, visit:
  - OpenSource.com: *https://opensource.com/resources/organizations*
  - Wikipedia.org: *https://en.wikipedia.org/wiki/List_of_free_and_open-source_software_organizations*
- For getting-started guides to joining and working in open-source projects, visit:
  - FreeCodeCamp.org: *https://www.freecodecamp.org/news/the-definitive-guide-to-contributing-to-open-source-900d5f9f2282/*
  - OpenSource.guide: *https://opensource.guide/how-to-contribute/*

Focus on Open-Source Methodologies

Starting an Open-Source Project

- An open-source project can start in several ways.
  - An individual who senses the need for a project announces the intent to develop a project in public.
  - A developer working on a limited but working codebase, releases it to the public as the first version of an open-source program.
  - The source code of a mature project is released to the public.
  - A well-established open-source project can be forked by an interested outside party
- In his “The Cathedral and the Bazaar” essay, Eric Raymond noted that announcing the intent for a project is usually inferior to releasing a working product to the public.

Focus on Open-Source Methodologies

Starting an Open-Source Project

- There are several ways to start an open-source project:
  - Start a new project at well-established organizations.
    - Advantages:
      - Widely recognized
      - Easier to gain a following
    - Disadvantages:
      - More difficult to start project
  - Create your own open-source project on GitHub.com.
    - Advantages:
      - Easy to get started
    - Disadvantages:
      - Difficult to find your project
      - Difficult to gain a following

Focus on Open-Source Methodologies

Starting an Open-Source Project

- For a getting-started guide to starting open-source projects, visit:
  - The Linux Foundation: https://www.linuxfoundation.org/resources/open-source-guides/starting-open-source-project/

Focus on Open-Source Methodologies

Lessons for Open-Source Projects

- In his essay, Eric Raymond proposed 19 “lessons” for good open-source software:
  1. Every good work of software starts by “scratching a developer's personal itch.”
  2. Good programmers know what to write. Great ones know what to rewrite (and reuse).
  3. Plan to throw one [version] away; you will, anyway. (Copied from Frederick Brooks' *The Mythical Man-Month*)
  4. If you have the right attitude, interesting problems will find you.
  5. When you lose interest in a program, your final duty to it is to hand it off to a competent successor.

Focus on Open-Source Methodologies

Lessons for Open-Source Projects

- In his essay, Eric Raymond proposed 19 “lessons” for good open-source software:
  6. Treating your users as co-developers is your least-hassle route to rapid code improvement and effective debugging.
  7. Release early. Release often. And listen to your customers.
  8. Given a large enough beta-tester and co-developer base, almost every problem will be characterized quickly, and the fix will be obvious to someone.
  9. Smart data structures and dumb code works a lot better than the other way around.
  10. Treat your beta-testers as if they're your most valuable resource, and they will respond by becoming your most valuable resource.

Focus on Open-Source Methodologies

Lessons for Open-Source Projects

- In his essay, Eric Raymond proposed 19 “lessons” for good open-source software:
  11. The next best thing to having good ideas is recognizing good ideas from your users. Sometimes the latter is better.
  12. Often, the most striking and innovative solutions come from realizing that your concept of the problem was wrong.
  13. Perfection (in design) is achieved not when there is nothing more to add, but rather when there is nothing more to take away. (Attributed to French aviation pioneer Antoine de Saint-Exupéry)
  14. Any tool should be useful in the expected way, but a truly great tool lends itself to uses you never expected.

Focus on Open-Source Methodologies

Lessons for Open-Source Projects

- In his essay, Eric Raymond proposed 19 “lessons” for good open-source software:
  15. When writing gateway software of any kind, take pains to disturb the data stream as little as possible—and never throw away information unless the recipient forces you to!
  16. When your language is nowhere near Turing-complete, syntactic sugar can be your friend.
  17. A security system is only as secure as its secret. Beware of pseudo-secrets.
  18. To solve an interesting problem, start by finding a problem that is interesting to you.
  19. Provided that the development coordinator has a communications medium at least as good as the Internet, and knows how to lead without coercion, many heads are inevitably better than one.