Introduction to open-Source Software (OSS)

Concepts, strategies, and methodologies related to open-source software development

Week 03 – Lecture 05

Jamil Hussain
jamil@sejong.ac.kr
010-6252-8807

Office: 421, Innovation Center
Sejong University

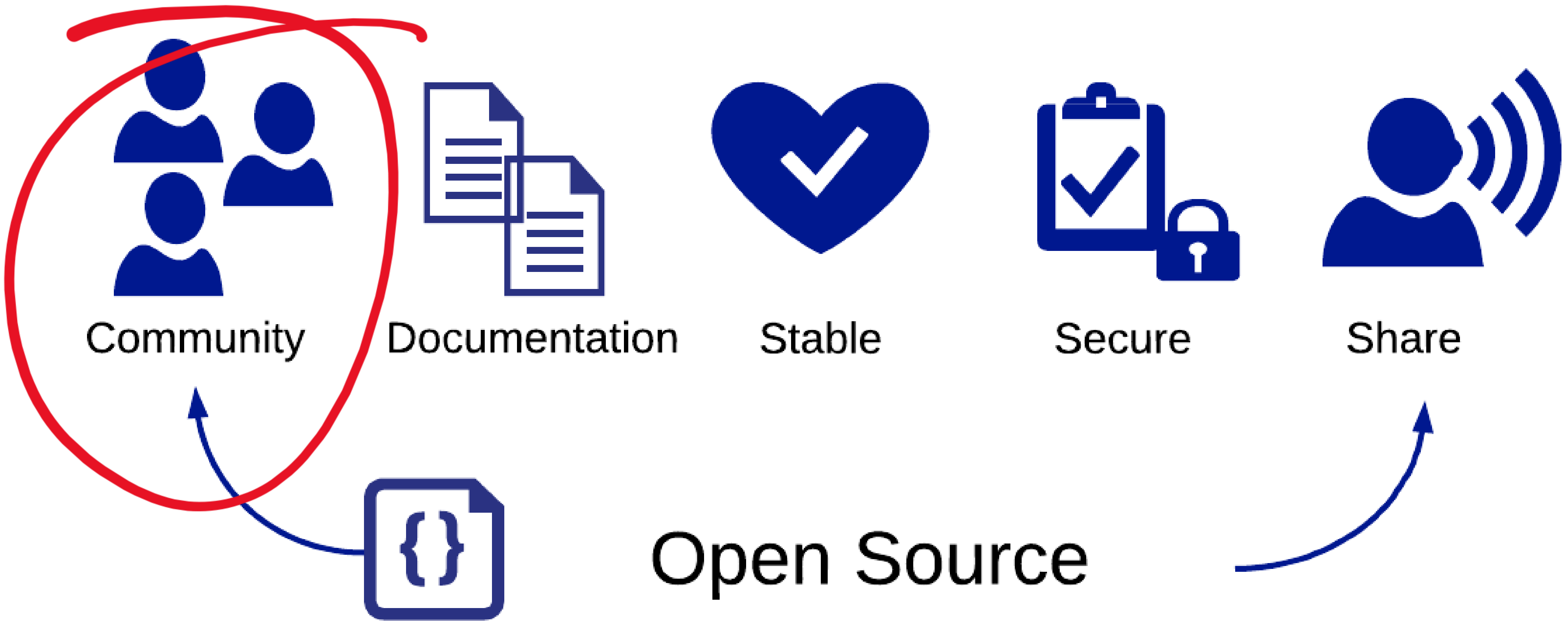
Recap

- Overview of Copyright and Licenses
- Understanding the legal implications of open-source
- Licenses in a Nutshell
- Software license categories
- Aspects of Licenses
- The GPL and License Compatibility
- Choosing a License

Today, Agenda

- ◯ The community and its structure
- ◯ Why contribute to open source?
- ◯ Class Activity

세종대학교 SEJONG UNIVERSITY



Source: https://hackernoon.com/what-i-learned-from-building-my-first-successful-open-source-project-72f7429145a0

Open Source Project Community

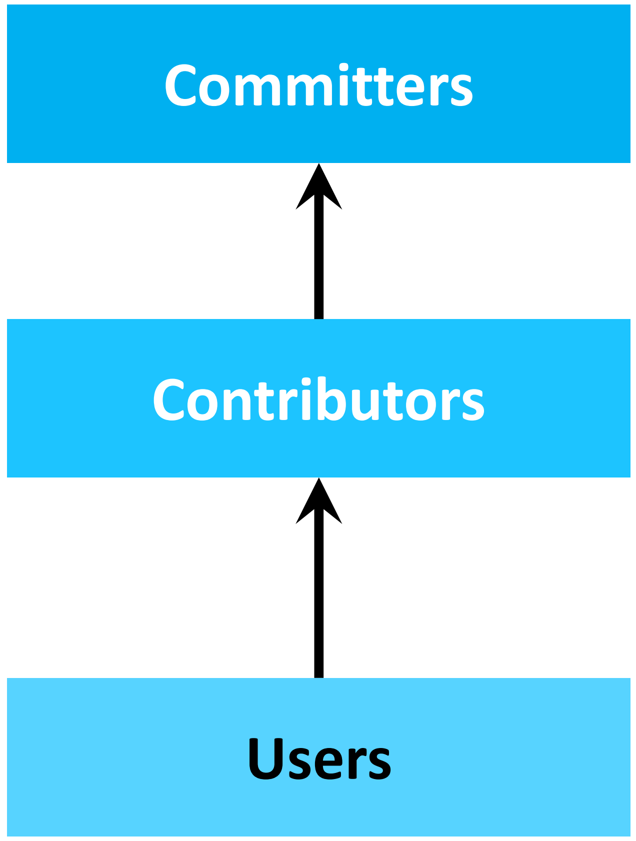
- A software that is collectively developed by a community of **technologists** interested in a **particular application** or **tool** and then **distributed** at no cost to the **broader community** of individuals who can find a use for it.

- An Open-Source project community is
  - The **group** of **peoples and companies** engaged in an open-source project
- The **developer community** is
  - The subset of the project community that is **developing** the software
- The user community is
  - The subset of the project community that is **using the software**

Open-Source Project Community- The Onion Model of FOSS

- To understand the structure and dynamics of FOSS communities, the “Onion model” provides a useful framework.
  - **Leaders**: Project founders or long-term maintainers. Set the vision and make key decisions.
  - **Core Members**: Trusted contributors managing key sections of the codebase.
  - **Active Developers**: Regular contributors involved in coding, bug fixing, and feature development.
  - **Peripheral Developers**: Occasional contributors focusing on specific tasks.
  - **Bug Fixers**: Identify and fix issues, enhancing stability.
  - **Readers**: Engage with documentation and code without contributing.
  - **Passive Users**: Use the software without participating actively.

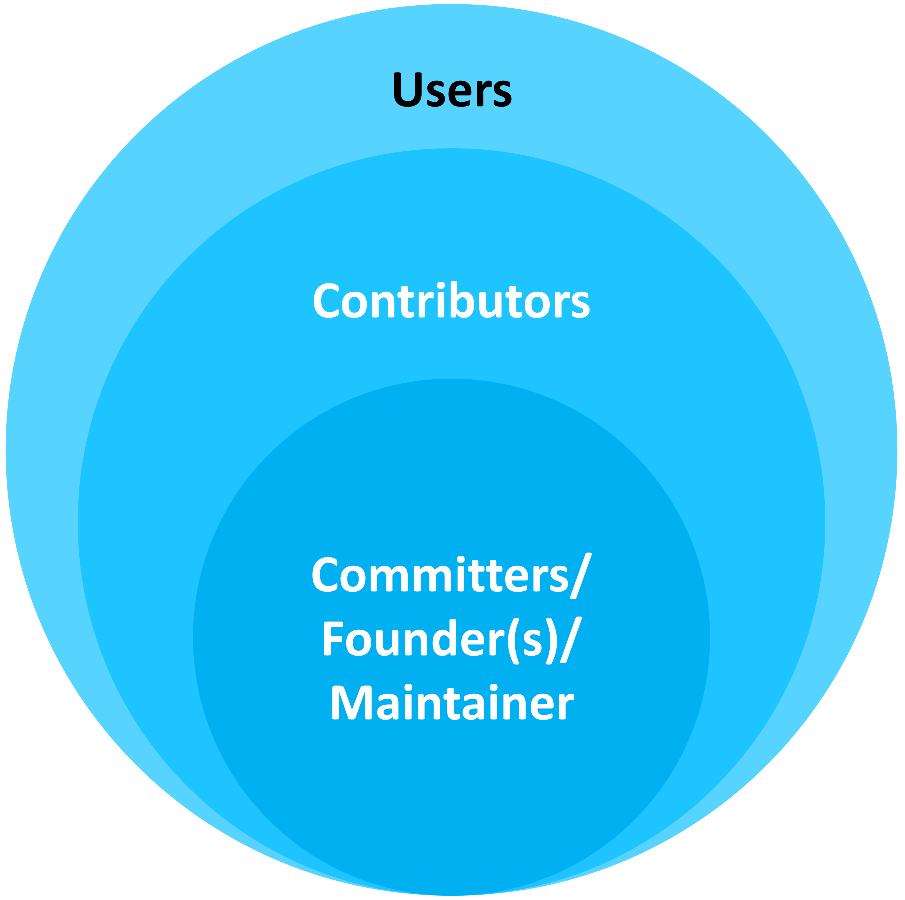
Open-Source Project Community- The Onion Model of FOSS



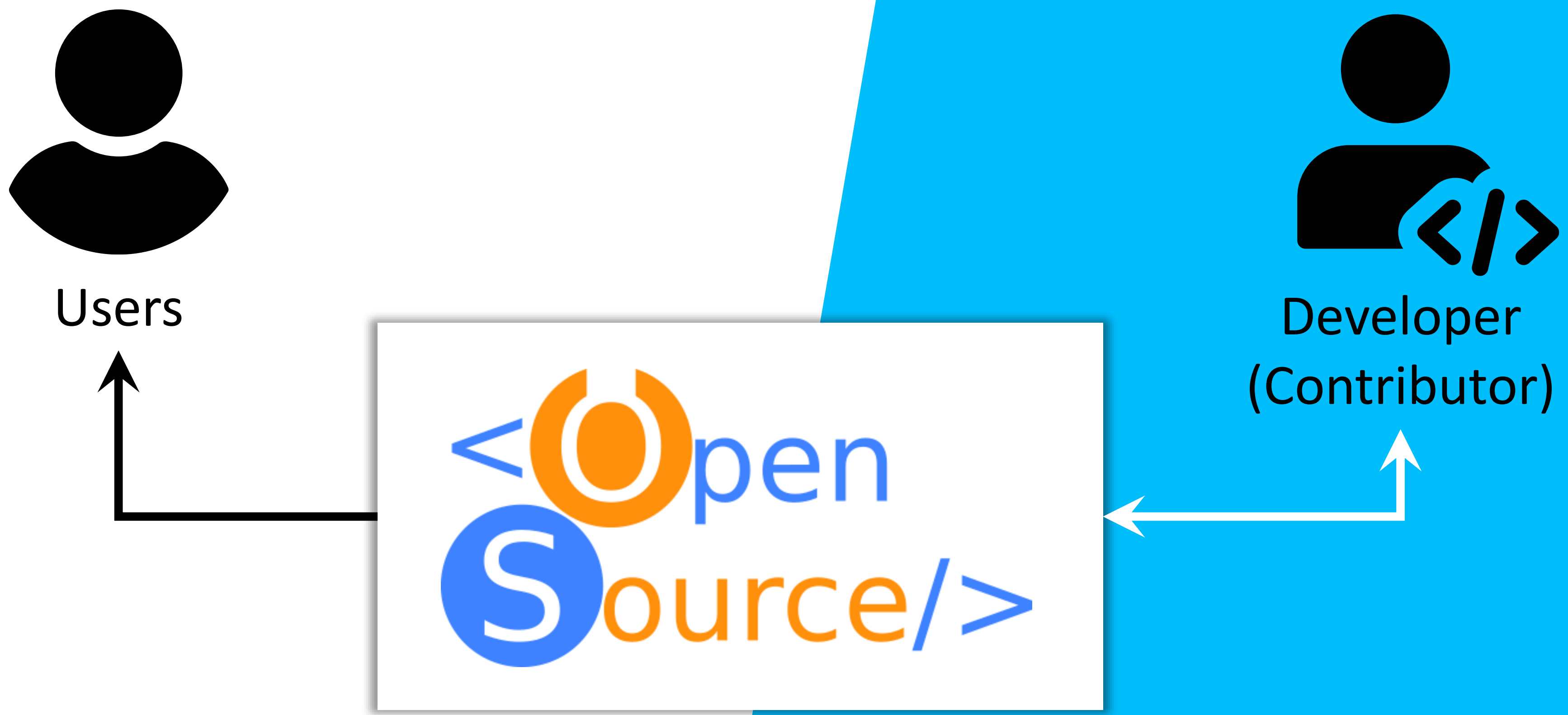
- Project Leader
- Core member
- Has commit(write) rights
- Perform bulk of the work, patch review

- Developer
- Provide small features, bug fixes
- Submit patches (no commits rights)

- People who know and use the project
- If so, help with comments, feedback..



Open-Source Project Community- The Onion Model of FOSS



Open-source projects should be run in such a way as to make that transition welcoming to anyone interested

Open-Source Project Community- The Onion Model of FOSS

Contributors Types

Core Contributors

- Most senior and experienced people in the project
- Provide guidance and mentorship to the newer community members
- Hold the *commit bit* - they are able to approve changes made to the project

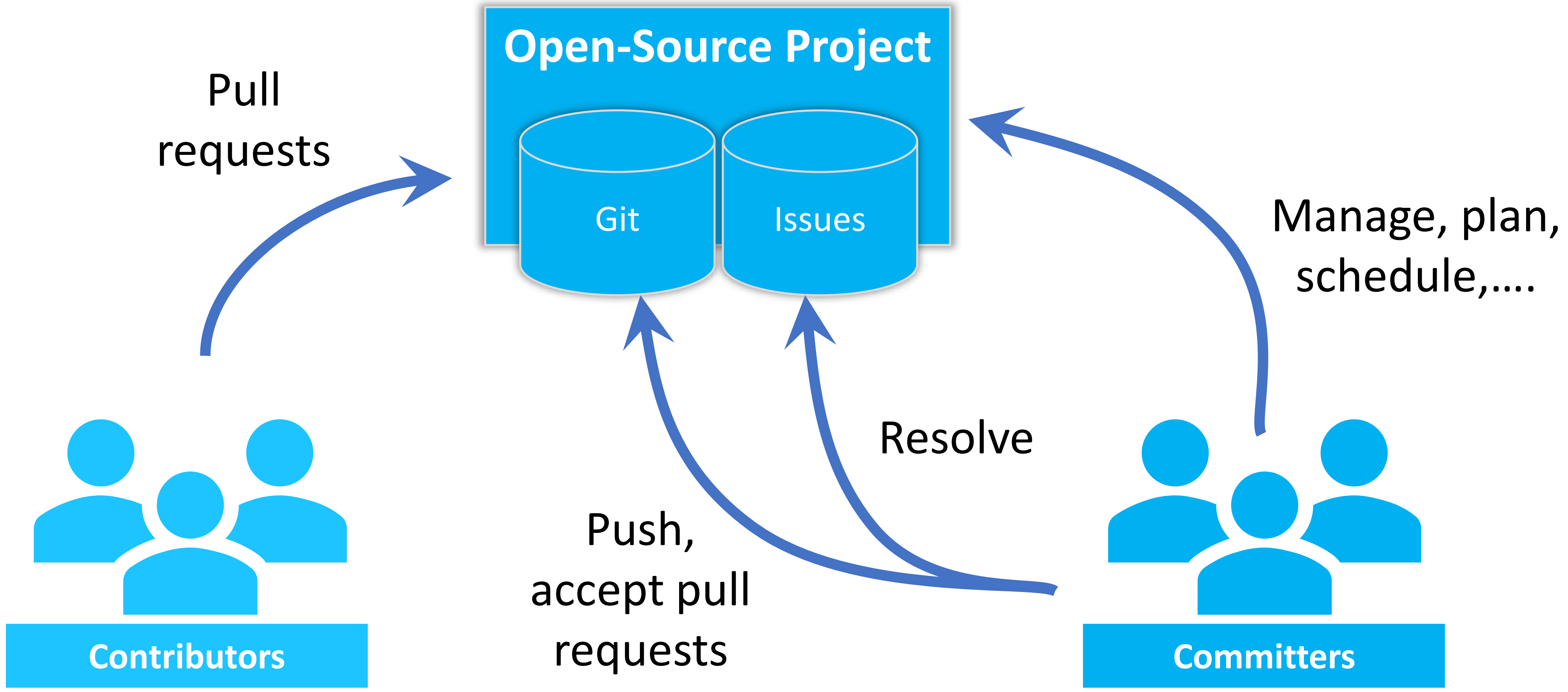
Non-core contributors

- Make regular contributions
- Active in issue discussions
- Provide feedback to the newer members of the community

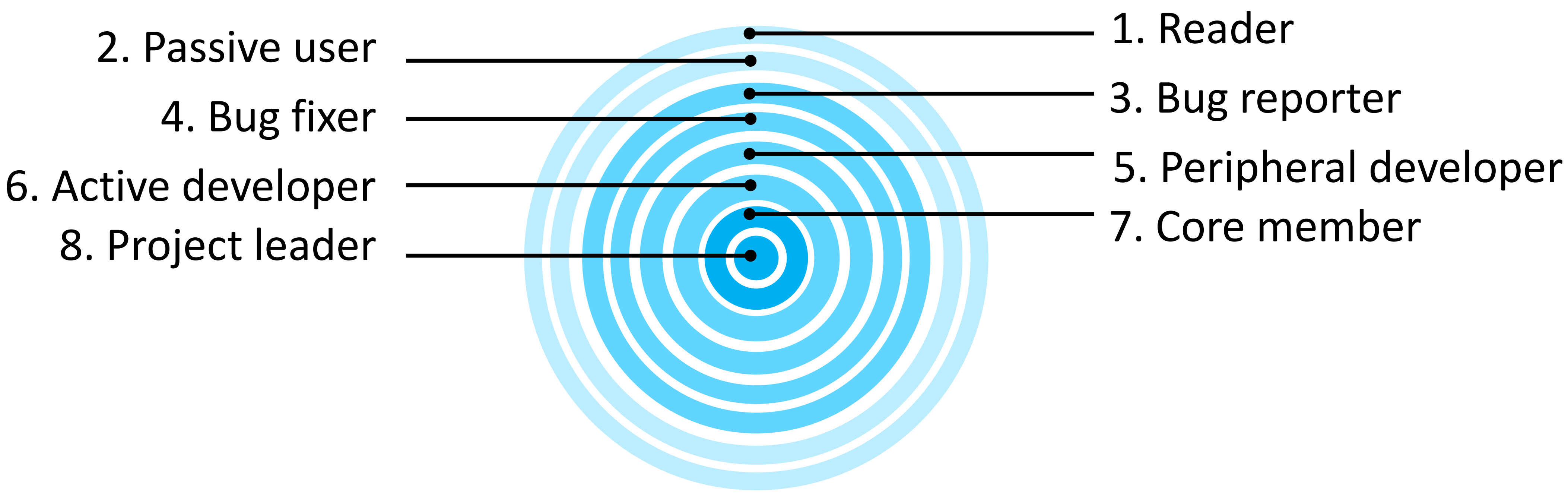
New contributors

- People interested in contributing to the project
- Still learning the structure of the project and how to interact with the community

Open-Source Project Community- The Onion Model of FOSS

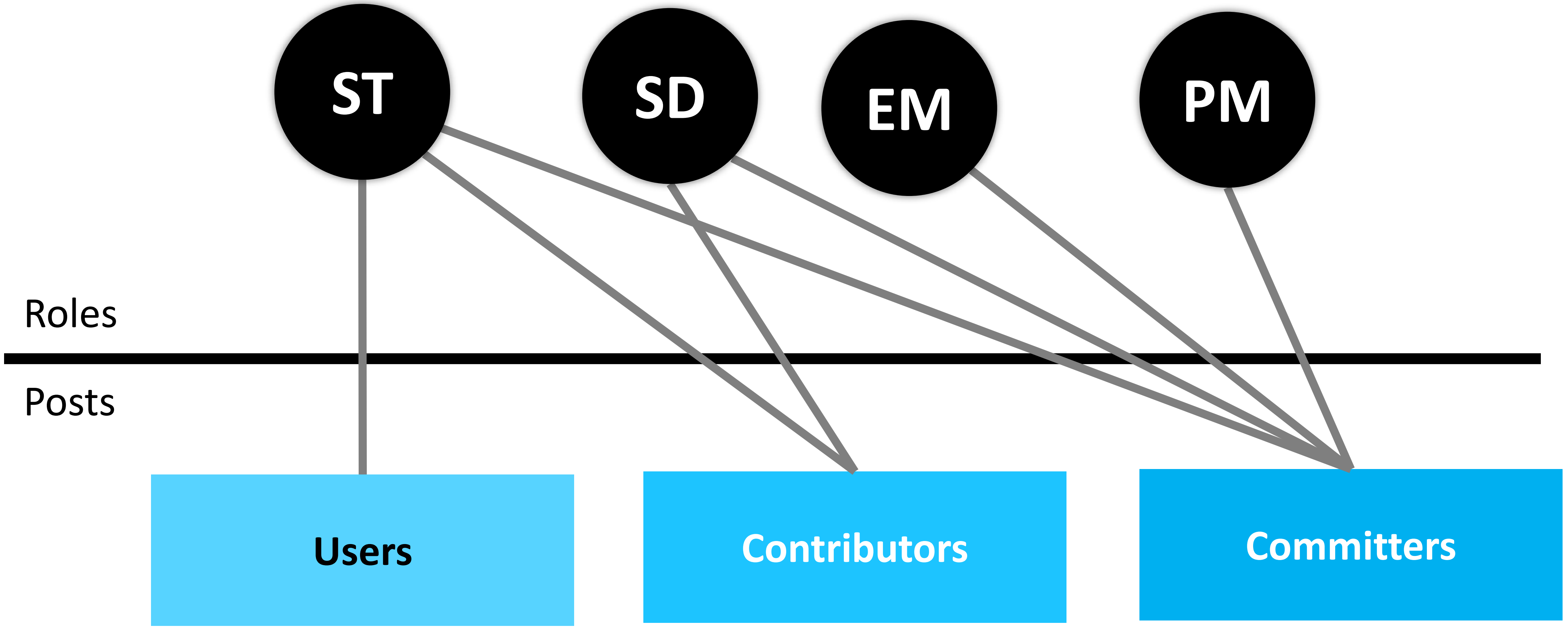


Open-Source Project Community- The Onion Model of FOSS



Users | **Contributors** | **Committers**

Roles and Post in Open-Source Projects



Why contribute to open source?

- Contributing to open source can be a rewarding way to learn, teach, and build experience in just about any skill you can imagine.

- Why do people contribute to open source? Plenty of reasons!
  - Improve software you rely on
  - Improve existing skills
  - Meet people who are interested in similar things
  - Find mentors and teach others
  - Build public artifacts that help you grow a reputation (and a career)
  - Learn people skills
  - It’s empowering to be able to make changes, even small ones

What it means to contribute

If you’re a new open-source contributor, the process can be frightening. How do you find the right project? What if you don’t know how to code? What if something goes wrong?

Not to worry! There are all sorts of ways to get involved with an open-source project, and a few tips will help you get the most out of your experience.

What it means to contribute

- **Do you like planning events?**
  - Organize workshops or meetups about the project, like @fzamperin did for NodeSchool
  - Organize the project's conference (if they have one)
  - Help community members find the right conferences and submit proposals for speaking
- **Do you like to design?**
  - Restructure layouts to improve the project's usability
  - Conduct user research to reorganize and refine the project's navigation or menus, like Drupal suggests
  - Put together a style guide to help the project have a consistent visual design
  - Create art for t-shirts or a new logo, like hapi.js's contributors did

What it means to contribute

- **Do you like to write?**
  - Write and improve the project's documentation
  - Curate a folder of examples showing how the project is used
  - Start a newsletter for the project, or curate highlights from the mailing list
  - Write tutorials for the project, like PyPA's contributors did
  - Write a translation for the project's documentation

- **Do you like organizing?**
  - Link to duplicate issues, and suggest new issue labels, to keep things organized
  - Go through open issues and suggest closing old ones, like @nzakas did for ESLint
  - Ask clarifying questions on recently opened issues to move the discussion forward

What it means to contribute

- **Do you like to code?**
  - Find an open issue to tackle, like @dianjin did for Leaflet
  - Ask if you can help write a new feature
  - Automate project setup
  - Improve tooling and testing
- **Do you like helping people?**
  - Answer questions about the project on e.g., Stack Overflow (like this Postgres example) or Reddit
  - Answer questions for people on open issues
  - Help moderate the discussion boards or conversation channels

What it means to contribute

- **Do you like helping others code?**
  - Review code on other people’s submissions
  - Write tutorials for how a project can be used
  - Offer to mentor another contributor, like @ereichert did for @bronzdoc on Rust

Finding a project to contribute to

- Now that you've figured out how open source projects work, it's time to find a project to contribute to!
- You can also use one of the following resources to help you discover and contribute to new projects:
- GitHub Explore
- Open Source Friday
- First Timers Only
- CodeTriage
- 24 Pull Requests
- Up For Grabs
- Contributor-ninja
- First Contributions
- SourceSort

Class Activity

Open-Source Project Roles and Responsibilities

Open-Source Project Roles and Responsibilities

– Card Sorting Activity

- **Case Study: Managing an Open-Source Project Amid Challenges**
  - You are part of an open-source project that manages a popular JavaScript library used for web development. The project is growing rapidly, but new challenges have emerged as the community expands. You will work in groups to sort roles and responsibilities while navigating complex project scenarios.

Open-Source Project Roles and Responsibilities

- **Instructions:**
  1. **Roles:**
     1. Users
     2. Contributors
     3. Committers
     4. Project Leaders
  2. **Responsibilities:**
     1. Report Bugs
     2. Develop Features
     3. Review Code
     4. Manage Releases
     5. Write Documentation
     6. Perform Security Audits
     7. Plan Project Roadmap
  3. **Scenarios:**
     Each group will receive scenario cards that present real-world challenges, such as:
     1. Major security vulnerability
     2. Conflicting feature requests
     3. New contributor code quality issues
     4. Bug backlog crisis
     5. Community conflict

Sample Scenario : Influx of New Contributors Overwhelming the Core Team

- Your open-source project has gained popularity, attracting many new contributors who are submitting code. However, a lot of their **contributions are poorly tested**, leading to **more work** for the core team to **review and fix issues**. The core team is **overwhelmed** with the volume of pull requests and is struggling to **keep up**, causing delays and frustration among contributors.
- **Challenges:**
  - Managing large volumes of **low-quality contributions**.
  - Keeping new contributors engaged while maintaining project standards.
  - Ensuring code quality while speeding up the review process.

Sample Scenario : Influx of New Contributors Overwhelming the Core Team

- **Roles Involved:**
  - **New Contributors:** Provide small features or fixes and undergo mentorship.
  - **Experienced Contributors:** Act as mentors, reviewing and guiding new contributors.
  - **Committers:** Implement a staged review system and focus on high-priority pull requests.
  - **Project Leaders:** Develop and enforce contribution guidelines to maintain quality and efficiency.

Sample Scenario : Influx of New Contributors Overwhelming the Core Team

- **Action Plan:**
  1. **Onboarding and Mentorship:**
     - Assign **experienced contributors** as mentors to new contributors. This will help train them on project standards and improve the quality of their code submissions.
     - Create **onboarding documentation** and video tutorials to help new contributors understand the workflow and coding standards before they submit code.
  2. **Streamlined Review Process:**
     - Implement a **tiered code review process** where simple contributions are first reviewed by newer contributors or junior developers before being escalated to core committers. This will reduce the load on senior reviewers.
     - Automate checks using **Continuous Integration (CI)** tools to detect basic issues such as style errors, missing tests, or failed builds before human reviewers spend time on them.
  3. **Improving Contribution Guidelines:**
     - **Project leaders** should create detailed contribution guidelines, including clear instructions for submitting pull requests, code style rules, and testing requirements. These guidelines can be shared with new contributors to ensure submissions meet a minimum quality standard.
  4. **Engagement and Feedback:**
     - Provide regular **feedback** to new contributors on their pull requests, pointing out areas of improvement and encouraging best practices. This will help them grow as contributors and feel supported.
     - **Recognize contributions** from new contributors to keep them motivated, even if their contributions are small. This could include shout-outs in community meetings or blog posts.

Scenario 1- Team 1: Major Security Vulnerability

- A critical security flaw has been discovered in your project. Users are reporting that the flaw can potentially expose sensitive data. The issue requires immediate attention, but some key contributors are unavailable for the next few days. You need to prioritize this security fix and release a patch as soon as possible.
- **Challenges:**
  - Time pressure: You have 48 hours to fix the issue.
  - Lead security maintainer is unavailable for code review.

Scenario 2- Team 2: Feature Request from a Major Company

- A large enterprise that relies on your project has requested a new feature. However, implementing this feature requires significant changes to the existing architecture. Some contributors are concerned about the potential risks of these changes, and others feel it will slow down development on other important features.
- **Challenges:**
  - Balancing stakeholder needs with ongoing project development.
  - Ensuring the change doesn't break existing functionality.

Scenario 3- Team 3: Codebase Restructuring Proposal

- Some of your core contributors have proposed restructuring the entire codebase to improve long-term maintainability. However, this restructuring would delay feature development and introduce the risk of breaking backward compatibility for current users.
- **Challenges:**
  - Balancing long-term maintainability with immediate user needs.
  - Preventing backward compatibility issues during the restructuring.

Scenario 4- Team 4: Bug Backlog Crisis

- The number of bugs reported by users has significantly increased, and your project is struggling to keep up. Some users are starting to lose confidence in the project's reliability. Your team needs to figure out a way to handle the bug backlog efficiently while keeping other important tasks moving forward.
- **Challenges:**
  - Managing a growing backlog of bugs.
  - Allocating resources between fixing bugs and feature development.

Scenario 5- Team 5: Conflict in the Community

- Two of your core contributors are having a heated disagreement over the direction of the project. This conflict is slowing down decision-making and causing tension within the community. You need to mediate the situation and find a resolution that aligns with the project’s goals.
- **Challenges:**
  - Mediating conflict within the community.
  - Maintaining project momentum while addressing disagreements.

Scenario 6- Team 6: New Release Deadline with Limited Resources

- You've planned a major release of your project, but several contributors are unable to work due to personal reasons. Additionally, the project has limited resources for testing and QA. The release deadline is approaching quickly, and there is pressure from users and stakeholders to meet it.
- **Challenges:**
  - Limited contributor availability.
  - Insufficient testing resources.
  - Pressure to meet the release deadline.

Scenario 7- Team 7: External Dependency Update Causing Issues

- Your project relies on several external libraries, and a recent update to one of those libraries is causing compatibility issues. Some contributors are advocating for reverting to the previous version, while others believe that the project should adapt to the new changes.
- **Challenges:**
  - Balancing the adaptation to external changes vs. stability.
  - Managing potential disruptions caused by external dependencies.

Scenario 8- Team 8: User Requests for Rapid New Features

- Your project has gained popularity, and users are requesting new features at an overwhelming pace. The contributors are struggling to keep up with the demand, and some features are rushed, introducing bugs and performance issues. You need to find a way to balance rapid feature development with maintaining quality.
- **Challenges:**
  - Managing user expectations for rapid feature development.
  - Maintaining code quality while introducing new features quickly.

Scenario 9- Team 9: Contributor Burnout

- Several key contributors have expressed feelings of burnout due to the increasing workload. They are essential to the project, but they are starting to step back. You need to figure out a way to reduce burnout while ensuring the project continues moving forward.
- **Challenges:**
  - Addressing contributor burnout and retaining key contributors.
  - Ensuring continued project momentum despite reduced contributor availability.

Reading Materials

- Book
  - **Producing Open-Source Software How to Run a Successful Free Software Project** → Chapter 9
  - **Understanding Open Source and Free Software Licensing** → Chapter 1: Open Source Licensing, Contract, and Copyright Law
- https://www.youtube.com/watch?v=02m1T9c7CFk
- https://www.coursera.org/lecture/open-source-software-development-methods/open-source-governance-models-TEhwM
- https://hackernoon.com/what-i-learned-from-building-my-first-successful-open-source-project-72f7429145a0
- https://opensource.guide/how-to-contribute/
- https://fossa.com/blog/apply-license-open-source-software-project/
- https://opensource.guide/legal/

Thanks

Office Time: Monday-Friday (1000 - 1800)

You can send me an email for meeting, or any sort of discussion related to class matters.

jamil@sejong.ac.kr

세종대학교
SEJONG UNIVERSITY