# Intro to FOSS Project Anatomy (Activity)

Organization

This activity will be organized using the Process Oriented Guided Inquiry Learning (POGIL) approach. You will work in groups of three to complete the activities below in the amount of time that is suggested, and then report your findings and observations to the rest of the class.

Each student in your group should take on one of the following roles:
* Manager: Manages the group. Ensures that members are fulfilling their role, the assigned tasks are being accomplished on time, and all members of the group are participating in the activities.
* Recorder: Records the important aspects of the group discussions, observations, insights, etc.
* Presenter: Presents oral reports to the class for the group if required.
If there are only two people in your group, the Manager and Presenter roles can be filled by one person.

1. **Read & discuss! (20 minutes)**

**Background**

Producing Open Source Software by Karl Fogel

FOSS projects have a distinct culture and set of tools that support project development. The form of the culture and the specific tools vary somewhat across projects, but there is significant commonality such that many FOSS developers migrate easily among FOSS projects.

Now that you have an understanding of some of the communication tools used in FOSS environments, this activity will provide a high-level understanding of some of the culture and development tools that you will use when participating in a FOSS project. The goal of this activity is not to provide depth in any one tool or aspect of culture, but to provide a high-level view of what a typical FOSS project looks like.

**Directions**

Read the information below which provides a foundation for understanding a FOSS project. In this section we provide basic definitions for the common aspects of FOSS projects. The Guided Tour will walk you through an HFOSS project and highlight some of the aspects described below. We begin with a discussion of community as FOSS development is as much about community as it is about code.

**Community -** As you have discovered, FOSS projects operate on the basis of a meritocracy where a person defines their position within the community based on the merit of their contributions to a project. These contributions do not necessarily have to be code. As an individual continues to make valuable contributions, that person will be given (or will take on) additional responsibility.

There is a typical progression of participants in a FOSS community. Most FOSS developers start as users of the application. They then identify some small change that they want to make and make the change. If the change is accepted, they progress to making a larger change and eventually become responsible for a portion of the project. A committer is a developer who has risen to the level in the meritocracy where that individual is responsible for committing any changes submitted to the code base. This is a large responsibility as the committer must review all changes and make sure that the change is of sufficient quality to be incorporated into the code base.

**Leadership -** You may be thinking "But how are decisions made?" In many FOSS projects, there is a "Benevolent Dictator", AKA "BD". This is the person who has the final say on a decision. And while this person has the final say, they typically rely on one or more developers who have risen in the meritocracy to the point where they have the ability to commit significant changes. As such, the benevolent dictator doesn't actually dictate much, but instead comes to a collaborative agreement with the major developers who have expertise in the area in which the decision must be made. Most BDs actually do not like making decisions by fiat and are reluctant to put their foot down.

As projects mature, they frequently migrate from a BD model to a democratic model where decisions are made by consensus. A consensus is a decision about an issue that everyone agrees is a reasonable solution. The consensus is usually reached via a discussion about a particular issue and how to solve it. The important thing about consensus organization is that when consensus is reached, it must be posted publicly.

**Forking -** One of the important cultural and technical aspects of a FOSS project is its forkability. Forkability is the ability for anyone to make a copy of the code and start their own project based on that code. That new project is known as a "fork" of the old project. For instance, LibreOffice was forked from OpenOffice. When a project is forked, the source code is copied to a new project and usually some members of the original development team leave the original project and migrate to the new project.

Forking is not an ideal situation as it may result from dispute with in the development team that cannot be resolved.

**Communication** - FOSS projects use common communication tools. You have already been exposed to wikis and IRC. In addition, most FOSS projects have one or more mailing lists. In addition, bug trackers and version control systems (discussed below) are also frequently used for communication.

**Roadmaps** - FOSS projects need some way of defining the future of their project. Most FOSS projects have a Roadmap or Blueprint that does this. The Roadmap may be a detailed schedule of releases with the specific bugs fixed and enhancements added for each release detailed. In smaller projects, the Roadmap may simply be a list of the next enhancements to be added to the project.

**Releases** - A "release" happens when a project has been developed to the point where the developers want to distribute the code and documentation. Releases are typically planned based on the roadmap for a project. Software development is typically "frozen" at a particular point in time so that testing of the software may proceed before the release. This does not mean that development stops. Version control is used to manage different branches of development that allow some developers to continue to work while others test the code to be released.

Releases are numbered and the convention is to use 1.X for major releases. Numbering 1.X.x is used for more minor issues. The next major architectural change would be in 1.2. A complete upgrade of the code (e.g., from python 2 to python 3) would result in an entirely new number, e.g., 2.1.1.

**Repositories** - Most FOSS projects save their code in a repository (AKA repo). A repository is simply a place where software packages are stored and from where they may be retrieved and installed. Some projects use a web-based common repository such as Launchpad, SourceForge, GitHub, or a local repo. The repo is where users go to download a software application.

**Packaging** - FOSS projects are distributed as source code. In fact, access to the source code is one of the major benefits of FOSS. The source code should be distributed in a standard format (e.g., tar compressed by gzip for *nix). Additional reading: Karl Fogel's section on Binary Packages.

However, code that is distributed in this manner must be downloaded, unzipped, untarred, and then installed using a manual process. As a result, most FOSS projects create "binary" packages to support easier installation. In this case, "binary" doesn't necessarily mean compiled. It means that the package is pre-configured to be installed on a computer without having to build the project from the source code directly.

**Upstream/downstream** - Note that development in a FOSS project is not entirely linear. There are typically many people working on the project simultaneously. The main branch of the project is said to live "upstream" of all efforts to make changes. The upstream version is the main version. Developers "downstream" check out a copy of the code to work on. When a change is made, the change is pushed upstream to the main branch so that everyone is working with a consistent code base.

Upstreaming and downstreaming may also refer to the build process where packages are built for release. In addition to upstreaming in an individual project, a project may depend on another project. If project A depends on project B, then the release of the upstream project B must be complete before the release of project A can happen.

**Version Control** - Most FOSS projects have multiple people working on the project simultaneously. Developers may even be working on the same area of a project at the same time. A version control system is a system for controlling the revisions made to software. Version control software tracks and controls changes to a project's files. Most FOSS projects use some form of version control to control change in the software. Commonly used version control systems include Git, SVN, CVS, and Mercurial. The version control system provides both a method of communication as well as a way of coordinating changes in the software. There is a learning activity later on version control. Additional reading: Wikipedia page on version control.

**Trackers** - In most FOSS projects, there is a need to keep track of all of the bugs found as well as the new enhancements suggested. A bug tracker (AKA issue tracker) is used to track change requests for a project. These change requests could be bug fixes, new feature requests, patches from outside developers, and more. A bug tracker allows issues to be logged, described, prioritized, reproduced, diagnosed, fixed and tested. There is a learning activity later on bug trackers. Additional reading: Karl Fogel's section on Bug Trackers.

***To Do: Work through the guided tour and add your findings and results to your documentation. (40 minutes)***

**Guided Tour**
**The Sugar Labs Project (http://sugarlabs.org/)**

Read the information found here to get an overview of the goals of the project and the latest news. You will be asked to answer questions and to make observations. Your responses should be placed in your documentation.

**Contributions --** Read the Getting Involved page which describes the roles of various contributors to SugarLabs. Note that there are a variety of different types of contributions that may be made by people in different roles. In your documentation:

- Summarize the roles that you think would be most applicable for you as student.
- What are the commonalities across roles? What are the differences?

**Tracker --** An overview of the Sugar Labs bug tracker may be found here. A specific query on the Sugar Labs bug tracker can be found here. In your documentation:

- Describe the general process for submitting a bug.
- Indicate the types/categories of tickets listed on this page as well as the information available for each ticket.

**Repository --** https://github.com/sugarlabs/sugar/ Click the "Commits" link and determine the date of last commit (an update of the repository).

- Record the date in your documentation.

**Release cycle --** Information about Sugar's release cycle and roadmap can be found here. In your documentation:

- Describe how the release cycle and roadmap update are related.

**Communication --** Sugar Labs promotes communication among its community members in the following ways.

- IRC: https://wiki.sugarlabs.org/go/Internet_Relay_Chat
- Mailing lists: http://lists.sugarlabs.org/
- Blog: http://planet.sugarlabs.org/
- Wiki: http://wiki.sugarlabs.org/go/Welcome_to_the_Sugar_Labs_wiki

**The Sahana Eden Project (https://sahanafoundation.org/eden/)**

Read the information found here to get an overview of the goals of the project and the types of contributions one can make.

**Community --** In the section titled *Want to Contribute to Sahana Eden?*, you will find a list of ways in which one can contribute. Again, you will note that there are a variety of distinct groups, each with a distinct responsibility. In your documenation:

- Follow the links to each of the groups listed below and summarize the information you find there. For example, are there any commonalities? Is there something distinct for each type of contributor? How is this structure different than the one you found on the Sugar Labs website?

**Tracker --** The Sahana Eden bug tracker can be found here. Place your answers to the following in your documentation.

- How is the information here different than the information found on the Sugar Labs tracker page?
- Click the Active Tickets link. Indicate the types/categories of tickets listed on this page as well as the information available for each ticket.

**Repository --** https://github.com/sahana/eden Click the "Commits" link and determine the date of last commit (an update of the repository).

- Record the date in your documentation.

**Release cycle --** Information about Sahana Eden's release cycle and roadmap can be found here.

- Include a brief entry in your documentation that summarizes the information you find here.

**Communication --** Sahana Eden promotes communication among its community members in the following ways.

- IRC: http://eden.sahanafoundation.org/wiki/Chat
- Mailing lists: http://wiki.sahanafoundation.org/community/mailing_lists
- Google Groups: https://groups.google.com/forum/?fromgroups#!forum/sahana-eden