Extracted from:

Forge Your Future with Open Source

Build Your Skills. Build Your Network.
Build the Future of Technology.

The Pragmatic Bookshelf
Raleigh, North Carolina

The Pragmatic Programmers

Forge Your Future with Open Source

Build Your Skills
Build Your Network
Build the Future of Technology

VM (Vicky) Brasseur
edited by Brian MacDonald

Forge Your Future with Open Source

Build Your Skills. Build Your Network.
Build the Future of Technology.

VM (Vicky) Brasseur

The Pragmatic Bookshelf
Raleigh, North Carolina

The team that produced this book includes:

Publisher: Andy Hunt
VP of Operations: Janet Furlow
Managing Editor: Brian MacDonald
Copy Editor: Paula Robertson
Indexing: Potomac Indexing, LLC
Layout: Gilson Graphics

ISBN-13: 978-1-68050-301-2
Encoded using the finest acid-free high-entropy binary digits.
Book version: P1.0—October 2018

APPENDIX 1

Glossary

While many of the terms below can have multiple meanings, the definitions provided all assume the context of free and open source software projects, communities, and contributions.

accessibility
: The process of opening up access to the software to as many people as possible. This could mean making sure the color scheme is good for people who are color blind, confirming that the software is usable by a screen reader, providing text alternatives to audio content, and other potential actions that can enable more people to use the project.

ad hominem
: A Latin phrase meaning, "to the person," ad hominem statements are those that address qualities of the person at the receiving end rather than the qualities of someone's contributions. For instance, saying, "You're dumb for thinking that" is addressing the person and can rightfully be seen as a personal attack. "That is not a good idea" addresses the concept being discussed rather than the person who proposed it. Avoid ad hominem statements whenever communicating in FOSS projects and communities.

API
: Short for *Application Programming Interface*. An API is the external "face" of a piece of software. It details how to communicate with the software programmatically and allows different software packages to connect and interact.

atomic commits
: A version control commit that addresses a single (usually small) topic, fix, or feature. Atomic commits are considered safer than large, unwieldy commits. The relatively small size and scope allow an atomic commit to

be reviewed more easily and thoroughly and is easier to roll back should something go wrong. Both the review and the easy rollback mitigate the risk of fatal bugs slipping into the project.

BDFL

Short for *Benevolent Dictator For Life*. BDFLs are rare in FOSS but they do exist. For example, Guido van Rossum was the BDFL of Python and Dries Buytaert is the BDFL of Drupal. A BDFL is typically the founder of the project. They have final say in and can veto all decisions related to the project, but it's very rare that they use this power. Typically a BDFL will lean on the *Benevolent* part of the title by seeking consensus and always working toward what's best for both the project and its community.

birds of a feather

Also known as *BoF* or *Open Space*. A BoF is an informal gathering of people interested in a similar topic. Many FOSS conferences and events provide BoF meeting space to give communities a way to gather, meet each other, and discuss matters related to their specific topic.

branch

In a version control system, a branch is simply a pointer to a specific commit in a repository, creating a new path for work on the repository. Working on a branch allows you to isolate development from other parts of the repository, so you can work without risk of affecting unrelated features or code.

breakout session

Often called simply *sessions*, these are scheduled training events at FOSS conferences. Each breakout session features one or more presenters who deliver information to the audience. Sometimes a session will be a panel of people answering questions posed by a moderator.

bug

Something that doesn't look or act right in a system. Typically bugs are found in software, but the term is also applied to things like human interactions, FOSS governance, or other facets of FOSS development.

build

Refers either to the process of creating a distributable version of the FOSS project or the distributable version itself. The build process will be different for most projects. It may require compiling code, running a test suite, or other steps. Causing an error in the build process is called "breaking the build."

bus factor

A number equal to the number of team members who, if run over by a bus, would put the project in jeopardy. The worst possible bus factor for a project (or part of a project) is *one*. If only one person is familiar with that piece of the project, and that person goes away for some reason, the project will find itself in a very uncomfortable position.

CI/CD

Short for *Continuous Integration/Continuous Deployment (or Delivery)*. In a CI/CD process, merging a commit into a repository automatically starts running the entire test suite. If all tests pass, then the repository (including the newly merged change) are automatically deployed to either a test or production system.

clone

A standalone copy of a repository, or alternatively, the process of copying a repository.

commit

The process of submitting a change to a version control system, or alternatively, another name of the change itself.

commit bit

Having a *commit bit* means that someone is allowed to merge changes into the project version control repository. There is no physical thing (*bit*) involved. It's simply a phrase that originates in the access control systems of legacy version control systems, where a commit access was controlled by the value of a single binary digit (a bit). Those bits are gone, but the *commit bit* term remains.

commit message

Text describing what's changed and/or fixed in a commit to a version control system. The commit message should be as detailed as necessary and include not only what was changed but also why. If the work in the commit is associated with an issue, the issue number should also be included in the commit message.

community

A self-organized and self-identified collection of people sharing a concern or interest. Many FOSS projects have communities associated with them.

continuous deployment

The *CD* in *CI/CD*. See the entry for CI/CD for more information.

continuous integration
: The *CI* in *CI/CD*. See the entry for CI/CD for more information.

contribution
: Documentation, testing, design, programming, event coordination, or any other action that helps a FOSS project.

Contributor License Agreement
: Also known as *CLA*. This is a legal document intended to certify that the person sharing a contribution has the right to do so, and that once the contribution is accepted, the project has a license to alter, distribute, and administer those contributions however it sees fit. Once in a while the CLA will also transfer copyright for the contribution from the contributor to the project or the project's organizing body. The intention of a CLA is to minimize potential legal complications of distributing the work.

copyleft
: See the entry for reciprocal license.

copyright
: The right given to the creator of a work to decide how and under what conditions that work may be used by other people and organizations. In most countries the creator automatically receives copyright over a work as soon as it's created, but in countries that have not signed the Berne Convention, the creator may be required to apply for copyright over their work.

core contributor
: Someone with extensive knowledge and experience in the FOSS project. Core contributors usually hold some sort of leadership position in the project, if only informally. They often are responsible for maintaining the quality of the project and guide the project's development roadmap.

design pattern
: An accepted best practice for solving a certain type of software design problem. A design pattern is a very general description of how the problem is best solved. The generality of the description makes it applicable across different programming languages and applications.

Developer Certificate of Origin
: Also known as *DCO*. A confirmation by a developer that they have the right to share their contribution with the project. The developer provides their confirmation by "signing" their contribution using a -s flag on the git commit. The DCO is intended as a paperwork-free and low hassle alternative to the CLA. Because it requires use of git to sign a commit,

the DCO can only be used by projects that use the git version control system and on contributions that are tracked in version control.

diff

Either a specially formatted output showing the differences between two files (or two versions of the same file), the process of creating the output, or a utility that creates the output. Often used in conversations about a contribution, such as “Did you diff this before commiting it?” and “Check the diff and you’ll see it’s just whitespace changes.”

domain knowledge

Specialized knowledge about a certain topic, industry, or area of interest (*domain*). People outside of the domain are unlikely to be familiar with this knowledge. For instance, a knitter has domain knowledge about how to read a knitting pattern, the different types of needles available, and the different types of fibres used in yarns, among other related information. Domain knowledge is important when making judgement calls about the design and implementation of a project for a domain. People who have relevant domain knowledge are known as *subject matter experts*.

DRY

Short for *Don’t Repeat Yourself*. DRY is a best practice in software development. Any time you find yourself in a situation where you might need to reuse a piece of code, design element, or other component, DRY encourages you to pull it out into its own reusable fragment (function, method, file, or whatever makes sense). In this way, if a change is needed, it only needs to be made in a single place rather than in a series of repeated components. This reduces the chance of introducing bugs.

employment agreement

Sometimes called an *employment contract* or simply *contract*. This is the thing you sign or otherwise agree to when you start working for a person or organization. The employment agreement defines details like how much you’ll be paid, whether you get vacation time, and—most importantly in the context of a FOSS contribution—who owns the copyright over what you create while on the job or while using the employer’s equipment.

environment

In a FOSS context, *environment* doesn’t refer to trees, oceans, and the like. It refers to the combination of software and hardware where a FOSS project runs. If you’re developing the FOSS project, you may have a *developer environment* composed of your laptop, a local installation of the software, and your IDE or text editor. Once you’re done developing, you may install the project in a *testing environment* running on a single server in the cloud with limited network ability and a sample database. After testing is complete, you may install the project in a *production environment*, running on a large collection of cloud-based servers that are accessed by many people and both reads from and writes to a large database.

feature branch

Also known as a *topic branch*. Simply a branch in the version control system, created, used, and destroyed as you would any other branch. What makes a feature branch a feature branch is that you've created it specifically so you can work on a single feature. Once the feature is complete and merged into the main branch of the version control system, you can delete the feature branch.

forge

Also known as *code forge*. A web-based service for hosting the source files for FOSS projects. A forge will usually provide features like access control, version control, and an issue tracker. Some forges will also provide online editing for the source files, a wiki, CI/CD services, and other features related to software development. GitLab, GitHub, and BitBucket are three popular forges.

fork

For such a short little word, *fork* carries an awful lot of baggage and responsibility in FOSS. The original and primary meaning of the word in this context is to take a copy of an existing project, rename it, and start a new project and community around the copy. It could also be used as a verb for that entire process. While this kind of fork requires some technical work (version control, renaming, things like that), it is primarily a social action. Despite this existing use of the term in FOSS, in 2008 GitHub decided to use the word *fork* to represent the action of a `git clone` command, instead of using the word `clone`. The word *fork* has now come to mean both copying a project to start a whole new project and community, as well as copying (cloning) a project simply to inspect or work on it.

FOSS

Short for *Free and Open Source Software*. Sometimes you'll see the same concept abbreviated *F/LOSS* for *Free/Libre and Open Source Software*, *OSS* for *Open Source Software*, or *OS* for *Open Source*. This book uses the abbreviation *FOSS*.

“Free as in…”

Spend any time in FOSS and you’ll very soon see statements that start with these three words. The three most common variations are “Free as in Speech,” “Free as in Beer,” and “Free as in Puppy.” The *speech* and *beer* variations are from a quote by Richard M. Stallman and are related and play on the multiple meanings of *free* in the English language. “Free as in Speech” uses *free* in its *libre* sense: few restrictions placed on the thing. “Free as in Beer” uses *free* in its *gratis* sense: no monetary cost. “Free as in Puppy” also plays on the *gratis* meaning of *free*, but with the added complication that comes from bringing a living, breathing thing into your life. The meaning here is that even if you pay nothing for the software (or puppy), you are on the hook for its maintenance and welfare. So while there is no up-front cost, there is an ongoing one.

free software

Software that provides the Four Freedoms. For supporters of free software, just as all people should be free from slavery, oppression, and abuse, all software should be free from any restrictions of inspection, use, reuse, and distribution. For them, software freedom is a moral matter. Nearly all free software is also open source software.

governance

The way that a FOSS project and its community are run and operated. This can include the roles in the project (core contributor, contributor, user), how decisions are made and communicated, whether there are elections for certain roles and if so how to perform those actions, among other social and political structures necessary for keeping the project and its community running smoothly.

hallway track

All of the learning that occurs outside of scheduled sessions (that is, in the hallway) of a FOSS community conference or meetup. Many people find the hallway track to be the most valuable part of any conference.

IDE

Short for *Integrated Development Environment*. A usually complicated piece of software used to develop other software. Most IDEs include a text editor, a diff tool, and a debugger along with related software development tools. Visual Studio and Xcode are two popular IDEs.

infosec

Short for *information security*. The practice of maintaining the privacy and security of data and systems, including preventing unauthorized access to them, securely deleting data when it's no longer needed, and being careful with what data and access is necessary in the first place.

inline reply
: When replying to an email, embedding your replies in the body of the original message, immediately below the piece to which you're responding, and optionally deleting the parts of the original message you're not addressing in your response. Some FOSS communities prefer that people use inline replies to mailing list messages.

integration test
: Tests to determine whether the individual parts of a system will still work as expected when you integrate them together. For instance, if you were running a basic integration test on a car, the wheels may each roll well individually, but when you attach them to the chassis and test again, you learn that the wheel wells are too small and the wheels no longer turn.

intellectual property
: Anything that's the result of you using your mind (*intellect*). Writing, drawings, technical inventions, music, and other creative works are a few of the things that qualify as intellectual property. There's a large body of law dealing with intellectual property. It covers things like patents, trademarks, and copyright. Because the work that goes into creating FOSS is copyrightable, intellectual property is a pretty big deal in free and open source software.

interface
: An interface is how a person or system interacts with a piece of software. This could be a user interface (UI), a graphical user interface (GUI), a command line interface (CLI), or an application programming interface (API). Interfaces are important to get right, as having a difficult interface of any sort means it's less likely someone will want to use the software.

IRC
: Acronym for *Internet Relay Chat*, a real-time chat system invented in 1988 and commonly used in FOSS projects and communities. While IRC is a popular avenue for communication in FOSS, it's only one of many different real-time chat options.

issue
: Also known as *ticket*. A general term for all of the bug reports, feature requests, and support questions related to a FOSS project.

issue tracking

The process of maintaining and monitoring a collection of issues. Issue tracking typically includes ways to comment on, tag, flag, or close/resolve issues. Nearly every FOSS project will use some sort of issue tracking and many forges include issue tracking functionality.

license

Also spelled *licence*. A legal document declaring the conditions under which a piece of intellectual property may be used. Licenses are the backbone of FOSS. Without an OSI-approved license, a FOSS project is not considered "open source". Without a license at all, it's illegal for anyone to use a software project, as doing so violates intellectual property laws.

linting

Static analysis of code (without needing to run it) to find common or potential errors. There are automatic linting tools for most programming languages. Linting often occurs as a part of the testing process. Sometimes a FOSS project will have a linter set up as a part of the CI/CD system, so each commit to the repository will be linted before it's merged.

listserv

Another name for a mailing list. The original mailing list software—which is still under active development today, but no longer used as frequently by FOSS projects—is named *LISTSERV* (all caps, yes). The name of this software became a general term often used for all mailing lists.

lurking

Joining a real-time chat room, mailing list, or other communication avenue used by a FOSS project, but only listening to the conversations rather than participating in them. Lurking is a good way to get a sense of the community around a FOSS project: how they interact, who the key players are, and whether they're welcoming to new contributors.

mailing list

An email group on a specific subject. Most FOSS projects use a mailing list (also known as a *listserv*) of some variety as one of its communication routes, and some FOSS projects prefer the listserv to other forms of communication.

meetup

A typically informal gathering of people. FOSS communities can be globally distributed, making it difficult to gather and collaborate. Frequently, instead of a large global gathering, community members who live near each other get together at meetups to discuss the FOSS project and get to know each other. Sometimes the meetups can be quite large, but usually they're no larger than a couple dozen people (and often much smaller than that).

merge request

See the entry for *pull request*.

open source software

Software that is released under a license that has been reviewed by the Open Source Intiative and certified as providing all the freedoms of open source as detailed in the Open Source Definition. Software released under a license that is not approved by OSI by definition cannot be "open source" software, since it is not guaranteed to provide the freedoms defined in the Open Source Definition.

Pac-Man Rule

When standing or sitting around in a circle having a conversation, leave a gap in the circle. This gap—which if viewed from above would make the circle resemble the Pac-Man video game character—provides an opportunity for others to join in the conversation. The Pac-Man Rule is an effective way to strengthen a community by creating a space that's more inclusive of people who may otherwise lack the confidence to step into a closed circle.

pastebin

A web-based service for pasting then sharing large blocks of plain text. Pastebins help to keep emails, issues, and real-time chats more readable. Rather than, for instance, pasting a long log file into an email, you can paste that log into a pastebin then share a link to it in the email. It's best practice in FOSS communities to use a pastebin to share plain text that's more than a few lines long.

patch

See the entry for *pull request*. This may also refer to the *patch* utility, which is used for creating a *patchfile* that can be submitted as a contribution to a FOSS project.

permissive license

A type of open source license that states that someone who makes a change and redistributes the software is *permitted* to change the terms and conditions under which someone can use the new distribution (also known as a *derivative work*). In other words, derivative work can be released under a different license from the original work, even if that license is proprietary. The Apache License and MIT License are two popular permissive open source licenses.

ping

Mentioning someone in a real-time chat system. So called because some real-time chat clients notify the person of the mention with a sound or a visible change to the client. Sometimes used as a verb for generally reaching out to someone: “I’m going to ping Ioana about the high load on the production server.”

platform

The environment that runs a piece of software. Platform can include operating system, browser, chipset, and other relevant components of the system. Which components are relevant depend on the software being run. For instance, for a web-based Javascript application, the user’s browser, browser version, and operating system may be the only relevant components, whereas for compiled software, chip architecture and operating system may be the most relevant components of the platform.

premature optimization

Spending a lot of time and effort to “improve” something before you know whether or what type of improvements are needed. Premature optimizations can devour a lot of valuable time and are considered a worst practice in software development.

Principle of Least Astonishment

A convention in software and system design which says that if a design has the chance to surprise people with an unexpected interface or result, then that design should be thrown out in favor of one that will not surprise anyone. While it’s usually applied to software development and user interface design, the Principle of Least Astonishment works in all situations where people might be caught off guard. For instance, if you would like to institute a new rule or policy in a community, discuss it and its reasons first, rather than simply popping it on people.

project

A collection of software and the people, policies, and procedures that come together to build and maintain that software. If the software is released under an OSI-approved license, then this collection is called an *open source project*.

proprietary license

A software license that is not approved by the Open Source Initiative. Most proprietary licenses are created by companies and organizations for the software that they sell and release as a part of their product or service offerings, and therefore are an important part of the business of software.

pull request

Also known as a *merge request* or a *patch*. A type of contribution to a FOSS project. By submitting the contribution, you are *requesting* that the project *pull* (merge) it into the main repository. Pull requests can be code, documentation, designs, or anything else that is stored in the project's version control system.

real-time chat

An online service that allows people to communicate in real time using text- and image-based messaging. Most FOSS communities use some sort of real-time chat system as one of their communication routes. IRC, Mattermost, Telegram, Discourse, and Rocket.chat are some popular real-time chat options used by FOSS communities.

reciprocal license

Also known as *copyleft license*. The conditions of a reciprocal license ensure that a work released under one can never be released under a license that may in any way remove or diminish any of the original rights and freedoms granted to the user by the license. A redistributed or derivative work released under a reciprocal license must also not add new restrictions to what the user may do with the work. This ensures that the work, once freed, will forever be free. Reciprocal licenses also have a requirement that if a work licensed under one of them is included in a derivative work that is then redistributed, that derivative work must be released under the same terms and conditions as the reciprocally-licensed work. That is the *reciprocal* nature of this type of license: if your creation benefits from a reciprocally licensed work, then anyone who receives your creation must similarly benefit. The GNU General Public License (GPL) is the most common reciprocal license. Some others are the GNU Lesser General Public License (LGPL) and the Mozilla Public License (MPL).

repository

Often abbreviated as *repo*. A version-controlled collection of code, documentation, images, and any other files necessary for the operation of the FOSS project. Repos usually have a single, central source. Each copy (*clone*) of that central source is also called a *repo*, but may be called a *local repo* to distinguish it from the central source.

roadmap

An ordered plan for the development of a FOSS project. A roadmap usually includes a list of features and bug fixes, loosely organized into a release schedule. Having a roadmap allows the FOSS project to plan the resources and time required for development, while also allowing it to establish feature delivery expectation for the users of the project.

RTFM

Short for *Read The F'ing Manual*. The meaning of the *F* in this acronym is left as an exercise for the reader. The acronym is often used in FOSS project communication as a not-so-gentle reminder to people that they should read the documentation prior to asking questions. While it can get the attention desired, using RTFM in conversation is often rude and unhelpful. It's far more effective to send a link to the appropriate documentation.

scope creep

When a feature or bug fix starts with a small set of requirements, but over time accumulates more and more requirements, greatly increasing the scope (and therefore also the risk as well as the time to complete) for the feature or bug fix. Scope creep is an anti-pattern in software development and should be avoided by any means possible.

scrollback

In real-time chat systems, the conversation that occurred while you were away from the chat session. Some systems store this for you automatically, while others (such as IRC) require a special setup to store and view scrollback.

significant whitespace

In programming languages, whitespace (space and tab characters) that has semantic and syntactic meaning in the code. This means that if you get the whitespace wrong, the program will not work (or not work as you expect). Python is the most popular programming language that uses significant whitespace. Some others are Haskell and YAML.

source control

See the entry for *version control*.

squash

Taking a series of version controlled commits and condensing (*squashing*) them into a single commit. Some FOSS projects require a commit squash before you submit your contribution.

subject matter expert

Also known as SME. See the entry for *domain knowledge* for more information.

test suite

A collection of unit, integration, and other tests run against software to ensure that it does what is expected (and does it in the right way). Running a test suite is typically an important step in CI/CD.

ticket

See the entry for *issue*.

top posting

When replying to an email, placing your replies at the top of the message and leaving the original message untouched below it. Some FOSS communities prefer that people use top posting when replying to mailing list messages.

topic branch

See the entry for *feature branch*.

triage

Reviewing an issue to confirm you understand the problem, can duplicate it, and it isn't already fixed elsewhere. Doing issue triage takes time up front, but it saves time during the implementation of the fix for the issue. While some FOSS projects prefer that more experienced contributors triage issues, others are thrilled to have less experienced people lend a hand as the first responders to any new issues that arrive.

unconference

A "conference" where the session schedule is emergent and defined by the attendees, who also provide the material for each session. The entire conference therefore is driven by its participants. The freeform schedule of an unconference allows a FOSS community to discuss topics that are most relevant to them at that very moment, unlike a regular conference schedule that may be determined months in advance.

unit test

A test of one discrete piece of a software system. The *unit* being tested should be the smallest reasonable piece of the overall piece of software, for instance a method or function.

upstream

The primary repository for a FOSS project. All clones of that repository are considered to be *downstream*. It's best practice to push changes to downstream repositories back *upstream*, sharing them as contributions to the FOSS project.

UX

Short for *User eXperience*. Every interaction that a person has with a FOSS project and its community contributes to that *user*'s *experience* with the project. Optimizing for a positive UX is the best way to increase a project's user base and therefore also its community.

version control

Also known as *version control system*, *source control*, and *VCS*. Processes and tools for tracking and maintaining a collection of files as they are modified. Version control can be used for source code, configuration files, documentation, design artifacts, or any other digital file. Version controlled files can be edited by multiple people—sometimes even simultaneously—and then all of the edits can be merged into a canonical version of the file. At the time of writing, `git` is the most popular version control system for free and open source software projects. Examples of other VCSs are Subversion, Mercurial, Perforce, and CVS.

whitespace

Space and tab characters. For some programming languages, the amount of whitespace in the code can make a big difference to how the software operates. See the entry for *significant whitespace* for more information.

word of mouth marketing

Marketing outreach that's encouraged by a person or organization but is implemented independently by individuals sharing their own opinions. Also known as, "telling your friends and colleagues about things that you like." Word-of-mouth marketing is very effective for building a positive brand; many companies participate in and support FOSS projects and communities to gain a good reputation among the community members who may then tell their friends about just how great the company is for providing its support.

WSL

Short for *Windows Subsystem for Linux*. A method for running Linux utilities and programs on the Microsoft Windows operating system. WSL is a critical tool for people who wish to contribute to FOSS projects but who do not have access to a computer that runs Linux or macOS, since most FOSS projects assume that their contributors are not running Windows and do not optimize their contribution processes for it.