NGI ZERO

BEST PRACTICES COLLECTION

RESOURCES FOR FREE SOFTWARE

Version 0.9.1 - 2020-10-26

D2.1 – WP2 – NLNet lead – A P.S.: & CCT production

This document is Deliverable 2.1 – Best practices resources FOSS projects.

This publication has received funding from the European Union's Horizon 2020 research & innovation program under grant agreements No 825310 and 825322.

Changelog

V0.9.1 – 2020-10-26 – Link to available deliveries and https://nlnet.nl/NGI0

V0.9 – 2019-10-26 – First public release

Introduction

The NGI0 Best Practices series compile resources for best practices dedicated to free software professionals and enthusiasts. This volume specifically introduces free technologies in the context of free culture, with a focus on human organization and long-term sustainability of peer production of software; it offers a comprehensive view for the interested reader (policy-maker, decision-maker, technical director, etc.) who holds a position where the choice of *free* digital technologies can make a difference.

At NGI0, we're looking ahead at what the Internet will resemble 10 years from now. We're interested in the techniques and the practices that may not exist yet, or if they exist, remain marginalized.

Free software now and in the future can be understood to promote:

- an ethical approach to sharing code and caring for each other
- a pragmatic approach to engineering great software
- an effective way to facilitate cooperation among strangers

Therefore this document, rather than adding up to the well-established and extremely detailed lists of best practices you can already find online, such as the Core Infrastructure Best Practices, will focus on three aspects:

1. The first part, **Five Things to Think About When Creating Free Technologies** introduces key notions of software freedom and their European context.
2. The second part focuses on **Funding and Sustainability** for peer-produced free software.
3. In the third part, **How NGI0 Can Help Your Free Software Project**, we introduce best practice topics critical to the production of high-quality software that the Next Generation Internet Zero (NGI0) consortium promotes and encourages throughout this series with regard to software distribution, community engagement, and security practices.

Part 1

Five Things to Think About When Creating Free Technologies

Technique
Culture
Community
Infrastructure
Sustainability

1. Technique

Technique is not just about code. And code is not just about writing *more* code. As free software is often produced by remote team members, one of the main tasks of a free software developer is to keep the line open: to communicate with each other, within the project, but also beyond its limits with other entities committed to the same ecosystem.

Free software works by iteration: start small, make the smallest possible thing that could possibly work, and iterate from there. More code does not mean better code, but a higher probability of bugs. A successful project should aim at minimalism, portability, and reproducibility.

As the project evolves, it should automate its quality assurance, build and packaging processes, so that errors are caught early, and so that the latest release is always improving on the software quality.

Public code and documentation are instrumental in the adoption of a software and attraction of new contributors. Often, a successful community will bring more resources and value to a project than the best possible code.

As a software grows its user base, smaller errors or oversight can have larger consequences. Therefore risk management and legal compliance become more important with more users, especially if the software is used in industry and other legally-binding settings: the software license may claim NO WARRANTY, it's important to build trust with your users, and avoid catastrophic consequences, such as the regular massive user data leaks seen at larger operators.

Technique implies a *milieu*, and a responsible developer should embrace it rather than hide behind a screen "just doing code".

2. Culture

The free software movement started with the premise of user freedom, evolved to embrace excellence in software production and methodologies, and has now reached mainstream acceptance. Meanwhile, as the Internet boomed into a daily part of the lives of half the population of the Earth, the developers culture also evolved to a more mature level, embracing accountability, continuity, interoperability, etc., meeting along the way the shared concerns of the EU for consent, privacy, and digital rights: best technical practices such as the Principle of Least Authority (POLA), privacy-by-default, and privacy-by-design match policy orientations to protect citizen rights.

Today's — and tomorrow's — software should embrace EU-protected digital rights[1] to:

- know what personal information exists
- see the actual content of personal information
- rectify false personal information
- audit who accesses and processes personal information and why
- obtain personal information and use it freely
- share or sell personal information to third parties
- remove or delete personal information

1 From Poikola, Antti, Kai Kuikkaniemi, and Harri Honko. 2015. "MyData, a Nordic Model for human-centered personal data management and processing," Ministry of Transport and Communications. CC-BY-SA 4.0 International. Available at (PDF): http://urn.fi/URN:ISBN:978-952-243-455-5.

3. Community

Software without users is a fish without water. Here we offer six keywords to keep in mind when addressing your community: accessibility, diversity, inclusion, respect, solidarity, well-being.

In the age of globalization, it becomes clearer that a mono-culture is not going to work, and will fail every time there's a need to take into account differences. If previous historical periods could make up with ideals and hide the atrocities targeted at an other considered inferior, if we're to succeed facing the challenges ahead of us, we must embrace among ourselves values that can bring us together despite — or thanks to — our differences, as we're on board the same spaceship.

It certainly means more work to think about how a diversity of people will use a software created by, and conceived for "the majority". Yet, it's a humbling experience to step down from the intellectual elite of software engineering and consider oneself among others, as equals. Making this effort today will transform your direct life experience as it pervades to other aspects of your well-being apart from your work.

Community starts with respect, and the ability to put oneself into another's shoes: see the world not *as it could be*, but *as it actually is for others*, considering not an ideal, but a material condition. From there stems solidarity and from there accessibility, leading to inclusion, and attracting diversity. In the end, well-being has been along all the way.

4. Infrastructure

Art, popular culture, academic research, and free software engineering share the premise that cooperation is desirable, and that we *sit on the shoulders of giants that preceded* us. In practice, developing free software is already sharing infrastructure.

The advantages of sharing infrastructure, although they may seem to run counter to the notion of *growth*, are actually its enablers: sharing costs, sharing expertise, avoiding vendor lock-ins, gaining user understanding, and easing response-ability and flexibility.

When creating free software, one should think about how it will complement what exists, rather than how it will beat it. Free software, understood as *public digital infrastructure*, has the potential to bring Europe beyond *surveillance capitalism*, and you are part of this adventure.

5. Sustainability

2019 has seen the *beginning-of-the-end* of the *all gratis* Internet. For about 15 years, everything online was marketed as *free* (as in free beer), and the public perception of this situation led many in the free software movement, including some of NGI0 grantees, to work a lot for pretty much nothing, as they were building a considerable part of the Internet infrastructure without awareness of their pauperized situation. Yet, with this illusion of a *gratis* Internet, even users of digital services and electronic products seem to have remained stuck in the 1970s, when computer companies did not consider software as valuable.

But even today, the idea of volunteerism continues steering the ideal of an independent digital crowd, abstract from the burdens of matter. Already you can see how absurd that is: nobody would walk into a bakery and pick some bread without paying for the life-supporting work of the bakers, unless they're themselves indigent.

Studies demonstrated that 97% of software dependencies were managed by a single person: what if this person burns out, loses interest, or is hit by a bus? Volunteerism is an enemy of perennity, a clutch on which too many users depend, at the expense of developers' health, sanity, and their own ability to continue using the software. Only recently the EU started reaching out to the developers of software it depends on to ensure them a better position.

Producing sustainable software implies paying for the work required to write and maintain the code and the necessary infrastructure to support that work. NGI0 is one transitory way to addressing this problem, but remains a drop in the ocean of what's required to build infrastructure proper. The next section informs about funding opportunities that can bridge the gap until the EU itself, like it does for agriculture or roads, considers software as a valuable public infrastructure.

Part 2

Funding & Sustainability

Getting Funded
Planning & Change of Plans
Sources of Funding

Every project lives in a unique environment of contributors, users and other stakeholders.

A typical mistake that we see when projects apply for financial support is a focus purely on technical matters. We advise projects to think about its environment and "soft" goals, like growth in the number of external contributors, user support, design discussions, documentation and other tasks that should be taken into account in your description of work (if only in the number of hours you plan to have available).

All these aspects are not "overhead" and don't have to happen on the side, these are essential parts of your work! It is useful to read best practices documents such as the Best Practices Badge by the Core Infrastructure Initiative with this in mind. Which of these apply to your project, and in what order? How can you best include them in your planning?

Financial sources usually do care about the bigger picture, and want to see secondary aspects taken into account when you describe milestones and goals. It is a big misunderstanding that developers feel they are asked to have the design nailed down already to accurately describe achievable goals. We've seen too many applications where way too much time has been spent (unpaid) on the preparation of the funding application, and we see way too many canceled proposals because developers lack the time to (in their eyes) accurately describe what the project wants to do next.

In general, this is not the expectation when funding sources ask for a plan. It makes your life easier to merely describe the current state of the project and what needs to happen next. These "next steps" very likely include more detailed planning and refinement of (later) milestones, but nobody expects you to already have an exact list of tasks already in place, with the discussions and prototyping already done. On the contrary, as you well know things will change faster than you think, and you might later feel compelled to stick to your original plan as described (in too much low level detail) at the beginning. This creates unhappiness, and don't we all be happy? (This leads to a second typical mistake, which we will cover later in this document.)

We are not aware of much good material that helps with such planning, but found "`Software Estimation: Demystifying the Black Art: The Black Art Demystified`[2]" by Steve McConnell a good entry point and recommend at least browsing the first hundred pages of it.

2 ISBN13: 9780735605350

Regardless of who you want to help with your project, and whether that requested help is financial or not, taking a step back to think of a high level roadmap can be very useful (and that in itself should be a proposed activity when you apply for funding!).

If your goal is to attract more developers, why not propose relevant activities that help you achieve this goal, instead of strict development tasks? Depending on the funding source, it might make everyone's life easier to propose a number of hackathons and suggest to fund travel for contributors.

With that in mind, think of such activities as time well spent on improving the long-term sustainability. And time well spent should be equally valued (and paid for!) as development hours...

Planning and Change of Plans

"Life is what happens to you while you're busy making other plans."

Another very typical mistake that we see projects do is to try too hard to stick to whatever plan was agreed upon. With many funding entities a successful proposal leads to scary legal language in a contract or grant agreement, and regardless of how often your contact person tells you otherwise, it is only natural to feel bound to what you agreed upon. Especially if it is based on what you came up yourself in the first place, the idiot You of a year ago.

This is wrong! In your planning, take into account the constant change of plans, and communicate changes early. A sign of a failing project is that it did not get in touch about changes, only to fail at the end. We've seen too many reports delayed and delayed again because projects are too afraid to tell us that they didn't make it. Don't be one of those projects. This advice is relevant regardless of what your funding source is. Your contact person is there to assist and your project was picked because someone believes in it. Changes are expected, even if the language of the contract suggests otherwise.

Sources of Funding

Think of "funding sources" as allies, not merely as money dispensaries. Even if a foundation cannot provide financial support for your project directly, they might be happy to connect you to someone who can help.

Nadia Eghbal maintains a nice overview of different ways to find financial support in her “Lemonade Stand" document: https://github.com/nayafia/lemonade-stand.

NLnet has a more detailed list of alternative funding sources for FOSS projects: https://nlnet.nl/foundation/network.html

Renewable Freedom Foundation maintains a broader list of foundations interested in digital human rights topics: https://renewablefreedom.org/grants/funding-sources/

Part 3

How NGI0 Can Help Your Free Software Project

Low-friction Grants
Mentoring & Best Practice Resources
Community Outreach
EU Visibility

NGI0 consortium members offer best quality services to the free software community covering a wide range of complementary professions: through the lifetime of a software project, all aspects are covered to ensure the best chances of success. This part gives an overview of these aspects that are developed in specific documents referenced hereafter.

1. Low-Friction Grants

Following the lead of NLNet, the NGI0 consortium handles all of the burden of managing grants, so that free software projects can focus on their work without distraction. With two *cascading funding* projects, Search & Discovery (EU Grant Agreement No. 825322) and Privacy & Trust Enhancing Technologies (EU Grant Agreement No. 825210), the NGI0 consortium offers low friction grants every two months.

Low friction means:

1. A low barrier of entry application process, that facilitates access to EU funding to smaller teams not usually seen at this level of funding. It only takes up to a day of work to apply!

2. An easy to follow process for reporting: grantees agree in advance on a set of milestones with clear goals and a price attached; upon completion, the money is released to the grantee. All overheads, including reporting to the European Commission, are handled by the consortium.

2. Mentoring and Best Practice Resources

NGI0 grantees benefit from the unique network of the consortium to cover all the needs of a successful software development: software distribution, community engagement, and security awareness.

Software Distribution

Software is only as good as it is available to its users. NGI0 ensures grantees have access to best practices and expertise in packaging their software for easy and consistent distribution, survey and recommendations for licensing agreements, and support to access standardization processes such as with the Internet Engineering Task Force (IETF) and the World Wide Web Consortium (W3C).

- D3.1 Best practices resources Packaging
- D6.1 Best practices resources Copyright & licensing
- D10.1 Best practices resources Standardization

AGPL Highlight

The GNU Affero General Public License (AGPL) is conceived to ensure users can obtain the corresponding source code when using software over a network.

AGPL code is not limited to social networking software. For example, OpenCog was born in 2008 from a donation of six-year-in-development software from Novamente LLC and released under the AGPL: today OpenCog is used to build Artificial General Intelligence (AGI) robotics around the world, creating a level-playing field for global research.

Community Engagement

Free software is a peer production, and as such relies on the strength of the community of its users. Therefore NGI0 supports grantees to ensure their software is accessible to users with impairment and disabilities, translated and localized according to their audience, and properly documented to ease adoption, support, and reproducibility. In addition, NGI0 strongly encourages diversity and inclusion in grantees' user communities to foster European values of respect for human dignity and human rights, freedom, democracy, equality and the rule of law.

- D4.1 Best practices resources Accessibility
- D5.1 Best practices resources L10n/I18n
- D9.1 Best practices resources Diversity & Inclusion
- D12.1 Best practices resources Documentation, educational materials and publications

Security

As software systems become more complex, attention to security becomes a critical matter for success. NGI0 provides resources and expertise to grantees so that they can develop more secure software based upon such principles as privacy-by-default, privacy-by-design, and security-in-depth.

Security is better understood in terms of balance between the value of protected assets and the will and means of an attacker to obtain these assets. Grantees are also informed about responsible ways to handle security breaches and discovered vulnerabilities in their software.

- D7.1 Best practices Open Source Secure Software Development
- D8.1 Best practices Guide on Operational Security (PDF)
- D11.1 Best practices resources Responsible Disclosure

3. Community Outreach

As grantees of NGI0 projects, talented developers have access to a large network of experts and peers. Insider information flows through this network to benefit grantees. Moreover, as funded developers, grantees can spend more time on the development of their projects, working at their own pace, attending developer-oriented conferences where they can showcase their work. NGI0 mentors are available to serve as information sources and orient grantees through the maze of possibilities beyond the production of code, e.g., to sustain their work beyond the grants.

4. EU Visibility

Until now, access to EU funding was mostly reserved to industries that can afford going through large-scale projects involving layers of administrative, business, and international relations that prevented smaller actors from participating.

NGI0 is especially successful in bringing new talents to EU funding, with the prospect of diversifying candidates to future EU funding programs over the coming years. As technology development, especially in the field of free software and open-source hardware, increasingly comes from smaller entities, these grants open the door to further funding opportunities to successful grantees.

Anti-Patterns

If free software is to be understood to promote an ethical approach to sharing code and caring for each other, a pragmatic approach to engineering great software, and an effective way to facilitate cooperation among strangers, the following anti-patterns must be avoided.

Burn-out

Coding may be addictive, it brings highly satisfying intellectual pleasure. But you require sleep, exercise, a social life, and proper food. Do not overestimate your power, and do not underestimate others': delegation is a great way not to burn out. The fine folks at Basecamp understood it early, and they thrive. Read their books! https://basecamp.com/books

"Scratching an itch"

If motivation is key to success, individual motivation should come second to solving actual world problems. Don't waste your talent procrastinating. Instead, realize that Gödel's incompleteness theorem also applies to individuals: and individual problems are better solved collectively. Maybe scratch someone else's itch?

Meritocracy

The "Benevolent Dictator For Life" (BDFL) and *hero coder* myths focus on excellence, not on collaboration. The former without the latter burdens the community. Not everyone is a genius coder, let alone a fantastic community leader, and the objective is not to foster a society led by highly specialized egotistic morons.

Sexism

As soon as computer science became valuable to the market, it was invaded by males who prolonged a culture dismissive of women. It's not cool to pretend being great when you're hurting people. Meritocracy and sexism often go hand in hand – and a way to keep among men is to dismiss women's work, however great it is. If you're male, learn about your privilege, seriously.

NGI0 Mentors

NGI0 prospects and grantees can reach out to the mentors for any question regarding the relation with the consortium (e.g., other mentoring teams) or to share their own situation, doubts, and project orientations.

Feel free to contact any of us anytime: we're here to help!

Center for Cultivation of Technology

Location: Augsburg / Berlin (Germany)

Contact: Moritz Bartl <moritz@techcultivation.org>

GPG Key: `7A3D AD44 08A0 009B 4DE9  C855 858E E1C3 B8A4 568D`

Languages: English, German

Petites Singularités

Location: Brussels (Belgium)

Contact: hellekin <ngi0-mentor@zoethical.com>

GPG Key: `4CE6 4F27 9931 0403 A903  E43F 0B14 96B7 E336 03EB`

Languages: English, French, Spanish