MAINTAINERATI

BERLIN 2019 Event Report

BERLIN 2019 – EVENT REPORT

MAINTAINERATI BERLIN 2019

Event Report

Prepared by the Maintainerati Foundation Board

Revision 1, 26 March 2020.

TABLE OF CONTENTS

EXECUTIVE SUMMARY

Maintaining the world's digital infrastructure can be both immensely rewarding, but also incredibly frustrating. There are often no answers to the long list of problems and obstacles maintainers face: lack of funding, burnout, governance, conflict, technical debt, *ad naseum*.

On 24 May 2019 we held Maintainerati Berlin, an Open Space event, at the Spreespeicher Center in Berlin, Germany. Close to 80 maintainers came together from all over the world to share their experiences maintaining digital infrastructure projects. This report presents the key challenges and lessons raised during the day.

The funding issue is complex and there is no one solution that works for everyone. However, as we discuss in this report, there are some innovative practical experiments taking place, and even more interesting questions being posed. What merits pay? Who should pay and for what? Are there contributions that money can't buy? And when projects find money, how on earth should they use it?

Onboarding new contributors and finding new maintainers is a recurring problem, including creating welcoming environments, and helping people understand that there are real non-code needs. Various projects have developed some great, human-friendly ways to do onboarding, but we have a long way to go.

Other important issues raised in Berlin include acknowledging and treating burnout, dealing with community pressure, and automating maintenance. There was also discussion about fundamental ideas, such as whether the open source philosophy of making code freely available for any purpose makes sense when maintainers are increasingly concerned about how their code might be used, for example, as part of weapons systems.

At the heart of all the issues on maintainers' minds is the question of sustainability. How do you keep projects and their people alive and healthy in the face of widespread burnout, technical debt, people shortages, skill shortages, and demanding users?

All the knowledge we need to achieve sustainability already exists within our community. But this knowledge is dispersed. The conclusion of the report presents some ideas for sharing this knowledge and invites maintainers to join us in this quest.

ABOUT THE EVENT REPORT SERIES

Welcome to the first Event Report from the Maintainerati Foundation. We're excited to have kicked off a new series of events and started a brand-new Knowledge Production initiative. Our goal is to capture the lessons, ideas, and inspiration of maintainers and share them broadly throughout the community.

We hope that maintainers can use this knowledge to solve their pressing problems, develop better practices, and support each other to create healthy and sustainable developer communities.

In this Event Report series, we aim to provide an account of each and every Maintainerati event held around the globe. As much as possible we aim to give you back the knowledge you shared on the day. You can expect to see a summary of the event, a brief analysis of the topics shared, presentation of key lessons, and ideas for the future. We also include a brief description of what happened in each session in the companion document, *Maintainerati Berlin—Session Notes*, available on the Maintainerati website.

These Event Reports are just the beginning. We aim to build on them through blog posts on specific topics and further reports that amalgamate the information collected at different Maintainerati events across the globe. We plan to use this knowledge to create workshops, seminars, and other initiatives that are helpful to maintainers.

Reporting isn't just a one-way stream: we need your help to keep the knowledge alive and spread it throughout the community. After all, if anyone knows how to address maintainers' problems, it's the maintainers themselves.

Gawain Lynch, Erin Taylor and Don Goodman-Wilson

The Hague and Amsterdam, March 2020

INTRODUCTION

Maintaining the world's digital infrastructure can be both immensely rewarding, but also incredibly frustrating. There are often no answers to the long list of problems and obstacles maintainers face: lack of funding, burnout, governance, conflict, technical debt, *ad naseum*.

After two and a half years of Maintainerati events, we know one thing for sure: all of the knowledge maintainers need to run healthy, sustainable projects already exist inside the community. You need only bring a group of maintainers together, and suddenly solutions, best practices, and workarounds start to bubble to the surface.

The problem is not a lack of knowledge, but rather that it is siloed. One maintainer might be an expert on governance but know little about managing conflict; another might be brilliant at funding models but uncertain how to onboard non-coding contributors.

Maintainers have done excellent work in getting their knowledge out into the public domain. There are a countless number of immensely valuable blog posts, conference talks, and open source guides on the Internet. But busy maintainers do not have the time to trawl through all this wisdom. And there is even less space to discuss how to apply it to real-life situations.

This is why the Maintainerati Foundation was created. We aim to provide spaces—real and virtual—where maintainers' wisdom can be collected, discussed, and finally disseminated back into the community. We believe that enabling maintainers to meet each other face-to-face is the best way to turbo-charge this process. And so we have launched a new series of events and a knowledge production process.

In this report we pull out the key themes and solutions conceptualized by our attendees. We also make suggestions for how to put this knowledge to work. Part of the answer is to collate and distribute "best practices" and other lessons efficiently.

But there is also a need for a culture change within the community. Many projects and individuals are already on their way to creating this culture change. We hope this report helps spur conversation around the future direction of digital infrastructure communities.

Maintainerati Berlin: The Event

Maintainerati Berlin was the first event in this new series. It took place on 24 May 2019 at the Spreespeicher Center in Berlin, Germany. Close to 80 maintainers came together from all over the world to share our experiences of running and maintaining digital infrastructure projects.

In attendance were representatives from a diverse range of digital infrastructure projects, including Babel (a key dependency of the World Wide Web), GitHub (which hosts a much of the code for digital infrastructure), Node.js, Open Collective, Mozilla, Google, Kubernetes and Formidable, among others.

Maintainerati events are Open Space unconferences, in which the attendees choose the topics themselves. During the course of the day we ran nine parallel tracks with 29 distinct discussion sessions that form the basis for this report. Each discussion lasted an hour and was initiated by the person who proposed the topic. We took notes for each session, and an edited version of these is available in a separate document, *Maintainerati Berlin—Session Notes*, available at https://maintainerati.org/berlin-2019/.

A Note on Terminology

Maintainerati supports the maintainers of the global digital infrastructure, no matter what kind of project they run or how they run it. Following community practice, we use the phrase “open source” (in lowercase) as a catch-all term to include software made available as part of three overlapping ideologies: the Free Software movement, the Open Source movement, and a highly variable and informal movement aimed at writing software whose source is available in some form that is not compatible with either Free or Open Source ideologies.

This is not a usage that is approved by any of these particular communities, but nevertheless is common parlance among maintainers when speaking informally. At the moment there is no good term or phrase that both adequately captures all of these communities of practice and that is in common usage, so we will adopt “open source” for these purposes in this report. We also use “digital infrastructure” when it is more relevant.

KEY CHALLENGES AND LESSONS

The goal of the discussions at Maintainerati is twofold: to uncover the challenges that Maintainers face, and to discuss solutions, or at least lessons that can be shared. Sometimes this requires connecting dots and reading between the lines. Some of the topics here were never explicitly chosen as topics for discussion, yet came up repeatedly in different sessions throughout the day.

The following summaries are intended to capture the general thinking and mood of the discussions. In reality, the specific viewpoints of individuals may differ from these descriptions—sometimes substantially. We explore the question of the diversity of mindsets, cultures and values later in this report.

Building sustainable business / funding models

Most digital infrastructure projects start as side projects, hobbies, or passion projects, and are generally worked on in people's spare time. However, once a project reaches a certain point, money becomes necessary to pay for server time and other services, and to compensate contributors. There is still not a clear path to success for projects that reach this threshold, and there is little help available for project leads in the form of training, advice or support services.

Causes

- Projects often begin as a hobby or "itch to scratch" rather than with a concrete plan for future growth or a feature roadmap.
- Many maintainers need to hold "day jobs" in order to live, therefore they can only work on their projects in their spare time (e.g. evenings, weekends).
- Maintainers lack knowledge about running a project professionally, including marketing and sales.
- Open source projects are not generally designed to attract direct revenue that could sustain a business, therefore other funding models are needed.
- Maintainers lack knowledge about possible business or funding models.

Recommendations

- Stimulate industry-maintainer collaboration surrounding solving funding problems and challenges. Maintainers continue to want to see more leadership out of the industries that rely on their projects on how we solve these problems together.
- Focus on funding projects instead of people. Help projects create the necessary tax entities and payment structures to redistribute funds to the individuals who work on it.

- Public and transparent funding management was generally agreed on as being the best practice, with the caveat that privacy should be prioritized when required/in the case of safety.

- Projects need to have clear, concise and enforceable rules around how funding can and should be used.

- Managing taxes and other fees is important, but often under-appreciated. If they are not well-managed, this adds to the burden and risk of maintenance.

- Having the ability to hire a project manager is something people would like, but have not yet seen a well-defined path to success for such a move.

Managing projects and teams productively

Most digital infrastructure projects are started by engineers, and early growth is sustained by other engineers either using or contributing to the project. However, all software projects require additional resources at a certain stage, including project, product and community management. Often a recognition of this need is slow in coming, and many engineers are neither trained in these subjects, nor do they know how to reach out to communities of practitioners to ask for help.

Causes

- Maintainers are often not trained in management.
- Some people did not want to become maintainers, but did so because there was no-one else to take on the position.
- While there are many free guides on maintaining projects, maintainers do not have time to read them; moreover, reading "how-to" guides is no substitute for experience.
- There is a lack of mentorship or training for new maintainers.
- Maintainers burn out before they learn what they need to know to run projects successfully.

Recommendations

- Set up a governance structure at the beginning of a project that provides a framework for project growth.
- Free training courses for maintainers in management skills.

Insufficient prioritization of work

Most digital infrastructure projects begin by addressing a small, concrete need. Because these projects are usually initiated by engineers, little thought is often put into what work needs to come next once the initial itch is scratched.

Causes

- People generally prefer to work on building new features rather than on triaging, documentation, reducing technical debt, governance, etc.

Recommendations

- Creating and maintaining are intimately related. Creating solutions should take into account both feature design, and who and how maintains that feature.
- Create and offer free training courses in product management skills for maintainers.

Frequent and widespread burnout

Maintaining even small digital infrastructure projects places heavy demands on maintainers, in terms of requests from users, building new features, and generally maintaining the code. These demands frequently lead to burnout. Burnout is itself a cause for concern, but when maintainers burn out, there is often a strong desire to abandon their projects, which in turn has in the past led to infrastructure security issues (cf. Dominic Tarr [who was at the event] and event_stream). Thus burnout is something that affects everyone.

Causes

- Volume of work, e.g. answering user requests, reviewing contributions, dealing with technical debt, building new features.
- Delegation of tasks can be difficult due to a lack of common knowledge among contributors.
- It is impossible to please everyone; there are always competing needs and interests.
- Some participants felt trapped into maintaining their projects, since people depended on their leadership for the project to continue.
- Some participants found themselves doing activities they had not anticipated: they got into their projects through coding, but now spent their time managing the project.
- As managers, they struggled to meet users' and contributors' expectations.

Recommendations

- Normalize the idea that a maintainer's time is more valuable than that of members of the community.
- Normalize the idea that maintainers are under no obligation to their users to fix their problems.

- Maintainers should have clear guidelines that explain what work they are and are not willing to do, and under what conditions.
- Collaborate and delegate!
- Maintainers should not be afraid to push back on folks to prevent from becoming “the janitor” of all things, rather than an engineer writing code.
- Maintain a separate work phone.
- Switch from push to pull notifications or disable notifications.
- Keep email about maintenance separate from personal email.
- Users should recognize that requests for work are better received when accompanied by an offer to help, or outright code contributions.
- Creating templates for issues and contribution guidelines makes it emotionally easier to close issues for those that don’t follow the guidelines/create repro steps.
- Use bots to close issues that aren’t properly filled out or to close issues that are over X days (people <3 StaleBot). People seem to be more generous with bots than humans, and using them can help to reduce conflict.
- To help others make meaningful contributions, communicate the context of the project and why specific decisions were made.
- Don’t hesitate to open issues asking for help, such as in looking for another maintainer or for help reproducing issues.
- Start new projects with at least one other co-maintainer to help share the load.
- A maintainer’s primary long term job is finding their replacement. This removes pressure by giving back a sense of control.
- Look for mentors from larger successful projects for guidance.
- Likewise, maintainers from larger, more successful projects should consider offering their expertise as a mentor, insofar as they are able.

- Senior developers tend to talk more about interpersonal issues than junior developers. Create a mentoring system for seniors to juniors to help teach them how to bring up constructive discussions on interpersonal issues rather than attribute them to technical causes.

Frequent communication issues

Misunderstandings occur in open source work all the time, even when people mean well. These can be about work that needs to be done, or about personal issues. When misunderstandings occur they lead to frustration and make it harder to get work done. If they are not cleared up, they can diminish people's enjoyment of open source work and even their mental health.

Attendees noted that they struggle with communication issues daily and try hard to find better ways to communicate—and better channels through which to connect with each other. The big question is whether online communication is inherently problematic, whether tools can be implemented to overcome these limitations, and whether maintainers should make more effort to meet each other face-to-face.

Causes

- Online and mobile channels are not always adequate for complex communication. It is too easy for misunderstandings to occur, leading to conflict or wasted time when people understand a tasks's purpose differently.
- Maintainers and contributors rarely meet each other in person, which limits their ability to form social relations with each other and thus establish better communication.
- Lack of shared vision, knowledge, and understanding among core maintainers in a project.
- Language and culture differences, especially around expectations of work.

Recommendations

- Real, face-to-face human interactions are necessary to recharge and grant positive energy for maintainers who gain a false sense of human interaction online, such as via pull requests or interacting with avatars or fellow maintainers.

- Translate important information, such as process/onboarding guidelines, to multiple languages used broadly in the community.
- Be aware that different people have different preferences for communication, e.g. channels and style.
- A communication protocol must be in place or discussed to communicate when a feature breaks. Otherwise, teams split up.

Too much pressure to fix bugs and add features

A frequent complaint is that consumers of open source code behave as though they are entitled to maintainers' time. Such entitled consumers demand bug fixes or enhancements without considering the cost to the maintainer. Alternatively, they use the code—freely given—at a large scale without thinking of returning something of value in exchange, be they contributions or money. This concern feeds into several of the other topics raised at the event, notably the issue of funding.

Causes

- A sense of entitlement among users.
- Users do not understand how much work goes into producing software.
- Users do not understand the pressure teams are under.
- Users either do not want to contribute to solving the problem, or are not able to due to lack of skills, time or incentive.

Recommendations

- 'Maintainers owe you nothing': maintainers need to feel empowered to strongly protect their own boundaries.
- Teams need to support and protect each other from user pressure.
- Have a dedicated person or team to triage bugs and plan feature addition.
- Have a dedicated person or team to communicate with users to set their expectations.
- Encourage users to contribute to the project through good communication, creating a welcoming environment, and ensuring they get recognition for their work.

- Create a road map for your vision, and clearly communicate what you’re going to be working on.
- Select projects to take on that function in ways that meet your expectations/values, for example being part of an existing ecosystem you are active in or that is community-oriented.
- Create matchmaking tools for open source, such as to find contributors in documentation and UX beyond GitHub. Look in places such as ServerFault or at Meetups (“online dating vs. spending time with like-minded folks”).
- Increase the visibility of the maintainers community.

Abuse of maintainers and contributors

Aside from the demands for time from users of open source projects, the generally anonymous relationship that users and maintainers have with each other is a breeding ground for rudeness and worse. Maintainers are often subjected to withering abuse from angry and entitled users.

Causes

- Disagreements over processes (e.g. license agreements).
- Lack of behavioural norms.
- Cultural differences.

Recommendations

- Create a bot to welcome contributors with text designed to set the tone for future interactions.
- Projects should provide maintainer's guidelines as well as contributor's guidelines. Additionally, there is a need for boilerplate guidelines so that projects can be easily adapted and used.
- In public discussions, maintainers should be introspective with respect to how they are contributing.
- Try not to assume that people know what they are doing: everyone has to learn sometime.
- On the flip side, it's also a good idea not to assume that people don't know what they are doing: they may not be doing something wrong, but doing it in a different way than expected.
- Maintainers need to take a lead in educating others. Lead by example: set norms and protocols for behaviour, follow them, and encourage others to do so.

- Keep an eye on team size and structure. Sometimes the larger the team, the greater the possibility there are issues with abuse.

Working environments are not welcoming and inclusive

Attracting contributors to a project is already a difficult task, as there is a large amount of competition, and there are very few people with both the needed skills and the free time to make meaningful, regular contributions. Even so, many projects fail to attract or retain contributors because the maintainers fail to recognize the importance of creating and cultivating an inclusive and welcoming environment, especially when the potential contributors are from under-represented groups or are otherwise from different backgrounds from the maintainers.

Causes

- Lack of diversity in groups.
- Lack of group norms, values, and so on that are geared towards being inclusive.
- Lack of onboarding procedures.
- Lack of procedures for retaining talent.
- Tools set up for coders, not for people doing other roles.

Recommendations

- Be mindful that experience requirements in job descriptions can be problematic because people judge themselves differently.
- Be mindful of language:
  - Use simple language.
  - Don’t nitpick on grammar.
  - Use emojis.
  - Avoid metaphors.
- Keep bikeshedding low, stay pragmatic.

- Create a welcoming environment, especially for new contributors.
- Create safe spaces for underrepresented groups.
- Have a code of conduct that outlines how to escalate uncomfortable or unsafe situations, and have an enforcement plan in place.
- Positive feedback feels better but has a tendency to create “praise junkies”, cults of personalities, and manipulating others. Therefore it needs to be used more carefully, and structured to reinforce future behavior.
- Positive feedback during a tech conference creates a positive environment and rapport with speakers and allows for interaction and creating of future collaborations.
- Carefully worded and considerate negative feedback is essential in the system because it provides an action item that sets up incentives for certain behaviours, unlike positive feedback.

Low-quality contributions increase workload

When users do take the time to contribute code or other assets back to an open source project, their contributions may inadvertently create disproportionately more work for maintainers than was intended. For example, a code contribution with no tests, or without comments that explain the purpose of the code can require the maintainer to write the tests, or take the time to puzzle through what the code is supposed to do.

Causes

- People don't read contribution guidelines.
- Anyone can submit a code contribution.
- Maintainer accountability is poor.

Recommendations

- Bots that examine contributions to see if they meet certain requirements can help in instances where requirements can be checked automatically; for example, checking to see that tests have been written to cover contributed code, or that the code has been formatted correctly.
- Human guidance is often necessary, but does not scale well.
- Make it a requirement for people to read review project guidelines, such as review guidelines. There are bots for automating this kind of check by requiring contributors to certify that they have read the guidelines before a contribution is accepted.

Maintainers struggle to find new contributors

Rapidly growing projects require additional help to keep up with increased workload, but locating and identifying suitable contributors is difficult for maintainers. It can be particularly difficult to find contributors for non-coding tasks (such as project management or design work), or tasks that are seen as not fun (such as triage and documentation). Making your project visible to those looking, attracting developers to your project, and ensuring that there is a good initial fit are all extremely difficult.

Causes

- It is not easy for non-coders to discover projects to work on that are appropriate to their skill levels and interests.
- Projects lack tooling to make them visible in the right way at the right time.
- Projects do not know how to reach potential contributors, especially non-coding ones.
- There are far more coders than people to undertake other important tasks, such as design, events, project management, etc.

Recommendations

- Add a banner to the top of your issue to indicate that you are looking for new contributors or maintainers.
- To help beginners know where to dig in, clearly label issues in your project “good first issue” or “help needed”. Many code hosting providers like GitHub will surface those issues on special landing pages.
- Consider reaching out to coding schools to find new contributors; they often encourage or require their students to participate in open source development, and need help knowing where to start.

- Maintainers should take care with their communication style with new contributors. It is easy to come across as rude when you don't mean to. Use "Yes, and" instead of "No, but".

Onboarding and training new contributors

Open source projects need to be able to onboard new contributors if they are to survive and thrive. Some projects are lucky enough to have many people willing to contribute to their project; others struggle to attract contributors.

In either case, new contributors need to be brought up to speed: they need to learn about the goals of the project, team structure and culture and the project's ways of working, as well as a host of technical details with respect to the code itself and documentation.

The problem is that most maintainers do not have the resources to invest in these critical new contributors. Skills transfer is highly time-intensive and takes resources away from the other things that maintainers need to do, such as managing the project, writing code, dealing with technical debt and producing documentation. Even with respect to coding, what's needed can be advanced and specific.

Many great projects struggle to move forward because they cannot get new contributors up to speed fast enough. This impacts negatively on the advancement of open source and also the careers of contributors.

Causes

- Projects lack proper onboarding procedures.
- Projects lack resources to develop onboarding or training.
- While it is quite easy to get people to submit their first contribution (e.g. by offering incentives), there is a high rate of attrition.
- Even if people join a project, they may contribute irregularly.
- Learning development practices takes time.
- Projects start searching for talent too late.
- Sometimes contributors are willing to increase their contribution to a project, such as becoming a maintainer, but are not willing to be mentored publicly. There is a lack of tools for private mentoring.

Recommendations

- Maintainers should be personally involved in each new contributor's first contribution, to help them feel comfortable and to teach them your community norms.
- Maintainers should be sensitive that contributors often will not speak English as a first language, if at all.
- Code hosting providers should offer private communications channels to facilitate private training and mentoring.
- As a maintainer, find ways to cultivate future maintainers among your contributors.
- Projects should offer onboarding and contribution guidelines, including a code of conduct, a technical guide to contribution processes, and a guide to reviews.
- Maintainers should set clear expectations in all directions for contributors, reviewers, and maintainers.
- Projects need resources and dedicated team members to train contributors.

Maintainers underestimate the importance of "glue work"

Maintaining an open source project requires more than just writing code. Successful projects require documentation, support, outreach, community management, and more: what we here call "glue work". Most importantly of all, successful projects don't only perform these activities on an ad hoc basis, but require that maintainers coordinate and manage all of these activities carefully.

A single-minded focus on code leads maintainers and contributors to undervalue non-code contributions. This includes not just design and graphics, but contributions of governance documents, automation aimed at creating welcoming environments; indeed, any of a very broad class of contributions that simply aren't code on the main project. As a result, this kind of work gets pushed to the edges, when it is done at all—it's performed on the weekends or during other valuable free time, or it's pushed off on others whose contributions are often devalued because it is not code.

Causes

- Glue work is not recognized as necessary.
- Glue work is not popular among coders because it doesn't contribute to prestige.
- Glue work is usually given less recognition than coding.
- Glue work is undervalued and affects promotion metrics and morale, especially for female developers.

Recommendations

- Maintainers should demonstrate the value of glue work to all contributors.
- Maintainers should create and highlight issues requesting non-technical contributions, rather than waiting for someone else to provide the necessary work.

- Having a diverse maintainer team will help cover blind spots when it comes to non-code work needs. Including people with humanities degrees can also help.
- Maintainers should make an extra effort to publicly recognize non-code contributions.

CREATING CULTURE CHANGE

At the heart of all the issues on maintainers' minds is the question of sustainability. How do you keep projects and their people alive and healthy in the face of widespread burnout, technical debt, people shortages, skill shortages, and demanding users?

During Maintainerati Berlin this theme arose again and again in the context of funding, burnout, working with contributors—in fact, in nearly every conversation.

This worry manifests in different ways depending on the subject and the person. Sometimes we hear it expressed as a tiredness, as individual developers find that they cannot keep pace with demands. Sometimes it comes up as anger at a growing sense of entitlement by the broader community of open source consumers.

Complaints abound about the devaluation of open source work. Burnout remains a perennial issue, exacerbated by a culture that expects maintainers to work in their spare time, that is when they're not on the clock.

At a more fundamental level, maintainers worry that the optimism that fueled the initial open source movement in the 1990s and early 2000s cannot survive the reality of software development in the 2020s.

As Clive Thompson describes in his book *Coders: Who They Are, What They Think and How They Are Changing Our World*[1], in the early days few people coded, and those who did were often motivated by the desire to "scratch an itch"—to create something for the love of it.

We are seeing a massive influx of developers into open source. This increase is great for the future of software development and for society as a whole. But it makes projects much harder to manage. In fact, the biggest barriers to creating sustainability in open source are human factors.

Resourcing is of course supremely important, but as Nadia Eghbal notes in her report *Roads and Bridges: The Unseen Labour Behind Our Digital Infrastructure*[2], even if the resource issue were magically fixed, we would still have to face issues with interpersonal relations,

[1] Thompson, Clive. 2019. Coders: Who They Are, What They Think and How They Are Changing Our World. Pan MacMillan.

communication, management, training, succession, mental health, and cultural difference.

And, with the increase in the number of developers, we are starting to see a change in open source culture. There has been a shift away from an ideological focus on building legal tools to make open source possible, to a pragmatic focus on simply finding ways to use open source code, regardless of license. It's much easier to get into coding now due to the widespread availability of tutorials and tools like GitHub (although it's still hard to get into software development as a non-coder). This shift is further highlighted by the uptick in the number of people wondering whether it makes sense to add restrictions to open source licensed code, for example, to prevent military use.

How can we work together to build more healthy and sustainable communities in which both projects are properly managed and people are cared for and valued? How, in other words, do we create, shape, and nurture a culture change away from purely technical concerns, to one also focused on human concerns?

The Maintainerati Foundation was created to seek answers to these questions as a community. We don't claim to have all the solutions, but Maintainerati Berlin was certainly a good first step in collecting ideas.

Both maintainers and contributors need to be aware that they are not alone. All too often people feel that they are the only one struggling with burnout, upskilling, and other issues. This is not the case: these problems are widespread. People need to know they are not alone, that it is OK to set boundaries, to ask for money for their work, and to say "no".

As Mike McQuaid puts it, "Maintainers owe you nothing" (and neither do contributors).[3] This attitude needs to be normalized if people are to survive themselves and be able to support others.

Not only are these problems widespread, but people have been dealing with them for decades. This means that maintainers and contributors have built up an immense amount of knowledge about detailing with each and every one of these problems.

All the knowledge we need to achieve sustainability already exists within our community. But this knowledge is dispersed. One person

[2] Eghbal, Nadia. 2016. Roads and Bridges: The Unseen Labour Behind Our Digital Infrastructure. https://www.fordfoundation.org/work/learning/research-reports/roads-and-bridges-the-unseen-labor-behind-our-digital-infrastructure/

[3] McQuaid, Mike. 2018. "Open source maintainers owe you nothing", 19 March, https://mikemcquaid.com/2018/03/19/open-source-maintainers-owe-you-nothing/

may have developed a fantastic way to deal with work overload. Another may have gained a deep knowledge of burnout and how to handle it. There are plenty of open source guides, but few dealing with tricky interpersonal or management issues. It is essential that we share this knowledge.

How can we undertake this task? There are many ways. We can have a broader conversation about solutions that have worked in the hands of others. We can share best practices through more Maintainerati events, workshops, videos, and so on. Those of us with the bandwidth can step up to do more mentoring. We can campaign for more awareness and support for mental health.

We can take personal responsibility by asking for help when we need it, rather than hoping that everything turns out OK. Talking about the problems we face personally is valuable as it makes others feel they can also speak up. We can also monitor our own behaviour and think before we judge others, who may be struggling or come from a different cultural background.

Cultural change is a big project and we cannot do it alone. We need your help! Whether you are a coder, designer, writer, facilitator, or bring other skills to the table, we welcome your involvement. Please volunteer to help us out, by signing up with this form.

ACKNOWLEDGEMENTS

Maintainerati Board 2019–2020:

President: Gawain Lynch
Secretary: Erin Taylor
Treasurer: Don Goodman-Wilson

Report production:

Don Goodman-Wilson | Erin Taylor | Gawain Lynch
Alexia Maddox | Melanie Uy | Nicole Weber

Note-takers:

Ben Balter | Tierny Cyren | Katie Delfin | Brian Douglas
Don Goodman-Wilson | Andrea Griffiths | Justin Hutchings
Mike Kavouras | Wilhelm Klopp | Mace Ojala | Paul Oliver
Lee Reilly | Erin Taylor | Nicole Weber | Henry Zhu

ABOUT THE MAINTAINERATI FOUNDATION

The Maintainerati Foundation supports the maintainers of the global digital infrastructure to build healthy, productive, inclusive, and sustainable communities. We act in collaboration with maintainers and other relevant stakeholders to:

- Collate knowledge about running healthy, productive, inclusive and sustainable communities that produce digital infrastructure.
- Package this knowledge in ways that maintainers can apply it to build healthy, productive, inclusive, and sustainable communities.
- Distribute this knowledge broadly among the community.
- Evaluate our progress in consultation with the community and re-frame our strategy and practices accordingly.

Stichting Maintainerati Foundation is a social benefit foundation incorporated in The Netherlands, KvK number 76011690.

Contact Us

Website: https://maintainerati.org

Email: info@maintainerati.org

Twitter: @maintainerati

EVENT SPONSORS

Thanks to our event sponsors for making Berlin 2019 possible!

GitHub

open collective