The Journal of Systems & Software 204 (2023) 111774

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

"We do not appreciate being experimented on": Developer and researcher views on the ethics of experiments on open-source projects☆,☆☆

Dror G. Feitelson

Department of Computer Science, The Hebrew University of Jerusalem, 91904 Jerusalem, Israel

ARTICLE INFO

Article history:
Received 12 June 2022
Received in revised form 6 April 2023
Accepted 1 June 2023
Available online 7 June 2023

Dataset link: https://doi.org/10.5281/zenodo.5752053

Keywords:
Experiments
Ethics
Open source

ABSTRACT

A tenet of open source software development is to accept contributions from users–developers (typically after appropriate vetting). But should this also include interventions done as part of research on open source development? Following an incident in which buggy code was submitted to the Linux kernel to see whether it would be caught, we conduct a survey among open source developers and empirical software engineering researchers to see what behaviors they think are acceptable. This covers two main issues: the use of publicly accessible information, and conducting active experimentation. The survey had 224 respondents. The results indicate that open-source developers are largely open to research, provided it is done transparently. In other words, many would agree to experiments on open-source projects if the subjects were notified and provided informed consent, and in special cases also if only the project leaders agree. While researchers generally hold similar opinions, they sometimes fail to appreciate certain nuances that are important to developers. Examples include observing license restrictions on publishing open-source code and safeguarding the code. Conversely, researchers seem to be more concerned than developers about privacy issues. Based on these results, it is recommended that open source repositories and projects address use for research in their access guidelines, and that researchers take care to ask permission also when not formally required to do so. We note too that the open source community wants to be heard, so professional societies and IRBs should consult with them when formulating ethics codes.

That which is hateful to you do not do to another

[Hillel the Elder, Talmud Bavli, Shabbat 31:a]

1. Introduction

The advent of open-source software development, and moreover the creation of repositories where numerous open source projects are hosted, has been a boon to empirical software engineering research. Large volumes of high-quality code and related artifacts have become accessible for analysis (Harrison, 2001; von Krogh and von Hippel, 2006). The next step was to use the record of changes to the code to also study the software development process. A third step was to not only observe the code and the process, but to also interact with them. For example, many research projects aimed at building tools for code improvement report on applying their tools on open source projects, and then submitting the improvement suggestions to the projects' developers. The fraction of suggestions that is accepted is then used as an indication regarding the efficacy of the tool (e.g. Tartler et al., 2012; Liu et al., 2013; Monperrus et al., 2019). While this practice exploits the time of the developers to assess the tool, it is considered acceptable as it conforms with the underlying principles of being "open source", which specifically include openness to contributions from anyone.

But in April 2021 the maintainer of the Linux kernel, Greg Kroah-Hartman, banned the University of Minnesota (UMN) from making contributions to the Linux kernel (for a detailed description of the whole affair, see e.g. Chin, 2021). He also reverted previous contributions from the university, pending a check that they are valid contributions. The reason was a research project from the Lu lab in the university, dubbed the "hypocrite commits" (HC) study. In this study patches that included bugs were intentionally submitted to kernel developers to see whether they would be accepted, under the pretext of demonstrating that open source development is vulnerable to malicious contributors (Wu

☆ Editor: Daniela Damian.

☆☆ Dror Feitelson holds the Berthold Badler chair in Computer Science. This research was supported by the ISRAEL SCIENCE FOUNDATION (grant no. 832/18).

E-mail address: feit@cs.huji.ac.il.

https://doi.org/10.1016/j.jss.2023.111774
0164-1212/

and Lu, 2020b). In the email announcing the ban, Kroah-Hartman wrote "Our community does not appreciate being experimented on".[1]

While many find this study clearly unethical, there are many other situations that are not so clear cut. So when exactly do such studies constitute an unacceptable experiment? This is not an easy question to answer, and there are myriad considerations including how exactly you define "human subjects research" and in fact whether the discussion should be limited to only "research" (e.g. Watts and Brightman, 2017). Rather than theorizing on the definitions and the ethics considerations involved, we decided to elicit opinions straight from the horse's mouth. We therefore conducted a survey among open source developers and asked about their reaction to various experimental scenarios. For comparison, we also sent the survey to researchers involved in empirical software engineering research, and specifically those who use open source projects and experimentation. As far as we know such a survey was never conducted among practitioners, and the last time a survey concerning ethics in computer science was conducted among researchers (or rather, university department heads) was 20 years ago (Hall and Flynn, 2001).

The goal of the survey was to answer two main research questions:

(1) What do developers care about in terms of ethics in research?
(2) To what degree do researchers appreciate what developers care about?

The survey started out by asking whether respondents knew of the Linux-UMN incident. Subsequent questions asked about 16 possible concerns, ranging from looking at code without asking for permission to voicing an opinion about the quality of the work of identified developers. The next section outlined 16 development and research scenarios, and asked respondents to judge the degree to which they were acceptable. We collected responses from 168 developers and 56 researchers. The response rate was reasonable for this type of survey: about 9% for the developers and 30% for the researchers. This attests to an awareness of and interest in ethics issues in such research, but also raises the danger of a selection bias, where the respondents are predominantly those with higher awareness and stronger opinions.

The results indicate that developers are quite open to various types of activity, be it novices who want to build a reputation, students who want to learn about open source, or researchers who want to study open source projects. However, they also tend to expect that freedoms associated with the software and project be respected — both their freedom to choose whether to participate in experiments, and maintaining the freedom of the code in line with its license. Researchers largely see things eye to eye with developers, but sometimes do not fully appreciate the nuances of developers' opinions. It is therefore recommended that repositories and projects explicitly address research use in their access guidelines, and that researchers approach them to ask for permission even if not formally required to do so.

The rest of the paper is structured as follows. The next section provides background on ethics in research, starting with general ethics considerations and continuing with how they are applied to on-line research and to software engineering research. Section 3 then describes the survey and how participants were recruited. Section 4 presents the results of the survey, followed by a discussion and recommendations in Section 5. Finally, Section 6 lists threats to validity, and Section 7 presents the conclusions.

[1] lore.kernel.org/lkml/YH%2FfM%2FTsbmcZzwnX@kroah.com/

2. Ethical perspectives in software engineering research

Experimental research on open source projects exposes some ethics considerations that have not been explored in the literature on research ethics.

2.1. Background on ethics in research

Experiments involving software developers fall under the regulations for ethical research involving human subjects. The regulations covering such research were developed primarily in the context of biomedical research (The National Commission for the Protection of Human Subjects of Biomedical and Behavioral Research, 1979). While each country naturally has its own precise regulations, the most commonly cited are those of the U.S. Department of Health and Human Services known as "45 CFR 46" (Code of Federal Regulations, Title 45, Part 46) (U.S. Dept. Health & Human Services, 2018), which apply to research conducted or supported by U.S. government agencies. These regulations define research on human subjects as "systematic investigation, including research development, testing, and evaluation, designed to develop or contribute to generalizable knowledge", where "an investigator (whether professional or student) conducting research: (i) Obtains information or biospecimens through intervention or interaction with the individual, and uses, studies, or analyzes the information or biospecimens; or (ii) Obtains, uses, studies, analyzes, or generates identifiable private information or identifiable biospecimens".

Using the HC study as an example, Wu and Lu wrote in their paper that "The IRB[2] of University of Minnesota reviewed the procedures of the experiment and determined that this is not human research. We obtained a formal IRB-exempt letter" (Wu and Lu, 2020b). More specifically, in a FAQ related to the paper they wrote "This is not considered human research. [...] We send the emails to the Linux community and seek community feedback" (Wu and Lu, 2020a). However, a community is composed of individuals, and in the end they did indeed interact with individuals and obtained information about their behavior. Moreover, they themselves note the need to protect the identity of the Linux maintainers who handled their patches, as being identified as having approved faulty code may cause embarrassment or other inconvenience (Wu and Lu, 2020b). In addition, they acknowledge the problem of wasting maintainers' time, which was the more important issue for at least some of the Linux maintainers. It seems that the ethics of a scenario of observing people perform their work, and adding to that work as part of the experiment, has not been specifically discussed in the literature.

The basic document on the ethical use of human subjects in research is the Belmont Report from 1979 (The National Commission for the Protection of Human Subjects of Biomedical and Behavioral Research, 1979). The Report first makes a distinction between research and normal practice, and states that if an activity contains any element of research, it should undergo an ethics review. It then identifies three basic ethics principles:

(1) **Respect for Persons**, so experimental subjects should be informed about the experiment and are entitled to decide for themselves whether to participate in it;
(2) **Beneficence**, comprised of the obligations to do no harm and to maximize possible benefits; and
(3) **Justice**, meaning that the benefits of the research as well as its costs should be shared equally by all.

[2] Institutional Review Board, the committee charged with reviewing research proposals to ensure they are ethical.

The HC study clearly fails the first, in that informed consent was not sought. Moreover, a central tenet of open source development is trust, based on reputation and a hacker ethic of sharing (de Laat, 2010, 2014). The report on the incident by the Linux Foundation Technical Advisory Board specifically cites the breach of trust that contributions to the kernel are well-intended as a major offense, and in fact classifies the whole affair as a "breach-of-trust incident" in its title (Linux Foundation Technical Advisory Board, 2021). One can also argue that the HC study fails the second principle, as it wasted maintainers time. This was aggravated by the trust problem, as considerable work was needed to re-assess all previous UMN contributions that had been reverted (Linux Foundation Technical Advisory Board, 2021). The study does seem to comply with the third principle, in that the maintainers whose time was wasted were not singled out for any particular reason.

Professional societies publish ethics codes for their members which often mention the Belmont Report. One example is The American Psychology Association Ethical Principles of Psychologists and Code of Conduct. While the bulk of this code concerns the professional conduct of psychologists, it also includes a section about research, with a detailed description of what should be included in informed consent, and how to deal with cases where deception is necessary to elicit spontaneous behavior (American Psychological Association, 2003).

The IEEE Code of Ethics, in contradistinction, focuses exclusively on professional conduct and does not mention research at all (IEEE, 2020). The ACM Code of Ethics and Professional Conduct also does not address the issue of ethics in research, except in saying that "The public good should always be an explicit consideration when evaluating tasks associated with research" and many other activities (Association for Computing Machinery, 2018). Informed consent is mentioned only in relation to respecting privacy: as a computing professional, if you build a system that collects data, you should ensure "individuals understand what data is being collected and how it is being used". This is also the case for the IFIP Code of Ethics and Professional Conduct, which is based on the ACM code (Intl. Federation for Information Processing, 2021).

2.2. Ethics in online settings

Mining repositories of open source software, and in particular collecting data on software developers, is just a special case of collecting social-science data about the behavior of individuals in some context. In some cases this has clear ethical implications, for example when studying the consumption of porn (Thomas, 1996b), postings in support groups for trauma victims (King, 1996), or an LGBT forum including discussions of coming out (Bassett and O'Riordan, 2002). But even if there are no clear repercussions to divulging data about participants, there is still the question of the expectation for privacy and the boundary between "public" and "private" in an online setting (Thomas, 1996a; Bassett and O'Riordan, 2002). Such expectations exist despite Google's search capabilities, which often allow the identification of even short quotations from text (or code) which is openly accessible (Oezbek, 2008; Gold and Krinke, 2022).

As the above examples imply, online ethics often apply to the *users* of software systems, especially web-based systems. Facebook in particular provides several relevant case studies with increasing severity. The first level is using A/B testing to improve the product (Feitelson et al., 2013). This means that new features are added only after being tested with real users in live action, unbeknownst to those users, to see that they lead to a favorable response. This practice is widely used in practically all web-based companies (Kohavi et al., 2009; Fabijan et al., 2018; Li et al., 2021), and Google even offers it as a free service on its marketing platform.[3] However, it introduces two ethical difficulties. First, there is no informed consent by the users to participate in the experiment. This may be excused on the grounds that users do not know what version of the system they are using anyway, and this changes daily. But this leaves the second issue, which is what the experiment is really trying to optimize. It may be that the benefits are for the company and not for the users, e.g. when the desired effect is to improve the monetization from using the product, even if this comes as the expense of the users in terms of expenses (on ecommerce sites) or detrimental effects like addiction.

The second level is facilitating other research, which is not directly related to improving the service to users (King, 2015). For example, a much-cited research done using Facebook data concerned the transmission of emotions among social network users (Kramer et al., 2014). To do so, "the experiment manipulated the extent to which people (N = 689,003) were exposed to emotional expressions in their News Feed". This manipulation was justified by Facebook being a private company, hence not bound by regulations on research supported by the government, and by the fact that the analysis performed "was consistent with Facebook's Data Use Policy, to which all users agree prior to creating an account on Facebook, constituting informed consent for this research". However, this lenient interpretation of informed consent elicited an expression of concern from the Editor-in-Chief of the journal where the research was published (Verma, 2014).

The third level is facilitating direct manipulation of the public, as in the infamous Cambridge Analytica affair (for a detailed description of this affair, see e.g. Meredith, 2018). On a superficial level the ethical problem was that around 270,000 Facebook users gave informed consent to use their Facebook data for psychological profiling, but using Facebook's Open Graph platform the profiling app actually collected data also from their Facebook friends' accounts, totaling 87 million users. The more insidious problem was that Cambridge Analytica claimed to have used this data to manipulate voters in the 2016 USA elections, which may have contributed to the campaigns of Senator Ted Cruz and President Donald Trump.

2.3. Application to software engineering research

Vinson and Singer attempt to adapt the Belmont principles for the special case of experiments in software engineering (Vinson and Singer, 2008; Singer and Vinson, 2002). They retain the first two principles, of informed consent and beneficence, and add two more: maintaining the **confidentiality** of all information shared by the experimental subjects, and ensuring **scientific value**, in particular by using established methodology.

The Menlo Report on Ethical Principles Guiding Information and Communication Technology Research is also based on the Belmont Report. It adopts the three original principles, and adds **Respect for Law and Public Interest** (Bailey et al., 2012). However, in doing so the Menlo Report actually conflates ethics with legal issues and with safety. This is a result of two factors. First, their focus is on security research, including whitehat hacking and the study of malware. Studying malware can be dangerous, but this is not an ethical issue but a safety issue, just like studying explosives in a chemistry lab or viruses in a biology lab are not ethical issues but safety issues. Second, the law may indeed step in when ethics standards fail. For example, the European GDPR and the California CPA were both responses to lack of respect for privacy and information security on the side of high-tech companies. But while this may affect research, the motivation was unrelated to research. And ethics is more than just abiding by the law. In fact, ethics guidelines are specifically an attempt to codify what is right or wrong *beyond* what the law demands. So the Menlo Report is not so much about the ethics of human subjects

[3] marketingplatform.google.com/about/optimize

Table 1
Potential ethics issues in studies using open source projects.

Study type	Potential issues
Studying the code	Use of the code not for the purpose for which it was opened Harming a project by judging it or publishing its faults
Studying the developer community	Violating developers' privacy Harming developers by exposing inappropriate practices
Interacting with a project's development	Harming the project by adding inappropriate code Wasting the time of the projects' original developers

research, as about all aspects of research with the **potential to harm humans**, mainly by exposing data about humans.

Experimenting with software engineers can lead to various ethical issues, including inconvenience due to frustration or boredom during the experiment, worrying about it, and disapproval or stigmatization by co-workers due to disclosed information (Sieber, 2001b). There is also a growing body of software engineering research that interacts with the subjects directly, e.g. using fMRI or psychological tests (Floyd et al., 2017; Peitek et al., 2020; Graziotin et al., 2022). These are adequately covered by procedures used in other fields which use such devices, e.g. cognitive science.

Additional vexing ethical questions may come up in the context of collaborations between researchers and industry (Lethbridge, 2001). For example, consider a company-based study where developers are ranked based on a quality analysis of their code. Should the researchers be loyal to the company, which invited them to collaborate and may be financing them, and would be harmed by keeping sub-par employees — or to employee-subjects, who trust them with their data, and might be fired? (Vinson and Singer, 2001). As it is hard to anticipate and regulate all such potential conflict situations, they imply a need for open discussions among the collaborators to map out the issues and decide on how to deal with them.

A recurring subject in previous investigations on ethics in software engineering research is the use of student subjects in experiments. The risk here is that the researchers are also the professors, and the situation might be perceived by the students as coercive (Liebel and Chakraborty, 2021; Singer and Vinson, 2002; Sieber, 2001b). Thus at a minimum one needs to uphold anonymity, and allow the option to opt out, thereby negating the fear of influence on grades.

There has also been concern regarding harm to the students' academic progress. Therefore, especially when experiments are carried out as part of compulsory classes, they should have educational goals (Singer and Vinson, 2002; Carver et al., 2003; Berander, 2004; Carver et al., 2010). Examples include the opportunity to learn or exercise some technique or methodology, being exposed to cutting-edge ideas and procedures, and more (Staron, 2007). At the same time, one should consider whether the students could have learned the same things more efficiently in some other way, and whether it is fair to grade them on their performance in an experiment, especially if they were divided into groups that used different treatments?

2.4. Considerations for open source

Most of the research using open source projects is based on repository mining. In addition to the repositories themselves, Internet sites like OpenHub[4] are devoted to the tabulation and display of the activity by different developers in different projects, adding to their exposure. While often thought to be benign, such exposure can in fact have detrimental implications (Table 1) (Gold and Krinke, 2022). For example, OpenHub ranks developers by "kudos", which is assigned by members of the site to each other. Research may identify "influential" or "core" developers, using measures of impact usually based on levels of activity (e.g. Wu et al., 2018; Yan et al., 2020; Song et al., 2022). Such rankings may shame or offend those who receive a low ranking. Even more harm can ensue if developers or companies are ranked based on a quality analysis of their code, or if intellectual property is revealed (Vinson and Singer, 2001).

Open source projects are also vulnerable to situations where researchers interact with the developers, as in the HC study. The goal of the HC study (which led to the ban on UMN contributions to the Linux kernel) was to raise awareness of how vulnerabilities can be introduced to open-source software on purpose by malicious agents (Wu and Lu, 2020b). The idea was to analyze the code and create "hypocrite" patches that added a missing condition for a vulnerability, thus turning "immature vulnerabilities" into real vulnerabilities. As a proof-of-concept, they prepared 3 patches that created "use after free" bugs in Linux.[5] They claimed that this was done safely, and that the buggy code was only exchanged in emails, and indeed the 3 buggy patches were rejected by Linux maintainers. But the main fault developers found with the study was the lack of obtaining consent.

Other unique ethics issues with open source concern whether and how the research interacts with the open source philosophy. At the most basic level the issue is one of openness and freedom. On the face of it there should not be any problem. A well-known adage regarding free software likens it to free speech as opposed to free beer. The implication is that anyone can do whatever they want with the code (Chopra and Dexter, 2009; Wolf et al., 2009), and the problem is not in exposing it but only when it is confined and restricted. Indeed, the hacker ethic of free information justifies using systems in unintended ways to uncover their inner working (Chopra and Dexter, 2009). However, one may question whether it is ethical to use public data for purposes other than intended by its authors (El-Emam, 2001). Opening source code is not done to facilitate research — it is done to improve the code and benefit its users. So using it for other purposes may require the consent of the authors. But who do you ask for consent for using code from a long lived project where developers come and go with time? (Gold and Krinke, 2022).

The flip side of openness is the danger of compromising privacy (Zimmer, 2010). Vinson and Singer comment that developers who identify themselves as authors of open-source code cannot expect privacy, and therefore using their code — and even identifying them — does not fall under the usual limitations of human subjects research (Vinson and Singer, 2008; Singer and Vinson, 2002). But according to Berry, the question is "Is the Internet

[4] www.openhub.net

[5] In total it appears that 5 patches were submitted, but only 3 were buggy (Linux Foundation Technical Advisory Board, 2021).

a space in which embodied human beings interact? Or is it a textual repository where authors deposit work?" (Berry, 2004). The difference brings up considerations like copyright and fair use of artifacts, not necessarily from the legal perspective, but from the social perspective. For example, software published under the GNU public license requires any derivative work to be published under the same open license. Legally speaking, quoting from such software in research probably falls under fair use. But ethically, is it acceptable to republish even just short pieces of code in copyrighted papers, given the specific license provisions? (Berry, 2004).

Moving beyond privacy, removing restrictions on using open source data may lead to risks that developers may not be aware of. One example of the need to keep information confidential is when studying the development process, and some employees do not follow prescribed practices (Becker-Kornstaedt, 2001); exposing this is part of the research goal, but if the employees are identified they might be punished or even fired. As this example shows the risk also depends on the style of the research. There is a difference between actual reading and analysis by human researchers and the publication of specific identified quotes, and massive-scale automated analysis using machine learning, where the end result is just statistical observations. And indeed, developers are sometimes sensitive to their privacy, and even may disassociate themselves from code they had written, as evidenced by the existence of services like Gitmask.[6] This casts a shadow on research practices such as analyzing developers' emails (Bacchelli et al., 2010).

A more philosophical level concerns the issue of contribution. At the core of open source software development lies the notion that anyone is invited to contribute to the project. So the basic expectation is that people will give to the project, especially if they also benefit from it. For example, Grodzinsky et al. write about the ethical responsibilities of open source developers, including the obligation of organizations who use open source software to also contribute to the community, and the obligation to produce high-quality code (Grodzinsky et al., 2003). Yu writes about the reciprocity in firms' open source policies, and how it contributes to their business performance (Yu, 2020).

Finally, it is not clear that a legally or even ethically centered discussion is the right approach. As Bakardjieva and Feenberg write, "alienation, not privacy, is the actual core of the ethical problems of virtual community research" (Bakardjieva and Feenberg, 2000). Our survey is specifically designed to obtain input from the community itself about what really concerns them.

3. Survey design and execution

To the best of our knowledge the closest work to ours is a superficial survey of department heads regarding awareness and procedures for ethics approval of software engineering experiments with human subjects, published by Hall and Flynn twenty years ago (Hall and Flynn, 2001). We conducted a deeper survey of the considerations involved, as seen by the researchers themselves, and, more importantly, by the potential experimental subjects.

3.1. Survey structure

The survey contained four sections:

(1) Questions about the Linux-UMN incident;
(2) Questions on general ethical considerations when performing research on open-source projects;
(3) A list of short scenarios describing contributions to open-source projects or research on such projects, asking for judgment on whether they constitute acceptable behavior;
(4) Demographic questions used to characterize and classify the respondents.

The questions are detailed below together with the results. Each section ended with an option to provide general comments.

In writing the questions about ethics considerations, we used concrete questions rather than asking about abstract principles. For example, instead of asking about the principle of not putting experimental subjects at risk, we asked about voicing an opinion about the quality of developers' work. Likewise, instead of asking about the principle of maintaining privacy we asked about reading and analyzing the text of communications between developers to better understand their social interactions. The respondents were asked to rank how much these actions are an ethical concern on a 7-point scale.

The questions on acceptable behavior also used concrete scenarios that may happen in the work on an open-source project. Some of the scenarios focused on developers, for example a developer who contributes code that was not adequately tested. Other scenarios were about researchers, for example identifying a project whose development they had analyzed. The respondents were again asked to judge whether they are acceptable behaviors on a 7-point scale.

The survey was implemented on the Google forms platform. None of the questions were mandatory, and no identifying information was collected.

Table 2
Summary of recruited participants.

Inclusion criteria	Invited	Bounced	Responded	Resp. rate
Developers				
≥20 commits since 2020 in project with ≥50 commits	2000	17	180	9.08%
Researchers				
Published on experiments or open source; H-index ≥ 10	151	3	44	29.7%

3.2. Recruiting subjects

The recruiting procedure and its outcome are summarized in Table 2. Given the nature of the topic we aimed to collect the opinions of more experienced developers and researchers, rather than novices and students. To ensure we had subjects of both types we employed separate procedures to recruit developers and researchers. However, the distinction is not really binary, as researchers may also make code contributions to open source projects, and developers may participate in research. We therefore also asked the subjects about how they identify themselves, and this was used to adjust the final classification as described below.

For developers we wanted those who were involved with the project and not just making a casual contribution. We first selected active recent GitHub projects, identified by a threshold of having at least 50 commits since 2020. To ensure that they are software projects we used the CCP (corrective commit probability) metric, and used only projects where this was between 0 and 1 (Amit and Feitelson, 2021). This excludes projects which do not have commits that are identified as bug fixes. From these projects we extracted developers who had public emails and at least 20 commits since 2020. There were 16,559 such developers. From them we randomly selected 2000 and invited them to participate in the survey. The invitations were made by personal emails. In most cases these were addressed using the first name, after

[6] A project which facilitates anonymous contributions to open-source projects, www.gitmask.com.

manual checking. When the first name could not be identified the email was addressed to "developer". 17 emails bounced, and 180 responded, leading to a response rate of just over 9%. This is significantly higher than the 2%–4% reported by Wagner et al. (2020).

To identify relevant researchers we performed a Google Scholar search with the query "''software development'' experiment ''open source'' github". The query was restricted to papers published in the last 5 years (since 2016). We then verified that the paper is indeed on topic based on its title (there were a few irrelevant ones, e.g. reporting on open-source software developed to analyze a physics experiment) and published in a leading software engineering venue. From these papers we selected authors who have an H-index of 10 or above. The first 150 papers returned by the query yielded 48 usable papers and 102 authors. Some of the papers were not used because all their above-threshold authors had already been identified from previous papers. The authors were again invited to participate using personal emails, addressed to their first names.

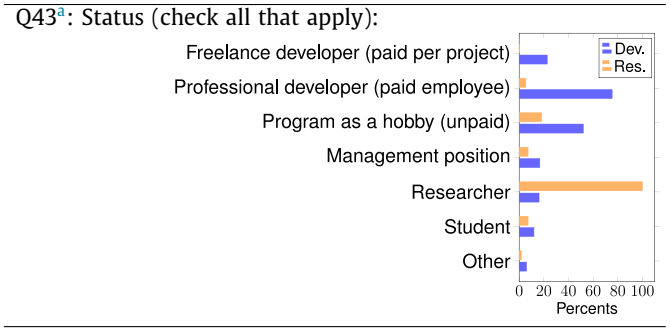
To increase the number of research participants, we conducted a second wave of invitations, based on the "empirical software engineering" label which researchers can attach to their Google scholar profile. The top 100 researchers with this label were scanned, and those with papers with titles indicating work on experiments and open source in either the first page or the last 5 years were identified. There were 57 such researchers, but 8 of them had already been identified in the previous round, so only 49 additional invitations were sent. In total then 151 invitations were sent, of which 3 bounced. 44 authors responded, leading to a response rate of nearly 30%.

Participants in the survey were not paid or given any other reward. The instructions indicated that advancing to the questions constitutes consent to participate. The survey and recruitment procedure were submitted to the ethics committee for non-medical research on human subjects of the faculty of science, and received approval. Due to rate limitations on sending emails, the invitations were sent over several weeks from the end of October to the end of November 2021, about 6 months after the Linux-UMN incident.

We are aware of the problems with sending unsolicited emails to invite potential participants to a survey (Baltes and Diehl, 2016; Cho and LaRose, 1999). The above procedure was meant to reduce the danger of sending such emails to irrelevant people, and the relatively high response rates indicate that indeed many recipients found the survey relevant and important. Some even said so explicitly and expressed interest in the results. This issue is discussed further below, in the context of a survey question that addressed it and in the recommendations.

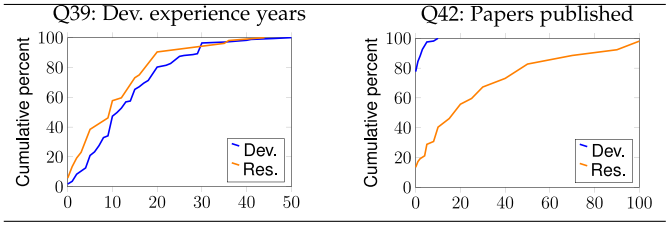
3.3. The survey respondents

A total of 180 active GitHub users responded to the developers survey, and 44 Google Scholar authors to the researchers survey. However, some of the developers identified themselves as also being researchers and reported having authored multiple papers on empirical software engineering. And some of the researchers reported significant activity in open-source development. We eventually reclassified 12 respondents to the developers survey as researchers, as they had self identified as researchers and not as paid developers. Subjects who reported being both researchers and developers were left in their original classification. All the results presented below are after this reclassification.

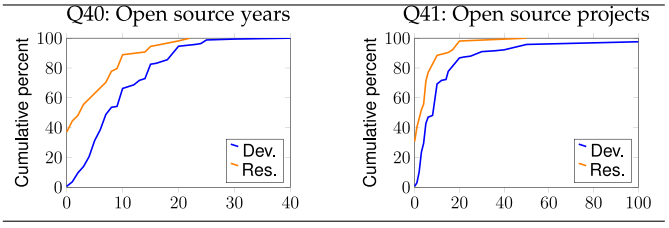


[a]The question numbers indicate the order in which they were presented to participants, which is different from the order used here. In particular, the demographic questions originally appeared at the end.

Interestingly, the distribution of years of development experience was similar for developers and researchers, with developers having only a slight edge. However there was a large difference in the distribution of number of papers published: 78% of developers had published none, and the maximum was 10. Only 14% of researchers had not published papers (or did not reply), and the median was 20. The distributions are shown as CDFs (cumulative distribution functions). This enables an easy comparison of the distribution of responses of developers and researchers. A CDF that is below and to the right of another implies a distribution biased towards higher values.



Regarding experience with open source projects, more than 30% of researchers reported having none. And the distributions of years of experience and number of projects worked on by developers dominated the respective distributions of researchers.



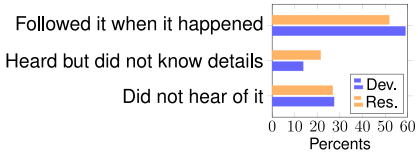
4. Survey results

The results for the survey questions are shown as histograms of the selected options. In most questions the options form a scale. In these cases a CDF is shown too.

4.1. The Linux-UMN incident

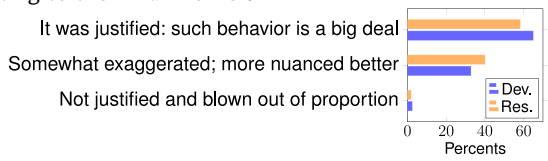
The first part of the survey concerned the Linux-UMN incident. The first question was whether the survey participants had heard about this incident. As shown below, a substantive majority had heard about it, and most — and even slightly more so among developers — had followed it when it happened. In other words, this incident was an important news story for our participants.

Q1: Did you hear of the Linux-UMN incident?



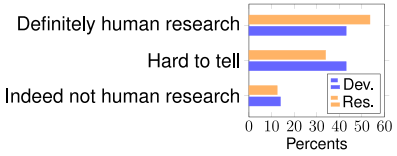
Delving into the details, the majority of respondents thought it was justified to ban UMN from contributing to the Linux kernel. Nearly all the rest thought this response was somewhat exaggerated,[7] and only a handful thought it was wrong. In added comments, quite a few respondents referred to the gap between the open source community and academic researchers, some even using quite strong language concerning researchers in their ivory towers. One wrote: "If you wouldn't perform the same experiment on commercial developers without management consent, or on academic colleagues without university consent, and you didn't get permission from project owners, don't do it". Particular ire was reserved for the perception that researchers were using developers but then were not interested in hearing their opinions and did not solicit feedback.

Q2: What do you think of the decision to ban UMN from contributing to the Linux kernel?



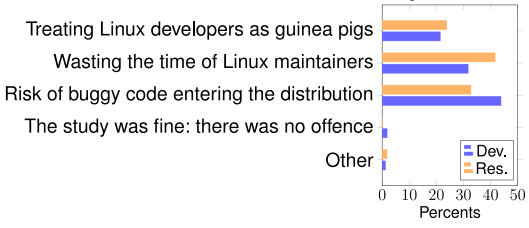
Responses were more evenly distributed concerning the question whether the UMN IRB had erred in determining that this is not human research. Around 13% of developers and researchers thought they got it right. The rest of the developers were evenly split between claiming they were wrong and saying it is hard to tell; among researchers a significant majority said they were wrong.

Q3: The UMN IRB (Institutional Review Board) had given an exemption to this research based on the perception that studying the kernel patch process is not human research. Do you agree with this judgement?



When asked to identify the worst offense in the UMN study, the top spots were wasting the time of maintainers (preferred by researchers) and the risk of distributing buggy code (preferred by developers). Some explicitly cited the lack of informed consent and violating trust; in the graph these are included under "treating the Linux developers as guinea pigs". Others said all 3 offenses were equally bad; they were counted as contributing $\frac{1}{3}$ to each.

Q4: What was the worst offense in the UMN study?



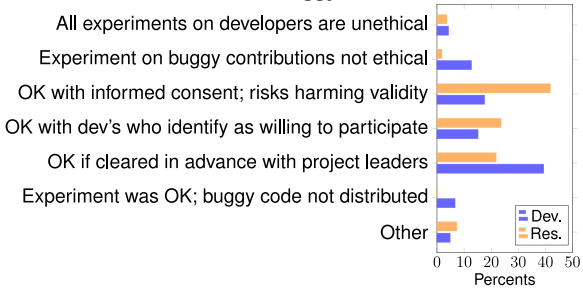
[7] This and some of the options in other questions are shown in abbreviated form to fit in the available space.

Several comments made at the end of the survey can be interpreted as alluding to this question. Some referred to the special status of Linux as a major global resource, so any interference with it is especially troubling. Another interesting comment was: "The biggest sin of the research was that it was done without cooperation with the community". The relationship between academic researchers and open source developers figured in many other comments as well, and is discussed in Section 5.

It is interesting to note that hardly anyone thought the study was OK. This does not contradict the previous question, where around 13% said it was not human research at all, because of the option to find fault with the study for reasons other than ethics — and specifically because of the danger that buggy code would enter the distribution.

Finally, there was a wide range of opinions on whether this study could have been executed in an ethical manner. Specific interesting results were that many researchers and not few developers thought it would be OK only if informed consent was given, even though such consent may harm the validity of the experiment. On the other hand many developers were also content with having the experiment cleared with project leaders without explicit informed consent from the affected maintainers.

Q5: With regard to Kroah-Hartman's comment that "Our community does not appreciate being experimented on", is it at all possible to conduct an ethical experiment on whether open-source maintainers detect buggy code contributions?



In comments one developer cited the practices of whitehat hackers to contact the security-focused maintainers of a project to coordinate the scope of the research up front. Others claimed the results could be obtained by other means. In addition, several developers suggested that if maintainers' time was wasted by an experiment they should be compensated, either directly or by a donation to the project.

4.2. Ethics concerns

The second part of the survey was about possible ethics concerns in isolation. The respondents were asked to rate these concerns on a 7-point scale, ranging from 1 = no concern to 7 = extreme concern. This directly reflects our first research question, of what developers care about. The results also pertain to the second research question by comparing the answers of developers to those of researchers.

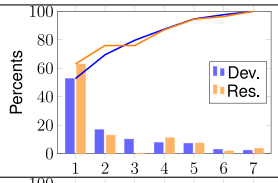
We present the results in separate subsections that each contains a set of questions related to the same general concern. Note that in the survey there was no such structure, and in some cases the questions appeared in a different order. The original order is given by the question numbers.

4.2.1. Inappropriate use of open source code

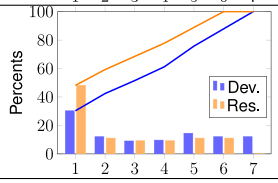
The first possible concern is about using open source code in a manner different from the intentions of its developers. One question asked about using code examples without asking permission. A large majority of both developers and researchers

thought this was of little or no concern. The second question was about quoting code in copyrighted articles. In this case developers were consistently more concerned than researchers. Quite a few added comments about open source licenses prohibiting this, as they require all derivatives of the code to remain free. Others commented that in general one should learn about a project's rules and culture before using it in research, and respect these rules.

Q6: Using code examples from the project without asking permission



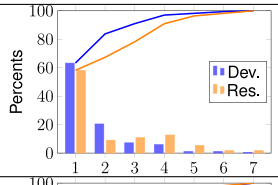
Q7: Publishing open-source code examples from the project in a copyrighted article



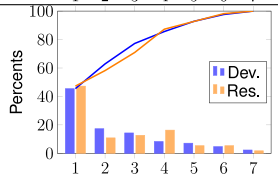
4.2.2. Expectations for privacy and confidentiality

Several questions concerned the exposure of the developers. A set of three questions were about using the texts written by developers to communicate among themselves. Developers were very open to having such texts read when the goal was to better understand technical issues; researchers were slightly more reserved. Both developers and researchers were somewhat more reserved when the goal was to understand social interactions.

Q8: Reading and analyzing the text of communications between developers of the project to better understand technical issues

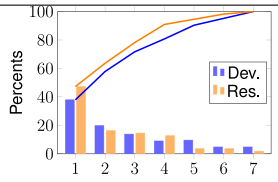


Q9: Reading and analyzing the text of communications between developers of the project to better understand the social interactions between them



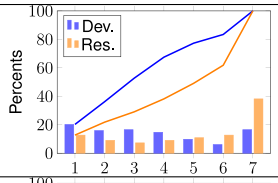
Interestingly, developers were even more reserved about the use of machine learning to analyze all the communications and derive a statistical characterization. One explained in a comment that machine learning may not catch all nuances of human communication and especially cases of non-native-English speakers or jokes.

Q10: Using machine learning to analyze the text of all communications between developers of the project and derive statistical characterizations (e.g. "73% of comments were negative")

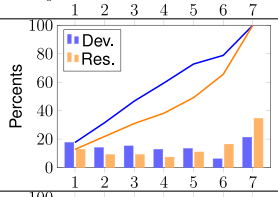


A second set of questions concerned the identification of developers in research reports. These questions elicited a wide range of responses, from those who saw no problem even with the version asking about researchers voicing an opinion about the quality of work of identified developers, to those who were gravely concerned with the version that just asked about identifying developers who had contributed to a project. Tellingly, there was much less concern regarding voicing opinions on the project as a whole as opposed to identifying individual developers.

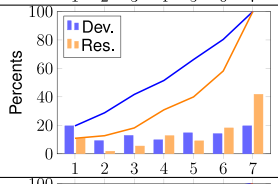
Q18: Identifying developers who contributed to the project in a paper about the research



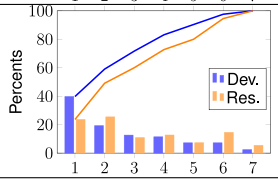
Q19: Identifying developers who wrote specific code or comments that were quoted in a paper about the research



Q21: Voicing an opinion about the quality of the work of specific identified developers



Q20: Voicing an opinion about the quality of the work on the project

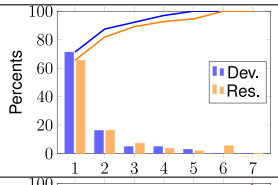


In all these four questions, researchers were significantly more concerned than developers, perhaps due to recent increased awareness of ethics issues in research. It could also be a matter of culture: one developer commented that in the open source culture criticism is welcome, but it should be a constructive discussion on how to improve and not a judgment after the fact. Another wrote "being a part of experiment or being judged by some non-even-a-contributor will probably lead to [...] ending any contributions".

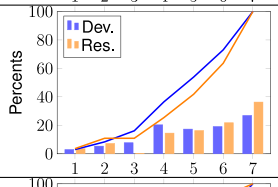
4.2.3. Interfering with developers' work

A related issue is interacting with developers in a way that might interfere with them. There was wide agreement that asking developers about their work is of no or at most little concern. However, the majority were opposed to engaging developers without explaining that this is part of an experiment and obtaining informed consent. In both questions, researchers were marginally more concerned than developers. A third question about deceiving developers so as not to affect their behavior elicited nearly uniform responses, with slightly higher concern by developers.

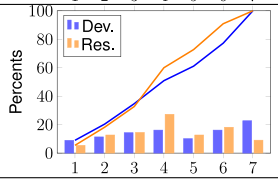
Q11: Approaching developers to ask them about their code or the considerations which guided its writing



Q12: Engaging developers without explaining that this is part of an experiment and obtaining informed consent to participate

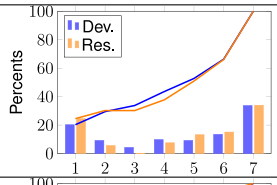


Q13: Telling developers that this is an experiment, but deceiving them about the details so as not to affect their behavior

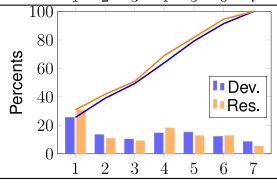


The concern regarding interfering with developers may be modulated by the merits of experiments. Questions about this led to a wide range of responses. The majority were concerned about the possibility of experiments with no scientific value, but there was also a sizeable minority who thought this was of no concern. And there was a nearly uniform response to the possibility of using a new experimental methodology, albeit with more respondents expressing no concern than any other single option.

Q14: Using the project for research with no scientific value



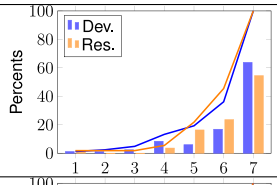
Q15: Using the project to test a new experimental methodology



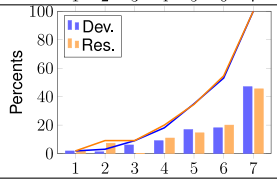
4.2.4. Risk of harming the project

Finally, developers and researchers alike were very concerned about conduct that may cause harm to a project. There were two questions about this, one asking about contributing buggy code and the other about wasting the time of maintainers to review and reject code.

Q16: Contributing buggy code to the project



Q17: Wasting the time of maintainers to review and reject code



4.3. Acceptance of scenarios

The next part of the survey presented several scenarios and asked whether they were acceptable. A 7-point scale was used, as follows:

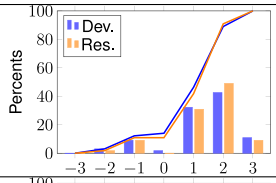
−3 criminal
−2 unacceptable
−1 preferably not done
0 I am not sure
1 not great but tolerable
2 reasonable and acceptable
3 best practice

Respondents were asked to try and be categorical, using 2 and −2. As in the previous section, we present the scenarios here in groups that do not necessarily correspond to the order in which they were presented in the survey.

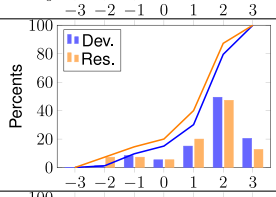
4.3.1. Contribution for self benefit

Three of the questions were about contributing to an open-source project for self benefit, rather than to advance the project. In essence this reflects the "scratch a personal itch" motivation identified by Raymond (2000). The results show that this is an acceptable or at least tolerable practice, with only a small minority of respondents indicating that preferably it would not be done.

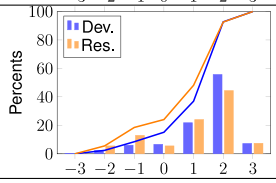
Q23: A novice programmer contributes beginner-quality code in an attempt to build up his or her reputation



Q29: A student contributes code to learn how open source development works

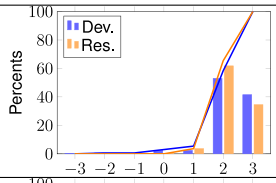


Q25: A developer contributes code that caters to a specific personal need

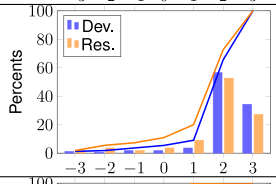


A special case of contributions that are not made just to advance the project is when the contribution is part of research on open source development (like the HC study was). There was wide acceptance of the use of open source projects to perform research about code and its development, with many even calling it a best practice. At the same time there was strong opposition to the idea of submitting buggy code to see if it would be caught, and in the case of security bugs many called such behavior criminal.

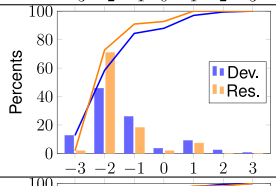
Q30: A researcher analyzes open source code in research on the use of certain programming language constructs



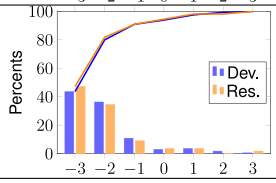
Q31: A researcher contributes valid bug fixes to see how long it takes to incorporate them into the codebase



Q33: A researcher contributes code with a minor bug to see if it would be caught



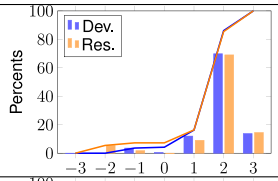
Q34: A researcher contributes code with a security bug to see if it would be caught



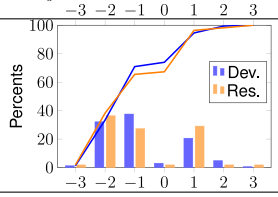
4.3.2. Contributing potentially problematic code

One of the dangers with contributions made for self benefit was that the submitted code may be sub-par. Making this explicit, a pair of questions concerned the role of testing. The responses indicate that contributing code that was tested is acceptable even if it still contains a bug (which the tests failed to uncover), while contributing code that was not adequately tested was usually thought to be unacceptable or at least something that should not be done, although some also thought it was tolerable.

Q24: A developer contributes reasonably tested code that unknowingly still contains a bug

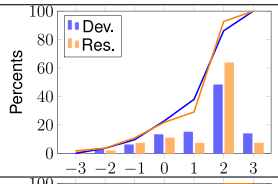


Q26: A developer contributes code that was not adequately tested

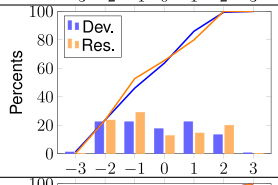


Continuing this sequence, three questions concerned the contribution of code generated by an automatic tool. Contributing such code after checking it manually was generally considered acceptable. But contributing code from an experimental tool to check whether it would be accepted, thereby obtaining an evaluation of the tools quality, elicited a wide range of less favorable responses. Interestingly, the responses were somewhat more accommodating for experimental tools by researchers than for novel tools by developers. In comments, several respondents suggested that automatically generated patches should be identified as such. This would run the risk that developers are prejudiced against tools (Monperrus et al., 2019). And another comment was that only the code is really important, and not who or what produced it.

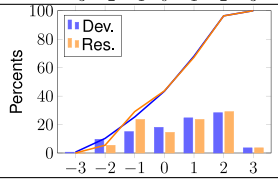
Q27: A developer contributes code based on an automatic tool after verifying it manually



Q28: A developer contributes code generated by a novel tool he or she is developing to see if the tool's output is good enough already



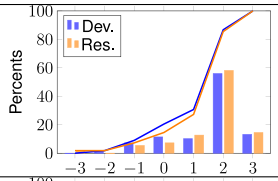
Q32: A researcher contributes code suggested by an experimental tool he or she is developing to assess the tool's possible contribution



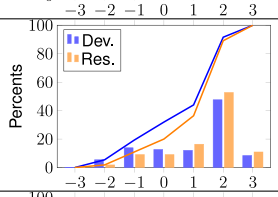
4.3.3. Identification of developers

Reports of research on open source development may identify the project and developers that were used in the research. Identifying the project was generally viewed as acceptable, but when the interactions among developers were studied, there were slightly more developers who thought that it should not be done. Identifying developers who wrote specific commit messages met with significantly stronger opposition, especially from researchers. This correlates with researchers being more sensitive to privacy as we saw above.

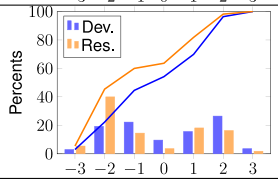
Q36: A researcher analyzes the development trajectory of the code in an open-source project, and identifies the project in the research report



Q37: A researcher analyzes the interactions among developers in a project, and identifies the project in the research report

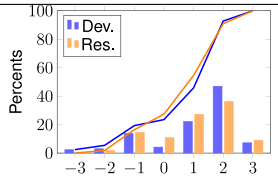


Q38: A researcher analyzes commit messages, and includes examples with the identity of the developers in the research report



A special case of identification is sending surveys to the emails of developers of open-source projects (like this survey was conducted). This was generally considered acceptable or at least tolerable, although a non-negligible minority said that preferably it would not be done. This result contradicts Baltes and Diehl, who quote a developer who said that such unsolicited surveys are "worse than spam" (Baltes and Diehl, 2016). And the present survey also received one response equating academic surveys to spam. But based on our results it may be that only a small minority indeed view such practice as unacceptable. However, if developers who are opposed to such invitations refrained from answering the survey (as the one we received said he does), this result is biased. We have no way to know how many potential respondents actually think that sending unsolicited questionnaires is unacceptable, but then did not register this opinion for this very reason.

Q35: A researcher sends a questionnaire about open source development to emails of developers listed as contributors to an open source project



Wagner et al. report on a similar situation, where an addressee of a survey approached GitHub to check on them (Wagner et al., 2020). The result was a determination that they had not violated GitHub's terms of service, but a suggestion to check beforehand in the future. GitHub documentation indicates that current practice is to set new users' emails to "private" by default.

5. Discussion and recommendations

From a perspective of over 40 years, and the vantage point of software engineering research, it seems that the Belmont report is perhaps not the best basis for discussing ethics. The terminology of "respect for persons" and "beneficence" obscures some of the real concerns of practitioners. Likewise, using US government definitions as a basis for discussions about what exactly constitutes "human research" is distracting and unproductive.

Open source developers in particular seem to want to promote good, which is what beneficence is all about. This starts, of course, with the drive to create good and useful software, and making it free for anyone to use. But it also includes giving others a chance to interact with the community and to develop professionally, as witnessed by the positive responses to our questions about a novice programmer contributing beginner-quality code, or a

student wanting to learn how open-source works. A relevant example is the response to one of the buggy patches submitted to the Linux kernel as part of the HC study, where the maintainer who handled it gave suggestions on what might be done to improve it,[8] in an apparent attempt to mentor what appeared to be a not-very-proficient junior contributor.

This openness and willingness to help naturally extends to researchers. It is OK to use code examples without asking permission. It is acceptable and even a best practice to analyze the code and the project's history. It is perfectly fine to read communications between developers, and even to approach them directly, in order to better understand the project. There is also no problem with contributing valid code to the project to follow how it is treated.

At the same time, what developers care about is

- **Maintain transparency**: If your goal is research and not contribution to the project, state this outright. Give details of your research, obtain informed consent, and take responsibility for your work. In general deception and masquerading are frowned upon; however, when justified by the nature of the research it may be acceptable to coordinate the level of disclosure with project leaders.
- **Do not harm the project**. In particular do not harm the code or put it at risk. The whole ethos of open source development is to improve the code; doing the opposite puts you on a collision course with the community.
- **Do not clash with the developers and maintainers**. Many of a project's developers and maintainers are volunteers, and their time and good will are the scarcest and most valuable resources at the disposal of the project. Negative interactions may lead to frustration and reduced willingness to contribute. In particular, listen to the developers and do not pass judgment on the project or on the activity without giving them a chance to explain themselves. The goal should be to learn and improve, never to blame.
- **Follow the rules**, such as license restrictions[9] and customs. Open source development is a community. If you want to participate, you need to accept the community standards. For example, the fair use argument for publishing code excerpts may be irrelevant, because the developers do not see this as a legal issue but rather as a core values issue.

Upon reflection, these considerations can be generalized and summarized as requirements for **maintaining and justifying trust** in the good intentions of the researchers. One respondent explicitly wrote in a comment that "Some actions may be worth forbidding even if they are not immoral because they damage the reputation of science as an institution and limit future opportunities for cooperation". Trust can be further strengthened if the researchers reciprocate and contribute to the projects they study.

One issue which suffers from a significant divergence of opinions is privacy. This was especially apparent in the questions about identifying developers. Interestingly, there was no appreciable difference between general identification of developers who contributed to a project, and specific identification of developers whose work had been analyzed. The divergent opinions are probably a result of the clash between two ideals: that of openness and attribution, which favors the identification of developers, and that of avoiding harm, which may favor protecting their identity. Further support for this conjecture is given by the fact that researchers, who are probably less motivated by the ideal of openness, tended to oppose the identification of developers more than the developers themselves. A workable solution is not to follow a predefined guideline but to ask those you want to identify for their explicit preference and consent.

Table 3
Some recommendations for possible ethics guidelines.

- Public data, including code and communications, can be used for research.
- Projects used in research should be identified; this is based on the tacit assumption that the research does not harm the project.
- When excerpts of public data are quoted, their authors should be asked whether they prefer to be identified (for attribution) or not (for privacy).
- Developers may be contacted about their work, but the research setting must be disclosed.
- Experiments should be cleared with project leaders and must obtain informed consent.
- Submitting code to open source projects must only be done in good faith.
- Code patches generated by a tool, especially an experimental one, should be noted as such.
- License terms should be respected; if this seems to pose a problem, consult with project leaders.
- Unsolicited invitations to experiments and surveys should be reduced by
  - Targeting only potentially interested developers;
  - Stopping if the response rate is low (indicating lack of interest);
  - Stopping when results stabilize (rather than trying to reach a given number of respondents).

While our results indicate that researchers generally see things eye to eye with developers, the HC study incident shows that this is not always the case. It is therefore advisable for open source repositories and projects to draft and publish explicit rules of conduct for researchers who wish to perform research on them. However, one must remember that regulations only work if there is good will. For example, Sieber wrote about informed consent that "a signed consent form is a bureaucratic and legal maneuver that better protects the researcher's institution than it protects the subject" (Sieber, 2001a). So the real goal is to facilitate a culture of cooperation, not just to draft regulations.

Some concrete recommendations are given in Table 3. These can be used in three contexts. The first is research guidelines in open source repositories. Having such guidelines would ensure better compliance, and avoid the need for creative interpretation of general usage guidelines that do not consider research explicitly (as suggested in Gold and Krinke, 2022). They would also enable large-scale studies on many thousands of projects where it is impractical to verify the preferences of the leaders of each project. The second is ethics codes by professional societies such as the ACM and IEEE. Such societies cater not only to practitioners but also to researchers. They should therefore include research ethics in their guidelines, and they should reach out to affected communities for input about what to include in these guidelines. Finally, a third context is ethics committees and IRBs charged with approving experiments. Such committees should consider not only the legal framework, but also the emergent etiquette of the communities from which subjects are recruited. Communities have opinions and want to be heard. Our survey is an example of how such relevant guidelines were obtained for the case of the open source community.

A recurring problem is recruiting projects and developers to participate in research (Cho and LaRose, 1999; Baltes and Diehl, 2016). Ideally this should be based on an opt-in mechanism to avoid spamming, but such a mechanism does not exist. The suggestion by Wagner et al. (2020) that over 30,000 invitations should be sent to obtain 400 respondents for a survey seems excessive; with such a low response rate they probably have a strong selection bias which invalidates the statistical assumptions. And such a large number of invitations implies that they are sent over some period of time. This can be used to reduce

[8] lore.kernel.org/lkml/20200821081449.GI5493@kadam/

[9] Regrettably this is not as simple as it sounds as there are so many different variants (Laurent, 2004; Zacchiroli, 2022).

the spamming in two ways. First, if the response rate is found to be very low, this implies that a wide gap exists between what the researchers are interested in and think is important and what invitees care about. Asking them about things they do not care about makes it spam. So if the response rate is too low the survey should be stopped. Second, researchers should analyze the results as they are collected, and discontinue data collection as soon as they appear to stabilize enough for their needs. In addition, snowballing can be encouraged: if participants agree to forward the invitation to their contacts, this reflects an expectation that they will be interested. Another idea is that informing invitees of how their email was obtained would be courteous. Finally, one respondent suggested that invitations should be posted to development mailing list rather than approaching the developers directly, which would be more in line with open source development culture.

6. Threats to validity

Our survey, like any opinion survey, suffers from a potential threat to construct validity. Respondents spend only a few seconds forming opinions on hypothetical scenarios that they have not experienced. For example, developers may not fully realize the risks they take when research is performed on their code, e.g. if they are identified and presented in a negative light. This should be kept in mind especially with regard to privacy and confidentiality. In addition, as some respondents noted in their comments, survey questions cannot really fully describe a situation, and therefore in many cases the actual answer is "it depends". We note, however, that the vast majority of respondents did make selections from the given options and did not skip questions or select "other".

A bigger problem is the threat to external validity, namely whether our results are representative of developers and researchers in general. This has two facets: whether the sample is big enough, and who is included in it. Regarding the size of the sample we checked the results obtained from only half our respondents, 88 developers and 22 researchers, and found that the results for developers are essentially the same, and for researchers very close, despite their low number. It therefore seems that the sample size is not a problem, and the results are representative for developers and researchers who respond to such surveys.

However, external validity may still be compromised due to a possible selection bias. Our participants reflect a self-selection to accept the invitation and answer the survey. It is reasonable to assume that practitioners who knew about the Linux-UMN incident — and especially those who have strong opinions about being experimented on — had a higher tendency to participate. Indeed, some of the respondents included rather emotional comments such as "We are NOT computers" and "Consent. It's a thing now. Get it." and even much stronger language. At the same time, those who could not care less about ethics most probably just deleted the invitation email and did not participate. The results may therefore not be representative of the whole population of developers and researchers. Note, however, that this implies that our results about acceptable behavior may be conservative rather than being too lenient.

7. Conclusions

Both open source developers and software engineering researchers come from a technical background. As such they may have blind spots when it comes to social issues and to ethics. As Harrison wrote, "Physicists don't have to ask an electron if they can measure it, nor are they obligated to allow the electron to quit the experiment at any time" (Harrison, 2000). And awareness of this often leads to a reliance on (and confinement to) legal requirements and licenses (Bailey et al., 2012; Gold and Krinke, 2022). But the laws originate from a background of privacy issues, and the licenses from a background of code distribution, so their implications regarding ethics issues like consent are incidental rather than intended.

Current ethics guidelines were defined in the context of biomedical research, and their application to software engineering research requires some adjustments. For example, issues that are not well covered include

- Using existing publicly accessible artifacts (code, development history, documentation, and communications among developers)
- Observational studies (watching people work)
- Interacting with developers workflows (contributing code, contributing tools, collaborating in other ways)

One approach to reduce ethical friction is therefore to develop guidelines that are better aligned with the practices of software development. Some ideas along this line were proposed above in Table 3.

Another possible approach is to join forces: instead of imposing on the research subjects and potentially alienating them, involve them as participants in the research (Bakardjieva and Feenberg, 2000). Such an approach is in line with the open source philosophy, and may be expected to lead to better scientific results — results that are more correct and more relevant, being based on the developers' point of view. A further step is this direction is to use participatory research: given that many researchers are also contributing developers, they can study the projects they work on from within. This proactively returns to the community (Oezbek, 2008), in a way that may be better appreciated than the publication of an academic paper.

Yet another alternative is to try to use industrial collaboration (Rico et al., 2021). Such collaborations naturally enjoy high relevance, because they necessarily focus on real needs. In addition, for smaller research projects one can hire developers for experiments (Sjøberg et al., 2002, 2003), or even use "human computation" platforms like Mechanical Turk (Sabou et al., 2020). Like experiments on open source projects, these approaches have the advantage of having developers work in their normal environment.

Last, the reaction to ethics violations should be carefully considered. Research like the HC study is important, and should not be discounted outright. In this specific case, its main contribution was to show how potential vulnerabilities could be turned into real vulnerabilities, and suggest that this could be hidden in innocent-looking patches. Another unintended contribution was to show that the Linux vetting procedure works, and in fact prevented these innocent-looking commits from being accepted. But the study suffered from a significant ethics blind spot. To be acceptable, it should have been coordinated with the target project, and performed in a manner they approve. To prevent such incidents from repeating, we do not need to chastise UMN — we need to develop procedures and mechanisms to coordinate research on open-source projects.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data has been shared on Zenodo at DOI https://doi.org/10.5281/zenodo.5752053.

Acknowledgments

Many thanks are due to all the survey participants, especially those who invested extra effort to write comments and explain their positions.

References

American Psychological Association, 2003. Ethical principles of psychologists and code of conduct. URL www.apa.org/ethics/code.

Amit, I., Feitelson, D.G., 2021. Corrective commit probability: A measure of the effort invested in bug fixing. Softw. Quality J. 29 (4), 817–861. http://dx.doi.org/10.1007/s11219-021-09564-z.

Association for Computing Machinery, 2018. ACM code of ethics and professional conduct. URL https://www.acm.org/code-of-ethics.

Bacchelli, A., Lanza, M., Robbes, R., 2010. Linking e-mails and source code artifacts. In: 32nd Intl. Conf. Softw. Eng. 1, pp. 375–384. http://dx.doi.org/10.1145/1806799.1806855.

Bailey, M., et al., 2012. The Menlo report: Ethical principles guiding information and communication technology research. URL https://www.dhs.gov/sites/default/files/publications/CSD-MenloPrinciplesCORE-20120803_1.pdf.

Bakardjieva, M., Feenberg, A., 2000. Involving the virtual subject. Ethics Inf. Tech. 2 (4), 233–240. http://dx.doi.org/10.1023/A:1011454606534.

Baltes, S., Diehl, S., 2016. Worse than spam: Issues in sampling software developers. In: 10th Intl. Symp. Empirical Softw. Eng. & Measurement. http://dx.doi.org/10.1145/2961111.2962628, art. 52.

Bassett, E.H., O'Riordan, K., 2002. Ethics of Internet research: Contesting the human subjects research model. Ethics Inf. Tech. 4 (3), 233–247. http://dx.doi.org/10.1023/A:1021319125207.

Becker-Kornstaedt, U., 2001. Descriptive software process modeling—how to deal with sensitive process information. Empirical Softw. Eng. 6 (4), 353–367. http://dx.doi.org/10.1023/A:1011986902298.

Berander, P., 2004. Using students as subjects in requirements prioritization. In: Intl. Symp. Empirical Softw. Eng.. pp. 167–176. http://dx.doi.org/10.1109/ISESE.2004.1334904.

Berry, D.M., 2004. Internet research: Privacy, ethics, and alienation: An open source approach. Internet Res. 14 (4), 323–332. http://dx.doi.org/10.1108/10662240410555333.

Carver, J., Jaccheri, L., Morasca, S., Shull, F., 2003. Issues in using students in empirical studies in software engineering education. In: 9th Softw. Metrics Symp. pp. 239–249. http://dx.doi.org/10.1109/METRIC.2003.1232471.

Carver, J.C., Jaccheri, L., Morasca, S., Shull, F., 2010. A checklist for integrating student empirical studies with research and teaching goals. Empirical Softw. Eng 15 (1), 35–59. http://dx.doi.org/10.1007/s10664-009-9109-9.

Chin, M., 2021. How a university got itself banned from the linux kernel. The verge. URL www.theverge.com/2021/4/30/22410164/linux-kernel-university-of-minnesota-banned-open-source.

Cho, H., LaRose, R., 1999. Privacy issues in internet surveys. Social Sci. Comput. Rev. 17 (4), 421–434. http://dx.doi.org/10.1177/089443939901700402.

Chopra, S., Dexter, S., 2009. The freedoms of software and its ethical uses. Ethics Inf. Tech. 11 (4), 287–297. http://dx.doi.org/10.1007/s10676-009-9191-0.

de Laat, P.B., 2010. How can contributors to open-source communities be trusted? on the assumption, inference, and substitution of trust. Ethics Inf. Tech. 12 (4), 327–341. http://dx.doi.org/10.1007/s10676-010-9230-x.

de Laat, P.B., 2014. From open-source software to Wikipedia: 'backgrounding' trust by collective monitoring and reputation tracking. Ethics Inf. Tech. 16 (2), 157–169. http://dx.doi.org/10.1007/s10676-014-9342-9.

El-Emam, K., 2001. Ethics and open source. Empirical Softw. Eng. 6 (4), 291–292. http://dx.doi.org/10.1023/A:1011962213685.

Fabijan, A., Dmitriev, P., McFarland, C., Vermeer, L., Holmström Olsson, H., Bosch, J., 2018. Experimentation growth: Evolving trustworthy A/B testing capabilities in online software companies. J. Softw. Evol. Process 30 (12), e2113. http://dx.doi.org/10.1002/smr.2113.

Feitelson, D.G., Frachtenberg, E., Beck, K.L., 2013. Development and deployment at Facebook. IEEE Internet Comput. 17 (4), 8–17. http://dx.doi.org/10.1109/MIC.2013.25.

Floyd, B., Santander, T., Weimer, W., 2017. Decoding the representation of code in the brain: An fMRI study of code review and expertise. In: 39th Intl. Conf. Softw. Eng. pp. 175–186. http://dx.doi.org/10.1109/ICSE.2017.24.

Gold, N.E., Krinke, J., 2022. Ethics in the mining of software repositories. Empirical Softw. Eng. 27 (1), http://dx.doi.org/10.1007/s10664-021-10057-7, art. 17.

Graziotin, D., Lenberg, P., Feldt, R., Wagner, S., 2022. Psychometrics in behavioral software engineering: A methodological introduction with guidelines. ACM Trans. Softw. Eng. Methodol. 31 (1), http://dx.doi.org/10.1145/3469888, art. 7.

Grodzinsky, F.S., K.Miller, Wolf, M.J., 2003. Ethical issues in open source software. J. Information Communication & Ethics in Society 1 (4), 193–205. http://dx.doi.org/10.1108/14779960380000235.

Hall, T., Flynn, V., 2001. Ethical issues in software engineering research: A survey of current practice. Empirical Softw. Eng. 6 (4), 305–317. http://dx.doi.org/10.1023/A:1011922615502.

Harrison, W., 2000. An issue of ethics: Responsibilities and obligations of empirical software engineering researchers. Empirical Softw. Eng. 5 (1), 7–9. http://dx.doi.org/10.1023/A:1009870532419.

Harrison, W., 2001. Open source and empirical software engineering. Empirical Softw. Eng. 6 (3), 193–194. http://dx.doi.org/10.1023/A:1017379030770.

IEEE, 2020. IEEE code of ethics. URL http://www.ieee.org/about/corporate/governance/p7-8.html.

Intl. Federation for Information Processing, 2021. IFIP code of ethics and professional conduct. URL http://www.ipthree.org/ifip-code-of-ethics.

King, S.A., 1996. Researching internet communities: Proposed ethical guidelines for the reporting of results. Inf. Soc. 12, 119–128. http://dx.doi.org/10.1080/713856145.

King, J.L., 2015. Humans in computing: Growing responsibilities for researchers. Comm. ACM 58 (3), 31–33. http://dx.doi.org/10.1145/2723675.

Kohavi, R., Longbotham, R., Sommerfield, D., Henne, R.M., 2009. Controlled experiments on the web: Survey and practical guide. Data Mining Knowl. Discover. 18 (1), 140–181. http://dx.doi.org/10.1007/s10618-008-0114-1.

Kramer, A.D.I., Guillory, J.E., Hancock, J.T., 2014. Experimental evidence of massive-scale emotional contagion through social networks. Proc. Natl. Acad. Sci. USA 111 (24), 8788–8790. http://dx.doi.org/10.1073/pnas.1320040111.

Laurent, A.M.St., 2004. UnderstandIng Open Source and Free Software Licensing. O'Reilly Media.

Lethbridge, T.C., 2001. Mixing software engineering research and development—what needs ethical review and what does not? Empirical Softw. Eng. 6 (4), 319–321. http://dx.doi.org/10.1023/A:1011974632340.

Li, P.L., et al., 2021. Evolving software to be ML-driven utilizing real-world A/B testing: Experiences, insights, challenges. In: 43rd Intl. Conf. Softw. Eng.. SEIP track, pp. 170–179. http://dx.doi.org/10.1109/ICSE-SEIP52600.2021.00026.

Liebel, G., Chakraborty, S., 2021. Ethical issues in empirical studies using student subjects: Re-visiting practices and perceptions. Empirical Softw. Eng. 26 (3), http://dx.doi.org/10.1007/s10664-021-09958-4, art. 40.

Linux Foundation Technical Advisory Board, 2021. Report on University of Minnesota breach-of-trust incident. URL https://lore.kernel.org/lkml/202105051005.49BFABCE@keescook/.

Liu, C., Yang, J., Tan, L., Hafiz, M., 2013. R2Fix: Automatically generating bug fixes from bug reports. In: 6th Intl. Conf. Softw. Testing, Verification & Validation. pp. 282–291. http://dx.doi.org/10.1109/ICST.2013.24.

Meredith, S., 2018. Facebook-cambridge analytica: A timeline of the data hijacking scandal. CNBC. URL http://www.cnbc.com/2018/04/10/facebook-cambridge-analytica-a-timeline-of-the-data-hijacking-scandal.html.

Monperrus, M., Urli, S., Durieux, T., Martinez, M., Baudry, B., Seinturier, L., 2019. Repairnator patches programs automatically. Ubiquity. http://dx.doi.org/10.1145/3349589, art. 2.

Oezbek, C., 2008. Research ethics for studying open source projects. In: Research Room At FOSDEM.

Peitek, N., Siegmund, J., Apel, S., Kästner, C., Parnin, C., Bethmann, A., Leich, T., Saake, G., Brechmann, A., 2020. A look into programmer's heads. IEEE Trans. Softw. Eng. 46 (4), 442–462. http://dx.doi.org/10.1109/TSE.2018.2863303.

Raymond, E.S., 2000. The cathedral and the bazaar. URL www.catb.org/ esr/writings/cathedral-bazaar/cathedral-bazaar.

Rico, S., Bjarnason, E., Engström, E., Höst, M., Runeson, P., 2021. A case study of industry-academia communication in a joint software engineering research project. J. Softw. Evol. Process 33 (10), e2372. http://dx.doi.org/10.1002/smr.2372.

Sabou, M., Winkler, D., Biffl, S., 2020. Empirical software engineering experimentation with human computation. In: Felderer, M., Travassos, G.H. (Eds.), Contemporary Empirical Methods in Software Engineering. Springer Nature, pp. 173–215. http://dx.doi.org/10.1007/978-3-030-32489-6_7.

Sieber, J.E., 2001a. Not your ordinary research. Empirical Softw. Eng. 6 (4), 323–327. http://dx.doi.org/10.1023/A:1011926716411.

Sieber, J.E., 2001b. Protecting research subjects, employees and researchers: Implications for software engineering. Empirical Softw. Eng. 6 (4), 329–341. http://dx.doi.org/10.1023/A:1011978700481.

Singer, J., Vinson, N.G., 2002. Ethical issues in empirical studies of software engineering. IEEE Trans. Softw. Eng. 28 (12), 1171–1180. http://dx.doi.org/10.1109/TSE.2002.1158289.

Sjøberg, D.I.K., Anda, B., Arisholm, E., Dybå, T., Jørgensen, M., Karahasanovic, A., Koren, E.F., Vokác, M., 2002. Conducting realistic experiments in software engineering. In: Intl. Symp. Empirical Softw. Eng.. pp. 17–26. http://dx.doi.org/10.1109/ISESE.2002.1166921.

Sjøberg, D.I.K., Anda, B., Arisholm, E., Dybå, T., Jørgensen, M., Karahasanović, A., Vokáč, M., 2003. Challenges and recommendations when increasing the realism of controlled software engineering experiments. In: Conradi, R., Wang, A.I. (Eds.), Empirical Methods and Studies in Software Engineering: Experiences from ESERNET. In: Lect. Notes Comput. Sci., vol. 2765, Springer-Verlag, pp. 24–38. http://dx.doi.org/10.1007/978-3-540-45143-3_3.

Song, Y., Wang, T., Shen, Y., Chang, J., 2022. A new method for evaluating core developers in open source software. In: 13th Intl. Conf. Softw. Eng. Serv. Sci.. pp. 48–53. http://dx.doi.org/10.1109/ICSESS54813.2022.9930230.

Staron, M., 2007. Using students as subjects in experiments – a quantitative analysis of the influence of experimentation on students' learning process. In: 20th Conf. Softw. Eng. Education & Training. pp. 221–228. http://dx.doi.org/10.1109/CSEET.2007.56.

Tartler, R., Sincero, J., Dietrich, C., Schröder-Preikschat, W., Lohmann, D., 2012. Revealing and repairing configuration inconsistencies in large-scale system software. Intl. J. Softw. Tools Tech. Transf. 14 (5), 531–551. http://dx.doi.org/10.1007/s10009-012-0225-2.

The National Commission for the Protection of Human Subjects of Biomedical and Behavioral Research, 1979. The Belmont report. URL https://www.hhs.gov/ohrp/regulations-and-policy/belmont-report/read-the-belmont-report/index.html.

Thomas, J., 1996a. Introduction: A debate about the ethics of fair practices for collecting social science data in cyberspace. Inf. Soc. 12 (2), 107–118. http://dx.doi.org/10.1080/713856137.

Thomas, J., 1996b. When cyberresearch goes awry: The ethics of the Rimm cyberporn study. Inf. Soc. 12 (2), 189–198. http://dx.doi.org/10.1080/713856140.

U.S. Dept. Health & Human Services, 2018. Regulations 45 CFR 46. URL https://www.hhs.gov/ohrp/regulations-and-policy/regulations/45-cfr-46/index.html.

Verma, I.M., 2014. Editorial expression of concern: Experimental evidence of massivescale emotional contagion through social networks. Proc. Natl. Acad. Sci. USA 111 (29), 10779. http://dx.doi.org/10.1073/pnas.1412469111.

Vinson, N., Singer, J., 2001. Getting to the source of ethical issues. Empirical Softw. Eng. 6 (4), 293–297. http://dx.doi.org/10.1023/A:1011966430523.

Vinson, N.G., Singer, J., 2008. A practical guide to ethical research involving humans. In: Shull, F., Singer, J., Sjøberg, D.I.K. (Eds.), Guide To Advanced Empirical Software Engineering. Springer-Verlag, http://dx.doi.org/10.1007/978-1-84800-044-5_9, chap. 9.

von Krogh, G., von Hippel, E., 2006. The promise of research on open source software. Manag. Sci. 52 (7), 975–983. http://dx.doi.org/10.1287/mnsc.1060.0560.

Wagner, S., Mendez, D., Felderer, M., Graziotin, D., Kalinowski, M., 2020. Challenges in survey research. In: Felderer, M., Travassos, G.H. (Eds.), Contemporary Empirical Methods in Software Engineering. Springer Nature, pp. 93–125. http://dx.doi.org/10.1007/978-3-030-32489-6_4.

Watts, R.D., Brightman, A.O., 2017. Crossing the line: When does the involvement of human subjects in testing of engineering capstone design projects require oversight by an IRB? In: ASEE Ann. Conf. & Expo.. http://dx.doi.org/10.18260/1-2--28091, art. 19741.

Wolf, M.J., Miller, K.W., Grodzinsky, F.S., 2009. On the meaning of free software. Ethics Inf. Tech. 11 (4), 279–286. http://dx.doi.org/10.1007/s10676-009-9207-9.

Wu, Z., Li, J., Fu, C., Xuan, Q., Xiang, Y., 2018. Network-based ranking for open source software developer prediction. Intl. J. Softw. Eng. Knowl. Eng. 28 (6), 845–868. http://dx.doi.org/10.1142/S0218194018500250.

Wu, Q., Lu, K., 2020a. Clarifications on the hypocrite commit work (FAQ). URL https://www-users.cse.umn.edu/~kjlu/papers/clarifications-hc.pdf.

Wu, Q., Lu, K., 2020b. On the feasibility of stealthily introducing vulnerabilities in open-source software via hypocrite commits. In: Accepted (with a Slightly Different Title) To the 42nd IEEE Symp. Security & Privacy, 2021, but withdrawn.

Yan, D., Qi, B., Zhang, Y., Shao, Z., 2020. M-birank: Co-ranking developers and projects using multiple developer-project interactions in open source software community. EURASIP J. Wirel. Commun. Netw. 2020, http://dx.doi.org/10.1186/s13638-020-01820-3, art. 215.

Yu, Y., 2020. Role of reciprocity in firm's open source strategies. Baltic J. Manag. 15 (5), 797–815. http://dx.doi.org/10.1108/BJM-12-2019-0408.

Zacchiroli, S., 2022. A large-scale dataset of (open source) license text variants. In: 19th Working Conf. Mining Softw. Repositories. pp. 757–761. http://dx.doi.org/10.1145/3524842.3528491.

Zimmer, M., 2010. But the data is already public: On the ethics of research in facebook. Ethics Inf. Tech. 12 (4), 313–325. http://dx.doi.org/10.1007/s10676-010-9227-5.

Dror G. Feitelson holds the Berthold Badler chair in Computer Science at the School of Computer Science and Engineering of the Hebrew University of Jerusalem, where he has been a faculty member since 1995. His research interests are in experimental computer science and empirical software engineering, with an emphasis on human aspects and experimental methodology. His current research focus is on program comprehension and what makes software hard to understand. This includes work on code complexity and on what makes variable names "meaningful".