I enjoyed Warren Harrison’s From the Editor column “Whose Information Is It Anyway?” (July/August 03). Including open source projects in research studies can provide useful feedback to the open source industry, widen the scope of existing software engineering research, and give us a better understanding of a larger variety of software products. However, using open source data is still a debated issue from an ethical perspective.

Ethics are about what a community accepts to be right and wrong. In this note, I describe what I find to be the right ways of getting involved in the open source world as a researcher. I think that further discussions on this issue will be useful to our community.

I believe that researchers, like other individuals, should have a right to get involved in open source projects and contribute to them in their own unique way. When researchers begin working on an open source project, they should email the project’s mailing list(s), introduce themselves, clearly explain the scope and goals of their study, and ask for help.

After that, they should feel responsible for contributing to the project while pursuing their research goals. This includes answering questions, exchanging ideas, contributing to the common knowledge, and keeping the project participants updated about the research results and final publications. By doing this, researchers automatically become active participants in the project. Then, they are entitled to comment on such issues as why the last release has more bugs, why progress is slow, and so on, just as other participants do.

In addition, I think researchers have a responsibility to make their results publicly accessible, so that many others can question their research methods and results. On the positive side, when the larger community can verify research results on an open platform, it might be possible to reach healthier conclusions.

Researchers should also avoid naming developers in correspondence and publications, although this doesn’t make tracing individual developers totally impossible. In my opinion, open source communities have already accepted the fact that nobody is error-free and integrated bug tracking systems into their processes. Therefore, developers should be open to the opinions of others.

Developers need to be conscious about the fact that the data and artifacts they leave behind will be publicly available; they can hide their identity if they want. If our research community feels strongly about this issue, we can ask important open source projects and project-hosting Web sites such as sourceforge.net and freshmeat.net to post warnings or disclaimers on their developer Web sites. Then smaller projects can follow what they do. Presently, developers might not be aware of the issues discussed here. Developer information is even made available on some projects’ Web sites (see http://people.kde.org).

A final note about the quality of data in open source repositories: Although the data might be
out there, it might not be suitable to be readily used in empirical studies. For example, if a researcher is interested in using defect data, preliminary studies about consistency in data collection, completeness of defect records, and so on, will be necessary.

A. Günes Koru  
PhD candidate  
Southern Methodist University  
gkoru@engr.smu.edu

Developers should be aware that the data and artifacts they leave behind will be publicly available; they can hide their identity if they want.

Warren Harrison responds:  
I agree that OSS is not only a convenient source of data, as more of it becomes available to the public, but also an important segment to study in itself. Some studies are statistical in nature; there’s no way to identify individual participants. On the other hand, there are tools and studies in which individuals are easily identified. Just because a tool’s author doesn’t identify individuals directly, I don’t think he or she is off the hook ethically if someone can use the tool (or the study) to identify individuals and it is made available to others.

For instance, suppose you have an OSS product created by one or two developers. If, in comparing various OSS products, you identify this product (or describe it in a way that makes its identification trivial) as having the highest bug rate, you have for all intents and purposes identified the individuals. Now suppose your analysis includes some judgmental aspect such as “most maintainable code,” which is really a synthesis of metrics and opinions as opposed to a simple count of artifacts such as lines of code or number of comments. Then you’ve added insult to injury.

However, I believe the real burden is on those organizations and people that host this kind of data to start with. Either contributors should be given an explicit informed-consent option, or (preferably) the data should be stored in such a way that extracting individual identities is simply not possible.

It would indeed be interesting if aca-

The B. Thomas Golisano College of Computing and Information Sciences (GCCIS) is pleased to invite applications for tenure-track Assistant and Associate Professor positions in its Computer Science, Information Technology, and Software Engineering departments starting September 2004. Successful candidates should possess the Ph.D. in one of the specified fields or closely related area; they should have interests in teaching and research, and should have an ability to contribute in meaningful ways to the Institute’s commitment to cultural diversity and pluralism. Exceptional candidates will be considered for the rank of Full Professor. Additional departmental requirements include:

• Computer Science (CS): Candidates should have an interest in intelligent systems in computer vision or machine learning, security, data mining, or human-computer interfaces.
• Information Technology (IT): Candidates should have strengths in one or more of the following areas: network design and protocols, database design and application development, security, gaming, visual object-based development, web and multimedia development, and computer-mediated experiences.
• Software Engineering (SE): Candidates must have a deep interest in education at the masters and undergraduate levels as well as a strong desire to conduct research on software process, software architectures and design, software quality assurance, software construction, and secure systems. Professional experience developing software is desirable.

GCCIS (www.rit.edu/~gccis) is RIT’s newest college at the 1,200-acre suburban university located in Rochester, New York, just 90 minutes from Niagara Falls, 3 hours from Toronto, and 6 hours from NYC. In addition to CS, IT, and SE, the college is home to the Lab for Applied Computing, the research arm of the college. The college, with 90+ faculty members, is housed in a new 126,500 square foot state-of-the-art building with 13 classrooms, student team rooms, and laboratories, including specialty labs for security, vision, embedded systems, networking, systems administration, and streaming multimedia. Our 2,300 undergraduate students are enrolled in BS programs in IT, Applied Networking and System Administration, IT New Media, Computer Science, and Software Engineering. Our 600 graduate students are enrolled in IT, Software Development & Management, and Computer Science MS programs.

Interested applicants should submit the following information to the email address below by January 30, 2004: a summary of education and professional background, a current list of publications, a summary of teaching and research experience, the names of three references, and a brief description of future research plans. Please indicate for which department(s) you would like your application to be considered.

B. Thomas Golisano College of Computing and Information Sciences  
Dean’s Office - Faculty Search Committee  
Rochester Institute of Technology  
20 Lomb Memorial Drive, Rochester, NY 14623  
Email: Faculty_Search@gccis.rit.edu

RIT is an Affirmative Action/Equal Employment Opportunity Employer.
demic studies of OSS began being reviewed by campus Human Subjects boards, as more traditional survey and controlled experiments already are.

Balancing both worlds

I enjoyed Bob Glass's last two Loyal Opposition columns. When I browse through so-called “software engineering” books, I find that authors rarely have much real-world experience. Of course there are a few exceptions, but rarely under the software engineering umbrella.

About the academic-versus-industry debate, I’ve found that balancing the two worlds is hard. Usually for professors, people in industry are just developers who write buggy code. And for industry people, professors only care about writing conference papers, not code. It’s a pity. At the end of the day, we must remember that we are basically part of the same business—like it or not.

Also, if you’re trying to go back to school, most computer science departments don’t care about your industrial experience; you’re just another student. It would be nice if universities could offer two PhD tracks: industrial and academic. If you chose the industrial one, you should be able to work on more real-world problems. Perhaps that’s the one for teaching software engineering.

Finally, I believe it’s possible to balance both worlds. This fall I’ll start my PhD in the field as a part-time student, much like James Cain said in the last issue’s Letters to the Editor: one day a week. It’s probably a long road, but I’m looking forward to it.

Omar Alonso
San Mateo, CA
oralonso@yahoo.com

The agile sweet spot

While it’s always nice to be noticed, I fear that the July/August Manager column “Scaling Agile Methods” by Don Reifer, Frank Maurer, and Hakan Erdogmus misrepresented my position on this topic. Naturally, I intended my sound byte “scaling agile methods is the last thing you should do” to be startling, but I also intended it to be precise. I’m not against scaling agile methods, but I think we should try other things first (http://martinfowler.com/bliki/LargeAgileProjects.html). In particular, we’ve seen a lot of evidence that large projects often show alarming levels of overstaffing and feature bloat. Dealing with these problems often turns a large project into one that fits nicely into the agile sweet spot.

Martin Fowler
Chief scientist, ThoughtWorks
http://martinfowler.com
More recent references


Holger Eichelberger
University of Würzburg
eichelberger@informatik.uni-wuerzburg.de

Correction

In our Nov./Dec. Quality Time column (“Putting Your Best Tests Forward” by Gregg Rothermel and Sebastian Elbaum), we erroneously switched the lines for “Optimal” and “Coverage” in the figure depicting fault detection rates for four prioritized test suites. The correct figure is shown here.