On Understanding How to Introduce an Innovation to an Open Source Project

Christopher Oezbek, Lutz Prechelt
Freie Universität Berlin
Institut für Informatik
Takustr. 9, 14195 Berlin, Germany
{oezbek, prechelt}@inf.fu-berlin.de

Abstract

Position Paper

We propose to research the introduction of Software Engineering inventions into Open Source projects (1) to help researchers with creating opportunities for evaluating their tools, methods and process designs in real-life settings, and (2) to help Open Source projects with improving their processes based on state-of-the-art knowledge. Such research will go beyond diffusion and dissemination of inventions to active introduction, and thus increase the chances of adoption. We will discuss the research approach, our preliminary insights, limitations of the approach, and why researchers interested in evaluating their own inventions should be interested in this research.

1 Introduction

Most software engineering research produces technology such as tools, methods, or processes to be applied during the construction of software systems. It has been gradually understood that the empirical evaluation of such inventions is necessary to judge research progress and generate acceptance outside of academia [25, 28].

There are two classic scenarios for how to conduct such empirical evaluations: First, there is the laboratory trial, often in the form of controlled experiments with student subjects. Such studies are difficult to set up in such a way that they are sufficiently impartial and realistic (in particular in their choice of task) to be credible — but credibility is what counts [19]. Controlled experiments with professional subjects are harder to set up, but often hardly more credible. Second, there is the industry trial, commonly performed as a case study in cooperation with a company. While such studies are certainly realistic, they have problems too: Cost and risk considerations make it hard to find industrial partners, non-disclosure constraints make it hard to fully describe the setting and results, and company idiosyncracies often make it hard to understand generalizability.

For many (though not all) evaluation purposes, some researchers consider observational studies in the context of Open Source Software (OSS) projects to be a third approach and one with almost ideal properties in many respects: Credibility can often be high, they are easy to observe, publication constraints hardly exist, risk considerations are more relaxed, and corporate cost considerations are replaced by (mere) group willingness hurdles.

Unfortunately, OSS projects are not interested in studies, they are interested in developing software. So, performing a study first requires to make the project *adopt* the invention in its normal work. However, as anybody knows who has ever tried to get any group of people to adopt an invention (that is, to *introduce* the invention as an *innovation*), this is rather difficult.

So, rather than letting a long row of researchers individually attempt, fail, attempt, fail, get frustrated, and give up, we suggest to make the adoption process itself the subject of research in order to provide these researchers with a proven methodology for introducing an invention to an OSS project.

Here the term *introduction* is used to signify the planned initiation of an adoption process within an organization or social system. *Adoption* then can be seen as the turning point where inventions become innovations that are actively used by individuals [7]. *Introduction* contrasts well with *diffusion*, which carries more passive connotations, and *dissemination*, which does not go beyond distributing information or resources related to an invention.

From the researcher's point of view, combining active introduction with OSS projects has several advantages. In contrast to industry settings, the public visibility of most of the working process, artifacts, and communication as well as the openness for outsiders to contribute to these projects allow the researcher to both capture and influence the project to a much larger degree. In contrast to dissemination and diffusion, the researcher can (1) observe

the adoption and use of the invention as it happens rather than performing post-hoc analysis, (2) tailor the invention to the particularities of the project and repair problems that often plague early versions of inventions on the spot, and (3) choose the project such as to maximize the insights gained.

From the point of view of the OSS community, such research increases their chances for benefitting from software engineering improvements, given the fact that conventional approaches to managing software process improvement such as CMMI [5], even approaches specialized to OSS [8], do not explain how the actual introduction of the improvements should be conducted, and traditional key success mechanisms such as management commitment and support [24] are unlikely to work.

The rest of the paper presents our research approach for gaining insights into the introduction of inventions in OSS projects as well as our preliminary results for the following research questions:

1. How to select target projects suitable for introducing software engineering inventions.

2. How to approach a project to offer an invention.

3. How to interpret reactions and make strategic and tactical decisions based on them in the course of the adoption process.

4. How to phase out involvement and exit the project.

5. How to obtain evaluation result data during and after the introduction.

2 Research approach

To develop an understanding of the introduction of inventions, we will perform a series of iterative case-studies [27] using action-research methodology [2], i.e., a circular, collaborative process of planning, acting and reflecting. These studies will be performed with three different inventions of different type and with a variety of different Open Source projects. We will not introduce several process improvements in the same project [9] in order to avoid synergies or cannibalization between improvements [11].
Inside each case we will gather qualitative data on action-reaction relationships and recurring patterns (using Grounded Theory data analysis methodology [6]) to obtain an understanding of the key interactions during an introduction effort.

We will work on minimizing risk toward the project and on protecting the autonomy of the subjects [4] by creating an atmosphere of collaboration, involvement and participation between project and researcher, and protecting privacy and confidentiality [3, 13]. Even though Open Source projects are very robust against negative influence from the outside, similar precautions must be taken by researchers who evaluate their inventions in projects to ensure proper ethical conduct.

3 How to choose a host project?

Choosing an appropriate Open Source project when evaluating a software engineering invention is important to establish a case that is (a) typical enough to generalize to other projects, (b) suitable for the given invention, and (c) has potential for interesting interaction regarding the introduction.

In particular, the project should be Open Source not only by license but also by development style: The project members need to be distributed rather than co-located at a single company site, communication must be public and preferably archived, it must be possible for external newcomers to join the project, and basic processes and tools (such as release process, issue tracker and version repository) should be established. The distribution, observability, and openness ensure that the researcher can study the use of the invention at all, while the presence of basic processes and tools indicates that the project probably fulfills basic professional software engineering standards so that study results may generalize to other software development projects. Fortunately, with the existence of project hosts such as SourceForge these tools and processes are now standard.

Regarding the size of the project a viable middle ground must be found between too small and too large. Small projects with less than three to four developers usually have little interaction, communication overhead, tool usage, and process inefficiencies or are still in the process of establishing basic process patterns. They are thus rather unsuitable for all but the most basic software engineering inventions. Large projects with more than fifty developers on the other hand have quite the opposite problem: They usually have well established processes, so that the "not invented here"-syndrome, explicit opposition, tedious consensus finding, low perceived benefit against the established processes, and high communication overhead might make it impossible for a single researcher to be heard. Accordingly, we suggested to chose a middle-sized project: five to fifty developers of whom at least five have been active during the last few months.

As a last project property, we believe it useful to target a project that has shown an affinity for change (or at least no opposition to it) in the past. In many cases this property will correlate with the openness of the project to accept new members, but it is still beneficial to study the history of in-

ventions adopted by the project; a typical example might be the transition from the CVS version control system to the newer and clearly superior SVN.

To acquire a project somewhat randomly yet within the limitations given above, a project news announcement site like Freshmeat, which aggregates projects independently of their hosting, or a project listing site like SWiK can be used. Both of these example sites offer the option to visit a project at random from the listing. While SWiK shows all projects that relate to Open Source, Freshmeat's notable limitation is its requirement for projects to run under an Open Source operating system; purely Windows-based OSS projects are not listed.

4 How to approach Open Source projects?

Some knowledge exists in the literature about how to approach an OSS project [10, 26]. Firstly, the concept of "gift culture"[21] suggests that the respect for the external participant and influence s/he carries are correlated to his/her contribution to the project. This raises the question whether the invention itself will be seen as a gift if disseminated to the project. A case study on the effects of offering a source code gift that requires further effort to integrate into the code-base of the project appears to indicate the following: Unless the gift is directly useful for the project and immediately comprehensible to the participants, chances are low that it will be accepted [20]. Thus, we hypothesize that the researcher should expect to spend a considerable amount of work generating these benefits until the invention is accepted and adopted.

Secondly, the researcher needs to decide whether to approach the project by contacting the maintainer and project leaders, individual developers, or by addressing the project community as a whole. Our working hypothesis is that the type of approach should be correlated closely with (a) the degree of independence of each member's adoption decision, and (b) the benefit structure of the invention. We will now explain these factors.

In *Diffusion of Innovations*, Rogers distinguishes three *types of innovation-decisions*: *optional innovation-decisions*, which each member of the project can make individually and independently, *collective innovation-decisions*, which require consensus within the project, and *authority innovation-decisions*, which are made by a small influential group within the project [22].

As an example, consider the adoption of a practice such as "mandatory peer review before committing patches to version control". Such an improvement usually starts as a collective innovation-decision to improve code quality, since a general consensus is needed that every member of the project will submit his or her patch first to a mailing-list for inspection, and thus the whole community should be addressed to promote the adoption. Additionally, it also involves an optional innovation-decision by each member to participate in the review of patches sent by others, and thus can be supported by the researcher by talking to individual developers. As an example of the third kind of innovation-decision and its implications for how to approach the project, consider the introduction of a feature freeze[1] two weeks prior to a release. This decision can be driven by the project leaders and maintainers in an authoritative fashion and supported technically by creating a local branch for the release in the version control system. Individual members can undermine the decision, but they need not take specific action to make it a reality. Thus, the researcher should communicate directly to the project leaders.

The second important property of the invention that affects the approach is the *benefit structure* of the invention offered by the researcher, i.e., the return on investment or relative advantage [22] for each project member in contrast to the return on investment for the whole project. The documentation of the project, for instance, does not provide a high return on time spent for the experienced developer who writes it, yet the information is highly useful for new developers (where they might provide large returns for the project). Inventors often understand the *increasing returns* [1] promised by their invention but tend to overlook that (a) individual project members driving the introduction might not benefit from the improvement sufficiently to compensate for the effort *they* spend on it and (b) the benefits might be hard to measure or only visible in the long-run.

We hypothesize that the researcher should start the approach with those project members who can gain immediate benefits. Instead of asking other project members to perform tasks with a negative bottom line in terms of their personal benefit, those tasks should be performed by the researcher initially. Later on, when the benefits become visible and affect individuals in the project, the researcher will have a much better chance to involve project members and withdraw from these activities.

5 How to interpret reactions and make strategic and tactical decisions?

When introducing inventions and novelties of any kind into a social system, the researcher should expect *rejection*, *adoption*, and *reinvention* as ultimate reactions to occur both on the individual and group level [22].

Rejection is the decision not to adopt an innovation. It might occur both actively, i.e. after considering the adoption or even conducting a trial, or passively, i.e. without any consideration at all [22]. Passive rejection, i.e. not getting

[1] In a software release process, a feature freeze is the point from which onwards no new features must be introduced; only defect corrections and documentation are allowed to be performed.

a response at all, is not uncommon even if the researcher explicitly expresses interest in joining the project [26].

Reinvention occurs if members of the project take up the invention and recast or reuse it in unexpected and unintended ways. Reinventions might prove highly beneficial for the researcher, as they may point to new fields of application for the invention.

Of course, there is still a lot of room for interaction between the project member, researcher and technology until these ultimate reactions are made. Social science literature provides various models for such discourse such as the theory of fields [12] or network-actor theory [17]. We have chosen to follow the innovation model developed by Denning and Dunham [7]. In this view, the innovation process starts with (1) the sensing of possibilities for change and (2) a vision of what might result from the change. (3) Offering this vision to the affected people (or other units of adoption) and receiving their feedback allows the idea to be shaped into something that can be (4) executed and implemented in concrete terms resulting in a product, process or social improvement. It is only after the invention has been (5) adopted by the desired target population and (6) sustained as a successful novelty that a successful introduction of innovation has occurred. In the setting discussed here, the first two stages will more focus on the tailoring of the existing problem, vision and invention rather than the generation of new ideas and implementation.

6 How and when to phase out involvement and leave?

Our current working hypothesis is that the researcher can leave a project when s/he has successfully established the innovation as self-sustaining, or if the adoption has failed and no clean-up work remains to be done. In successful cases, withdrawal from the project should be gradual rather than abrupt or it may endanger the success and cause harm to the project. Leaving a project after a failed introduction on the other hand obliges the researcher to clean up, say, revert changes to the code-base or reinstate previous infrastructure before a (gradual) withdrawal is in order.

7 How to obtain evaluation results?

The actual evaluation of the invention under investigation is highly dependent on the nature of the invention itself and on the particular evaluation research goal. For some inventions the successful adoption itself can be a sufficient success, while others can only be judged by comparing product, process, or usage metrics to their baseline values prior to introduction. A third kind of invention might require the developers to be surveyed about their experience with the new technology.

Independent of these three basic approaches, the researcher will probably gain the most practical, albeit qualitative, insights for improving and assessing the invention by communicating with the project during the introduction. A researcher using the action research perspective may view this as the primary result.

8 Chances, limitations and conclusion

In the end, the question remains whether the experiences gained with introducing software engineering inventions in OSS projects can be applied to other settings (*external validity*). These might include differences in project sizes, application domains, software architectures, non-volunteer personnel, management, distribution and work-place setting, prior experience with software engineering methods, etc. The most common target setting is a revenue-dependent corporate environment. The following arguments argue why evaluation results from OSS projects may transfer to such environments: 1) Open Source developers are notorious for being critical of academic results, (2) availability of management championship and extrinsic motivations (like pay) can often spur adoption and use, and (3) full-time employees will benefit more from economies of scale and learning effects than part-time OSS developers.

The most notable limiting factor of our research approach is the restriction on the type of invention feasible for investigation. The diffusion of innovation literature lists several attributes of invention that will affect their rate of success for being introduced: (1) The *compatibility* of the invention with existing technology, values, and beliefs[2], (2) the intellectual and technical *complexity*, (3) the *observability* of the resulting effects of the invention, (4) the possibility to experiment with the invention (*trialability*) before committing to it, and (5) the *uncertainty* about the invention [22]. Halloran and Scherlis hypothesize more specifically with regards to OSS projects that these tend to distinguish sharply between trusted and untrusted contributions ("walled server" metaphor) and that inventions need to preserve this distinction to be applicable to OSS projects [15]). This limits the approach as follow: while successful introduction suggests a valuable invention, failed introduction may be the result of specific properties of the OSS project (such as the walled-server) and may not say much about the real qualities of the invention.

As a second limitation we note that in contrast to fieldwork and ethnographic studies conducted with companies (see for instance [18]), it will be difficult to study the actual working processes and practices of each project participant since only the intermediates and process results, say, bug reports, CVS commits, and mailing list discussions are visible

[2]For instance, OSS projects may reject tools that are not licensed as Open Source software themselves.

to the researcher. To gather information about the actual usage of tools on the computers of the project members, these need to be instrumented appropriately [23, 16].

A third limitation of the approach concerns the speed at which adoption can occur. Open Source projects are to a large extent driven by volunteers who invest less than 10 hours per week and coordinate using asynchronous electronic means over different time zones [14]. The time scale of change should thus be expected to be much slower than in a commercial setting where employees work regular working hours and frequently interact synchronously.

Summing up, we have proposed to study the introduction of software engineering inventions to help researchers evaluate tools, methods, and processes developed in academic settings, and have offered our preliminary results. While the research community can benefit from access to real life settings and the possibility to "feed back the community", the Open Source community is introduced to state-of-the-art inventions tailored to their specific problems by the inventors.

References

[1] W. B. Arthur. *Increasing Returns and Path Dependence in the Economy*. University of Michigan Press, 1994.

[2] D. E. Avison, F. Lau, M. D. Myers, and P. A. Nielsen. Action research. *Commun. ACM*, 42(1):94–97, 1999.

[3] M. Bakardjieva and A. Feenberg. Involving the virtual subject. *Ethics and Information Technology*, 2(4):233–240, 2001.

[4] J. Cassell. Ethical principles for conducting fieldwork. *American Anthropologist*, 82(1):28–41, March 1980.

[5] CMMI Product Team. Cmmi for development, version 1.2. Technical Report CMU/SEI-2006-TR-008, Software Engineering Institute, 2006.

[6] J. M. Corbin and A. Strauss. Grounded theory research: Procedures, canons, and evaluative criteria. *Qualitative Sociology*, 13(1):3–21, Mar. 1990.

[7] P. J. Denning and R. Dunham. Innovation as language action. *Commun. ACM*, 49(5):47–52, 2006.

[8] S. Dietze. *Modell und Optimierungsansatz für Open Source Softwareentwicklungsprozesse*. Doktorarbeit, Universität Potsdam, 2004.

[9] G. W. Downs and L. B. Mohr. Conceptual issues in study of innovation. *Administrative Science Quarterly*, 21(4):700–714, 1976.

[10] N. Ducheneaut. Socialization in an open source software community: A socio-technical analysis. *Computer Supported Cooperative Work (CSCW)*, V14(4):323–368, Aug. 2005.

[11] M. L. Fennell. Synergy, influence, and information in the adoption of administrative innovations. *Academy Of Management Journal*, 27(1):113–129, 1984.

[12] N. Fligstein. Social skill and the theory of fields. *Sociological Theory*, 19(2):105–125, July 2001.

[13] M. S. Frankel and S. Siang. Ethical and legal aspects of human subjects research on the internet. Published by AAAS online , June 1999.

[14] R. A. Ghosh, B. Krieger, R. Glott, G. Robles, and T. Wichmann. Free/Libre and Open Source Software: Survey and Study – FLOSS. Final Report, International Institute of Infonomics University of Maastricht, The Netherlands; Berlecon Research GmbH Berlin, Germany, June 2002.

[15] T. J. Halloran and W. L. Scherlis. High Quality and Open Source Software Practices. In J. Feller, B. Fitzgerald, F. Hecker, S. Hissam, K. Lakhani, and A. van der Hoek, editors, *Meeting Challenges and Surviving Success: The 2nd Workshop on Open Source Software Engineering*, pages 26 – 28. ACM, 2002.

[16] P. M. Johnson, H. Kou, J. Agustin, C. Chan, C. Moore, J. Miglani, S. Zhen, and W. E. J. Doane. Beyond the personal software process: metrics collection and analysis for the differently disciplined. In *ICSE '03: Proceedings of the 25th International Conference on Software Engineering*, pages 641–646, Washington, DC, USA, 2003. IEEE Computer Society.

[17] J. Law. Notes on the theory of the actor-network: Ordering, strategy and heterogeneity. *Systems Practice*, 5(4):379–393, 1992.

[18] T. C. Lethbridge and J. Singer. Experiences conducting studies of the work practices of software engineers. In H. Erdogmus and O. Tanir, editors, *Advances in Software Engineering: Comprehension, Evaluation, and Evolution*, pages 53–76. Springer, 2001.

[19] D. E. Perry, A. A. Porter, and L. G. Votta. Empirical studies of software engineering: a roadmap. In *Proceedings of the conference on The future of Software engineering*, pages 345–355. ACM Press, 2000.

[20] L. Quintela García. Die Kontaktaufnahme mit Open Source Software-Projekten. Eine Fallstudie. Bachelor thesis, Freie Universität Berlin, 2006.

[21] E. S. Raymond. *The Cathedral and the Bazaar*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 1999.

[22] E. M. Rogers. *Diffusion of Innovations*. Free Press, New York, 5th edition, August 2003.

[23] F. Schlesinger and S. Jekutsch. ElectroCodeoGram: An environment for studying programming. In *Workshop on Ethnographies of Code*, Infolab21, Lancaster University, UK, March 2006.

[24] D. Stelzer and W. Mellis. Success factors of organizational change in software process improvement. *Software Process: Improvement and Practice*, 4(4):227–250, 1998.

[25] W. F. Tichy, P. Lukowicz, L. Prechelt, and E. A. Heinz. Experimental evaluation in computer science: A quantitative study. *Journal of Systems and Software*, 28(1):9–18, Jan. 1995.

[26] G. von Krogh, S. Spaeth, and K. Lakhani. Community, joining, and specialization in open source software innovation: A case study. *Research Policy*, 32:1217–1241(25), July 2003.

[27] R. K. Yin. *Case Study Research: Design and Methods*. Applied Social Research Methods. Sage Publications, Inc., 1988.

[28] M. V. Zelkowitz and D. R. Wallace. Experimental models for validating technology. *Computer*, 31(5):23–31, 1998.