HAL
archives-ouvertes.fr

Combining FOSS and Kanban: An Action Research

Annemarie Harzl

▶ To cite this version:

Annemarie Harzl. Combining FOSS and Kanban: An Action Research. 12th IFIP International Conference on Open Source Systems (OSS), May 2016, Gothenburg, Sweden. pp.71-84, 10.1007/978-3-319-39225-7_6 . hal-01369052

HAL Id: hal-01369052
https://hal.inria.fr/hal-01369052

Submitted on 20 Sep 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution| 4.0 International License

Combining FOSS and Kanban: An Action Research

Annemarie Harzl[1]

Institute for Software Technology Inffeldgasse 16b/II, 8010 Graz
aharzl@ist.tugraz.at

Abstract. Even though Free and Open Source Software (FOSS) and Agile Software Development (ASD) have been recognized as important ways to develop software, share some similarities, and have many success stories, there is a lack of research regarding the comprehensive integration of both practices. This study attempts to consolidate these methods and to answer if FOSS and ASD can be combined successfully. Action Reseach (AR) is conducted with one sub-team of a large FOSS project. We performed two action research cycles based on the Kanban method. This paper has two main contributions; first, it describes a real world situation, where Kanban is applied to a FOSS project, and second, it suggests two new Kanban practices. These two methods are targeted specifically at FOSS projects and their characteristics.

Key words: Free Open Source Software ·Agile Software Development ·Lean ·Kanban ·Action Research

1 Introduction

This paper examines the use of Agile Software Development (ASD), namely the Kanban method [3, 19], in the context of Free and Open Source Software (FOSS) development.

Since the publication of the agile manifesto in 2001 a large body of research has been published on agile methods [9], e.g. on adopting agile methods [4] and implementing agile methods in distributed settings [5, 22]. FOSS and ASD have become common software development processes over the last fifteen years [9, 6] and some studies have been done about combining FOSS and ASD [12].

Although Warsta and Abrahamsson [26] and Koch [18] already showed in 2003 and 2004 that FOSS development and the definition of ASD methods are rather close, research about agile development in the context of open source software was still identified as a future research area in 2009 [1]. In 2013 Gandomani et al. [12] conducted a systematic literature review on relationships between ASD and FOSS development. They concluded that the examined studies indicated that ASD can support FOSS development, mainly because they share several concepts and principles. However, the authors did not find a case study successfully integrating both methodologies. This research tries to comprehensively combine these two for a specific real world case. By doing this we attempt

to answer the following research question: "Can FOSS and ASD be comprehensively combined?". This paper describes two action research cycles and proposes two FOSS-related additions to the Kanban practices from the perspective of a real project, therefore making it scientifically and practically relevant.

2 Related Work

Even though the research fields of FOSS and ASD have been of interest for over a decade with a large number of publications, there is to the best of our knowledge no work on comprehensively integrating the two methods. This is supported by the findings of Gandomani et. al. [12]. Most studies show only collaboration between the two and most of the time only specific practices are applied to a FOSS project [7, 10]. Other studies use Kanban or FOSS for teaching computer science classes [2, 21], but never together. Koch [18] compares the methods based on some criteria, but the paper does not include a practical implementation.

3 Background

This section describes some details about the FOSS project under study, my role as a researcher within the project, and reasons for integrating Kanban into a FOSS project.

3.1 The FOSS Project

The project under study is a hybrid student FOSS project. Within the scope of this umbrella project, various teams develop mobile applications for different platforms and purposes. The software development method follows an agile approach with elements of eXtreme Programming (XP) and Kanban. Most of the contributors are participating in the project in the context of their studies (e.g. Bachelor thesis or Master project) and stay with the project between six months and approximately two years (with breaks between the Bachelor thesis and Master project). Unlike usual FOSS projects, where a small number of people develops the majority of the code [15], contributions to this project are more evenly spread. Developers in this project are constantly changing and there are no core developers, who stay with the project for multiple years. Students are often working off-campus and our international contributors are of course working all over the world. Individuals are not assigned to a team, everyone self-selects the team and topic he or she is willing to work on. There are only three distinct roles: contributors, seniors, who are allowed to accept code of others into the main repository and the team coordinator, who is the main contact person for other teams and has a good overview of the team's status. The team coordinator volunteers for this position and the team decides,

who is suitable for this position. There exists no central management, only the project head, who is mainly responsible for the overall orientation of the umbrella project and the sub-projects, and one person, who is responsible for organizational activities, e.g. managing accounts and infrastructure. Software development skills of the contributors range from beginner to intermediate and knowledge about agile methods ranges from very little to moderate.

Interesting to note is that this project was not started by "scratching a developer's personal itch" [23], but by a university professor with an idea. The developers of the software are not the target group of the developed software. To foster the understanding of user needs a Usability and User Experience (UX) team works alongside the development teams.

3.2 My Role as a Researcher within the Project

The study is designed as insider in collaboration with other insiders, but power relations may play a part. In the FOSS project under study I am responsible for organizational and supporting processes, for example creating user accounts and giving a short introduction into the overall project. I am neither programming with other contributors nor am I leading one of the programming teams. However, I am on good terms with the project head and founder, whose word has a lot of bearing in the whole community. He is also the professor grading the students, who do their Bachelor thesis or Master project in the FOSS project. Thus, although I am not in an official hierarchical position, contributors probably see me as someone with informal power within the organization.

3.3 Justification of Kanban

The Kanban method [3, 19] is about evolutionary change and strives to establish a culture of continuous improvement (kaizen). It consists of four principles: *start where you are; pursue incremental, evolutionary change; respect the current processes, roles, responsibilities and titles; promote leadership at all levels* and six practices: *visualize the workflow; make policies explicit; manage flow; limit Work In Progress (WIP); implement feedback loops; improve collaboratively, evolve experimentally (using models and the scientific method)*. It was mainly chosen for the following three reasons.

First of all, in 2014 Ahmad et al. [2] concluded that Kanban appears to be a good pedagogical tool and useful for teaching inexperienced software developers about software engineering. It appears to have a short learning curve and a low adoption treshold. It further helps students to improve their team work skills, for example communication and collaboration. The project under study is not only a FOSS project, with many inexperienced developers, it serves teaching purposes as well, making Kanban an appropriate and light-weight approach for teaching and FOSS development. Kanban fosters collaboration and keeps the entrance barrier of the project as low as possible, so potential contributors are not scared off.

Another reason was that Kanban is the most adaptive method [17]. It allows for small evolutionary changes, does not require week-long expensive trainings and job titles and responsibilities do not have to be altered. Moreover, small incremental changes do not require positional power, which is not available in a FOSS community anyway. People affected by these changes are not enduring them, but are involved in the process and their participation is an integral part. People are allowed and encouraged to use their own mind. Thus, Kanban is also a good fit for Action Reseach (AR) as research methodology, which is discussed in more detail in Section 4.

Last but not least, projects within the umbrella project already used a Kanban-like board. By choosing the Kanban method, the team would be able to keep existing tools and the initial alterations would not be overwhelming.

4 Research Methodology

4.1 Justification of Action Research

To accomplish not only scientific but also practical outcomes was an important part of the motivation to conduct this study. According to Dick [8] AR is well suited for this type of goal, thus, we chose AR as the research method. Researchers and practitioners collaborate to solve real world problems through theoretically informed actions [13]. To achieve outcomes, people affected by those actions have to commit to them. One way to ensure this commitment is through involvement [8], for which AR offers various participatory methods [14]. Another reason for choosing this methodology was that only little research has been done on the integration of ASD and FOSS development [12], suggesting that the theory about this matter is not fully developed. Therefore, according to Edmondson, McManus and Kampenes et.al. [11, 16], a flexible approach, like AR, would be appropriate.

4.2 Case Selection

The project in question was selected due to the following reasons:

- Personal contact: Direct personal contact to the people involved allows for more detailed observations on group interaction than analysis of e.g. mailing lists. Trust, which is needed to change a work process, is easier established through personal than written contact. Furthermore, it is easier to receive feedback on multiple levels and to refine the research methodology and researcher skills through personal contact.
- Experiments and evaluation: Although the setting with mainly student contributors is rather unusual, students work on many FOSS projects and are not atypical FOSS contributors. Moreover students are often used to research, thus more used to experimenting with different approaches and willing to evaluate them. Other contributors may be more reluctant to do so.

– Time and access: A basic trusting relationship to project members was already established, so the bonding period with the community, which could take a very long time, could be minimized and allowed to conduct the AR within a reasonable time frame. Topics can be discussed in a shorter period of time and one has access to various artefacts, e.g. whiteboards or flipcharts.

While this project should of course not remain the only case studied, in our opinion it is a good starting point to explore Kanban in the context of FOSS projects. We will elaborate on the limitations in Section 6.2.

The sub-project was co-selected by the participants of the AR. My supporting role (as described in Section 3.2) within the project may have led one team coordinator to ask me for help. The team experienced problems with motivation and their workflow and the team members did not know how to overcome these issues on their own. Therefore, one goal of this research is to achieve practical outcomes, which improve the working situation of the team. This bias for action contributed to the selection of the research methodology.

Asking for help shows some commitment, which is usually needed to achieve action outcomes. Knowing this, we decided to conduct the study with this team. Other factors for the decision were: the team (six to eight people, varying over time) has roughly the average size of teams in this FOSS project (six to twelve team members at the same time), it uses the same agile workflow as the other teams and direct personal contact with the members of the team is possible.

The sub-project develops a mobile application targeted at teenagers, which should enable its users to create small projects without prior domain knowledge. The application has not been released to the public, it is only tested by members of the FOSS project. From the beginning the team used elements of XP and a Kanban board, like all other sub-projects. The applied XP practices included *automated unit tests, pair programming, refactoring, release planning* (occurs in irregular intervals), *short releases, continuous integration, coding standards, collective code ownership simple design and regular meetings* (weekly). *Visualize the workflow* was the only Kanban practice applied, in the form of an agile board, but members did not know, that this was a Kanban practice.

4.3 Action Research Cycles

A modified version of Susman and Evered's approach [25] was used as a research method. The cyclical model contains the five stages *diagnosing, action planning, action taking, evaluating*, and *specified learning*. One diagnosing phase and two regular AR cycles consisting of action planning, action taking, evaluation and specified learning were realized. An additional cycle zero was added after an observation phase at the beginning of the study. A participatory AR approach was used, all steps were discussed with the study participants and decided jointly.

As already described in Subsection 3.1 the sub-project uses elements of XP and Kanban. However, an initial questionnaire showed, that team members

assess their knowledge about both methods quite differently. While all members think that they have average to very good knowledge about XP only 17% think that they have very good knowledge about Kanban. The other 83% think that they possess little to no knowledge about Kanban.

This supported my decision to conduct a Kanban coaching session at the beginning of cycle zero and the second cycle. In these sessions I talked about Kanban in general, its principles and practices, and explained terms like flow and kaizen. These sessions were based on two books [3, 19] and one video[1]. It was necessary to provide the participants with some theoretical background about Kanban so they could understand its principles and practices. Furthermore, it was also important to enable them to decide how to integrate Kanban practices into their workflow. Due to observations made during the diagnosing phase, I conducted a cycle zero, including a user analysis and a stakeholder analysis. The first AR cycle was designed to introduce two practices into the team, namely *visualize (the workflow)* and *make policies explicit*. The second AR cycle should then familiarize the team with the principles *limit WIP* and *manage flow*.

4.4 Data Sources

Various types of data sources were used as empirical basis: a questionnaire about the participants knowledge on agile practices, weekly notes from team meetings written by the participants, my notes taken during meetings and discussions, the artefacts produced as part of the user and stakeholder analysis, and the team's Kanban board. Six to eight people participated in the study. Two people joined the team after cycle zero and one person left the team after the second cycle was completed. I had no position in the team but as already discussed in Subsection 3.2 my role within the umbrella project may result in research bias. I made some suggestions and most of the time the research participants accepted them.

4.5 Data Analysis

Firstly, the questionnaire regarding the agile knowledge of the team was statistically analyzed. Secondly, all researcher notes taken during meetings and discussions, were examined for issues of interest to the research and recurring topics or problems. For this purpose statements were "coded" and grouped together, if they had a theme or problem in common. Thirdly, the results of the team's user analysis and information about the sub-project's target group, retrieved from the project head, were compared. For this purpose statements from the team and the project head were compared one by one and discrepancies identified. Finally, to accomplish a better visualization of the team's workflow, the Kanban board was analyzed for its adherence to Kanban principles and practices, e.g. pull instead of push and limiting work in progress.

[1] `https://youtu.be/6nOUa6E0250`

5 Action Research

This section explains cycle zero, the two AR cycles and their phases as described in Subsection 4.3.

5.1 Diagnosing

The sub-project of the FOSS project was inspired by a programming exercise done during a university programming course in 2012. The results showed some promising ideas for a new application, which would fit nicely into the portfolio of the umbrella project. However, the code was neither finished (many functions were only rudimentary implemented) nor was it as structured and neat as it should be, because the course was part of a Bachelor study program and lasted only one semester. Thus, only the ideas remained and the code had to be rewritten.

Some of the students of the programming course decided to do their Bachelor thesis within the scope of the FOSS project. Together with some other students they started to develop the software anew. After a while it became apparent, that the team had been too ambitious and had ignored agile principles, such as working in small iterations and implementing the simplest thing that could possibly work. The team tried to implement too many features in parallel and was overwhelmed by the amount of work necessary to finish it. This situation became even worse when some individuals decided not to use the XP practice pair programming and accomplished most of their work alone, leaving the rest of the team clueless about their contribution to the code. As a result the team ended up with a heap of unfinished code and after around a year the sub-project came to a halt. The team decided to start over again.

The second attempt to restart was not successful either. The application did not meet its usability goals and was abandoned again.

By this time the team members' motivation was understandably very low. They struggled with the code, their workflow and their team spirit. Disillusioned by the failed attempts the team coordinator of this sub-project thought that the team could not solve their problems all by themselves and asked me for help.

The main problems of this team seemed to be: underestimation of the tasks at hand, insufficient adoption of agile practices and the overall workflow resulting in frustration and lack of motivation.

5.2 Cycle Zero: Get to know Kanban, your users and your stakeholders

Before starting the AR cycles I regularly attended the weekly team meetings. I observed the interactions in the team and of course got to know the team members. By attending these meetings I got the impression that the team was targeting a different main user group than the project head envisioned. I had of

course previously talked with him about the sub-project and its target group. As far as I knew the project head wanted to target beginners in the domain and it seemed to me that the team was targeting people with at least intermediate knowledge in the domain. This assumption led me to add a cycle zero to the study. I wanted to clarify, if there was indeed a misunderstanding between the team and the project head, therefore I moderated a user analysis with the team. This user analysis yielded some unforeseen results and led to a repositioning of the sub-project within the umbrella project with some major changes for the team and its interaction with other teams. Therefore, I performed a stakeholder analysis with the team as well.

Action Planning and Action Taking The user analysis was done as a simple brainstorming exercise, where we collected all possible user groups team members could think of on a whiteboard. Afterwards, the team chose the main user groups for which they were developing the application.

The stakeholder analysis was conducted according to Leopold and Kaltenecker [19]. First, the team determined all stakeholders and listed them on individual pieces of paper. Each paper was sized differently, reflecting the importance of the stakeholder for the team's long-term success. Then, the pieces were put on a table and arranged around the team's mission, which is at the center of the analysis. Stakeholders, who are affected more by the team's day to day work and possible changes, are placed nearer to the center. Stakeholders, who are affected less, are placed further away. Afterwards, the frequency of relationships between all stakeholders was determined. The stronger the relationship, the more lines were drawn between two stakeholders. At last the quality of these relationships was determined as *friendly, adversarial, love-hate* or *unknown*. For an exemplary stakeholder analysis see Figure 1.

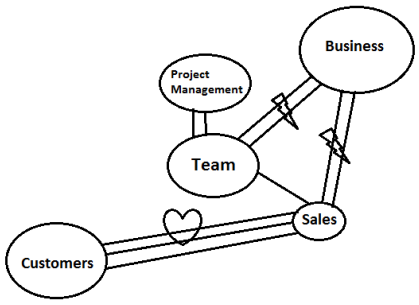

Fig. 1. Exemplary stakeholder analysis, adapted from Leopold and Kaltenecker [19]

Evaluating and Specified Learning The user analysis showed some discrepancy between the main user group the team was targeting and the main user group the project head envisioned. I articulated my impression to the team

and we concluded that the team had to talk to the project head. Luckily for us he was available at the time of the user analysis and we asked him to join the discussion. And indeed there was a misunderstanding. He explained to the team which user group they should target and why and it became obvious, that the team had been developing their application for a different target group. This was a very unexpected result for the team and the whole software development came to a halt. The team had to redefine their goals and to rethink their application. Two meetings with the project head followed and the UX team was contacted as well. The project head and the team discussed the goals for their sub-project. The UX team did a user inquiry with four members of the targeted user group. The results of the user inquiry were discussed with the developing team. Based on the input of the four possible users, the developing team and the project head, the UX team designed a digital mockup of the application. This mockup showed an absolutely different Graphical User Interface (GUI) than the current application. It was more intuitive for novice users, but more elaborate to develop. Thus, the team had to reimplement the whole GUI. More importantly the former independent stand-alone application was integrated into another application of the umbrella project. This decision was made jointly by the sub-team, the super-team and the project head. The main reasons for this decision were: the sub-project will extend the functionality of the super-application with some very important features. The sub-project will reach more users as an extension than as a stand-alone application because the super-application is already publicly available and has a steadily growing user base.

As a result of this repositioning from stand-alone application to extension and because it is recommended in Leopold and Kaltenecker [19], I performed a stakeholder analysis with the team. The analysis showed that the intensity and quality of some relationships between stakeholders were unknown to the team and that the team had to work on intensifying some relationships, especially the one with the new super-team.

5.3 First Cycle

Action Planning and Action Taking The team already used a Jira Kanban board to visualize the workflow. The board contained the following columns: *backlog, in development, done, done and accepted.*

Team policies were determined in an open discussion. Team members collected all policies on a white board and discussed their meaning and importance jointly. After agreeing on a set of policies, they were transferred to the project wiki.

Evaluating and Specified Learning The main focus in this cycle was *making policies explicit.* We regarded *visualize the workflow* as already finished, because the team already used a board. Additionally, as a consequence of cycle zero the software development was put on hold and the team focused on redefining their goals, hence there was no activity on the board at the time.

Making policies explicit yielded some interesting effects: Team members discussed their unspoken policies for the first time and discrepancies showed. Some were of semantic nature others reflected a different view of processes. During the discussion some problematic habits were identified and immediately discussed. To prevent these habits from reoccuring, new team policies were stipulated jointly and team members agreed to honor those policies.

5.4 Second Cycle

Action Planning and Action Taking *Limiting work in process* and *managing flow* were introduced to the team during the second Kanban coaching session. Different visualization possibilities were shown and the importance of *limiting WIP* and its effect on transition time, based on Little's Law [20], were explained. It was also discussed how WIP limits could be used to make problems visible and improve flow.

Evaluating and Specified Learning We revisited the Jira board to limit WIP. In the course of doing so we discovered something we already should have uncovered during the first cycle. The current board did not model a pull system. It was rather a push system. While team members pulled tasks from the backlog into development, the transition from development to code acceptance was a push process. Developers pushed the task from development to done where senior developers had to take them and move them to *done and accepted*. There was not even a state for being *in acceptance*. It was not possible to determine whether a task was already in the process of being accepted, or if it was still simply marked as *done*. I think this oversight can be ascribed to the focus on redefining the goal, which was still in progress during the first cycle.

To repair this, a new state was introduced into the workflow and the column *in acceptance* was added to the board, now reflecting a real pull system. The columns *in development* and *done* were merged into one column. Initially, the team wanted to create two subcolumns for *in development*, but Jira does not offer this functionality. Therefore we experimented with different possibilities of visualization. Figure 2 depicts on the one hand the desired visualization and on the other hand the current visualization, which is realizable within the constraints of Jira.

Afterwards, WIP limits were set and monitored. They soon revealed a bottleneck at the acceptance state. The root cause was quickly identified and team members were working hard to resolve this bottleneck. Due to the merge with the super-project the team now depended on the senior developers of the super-project to accept the sub-team's code. The senior members of the sub-team were and still are working hard to familiarize themselves with the slightly different acceptance process of the super-team and thus dissolving the bottleneck as soon as possible and improving flow. Another positive effect of the focus on the board is, the team now uses their Kanban board during each meeting, which it did not do prior to the AR cycles.

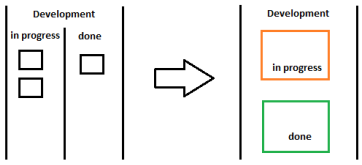

Fig. 2. The desired visualization on the left side and the actual visualization, which is possible in Jira, on the right side

6 Results and Discussion

While this study can not conclusively answer the research question "Can FOSS and ASD be comprehensively combined?" it shows some promising outcomes. Integrating Kanban and FOSS has so far been successful and beneficial for the research participants. New insights, e.g. on the target user group, have been gained and the team's workflow has become more effective. From my point of view, motivation has improved but this remains to be verified in the future.

6.1 Extending the Kanban practices

Based on the experiences described in Section 5, we propose two additional Kanban practices for FOSS projects.

- Conduct regular user interviews or feedback sessions with your users
- Review your assumptions about your current development practices

These recommendations are of course based on a hybrid student FOSS project, but usual FOSS projects could benefit from these additional practices as well. While most companies applying agile and lean practices have a marketing and sales team or even a user focus group, FOSS projects tend not to have this kind of resources. Nowadays, many FOSS solutions are employed by a large number of people, who do not contribute to the code, e.g. Mozilla Firefox or Linux. Thus, FOSS developers are not simply "scratching their own itch" anymore, they serve many people, who must not share the developer's requirements and domain knowledge, all over the world. Therefore, it could be beneficial for FOSS projects to investigate their users' needs. As for the second recommendation: Although, we agree with the Kanban principle "start where you are", we assume it could be beneficial for FOSS projects to review their current development practices before embarking on the endeavor of integrating Kanban, or any other agile or lean method, into their development process. FOSS projects usually do not have SCRUM masters, process experts or in general someone, who controls whether software development practices are exercised correctly. An honest and critical reflection about the current practices can clear some misconceptions, can further a joint understanding of the current situation and it is a first step into the direction of kaizen.

6.2 Limitations

This study is limited to a single case of a hybrid student FOSS project, therefore it has only very limited external validity. Only one team is part of this study therefore the results are not generalizable to other teams or projects without further research.

Another limitation may be my positionality [14] in the setting. Herr and Anderson [14] describe positionality as asking the question, "Who am I in relation to my participants and my setting?". Subsection 3.2 already identified power distance as a possible limitation. The informal power position may result in research bias, since suggestions I make could be accepted due to this perceived power distance and not only because team members agree with these suggestions. In an attempt to counterbalance this bias a bit, whenever possible at least two alternatives were proposed and the final decision was made by the team.

The setting of a hybrid student FOSS project might also be seen as a limiting factor, because some characteristics differ. Teams members contribute to receive course credits and not (only) to earn a reputation among other developers or out of altruistic motives. Student contributors change regularly and there exists no group of core or chief developers, which usually consists of 10% to 20% of a team, and which creates around 80% of the source code [18]. Future research could determine if or how these diverging characteristics impact a FOSS project. Students as main contributors to this project may also be considered a limitation, because they have not finished their studies. But people from different backgrounds contribute to FOSS projects regardless of their formal education and IT students work as normal developers on many FOSS projects. Many FOSS contributors have no formal software engineering education at all. Salman et al. [24] even observed that students and professionals show similar performances in carefully scoped software engineering experiments, when the development approach used is new to both groups.

7 Conclusion

This paper describes a practical integration of Kanban through AR in the context of a hybrid student FOSS project. Based on the findings of this work we proposed additional Kanban practices for FOSS projects. There is a lack of research regarding integration of ASD in the FOSS development context [12]. Thus, this paper contributes by offering some insights on the matter. As future work we plan on conducting more AR cycles with the same team to exercise the practices from the first and second cycle and further introduce the remaining Kanban practices into the workflow. To assure validity of the proposed additions to the Kanban approach, the cycles described in this paper should be conducted with other teams and within other FOSS settings.

References

1. Ågerfalk, P.J., Fitzgerald, B., Slaughter, S.: Introduction to the special issue - flexible and distributed information systems development: State of the art and research challenges. Information Systems Research 20(3), 317–328 (2009), http://dx.doi.org/10.1287/isre.1090.0244
2. Ahmad, M.O., Liukkunen, K., Markkula, J.: Student perceptions and attitudes towards the software factory as a learning environment
3. Anderson, D.: Kanban - Successful Evolutionary Change for Your Technology Business. Blue Hole Press (2010)
4. Boehm, B.W.: Get ready for agile methods, with care. IEEE Computer 35(1), 64–69 (2002), http://doi.ieeecomputersociety.org/10.1109/2.976920
5. Boland D., F.B.: Transitioning from a co-located to a globally-distributed software development team: A case study at analog devices, inc. In: Proceedings of 3rd Workshop on Global Software Development (2004)
6. Crowston, K., Wei, K., Howison, J., Wiggins, A.: Free/libre open-source software development: What we know and what we do not know. ACM Comput. Surv. 44(2), 7 (2012), http://doi.acm.org/10.1145/2089125.2089127
7. Deshpande, A., Riehle, D.: Continuous integration in open source software development. In: Russo, B., Damiani, E., Hissam, S.A., Lundell, B., Succi, G. (eds.) Open Source Development, Communities and Quality, IFIP 20th World Computer Congress, Working Group 2.3 on Open Source Software, OSS 2008, September 7-10, 2008, Milano, Italy. IFIP, vol. 275, pp. 273–280. Springer (2008), http://dx.doi.org/10.1007/978-0-387-09684-1_23
8. Dick, B.: A beginner's guide to action research (2000), http://www.aral.com.au/resources/guide.html
9. Dingsøyr, T., Nerur, S.P., Balijepally, V., Moe, N.B.: A decade of agile methodologies: Towards explaining agile software development. Journal of Systems and Software 85(6), 1213–1221 (2012), http://dx.doi.org/10.1016/j.jss.2012.02.033
10. Düring, B.: Sprint driven development: Agile methodologies in a distributed open source project (pypy). In: Abrahamsson, P., Marchesi, M., Succi, G. (eds.) Extreme Programming and Agile Processes in Software Engineering, 7th International Conference, XP 2006, Oulu, Finland, June 17-22, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4044, pp. 191–195. Springer (2006), http://dx.doi.org/10.1007/11774129_22
11. Edmondson A. C., M.S.E.: Methodological fit in management field research. Academy of Management Review 32, no. 4 (2007)
12. Gandomani, T.J., Zulzalil, H., Ghani, A.A.A., Sultan, A.B.M.: A systematic literature review on relationship between agile methods and open source software development methodology. CoRR abs/1302.2748 (2013), http://arxiv.org/abs/1302.2748
13. Greenwood D.J., L.M.: Introduction to Action Research: Social Research for Social Change. SAGE Publications (2007)
14. Herr K., A.G.: The Action Research Dissertation - A Guide for Students and Faculty 2nd Edition. SAGE (2015)
15. Kagdi, H.H., Hammad, M., Maletic, J.I.: Who can help me with this source code change? In: 24th IEEE International Conference on Software Maintenance (ICSM 2008), September 28 - October 4, 2008, Beijing, China. pp. 157–166. IEEE Computer Society (2008), http://dx.doi.org/10.1109/ICSM.2008.4658064

16. Kampenes V.B., Anda B., D.T.: Flexibility in research designs in empirical software engineering. In: Visaggio, G., Baldassarre, M.T., Linkman, S.G., Turner, M. (eds.) 12th International Conference on Evaluation and Assessment in Software Engineering, EASE 2008, University of Bari, Italy, 26-27 June 2008. Workshops in Computing, BCS (2008), http://ewic.bcs.org/content/ConWebDoc/19536
17. Kniberg H., S.M.: Kanban and Scrum - making the most of both. C4Media (2010)
18. Koch, S.: Agile principles and open source software development: A theoretical and empirical discussion. In: Eckstein, J., Baumeister, H. (eds.) Extreme Programming and Agile Processes in Software Engineering, 5th International Conference, XP 2004, Garmisch-Partenkirchen, Germany, June 6-10, 2004, Proceedings. Lecture Notes in Computer Science, vol. 3092, pp. 85–93. Springer (2004), http://dx.doi.org/10.1007/978-3-540-24853-8_10
19. Leopold K., K.S.: Kanban in der IT - Eine Kultur der kontinuierlichen Verbesserung schaffen. Hanser (2013)
20. Little, J., Graves, S.: Little's law. In: Chhajed, D., Lowe, T. (eds.) Building Intuition, International Series in Operations Research & Management Science, vol. 115, pp. 81–100. Springer US (2008), http://dx.doi.org/10.1007/978-0-387-73699-0_5
21. MacKellar B., Sabin M., T.A.: Bridging the academia-industry gap in software engineering: A client-oriented open source software projects course. In: Open Source Technology: Concepts, Methodologies, Tools, and Applications, chap. 99, pp. pages 1927–1950. IGI Global (2015)
22. Ramesh, B., Cao, L., Mohan, K., Xu, P.: Can distributed software development be agile? Commun. ACM 49(10), 41–46 (Oct 2006), http://doi.acm.org/10.1145/1164394.1164418
23. Raymond, E.S.: The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary. O'Reilly & Associates, Inc., Sebastopol, CA, USA (2001)
24. Salman, I., Misirli, A.T., Juzgado, N.J.: Are students representatives of professionals in software engineering experiments? In: 37th IEEE/ACM International Conference on Software Engineering, ICSE 2015, Florence, Italy, May 16-24, 2015, Volume 1. pp. 666–676. IEEE (2015), http://dx.doi.org/10.1109/ICSE.2015.82
25. Susman, G.I., Evered, R.D.: An Assessment of the Scientific Merits of Action Research. Administrative Science Quarterly 23(4), 582–603 (Dec 1978), http://dx.doi.org/10.2307/2392581
26. Warsta, J., Abrahamsson, P.: Is open source software development essentially an agile method? In: Proceedings of the 3rd Workshop on Open Source Software Engineering, 25th International Conference on Software Engineering. pp. 143–147. Portland, Oregon (2003)