Personal Extreme Programming: Exploring Developers' Adoption

IYAWA, Gloria Ejehiohen

Personal Extreme Programming: Exploring Developers' Adoption

Completed Research

Gloria Ejehiohen Iyawa
Sheffield Hallam University
gloria.iyawa@gmail.com

Abstract

Personal Extreme Programming (PXP), evolving from the popular agile methodology, Extreme Programming (XP) and Personal Software Process (PSP), has been suggested as a methodology for autonomous developers who aim to incorporate the principles of XP in the development of software applications. However, there is limited research exploring the adoption of PXP. The purpose of this paper was to explore developers' adoption of PXP. The constructs of Roger's (1962) Diffusion of Innovation (DoI) Theory and Technology Acceptance Model were used to investigate developers' adoption of PXP. Semi-structured interviews were conducted with developers who had experience in PXP practices. Although the findings revealed that PXP is beneficial in software development, the challenges of PXP in software projects include increased chances for errors and lack of skills diversity. Autonomous software developers should, however, evaluate their level of software development skills, experience and amount of work required when considering PXP as a software methodology.

Keywords

Personal extreme programming, adoption, developers, extreme programming, diffusion of innovation theory, factors, influencing, affecting, challenges, benefits.

Introduction

Agile methodologies have been applied in software projects that emphasise customer interaction in software development. An example is Extreme Programming (XP). XP is an agile software practice that focuses on a team of software developers interacting with customers at regular intervals (Dima & Maassen, 2018). The concept of XP has been explored in different settings; for example, XP has been explored in learning environments (Hedin, Bendix & Magnusson, 2005; Stapel, Lubke & Knauss, 2008; Murphy, Phung & Kaiser, 2008; Steghöfer et al. 2016). The concept of XP has also been explored in reducing rework in software development (Bin et al., 2003).

Iyawa, Herselman & Coleman (2016) explain that XP facilitates a deep level of interaction between customers and software developers. However, Agarwal & Umphress (2008) are of the opinion that in certain conditions, there are limited opportunities for developers to work as part of a team; hence, requiring the effort of a single developer in a project and in such cases, the application of XP practices is essential. Personal Extreme Programming (PXP) is the combination of modified XP and Personal Software Process (PSP) practices by autonomous developers in developing software applications (Agarwal & Umphress, 2008). The guidelines on how PXP should be implemented and a practical example of this approach was provided by Dzhurov et al. (2009).

While PXP has been introduced since the 2000s (Agarwal and Umphress, 2008; Dzhurov et al., 2009), there is a dearth of studies exploring PXP in practice. It is important to understand how PXP is being adopted by developers to help identify the benefits and challenges of adopting PXP in software practices. Furthermore, identifying the benefits and challenges of PXP practices could contribute to empirical evidence on the knowledge of PXP practice, factors influencing and inhibiting adoption, and provide developers with a direction on what is required when PXP is adopted as a methodology in software projects. The findings could also facilitate suggestions which could improve PXP practices.

The purpose of this paper was, therefore, to explore developers' adoption of PXP. The findings from this study could provide autonomous software developers, with the aim of applying PXP in software projects, a greater understanding of the factors influencing and affecting developers' adoption of PXP. This paper, thus, contributes to the body of knowledge on PXP practices.

The remainder of this paper is structured as follows: A review of related work and the theoretical framework guiding this study are presented. Next, the methodology and findings are presented. The next section presents a discussion of the findings. The paper concludes with a summary of the findings and recommendations for future work.

Theoretical Background

PXP, XP and PSP

Dzhurov et al. (2009) describe autonomous developers as individuals who develop software without the effort of other developers. Dzhurov et al. (2009) explain that the need for autonomous developers was as a result of the demand for Information Technology (IT) outsourcing and the need for lower cost of software delivery. Due to agile methodologies being flexible and popular (Altameem, 2015; Mohammed & Abushama, 2013), autonomous developers often aim to adapt agile methodologies in their projects. Dzhurov et al. (2009) suggest that software developers who aim to adopt agile methodologies such as XP should rather follow the PXP methodology as it is a better fit for the needs of such developers. According to Agarwal & Umphress (2008), PXP combines modified XP and PSP to improve the individual experience of the developer and the quality of software produced. Based on the descriptions provided in the literature, XP and PSP represent important components of PXP.

XP is one of several agile methodologies for developing software applications (Hneif & Ow, 2009). One of the reasons XP gained popularity is as a result of its flexibility regarding customer requirements (Qureshi & Ikram, 2015). Another reason why XP is widely supported is because it supports customer interaction in the development process which, in turn, facilitates constant feedback (Moniruzzaman & Hossain, 2013). Furthermore, Iyawa et al. (2016) found that it is less costly to adopt XP compared to traditional methodologies because there is less amount of rework involved when XP is applied. However, working with a team is not always feasible due to various reasons (Iyawa, 2016). Dzhurov et al. (2009) therefore explores PXP as a methodology relevant to autonomous developers.

PSP emphasises on individuals following a disciplined process in the development of software applications to improve the quality of software being developed (Pomeroy-Huff, Cannon, Chick, Mullaney & Nichols, 2009). Humphrey (2000) explains that each software engineer is unique and as a result, needs to develop a suitable plan that meets individual goals before work is carried out. Humphrey (2000) also states that individuals working as part of a team need to take responsibility in producing software which is of high standard.

While PSP focuses on improving the processes of individual work and XP focuses on improving interaction between team members and customers, the overall goal of both PSP and XP is to improve the overall quality of software products (Agarwal & Umphress, 2008). Thus, XP and PSP both represent a relevant approach for autonomous developers to attain quality in software development. For the purpose of this study, PXP can be defined as the extension of existing software practices such as XP and PSP to support autonomous developers when developing software applications.

Few studies have described the use of PXP in software projects. Asri et al. (2017) described the application of PXP in the development of a web application to support job training activities. It was emphasised that PXP was selected as the methodology for the project because it aimed at improving the quality of software projects while supporting autonomous developers. The study also provided evidence that the application of PXP in software projects could support the development of quality software within a short period of time. PXP has also been explored with other methodologies. For example, Marthasari et al. (2018) combined PXP with MoSCoW prioritisation in the development of a library information system. The reason for combining both methodologies was to rank requirements gathered based on certain priority metrics. However, the study revealed that combining PXP with MoSCoW prioritisation did not have any impact on the project completion timeframe.

The few studies that explored PXP as a methodology revealed that it improved the quality of software developed. For autonomous developers, PXP provides an approach to improve individual planning and at the same time improve the quality of software developed. PXP therefore presents a methodology to reduce the risk of software project failure while working as an autonomous developer.

Furthermore, Agarwal & Umphress (2008) highlight the differences between PXP and XP in terms of the twelve core practices. PXP allows autonomous developers complete tasks on their own rather than working as part of a team. For example, Agarwal & Umphress (2008) explain that while pair programming is used in XP, it is not feasible in PXP as autonomous developers work on their own; unlike the practice of collective code ownership in XP, in PXP, the source code is owned by autonomous developers. While every team member uses a common coding standard in XP, autonomous developers have the opportunity to choose their own coding standards. Hence, the major difference between PXP and XP is that PXP provides a personalised form of XP practices that involves modifying the twelve core XP practices to suit an autonomous developer.

Theoretical Models

While different theories have been used in technology adoption studies, the literature indicates that constructs of the Diffusion of Innovation (DoI) theory has been frequently used in examining the adoption of different technologies and innovation such as mobile payment (m-payment) systems (Al-Jabri and Sohail, 2012) and passive positioning alarms (Olsson et al., 2016). To the best of the researcher's knowledge, this is the first time the adoption of PXP is being studied. Constructs of the DoI theory include relative advantage, compatibility, complexity, trialability and observability (Rogers, 2003). These constructs are described as follows:

- Relative advantage refers to "the degree to which an innovation is perceived as being better than the idea it supersedes" (Rogers, 2003, p.229).
- Compatibility refers to "the degree to which an innovation is perceived as consistent with the existing values, past experiences, and needs of potential adopters" (Rogers, 2003, p.15).
- Complexity refers to "the degree to which an innovation is perceived as relatively difficult to understand and use" (Rogers, 2003, p.15).
- Observability refers to "the degree to which the results of an innovation are visible to others" (Rogers, 2003, p.16).
- Trialability refers to "the degree to which the innovation may be experimented with on a limited basis." (Rogers, 2003, p. 16).

Although the constructs of the DoI theory have been adopted in various studies (Olsson et al., 2016; Al-Jabri & Sohail, 2012), trialability will not be included in this study because PXP can be seen as methodological innovation rather than an artefact which needs to be tested before use on a limited basis, hence, not relevant to this study. Other adoption models include Technology Acceptance Model (TAM) developed by Davis et al. (1989). The model explores two key constructs that determines how users accept technology: perceived usefulness and perceived ease of use. Perceived usefulness is described as "the degree to which a person believes that using a particular system would enhance his or her job performance" (Davis, 1989, p.320). Perceived ease of use refers to the "the degree to which a person believes that using a particular system would be free of effort" (Davis, 1989, p. 320). Both constructs were included in this study.

Methodology

The target population for this study consisted of ten software developers who have applied PXP in software projects. The participants were purposively selected because they had experience in applying PXP principles in software development and understood the twelve core practices in XP. Purposive sampling was, therefore, relevant to this study as it aimed to include participants who are experienced in the knowledge area (Etikan, Musa & Alkassim, 2016). Names of participants were codified to Autonomous Developers (AD) AD1, AD2… AD10. Descriptions of the participants are provided below:

- AD1 is a recent Computer Science graduate and freelance software developer. AD1 has worked on three software development projects in which he applied the principles of PXP.
- AD2 works as a lecturer at a higher institution. AD2 also works on software projects in his spare time. AD2 has worked on two software projects in which he applied the principles of PXP.
- AD3 is a freelance software developer. AD3 has worked on four software projects in which he applied the principles of PXP.
- AD4 works in a software organisation, however, he works on private projects during his free time. AD4 has undertaken four independent software projects in which he applied the principles of PXP.
- AD5 is a full-time Masters student. AD5 works as a freelance software developer. Within the last six years, he has worked on seven projects in which he applied the principles of PXP.
- AD6 works a lecturer at a higher institution. AD6 also works as an autonomous developer in his spare time. He has worked on two projects in which he applied the principles of PXP.
- AD7 is a recent graduate. AD7 also works as an autonomous developer. He has also worked on two projects in which he applied the principles of PXP.
- AD8 works in a government organisation. AD8 works as an autonomous developer in her free time and she has worked on three software projects in which she applied the principles of PXP.
- AD9 is a full-time Masters student. AD9 also works as a freelance autonomous developer in his free time. He has worked on two software projects in which he applied the principles of PXP.
- AD10 is a recent graduate and a freelance software developer. AD10 has worked on three software projects in which he applied the principles of PXP.

Data Collection

In order to investigate the adoption of PXP in software projects among software developers, a qualitative method was applied. Hammarburg et al. (2016) further explain that qualitative methods can be applied in studies where the research aims to understand a concept. It was, therefore, important to apply the qualitative method in this study as the aim of the study was to investigate the adoption of PXP among developers. The study was conducted in Windhoek, Namibia.

Data was collected using semi-structured interviews. The interview was used to gain insights about the factors influencing and affecting the adoption of PXP by autonomous software developers. The interview questions were structured around the constructs of DoI and TAM to identify the relative advantage, compatibility, complexity, observability, perceived usefulness and perceived ease of use to identify the factors influencing and affecting developers' adoption of PXP. Prior to the interview, the interviewer explained the purpose of the study and had a discussion with the participants regarding their experiences and knowledge of XP and PXP practices. Participants were interviewed individually and each interview lasted approximately twenty minutes.

Data Analysis

The analysis aimed at understanding and interpreting what the participants' perceived as factors influencing and affecting the adoption of PXP based on the constructs (relative advantage, complexity, compatibility, observability, perceived usefulness and perceived ease of use). The interviews were recorded and later transcribed. Thematic analysis was used to code the text from the interviews and translated into themes (Braun, Clarke & Rance, 2014). The themes generated are as follows:

- Relative Advantage and Perceived Usefulness
    - Flexibility
    - Independence
    - Management
    - Comprehension of user stories
- Complexity
    - Increased chances for error

  - Lack of backups during emergencies
  - Not compatible with large projects
  - Lack of skills diversity
  - Time constraints for large projects
- Observability
  - Customer involvement
  - Software project management experience needed
  - Efficient for small projects
- Compatibility
  - Feedback
  - Incremental requirements gathering
- Perceived Ease of Use
  - Convenient when working on small projects
  - Task completion

Results

The findings of the study are categorised under five themes namely, relative advantage/perceived usefulness, complexity, observability, compatibility, and perceived ease of use.

Relative Advantage and Perceived Usefulness

Participants were asked to describe the benefits they experienced using PXP as a methodology for software development. The majority of the participants believed that applying the principles of PXP enabled them to fully focus on the job as there were fully aware that there would be no external support. Some of the participants indicated:

> *"It gives me the **flexibility** to focus on the job when necessary"* AD3
>
> *"With PXP, I am able to concentrate on the job as I am the only one involved in the requirements gathering and design phase, so it gives me the opportunity to focus since there is no other person to distract me"* AD2
>
> *"Applying PXP gave me the opportunity to concentrate on the tasks as I work solely on the development of the application"* AD6

Some of the participants who took part in the study pointed **independence** as one of the benefits of applying PXP practices in software development. Some participants explained:

> *"One thing I like most about using extreme programming as a single developer is that I am able to decide when it is time to work. I work with my own calendar"* AD10
>
> *"I was able to schedule when things would get done as I had no body to report to apart from the customer"* AD8

It was reported that one merit of adopting PXP is that developers can keep track and effectively **manage** the project because they are involved in carrying out all the activities in the project.

Some of the participants also indicated that the application of PXP made the software development process easier since core practices such as user stories and frequent feedback enabled them **comprehend** user needs faster. User stories is an approach that the user adopts in explaining the proposed components of a software (Chance, 2011).

Complexity

Participants were asked to describe the challenges they experienced when adopting PXP. An issue was raised regarding **not being able to participate in pair programming, hence increasing the chances for errors**. Pair programming is a process whereby two programmers work on a code, with one programmer writing the code and the other programmer reviewing the code being written (Maguire et al., 2014). Some of the participants indicated that they had to write and review their code themselves leaving chances for errors.

> *"The major challenge I had was having to work on my code myself and finding errors myself"* AD1
>
> *"Well, my problem with applying Extreme Programming in my individual work is that I don't get to code with other developers. The challenge is that there may be things which could be detected on time when there is someone looking at your code while you code"* AD7
>
> *"Unlike being in a team, as a single developer, there is nobody to discuss the progress with, as you are the only one who does the work"* AD9

Participants also indicated that it took them **longer time to produce any deliverable** compared to when they worked in groups. Some participants explained:

> *"I found that it takes a longer period to develop an application when I work on my own"* AD5
>
> *"Working alone means doing everything yourself, which could be time-consuming to produce any functional software component"* AD3

Another challenge found while adopting PXP practices as an autonomous developer as indicated by the participants is a possible **delay because of emergencies**. The participants added that it was difficult to meet deadlines when they experienced emergencies such as illness, as such, the work would be on hold until they were fit to complete the job.

Participants who took part in this study also indicated there was a **lack of skills diversity** when they worked alone. Some participants stated:

> *"Working as an independent developer meant that I had to depend on my skills alone and I couldn't learn from other developers"* AD7
>
> *"I noticed lack of diversity of skills in the development process, as I had to learn new things on my own, which took my time"* AD8

Other participants believed that PXP was **time-consuming when working on large projects** as they had to perform tasks which were meant to be distributed among different people. Some participants explained:

> *"Time, I believe, is a major drawback when using the principles of Extreme Programming in an individual project because you have to do the programming, interact with the customers, test the application and carry out other administrative functions…."* AD2
>
> *"It gets complicated when you have to so many tasks at the same time, it gets even more complicated when it's a large project which has to be completed within a short period of time, taking into consideration that you have to deal with customers in the process and make changes" AD 1*
>
> *"In my opinion, I wouldn't work alone when it's a big project even when I'm applying Personal Extreme Programming because it could be very demanding and would compromise on quality" AD 6*
>
> *"When the activities to be done are quite a lot, I would rather stick to working with three or four friends that are also developers rather than taking on huge projects I cannot complete on my own" AD 9*

Observability

Despite having only one developer involved in the software development, participants believed that adopting PXP improved **customer involvement** as much as it would with XP. However, the majority of the participants believed that efficient customer involvement would require **extensive software project management experience**. It was highlighted that PXP is **efficient for small projects** as they are easier to manage.

Compatibility

The majority of the participants found PXP compatible with similar experiences in agile development. Some participants highlighted being able to gather **customer feedback** and improve the development process was a key factor which motivated them to adopt PXP despite working as an autonomous developer. One of the participants explained:

> *"I still get feedback from my customers as I get them involved in the software development process and this is important when improving the application being developed" AD 6*

The participants also believed that **incremental requirements gathering** can be achieved with PXP similar to other agile methodologies.

Perceived Ease of Use

The majority of the participants considered PXP methodology as easy to adopt when working on **small projects** as it eases the **completion of various tasks** in the project. Some of the participants explained:

> *"It is very easy to use PXP when you are dealing with a small project which doesn't take time. Then you know you can easily complete the tasks because it is a small project..." AD 8*

> *"Well, it all depends, the big projects can be very stressful to complete when you work alone, but if you are dealing with those small ones, PXP can be very easy to use in such scenarios" AD 6*

Discussion

The objective of this study was to explore the adoption of PXP among autonomous software developers. To the best of the researcher's knowledge, this is the first time an empirical study is being conducted on the adoption of PXP. This study also contributes to the academic literature on the application of PXP in software projects. The findings of this study extends the current knowledge on PXP practices and developers' adoption.

The findings revealed various factors that influence developers' adoption of PXP which includes independence and management. The findings differ from other studies on XP which suggests that while team members are independent, they are, to some extent, controlled by management (Rumpe & Scholz, 2003). Participants in this study highlighted effective management as one of the main highlights of PXP. This can be attributed to the fact that only one developer is involved in both the administration and development of the project. This is in contrast with other studies that explain that software developer teams are different from management teams (Rumpe & Scholz, 2002). It was also noted that understanding user stories becomes easier when PXP is applied. This can be attributed to the fact that there is a single line of communication between the customer and the developer.

The study findings also revealed that it was easier for autonomous developers to complete tasks in small projects when PXP is applied. The findings of this study are congruent to a study which reported convenience in terms of planning tasks when PXP was applied in the development of a software application (Asri et al., 2017).

It was also identified that with PXP, there was a lack of diversity of skills since only one developer is involved in the software development process. This is in contrast with other studies that suggest diversity of skills as a feature of XP (Wood et al., 2013). This could be attributed to the fact that only one developer is involved when PXP is applied in software projects. It was also revealed that there is an increased chance for errors when PXP is adopted as a result of pair programming not being practised. This is also in opposition to previous findings which suggest that XP allows pair programming (Wood et al., 2013). This is also attributed to the fact that there is only one developer involved in the software development. However, this can be improved through experience in writing software codes and setting up extra time for cross-checking codes already written.

Some of the participants believed that adopting PXP could be time consuming when embarking on a large project, while others believed that PXP is easy to use when working with small projects. The findings from this study is congruent with other studies that suggest that XP is suitable for small scale projects (Cao et al., 2004; Schalliol, 2003). It is, therefore, necessary for autonomous developers to improve time management skills and work on small to medium sized projects when working as an autonomous developer. It is recommended that the scope of work should be considered first before adopting PXP in software projects.

Although the lack of skills diversity was experienced among developers adopting PXP, it can be improved with time through experience and practise. When the developer engages in different kinds of projects, they can develop different skills which can be applied in different projects. Similar to other studies in the literature on the benefits of PXP (Marthasari et al., 2018), this study found that PXP facilitates flexible requirements gathering.

In addition, the advantages of adopting PXP as found in this study provides empirical support that PXP can be rewarding for autonomous software developers.

Conclusion

The main objective of this paper was to explore the adoption of PXP among autonomous software developers. The study provides useful insights into the adoption of PXP using the constructs from the DoI theory (relative advantage, complexity, compatibility and observability) and Technology Acceptance Model (perceived usefulness and perceived ease of use). This study established that despite the challenges identified when applying PXP in software projects, independence and effective management can be achieved with the adoption of PXP. The study also found that PXP reduces the line of communication and makes it faster to understand user stories.

From a managerial perspective, PXP would be effectively adopted when an autonomous developer has gathered a wide range of skills in software development.

It would be interesting to investigate the perspectives of other stakeholders such as customers to understand their perspectives on PXP practices. In addition, there is a limited number of studies investigating PXP practices, as such, longitudinal studies can be valuable in examining the progress of PXP practices over time. Furthermore, one approach to improve the application of PXP in software projects is to incorporate PXP into learning environments, similar to XP (Murphy, Phung & Kaiser, 2008). Future research include carrying out longitudinal studies investigating PXP practices and incorporating PXP into learning environments.

REFERENCES

Agarwal, R. and Umphress, D. 2008. "Extreme Programming for a Single Person Team," in *Proceedings of the 46th Annual Southeast Regional Conference,* Auburn, CA, pp.82-86.

Al-Jabri, I. M., and Sohail, M. S. 2012. "Mobile Banking Adoption: Application of Diffusion of Innovation Theory," *Journal of Electronic Commerce Research* (13:4), pp. 379-391.

Altameem, E. 2015. "Impact of Agile Methodology on Software Development," *Computer and Information Science* (8:2), pp.9-14.

Anwer, F., and Aftab, S. 2017. "SXP: Simplified Extreme Programming Process Model," *International Journal of Modern Education and Computer Science* (9:6), pp. 25-31.

Asri, S. A., Sunaya, I. G. A. M., Rudiastari, E., and Setiawan, W. 2018. "Web Based Information System for Job Training Activities Using Personal Extreme Programming (PXP)," in *Proceedings of the 2nd International Joint Conference Science and Technology*, Bali, pp. 1-9.

Bin, X., Xiaohu, Y., Zhijun, H., and Maddineni, S. R. 2004. "Extreme Programming in Reducing the Rework of Requirement Change," in *Proceedings of the Canadian Conference on Electrical and Computer Engineering*, Niagra Falls, Ontario, pp. 1567-1570.

Braun, V., Clarke, V., and Rance, N. 2014. How to use Thematic Analysis with Interview Data. In *The Counselling & Psychotherapy Research Handbook*, A. Vossler and N. Moller (eds.), London: Sage, pp. 183-197.

Cao, L., Mohan, K., Xu, P., and Ramesh, B. 2004. "How Extreme does Extreme Programming Have to be? Adapting XP practices to large-scale projects," in *Proceedings of the 37th Annual Hawaii International Conference*, Big Island, pp. 1-10.

Chance, K. 2011. "User Stories in Practice: A Distributed Cognition Perspective," Doctoral dissertation, Auckland: Auckland University of Technology.

Davis, F. D. 1989. "Perceived usefulness, perceived ease of use, and user acceptance of information technology," *MIS Quarterly*, (13:3), 319-340.

Davis, F. D., Bagozzi, R. P., and Warshaw, P. R. (1989). "User acceptance of computer technology: a comparison of two theoretical models," *Management Science* (35:8), 982-1003.

Dima, A. M., and Maassen, M. A. 2018. "From Waterfall to Agile Software: Development Models in the IT Sector, 2006 to 2018 Impacts on Company Management," *Journal of International Studies (*11:2), pp. 315-326.

Dzhurov, Y., Krasteva, I., and Ilieva, S. 2009. "Personal Extreme Programming – An agile Process for Autonomous Developers," in *Proceedings of the International Conference on Software, Services & Semantic Technologies*, Sofia, pp. 253-259.

Etikan, I., Musa, S. A., and Alkassim, S. 2016. "Comparison of Convenience Sampling and Purposive Sampling," *American Journal of Theoretical and Applied Statistics (*5:1), pp. 1-4.

Hedin, G., Bendix, L., and Magnusson, B. 2005. "Teaching Extreme Programming to Large Groups of Students," *Journal of Systems and Software* (74:2), pp. 133-146.

Hammarberg, K., Kirkman, M., and De Lacey, S. 2016. "Qualitative Research Methods: When to use them and How to Judge them," *Human Reproduction* (31-3), pp. 498-501.

Humphrey, W. S. 2000. "The Personal Software Process (PSP)," Pittsburgh, PA: Carnegie Mellon Software Engineering Institute, November, 2000.

Hneif, M., and Ow, S.H. 2009. "Review of Agile Methodologies in Software Development," *International Journal of Research and Reviews in Applied Sciences* (1:1), pp. 1-8.

Iyawa, G.E. 2016. "A Framework for Improving Knowledge Management Practices in Namibian Software Companies," *Journal of Information and Knowledge Management* (15:1), pp. 1-13.

Iyawa, G.E., Herselman, M., and Coleman, A. 2016. "Customer Interaction in Software Development: A Comparison of Software Methodologies Deployed in Namibian Software Firms," *Electronic Journal of Information Systems in Developing Countries* (77:1), pp. 1-13.

Maguire, P., Maguire, P., Hyland, P., and Marchall, P. 2014. "Enhancing Collaborative Learning using Pair Programming: Who Benefits?," *All Ireland Journal of Teaching and Learning in Higher Education (*6:2), pp. 1411-14125.

Marthasari, G., Suharso, W., and Ardiansyah. F. A. 2018. "Personal Extreme Programming with MoSCoW Prioritization for Developing Library Information System." in *Proceeding of the Electrical Engineering Computer Science and Informatics*, Malang, pp. 537-541.

Mohammed, A. M., and Abushama, H. M. 2013. "Popular Agile Approaches in Software Development: Review and Analysis," in *Proceedings of the International Conference on Computing, Electrical and Electronics Engineering*, Khartoum, pp. 160-166.

Moniruzzaman, A.B.M., and Hossain, S.A. 2013. "Comparative Study on Agile Software Development Methodologies," *Global Journal of Computer Science and Technology Software & Data Engineering (*13:7), pp. 5-18.

Murphy, C., Phung, D., and Kaiser, G. 2008. "A distance learning approach to teaching extreme programming," in *Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education,* Madrid, pp. 199-203.

Olsson, A., Engström, M., Lampic, C., and Skovdahl, K. 2013. "A passive positioning alarm used by persons with dementia and their spouses–a qualitative intervention study." *BMC Geriatrics*, (13:1), pp. 1-9.

Pomeroy-Huff, M., Cannon, R., Chick, T. A., Mullaney, J., and Nichols, W. 2009. "The Personal Software Process SM (PSP SM) Body of Knowledge," Software Engineering Institute.

Qureshi, R.J., and Ikram, J.S. 2015. "Proposal of Enhanced Extreme Programming Model," *International Journal of Information Engineering and Electronic Business* (1), pp. 37-42.

Rogers, E.M. (2003). *Diffusion of innovations* (5th ed.). New York: Free Press.

Rumpe, B., and Scholz, P. 2002. "A Manager's View on Large Scale XP Projects," in *Proceedings of the 3rd International Conference on Extreme Programming and Flexible Processes in Software Engineering*, Sardinia, pp. 1-4.

Rumpe, B., and Scholz, P. 2003. "Scaling the Management of Extreme Programming Projects," Special Issue on *Management of Extreme Programming Projects* (3:8), pp. 11-18.

Schalliol, G. 2003. "Challenges for Analysts on a Large XP Project." in *Extreme Programming Perspectives*, M. Marchesi, G. Succi, D. Wells, L. Williams, J. D. Wells (eds.), Indianapolis: Pearson Education, pp. 1-5.

Sharp, H., and Robinson, H. 2008. "Collaboration and Co-ordination in Mature eXtreme Programming Teams," *International Journal of Human-Computer Studies* (66:7), pp. 506-518.

Stapel, K., Lübke, D., and Knauss, E. 2008. "Best practices in extreme programming course design*,"* in *Proceedings of the 30th International Conference on Software Engineering,* Leipzig, pp. 769-776.

Steghöfer, J., Knauss, E., Alégroth, E., Hammouda, I., Burden, H., and Ericsson, M. 2016. "Teaching Agile: Addressing the Conflict between Project Delivery and Application of Agile Methods." in *Proceedings of the 38th International Conference on Software Engineering Companion,* Austin, pp. 363-370.

Wood, S., Michaelides, G., and Thomson, C. 2013. "Successful Extreme Programming: Fidelity to the Methodology or Good Teamworking?," *Information and Software Technology* (55: 4), pp. 660-672.