# Agile Methods for Software Engineering Students Project:

# A Proposed Hybrid Methodology

**Abdulwahab[1] L, Abdalla A. Abdalla[2], Bashir S. Galadanci[3], Marshal Algudah[4] and Murtala[1] M**

1. Visiting Lecturer, Department of Computer Science Northwest University, Kano, Nigeria
2. School of Computer Science and Information Technology, Linton University College, Malaysia
3. Department of Software Engineering Bayero University, Kano, Nigeria
4. Faculty of Information Science and Technology, University Kebangsaan Malaysia
Corresponding Author e-mail abd_wahhb@yahoo.com

## ABSTRACT

Agile methods' principles and practices are nowadays becoming more and more useful in the software industry. For most software engineering students, traditional software development methods have been a core part of their development processes. Thus, both students at graduate and undergraduate levels may have limited exposure to the vital agile philosophy, its practical applications, principles and practices. In this paper, the researchers examine, in depth, the agile lifecycle through a real-time application of certain agile concepts and practices in a final software engineering project at Linton University College, Malaysia. A hybrid agile method was developed to describe the processes and applicable agile concepts and practices that could be potentially handy for software engineering students. Initial findings of this research showed that students could apply agile method practices successfully in implementing small and medium scale projects.

## KEYWORDS

Agile method, Agile practices, Extreme programming, Throwaway prototyping, Information Technology.

## 1 INTRODUCTION

In today's contemporary software industry, software engineering students are obliged to have knowledge of agile software development processes. As suggested by the Institute of Electrical and Electronics Engineers (IEEE) agile software development is considered to have significant importance [1]. Technical courses related to software development such as programming, software modeling and database management systems may help students to work in developing different kinds of systems. However, software engineering projects present even greater opportunities for students to apply both technical and methodological aspects of software development.

Selecting appropriate software development methodology can be a challenging task for student [2]. The emerging use of agile methods has presented a need for software engineering students to go agile [3].

Every software project may find a particular development methodology that is more or less appropriate and suitable [5]. The methodology can be an agile method, heavyweight or a hybrid of the two [6]. Predefined methodologies are often not followed by the developers. Thus a hybrid method or the adaptation of an existing method is mostly preferred to suit the type of system to be implemented [7]. However, even though agile methods continue to provide an unprecedented support throughout the software development life circle, it's major adhering practices poses yet a certain degree of difficulty for developers to follow thus making the software development projects vulnerable to failure [8]. Hence, adopting all the practices of one method could be challenging especially for a non-Agile practitioners. The agile practices that were followed in order to ensure end-users

satisfaction and efficient end products while maintaining agility of the development process includes the planning game, pair programming, partial on-site customer small releases, refactoring, simple design, and testing while the adopted practices of throwaway prototyping were applied in the analysis and the design prototype phases[4,9].

In recent years, agile development has been gaining high momentum and acceptance in the software industry [10]. This evolving factor necessitates the need for software engineering students to have good knowledge and practical experience of using agile methods. Some studies suggested agile methods such as extreme programming and scrum with little or no modification to their core practices as being adaptable by graduate and undergraduate students [2, 3]. However, it can be argued that agile software development for students requires more than just application of an agile method [4] with other researchers such as [11] describing agile methods as the best for small projects and highly experienced development team. This study presents a realized hybrid method of extreme programming and throwaway prototyping which combine the techniques, concepts, principles and practices of agile and rapid application development methods.

The manifesto of agile methods has laid out the core practices of the agile family [12]. Some of these practices include planning game, small releases, metaphor, simple design, testing, refactoring, pair programming, collective ownership, 40 hours per week (coding), on-site customer involvement and coding standard. However, not all the practices of each method were applied in this project due to the combination of two methods. For instance, the main XP practices that the researcher applied are planning game, partially on-site customer, small Releases, pair programming, refactoring, simple design and testing [4] were the practices of XP applied while Design Prototype, Joint Application Development Session (JAD) and Informal-Benchmarking were taken from Throwaway Prototyping methodology. The system was developed in several updated versions with each version having more functions. Different types of testing were used to make sure that the provided hybrid method can be used effectively and efficiently. This paper is an attempt to find an appropriate agile method as well as applicable agile concepts, principles and practices that can be suitable for both graduate and undergraduate students.

## 2 LITERATURE REVIEW

Numerous literatures have suggested the need for students to go Agile [2, 3], while several institutions are making agile development as part of their curriculum especially in the software engineering field [14, 15].

Study has shown that Extreme programming is the most popular and widely used method in the agile family [16]. It has all the values from the agile manifesto [12]. However, "Radical Software Development" methods [17] are the predecessors of agile software development methodologies. Since the latter is derived from the former [11], the researchers combined both to come out with a hybrid agile method.

The heart of Agile Modeling (AM) lies in its practices. It is agile methods' practices that are applied in projects which are guided by the values and principles of agile methods [10]. In practice, however, the principles underlying agile methods are difficult to realize especially for non-agile practitioners [18]. Hence, to ensure adequate agility in this project, most of the practices of agile methods were adopted and applied based on [4], as listed below:

- ✓ The planning game
- ✓ Pair Programming
- ✓ On-site Customer
- ✓ Small Releases
- ✓ Refactoring
- ✓ Simple Design, and
- ✓ Testing

### 2.1 Extreme Programming Method

The main values of Extreme Programming (XP) are simplicity, communication, feedback, and courage. XP guides software development through an incremental process and the application of different practices of extreme programming may differ from one project to another [19]. Extreme Programming is suitable for small projects with highly motivated, cohesive, stable and experienced team [11]. Therefore, this study adopted part of the XP while maintaining iterative and concurrent design and implementation.

### 2.2 Throwaway Prototyping Method

Throwaway prototyping mainly focuses on the development of design prototypes. However, the uses of the prototypes are adopted to explore design alternatives instead of the actual system [4, 11]. In the hybrid method, throwaway prototyping was adapted to suit the GUI design to be used in implementing the system. Throwaway prototyping helps to determine a complete set of requirements. During the requirements phase, a "quick and dirty" (hence rapid) partial design of the project is built [8].

### 2.3 Adapting the Throwaway Prototyping Method

Adapting the throwaway prototyping method gives the researchers more room for analysis, design and implementation to come out with the graphical user interfaces. In addition, other techniques such as informal bench marking and Joint Application Development Method (JAD) session were useful in understanding the system requirements. The figure below shows how a throwaway prototyping method was adapted.

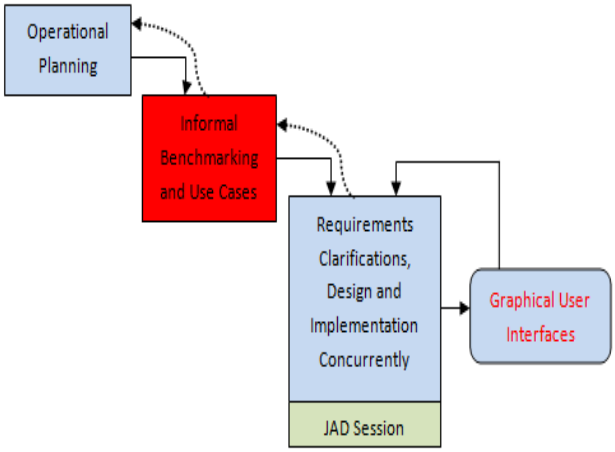


**Fig. 1**. Adapted Part of Throwaway Prototyping

The analysis and design phases of throwaway prototyping were adapted and in the analysis phase, Joint Requirement Planning was used to gather the requirements from the end-users, informal bench marking was used to help the developers to understand other awarded and similar systems. In the design phase, Graphical User Interface (GUI) design prototype of the implemented user stories for each version of the system was provided to the end-users and hence evaluated in the Joint Application Development (JAD) session.

## 3 RESEARCH METHODOLOGY

The study employ both qualitative and Quantitative techniques. A qualitative research was conducted to determine the satisfaction of the end-users and to demonstrate the appropriate interfaces and evaluate the final product [4]. The graphical user interfaces were developed based on an empirical case study of Basic English Language Tools for Beginners [9]. While for the quantitative techniques, questionnaire for this study was adapted from the research conducted by [22]. The quantitative techniques was conducted by the use questionnaire adopted from [4]. The questionnaire was administered to Basic English language students who are the main users of the system. Statistical Package for Social Science (SPSS) was then used to assess the reliability and validity of the questionnaire, as shown in Table 2.

ISBN: 978-1 -941968-10-9 ©2015 SDIWC

## 4 THE HYBRID METHOD

There are several types of software development methodologies that have been used in developing different kinds of systems that may vary in size and complexity within a wide variety of application domains. Choosing a development methodology for a particular system may be based on certain criteria such as clarity of user requirements, familiarity with technology, reliability, time and schedule visibility [11]. Moreover, Kassem [20] mentioned what an ideal life cycle model should be, "An ideal life cycle model is generic, flexible, adaptable, and scalable" [20]. The software development methodology that was chosen for this study is "Hybrid Methodology of Extreme Programming (XP) and Throwaway Prototyping".

According to the classification of software development methodologies, XP is an agile software development methodology while Throwaway prototyping is a Rapid Application Development Method. Both of these methods are useful in developing systems with unclear user requirements, systems that are reliable and systems with short time schedule [11]. It seems unclear at first to some developers on how these two methodologies will lead to a more flexible and adaptable software development methodology. However, by following the basis in which agile methods evolved, it may be understood that these methods were derived from rapid application development methods. Anderson [21] asserted that "Most agile methodologies, for example, Dynamic System Development Method are derived from earlier Rapid Development Approaches".

Therefore, if XP is derived from the family of Throwaway prototyping (RAD), it will be conceivable that both methodologies can be combined to come out with a hybrid software development methodology that will have the qualities of both methodologies thus making it reliable, faster and effective in developing software systems. The hybrid method is shown in the figure below

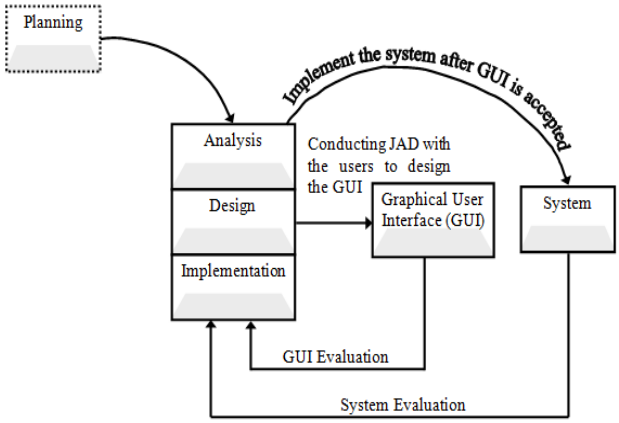


**Fig. 2**. The Hybrid Method of "XP and Throwaway Prototyping"

This method begins with a precise planning phase where the project team focuses mostly on user stories and use-cases of the system. In this phase, a tentative feasibility study of the project was analyzed to understand its technical, economical and organizational feasibility and the project team developed a work plan to set the project on the go. As with extreme programming, analysis, design and implementation will be done concurrently but the output would not be the system which is contrary to XP. In this methodology, the output of these phases will be the GUI of the system. Comparatively, the output of throwaway prototyping is a design prototype which is not used as the final system but in this methodology, the GUI prototype will be used for the final system. The GUI is therefore developed just to ensure that the developers are doing exactly what the end users are expecting because they will be involved in the process. This is even more important because it is difficult to keep users on-site during the entire development process and users can possibly get bored of waiting for the programmer to code the system and may result to a false feedback or make the end-users uncomfortable. If the end-users are not satisfied with the GUI, the developers will go back to the analysis, design and implementation phase to come out with a new GUI and the process goes on and on until the requirements are fulfilled. The next step is to start coding the functionalities of the system following an

essential XP practice which Sommerville [18] called "Pair Programming". This means that the developers will work in pair and provide support for each other to come out with the final system. However, if the users are not satisfied with the final system, the developers can go back again to the analysis phase proving that changes of the requirements are fully supported by this methodology. On the basis of stated procedure an extreme programming and Throwaway Prototyping is performed to come out with hybrid agile method which will be combined in implementing a medium scale system [4].

## 5 RESULTS

In this paper, the researchers are suggesting that students' project can either adopt agile methods or apply some of its practices to successfully complete their projects following the customers' requirements within a short schedule. Table **1** below summarizes the main processes of implementing a medium scale system by adopting some agile practices.

**Table. 1** Summary of how agile practices were adopted in implementing a medium scale system

| Simple Design | The GUI design was kept simple following what the end-users want. | At first, it was difficult to design the GUI exactly how the end-users want. |
|---|---|---|
| Testing | The users were allowed to test each function and provide feedback. User acceptance testing was performed to assess the final product [4]. | End-users faced some difficulties when testing the first two 2 iterations. However, they were satisfied after the subsequent iterations. |
| **Practices and Iterations Delivery** | **Medium Scale System** | **Difficulties Faced by Developers** |
| Number of Iterations | Six 6 iterations were delivered and evaluated by the end-users. They were asked to provide feedback and prioritise new backlog of requirements when completing each and every iteration. However, a big picture of the system was shown by conducting JAD session to discuss all the system requirements. | The end-users were not clear of what they want exactly. Hence, benchmarking technique was used to understand other similar systems and help the end-users in making decisions. |
| Type of Product | Standalone System | Not Applicable |
| Software Used | Front-end: Visual basic .Net<br>Back-end: SQL Server | Not Applicable |
| Customer Availability | 40% of the customers were on-site and they were always available at the university to provide all the system requirement. | Customer did not manage to be available at all time. Therefore, while implementing the third, fourth and fifth edition, the project manager had to call them and ask them to join while implementing the system. |
| Peer Programming | Partial Peer Programming [9]. | Peer programming was very helpful but both programmers did not get the chance to work together all the time. |
| Refactoring | Refactoring was done after the fourth iteration. | Not Applicable. |

ISBN: 978-1 -941968-10-9 ©2015 SDIWC

## 6 EVALUATIONS

Statistical Package for Social Sciences (SPSS) was used to analyze the data of the user acceptance testing. The case processing for this study was from 70 end-users of the system where about 58% of input were valid while 42% inputs were excluded. This was because the students were not able to understand and answer the questions correctly due to their proficiency level.

**Table. 2 Reliability Statistics**

| CRONBACH'S ALPHA | NO OF ITEMS/QUESTIONS |
|---|---|
| .624 | 37 |

The Cronbach's alpha (**α**) coefficient was within .60 to .70 and the results show that the end-users were satisfied with the system.

**Table. 3 The Construct Categorizations and Statistics**

| CATEGORIES | NO | MEAN | STD. DEV |
|---|---|---|---|
| Computer_ Proficiency | 41 | 4.3561 | .65194 |
| Computer_Integration | 41 | 3.6280 | .76469 |
| Effectiveness_of_CALL_vs_ Non_CALL | 41 | 3.5976 | .85491 |
| Effectiveness_of_CALL | 41 | 4.0366 | .40238 |
| Feedback | 41 | 4.1951 | .62127 |
| Skills_and_Subskills | 41 | 4.3577 | .27528 |
| Teacher_Influence | 41 | 4.0976 | .33536 |
| CALL_Based_Test_Exercise | 41 | 4.2053 | .33232 |
| Behaviour_and_Personality | 41 | 4.6585 | .56390 |

The questionnaire used in the study was divided into different categories as shown in the Table 3 above. The average mean of the categorical variable was 4.19 except for the third category (Effectiveness of CALL Vs Non CALL) where the questions were the questions were reversed and the lowering mean projects a positive view towards the use of the system.

Overall, result (See [4]) shows that the students have good perception towards the system and they will be able to use the system in real time proving that the hybrid method has been effective and was successfully applied in this project.

## 7 CONCLUSIONS

This study is an attempt to find out a suitable hybrid methodology that could be effectively used for both small and medium scale projects at the university level. The paper proposed the suitability of adopting a hybrid method for software engineering students at University level. The findings from this study suggested that: Key principles of Extreme programming and Throwaway prototyping must be applied when following the hybrid method of the two and these simple releases would help the students to receive regular feedback from the end-users and make necessary amendments accordingly. More so, users or customers remain integral part of the development team. When the hybrid method of "Extreme programming and Throwaway prototyping" is successfully incorporated to a software project at the university level, findings from this study suggested that agile practices are flexible and can be easily applied by software engineering students.

## REFERENCES

[1] SE2004 Steering Committee. "Curriculum guidelines for undergraduate degree programs in software engineering". [Online]. Available: at http://sites.computer.org/ccse/.

[2] K. M. Manamendra, K. N. Manathunga, K. H. D. Pereira, and N. Kodagoda, "Improvements for agile manifesto and make agile applicable for undergraduate research projects, International Conference on Computer Science & Education(ICCSE) , 2013.

[3] M. V. Mahnic, "A capstone course on agile software development using Scrum," Education, IEEE Transactions on, vol. 55, pp. 99-106, 2011

[4] M. Alqudah and A. A. Abdallah, "Implementing computer-aided language learning tool using hybrid agile method: A case study." International Conference of Informatics and Creative Multimedia, 2013.

[5] N. Gharaibeh, S. M. Abu-Soud, W. Bdour, and I. Gharaibeh, "Agile Development Methodologies: Are they suitable for developing Decision Support Systems," in Applications of Digital Information and Web Technologies. ICADIWT'09. Second International Conference, 2009.

[6] M. A. Awad, "A comparison between agile and traditional software development methodologies," University of Western Australia, 2005.

[7] L. Williams, "Agile software development methodologies and practices," Advances in Computers, vol. 80, pp. 1-44.
[8] J. Lemétayer, "Identifying the critical factors in software development methodology Fit," Victoria University of Wellington, 2010.
[9] M. Alqudah and A. Abdallah, "Basic English Language Tools For Beginners: Using Animations and Audio." International Journal of Scientific Engineering Research, vol 4, Issue 4, 2013.
[10] S. Ambler, Agile modeling: effective practices for extreme programming and the unified process: Wiley, 2002.

[11] A. Dennis, B. H. Wixom and R. M. Roth, Systems analysis and design, 5th ed.: Wiley, 2012.
[12] M. Fowler and J. Highsmith, "The agile manifesto," Software Development, vol. 9, pp. 28-35, 2001.

[13] G. W. Hislop, M. J. Lutz, J. F. and Naveda "Integrating agile practices into software engineering courses," Computer Science Education, vol. 12, pp. 169-185, 2002.

[14] K. Matsuo and S. Anzawa "Work in progress — Project practices of Agile Software development for undergraduate students", 40th ASEE/IEEE Frontiers in Education Conference (FIE) 2010, S2D-1 - S2D-2, Washington DC.

[15] T. Reichlmayr, "The agile approach in an undergraduate software engineering course project," Frontiers in Education, 2003. FIE 2003, 33rd Annual, Vol 3 (2003), pp. S2C-13-18 vol. 3.

[16] R. Pressman "Software Engineering: A Practitioner's Approach", 7th ed., McGraw-Hill, 2009
[17] S. Bayer and J. Highsmith, "RADical software development," American Programmer, vol. 7, 1994.
[18] I. Sommerville, "Software Engineering," 9th ed: Addison Wesley, 2010.
[19] K. N. Rao, G. K. Naidu, and P. Chakka, "A Study of the Agile Software Development Methods, Applicability and Implications in Industry," International Journal of Software Engineering and Its Applications, vol. 5.
[20] A. Kassem, "Quality software project management", 5th ed.: United Kingdom Harvard University Press, 2009.
[21] D. J. Anderson "Agile management for software engineering: Applying the theory of constraints for business results", Prentice Hall, 2004.
[22] M. Vandewaetere and P. Desmet, "Introducing psychometrical validation of questionnaires in CALL research: The case of measuring attitude towards CALL," Computer Assisted Language Learning, vol. 22, pp. 349-380, 2009.