I. J. Education and Management Engineering, **2021, 4, 43-50**
Published Online August 2021 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijeme.2021.04.05

Proposal of Enhanced FDD Process Model

Zahid Nawaz
Department of Computer Science, Punjab College, Dunyapur, Pakistan
Email: mss.zahidnawaz@gmail.com

Received: 13 July 2020; Revised: 12 November 2020; Accepted: 14 January 2021; Published: 08 August 2021

Abstract: Feature driven development (FDD) is an agile process model that develops software according to the client features. The FDD consists of five processes, several practices and both are providing benefits to improve the software development. Although the FDD provides lots of benefits, but still endures many flaws. In previous research, there have been made numerous modifications in FDD with different aspects. These modifications could not fix all type of flaws and FDD requires improvements in many aspects. These flaws reduce the agility to deliver increments continuously and make an inverse relationship between quality and agility. Due to this relationship, the FDD does not utilize enough time on making extensive documentation, robust design, client or user involvement, and efficient testing. To overcome these issues, an enhanced feature driven development model is proposed. EFDD introduces best practice of agile manifesto named as behavioral driven development used in FDD. In this way, the focus on delivering increments quickly is achieved without affecting the quality of the software. The proposed model provides maximum agility with continuous delivery according to client features and efficient testing strategy which have asses every feature according to client specified functionality.

Index Terms: Software engineering, Agile, Feature driven development, Software Testing, Software Quality, Simplified FDD Process Model, SFDD

1. Introduction

Software process models are famous in present era of software development. They provide organized way for developing software. Agile software process models are one of the categories of software process models. These models are rapidly popular in software industry and provide more agility for producing software [6]. The agile manifesto has defined twelve principles for agile. The principles are highly focused around customer satisfaction and involvement, incremental delivery of software and stakeholder's collaboration. These principles obviously help projects with changing requirements to succeed. For this reason, the agile methods are proper for adaptive and efficient model.

Feature driven development is an agile model which is used to develop a software according to the client valued functionality. It is an iterative and incremental process [13]. The functional requirements are based on demand features of the client or result in the eyes of client [13]. Assign a unique number to every feature and that number is based on the priority of client valued feature [13]. Every feature should be understandable, measurable and do able within two-week increment. FDD consists a series of steps which are providing many advantages to it. FDD consists of five primary phases: develops an overall model, builds a features list, plans by feature, designs by feature, and builds by feature. Every process describes different functions. The five processes of FDD are based on a well-known pattern or model is called ETVX [13]. ETVX is an important model for writing the template of process. The FDD processes template is written in ETVX model [13]. Initially, the FDD need to start an informal features list during development of an overall model. The domain experts and client write down features. The Chief Architect [11,13] is a main responsible person to measure the performance of each group and he may build alternative model. The modelling authority selects one best model for initiating further process [13]. The team communicates with client and domain experts to find features. The team should allocate priority and weight to features [13]. If a corresponding feature is large and complex, decompose it into small modules and complete the feature list [1,13]. The project manager, chief programmer and development manager develops a plan for implementation of features. Every feature with highest priority should be considered in first iteration. The development team identifies feature dependency and measure the complexity of every feature. The chief programmer allocates features ownership to every developer in the form of classes. The development team should build a design in form sequence diagrams, classes with prototypes and refine the overall model. After the design of feature then, the chief programmer conducts a design inspection. The developers write code according to every feature. Once the code is ready, the developers perform unit testing [10] and conducts code inspection. After successful code inspection, the code should be built, and features are implemented into a system.

Despite the benefits offered by FDD, several drawbacks are noted in previous researches. For instance, FDD suffers from weak documentation and lack of communication control with client and within team members. FDD also suffers from extensive modeling activity and less iteration among the phases and stakeholders. The client has right to change or add the requirements during the software development. FDD does not properly handle changing requirements and not provide efficient maintenance mechanism for software development. If the requirements are changed, the FDD allows to change overall plan and design of the product. FDD does not support fast modifications and it will be time consuming process which reduces the agility. It needs highly competent members in team. These members need to collaborate, trust, respect and be self-organized. The FDD practices and activities are require high collaboration, involvement, face to face meetings and customer involvement. We have already proposed a customized process model named as Simplified FDD Process Model (SFDD) [35] in our previous research. This model is simplified version of FDD for small to medium level projects [35]. The SFDD has built for small to medium level projects where the requirements are more likely to change [35]. It also introduces the requirement elicitation process of story cards and testing phase [1]. In extension of the previous research, we propose another customized model named as Enhanced Feature Driven Development (EFDD). EFDD is a design for medium to large scale projects and provides fast continues delivery of software. We use efficient practice of agile methodology is Behavior driven development [14] and its characteristics in our proposed EFDD model.

Remaining part of this paper includes related work in section II. Section III defines the problem questions and we propose EFDD process model as a solution. Section IV finally concludes this paper.

2. Related Work

In the last decades, many researchers have discussed the FDD with different aspects. Some of these are compared FDD with other agile methodologies [1,7,10], some are composed with other agile methods for devising the better hybrid model [1] and the rest of them modify it to develop customized model. Many researchers have worked for the improvement of FDD. Some of the previous research contributions are discussed below.

Nawaz et al. [35] presented Simplified FDD Process Model (SFDD). It is a simplified version of classical FDD process model that is designed for small scale projects. In this model, the requirement elicitation process of story cards and testing phase are introduced. The SFDD eliminates constraint the high reliance to experience staff and it provides the efficient mechanism for changing requirements. Doshi and Patil [1] presented Competitor Driven Development. It is hybrid process model and came in to existence with the integration of Extreme Programming and Feature Driven Requirement Reuse Development (FDRD). Mahdavi-Hezave and Ramsin [2] presented Feature-Driven Methodology Development (FDMD), an extension of Feature Driven Development. The authors presented the model for Situational Method Engineering (SME) where features are used to specify the requirements. Firdaus et al. [3] proposed an enhanced Feature Driven Development model for secure software development. They have proposed that features regarding the secure development of software must be added in the form of separate phases. According to authors, this activity will make the model suitable for the development of secure systems. Thakur and Singh [4] have proposed Feature Driven Reuse Development Process Model that is an enhanced version of FDD. They have integrated the software reusability feature with FDD. Kumar et al. [8] presented Cognizant Feature Driven Development (CFDD) a customized version of FDD. They have proposed change oriented adaptive software engineering model. In that model changes would be handled separately from the development phase. The authors have used combination of two development techniques Adaptive development and CFDD. Kanwal et al. [10] have proposed a hybrid software architecture evaluation method (SAEM) for feature driven development (FDD). The proposed SAEM consists of Quality Attribute Workshop (QAW), Architecture Trade-off Analysis Method (ATAM) and Active Review for Intermediate Designs (ARID). Rychlý and Tichá [11] presented a supporting tool for the implementation of FDD for software development. They also have discussed the different phases and practices of FDD model

Chowdhury and Huda [7] offered the comparison of adaptive software development and feature driven development both are the agile methodologies. This comparison is based on a software engineering body of knowledge (SWEBOK) with two knowledge areas such as software requirements and software construction and measures the result of both models. Cataldo and Herbsleb [9] discussed the factors influencing technical characteristics of product features and features team that cause the software integration catastrophes. The authors describe two factors like nature of module dependencies and geographical dispersion which affect the product features and features team. This article provides exploration that cross-feature communications measure the number of architectural dependencies between two products' features. Paetsch et al. [12] presented comparison of traditional requirements engineering techniques with agile software development methodologies. In this comparison, the agile software development is different as compared to traditional approaches with respect to requirement engineering process. The authors discussed different agile methodologies such as FDD, Scrum, XP, ASD, Crystal and they also contributed analysis between these methodologies regarding the importance of requirement engineering process and its other major activities such as documentation, modelling, management and validation etc. The summary of the literature review regarding this paper is shown in Table 1.

Table 1. Summary of the Related Work

Title	*Summary*
Simplified FDD Process Model (SFDD) [35]	Presented Simplified FDD Process Model (SFDD). It is simplified version of classical FDD process model that is designed for small scale projects.
Competitor Driven Development Hybrid of Extreme Programming and Feature Driven Reuse Development [1]	Presented Competitor Driven Development (CDD) an Integration of XP and FDRD.
FDMD: Feature-Driven Methodology Development [2]	Presented Feature Driven Methodology Development (FDMD) an extension of FDD.
Secure Feature Driven Development (SFDD) Model for Secure Software Development [3]	Proposed an enhanced FDD for secure software development
FDRD: Feature Driven Reuse Development Process Model [4]	Presented Feature Driven Reuse Development (FDRD) a customised FDD model.
Comparison between Adaptive Software Development and Feature Driven Development [7]	The comparison of adaptive software development and feature driven development both are the agile methodologies.
Change-Oriented Adaptive Software Engineering by Using Agile Methodology: CFDD [8]	Presented the Cognizant Feature driven development a customised FDD Model.
Factors Leading to Integration Failures in Global Feature- Oriented Development: An Empirical Analysis [9]	Factors leading to integration failures in global feature-oriented development: an empirical analysis.
A Hybrid Software Architecture Evaluation Method for FDD – An Agile Process Model [10]	Presented Single Software Architecture Evaluation Method (SAEM) for architecture evaluation of FDD.
A Tool for Supporting Feature-Driven Development [11]	Presented a supporting tool for the implementation of FDD
Requirements Engineering and Agile Software Development [12]	The comparison of traditional requirements engineering techniques with agile software development methodologies.

3. Materials and Methods

Feature driven development is an agile process model that is intended for adaptive software projects where requirements are based on client valued functions. One of the fundamental features of FDD is its closeness to reality and client expectations. Several papers are proposed for adaption of FDD in different scenarios [1,3,8,10]. Other papers are focused to improve and extend the FDD and they offer various case studies for different types of projects [4,5,7].

However, these proposed solutions lack to offer rapid responsiveness to change client requirements, efficient test strategy, and continuous delivery of project, strong communication and formal documentation. These proposed solutions are not provided efficient solution for medium to large scale project. Our first model SFDD [35] is for small scale projects and the proposed model (EFDD) supports medium to large scale projects. Both models increase the agility and they resolve the issues like poor documentation, lack of changing client requirements, lack of strong communication with client, efficient test mechanism, fast and continuous delivery. The research question arises is:

- How to introduce an effective fast delivery mechanism in FDD which can handle continues delivery of project?
- How to design a customized model for FDD that improves clients' change requests, efficient testing, and better documentation?

The enhanced FDD model (EFDD) is expected to deliver a high-quality software system in terms of design, code, test and documentation. Therefore, it can be implemented on projects of different sizes unlike the existing FDD that emphasis highly on large scale organizations. The proposed model is responsive to changing client requirements without changing overall plan and design of product. It provides robust testing, better documentation, strong communication. The traditional FDD offers major activities which are highly depending upon extensive design, modeling, coding, inspections, and unit testing. These problems affect the agility and quality of software projects. The existing FDD also does not provide robust documentation and rapid responsiveness to change client requirements in efficient manner.

EFDD is composed with Behavior Driven Development (BDD) [14] and it offers major characteristics of BDD. EFDD provides efficient performance using the BDD. EFDD mostly keeps the main phases of traditional FDD such as: Develop an Overall Model, Build a Feature List and Plan by Feature. It provides two new phases named as deployment and maintenance. We combine two phases named as 'Design by feature' and 'Build by feature' into one phase named as 'design and build by feature'. We implemented the BDD in Design and Build by feature phase shown in Fig 1. We shall not go on Develop an overall model phase, If the requirements or features are changed at any stage of development. We need to go on design and build by feature phase for incorporate changing requirements and updating of product.

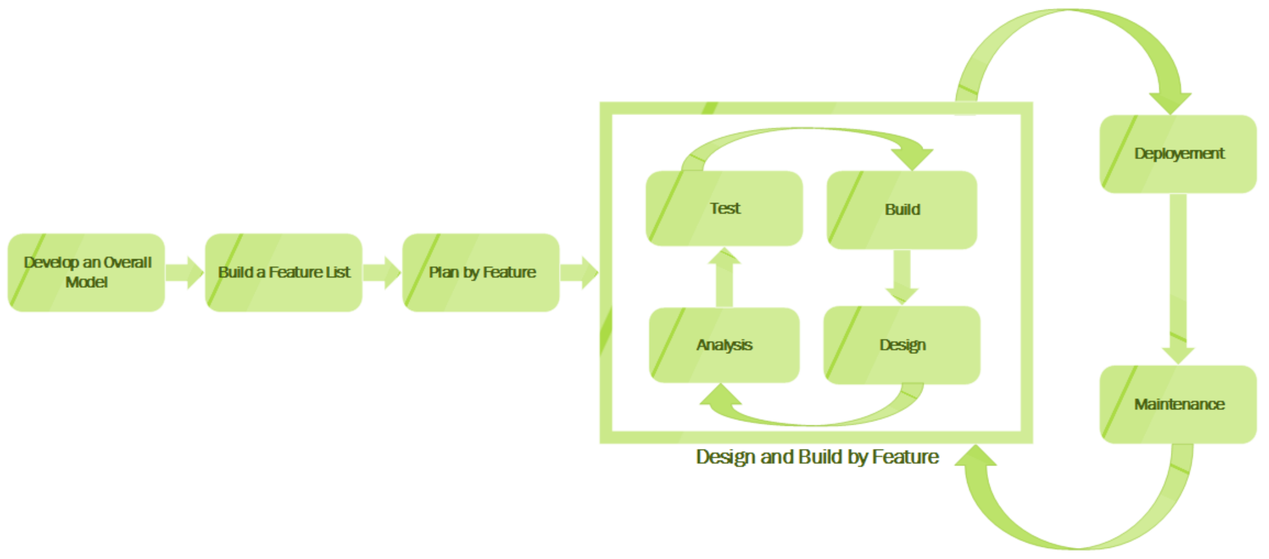

Fig.1. The Proposed Enhanced FDD Model (EFDD)

We need to more explanation of EFDD model. We elaborate all phases and their activities of EFDD with the help of activity diagrams.

3.1 Develop an Overall Model

The first phase of EFDD is similar to FDD but we introduced a log file for maintaining documentation. The development team build small groups. A model is assigned to every group and it is responsible to build a model. The Chief Architect is main the responsible person to measure the performance of each group and he may build alternative model. The Chief Architect provides guidance to domain and development members. At the ending stage of this phase, all the group data and other information stored on log file for maintain documentation. Finally, the modelling authority chooses one of the best models for initiating further process as shown in Fig 2.

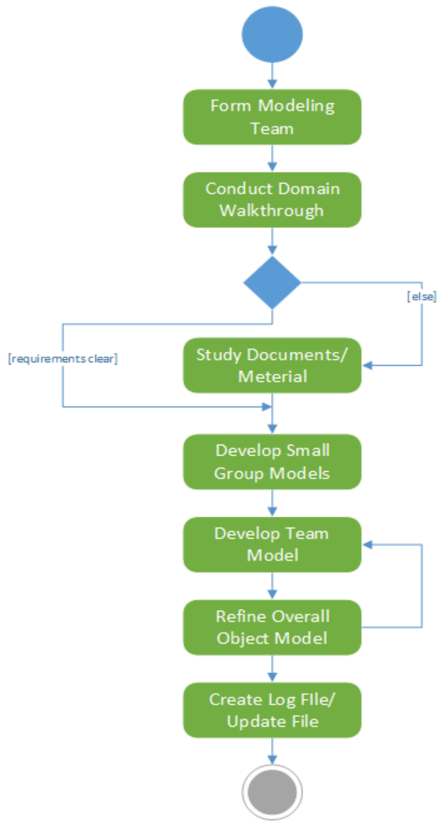
Form Modeling Team
Conduct Domain Walkthrough
[else]
[requirements clear]
Study Documents/ Meterial
Develop Small Group Models
Develop Team Model
Refine Overall Object Model
Create Log File/ Update File

Fig.2. Activity Diagram for Develop an overall Model

3.2 Build a Feature List

In the second phase of EFDD the team is required to find features through communication with client and prefer the client valued functionality. Each feature is required to complete within two weeks. The team should allocate priority and weight to features. If a corresponding feature is large and complex, it needs to decompose into small modules. Domain members participate in follow-up sessions. At the end of the process, all the required feature and other information stored on log file as shown Fig 3.

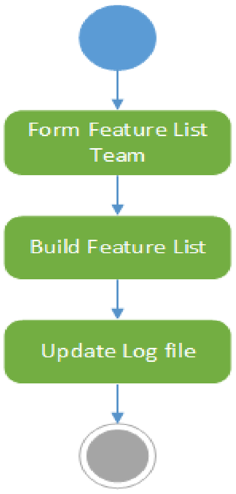

Fig. 3. Activity Diagram to Build a Feature List

3.3 Plan by Feature

The project manager, chief programmer and development manager are required to build a plan for implementation of features. Each feature with highest priority should be considered in first iteration. The development is needed to identify feature dependences and measure complexity of features. Finally, the chief programmer allocates feature ownership to every programmer in the form of classes. At the ending stage of this phase, it must update log file. We describe all activities of this phase with help of following Fig 4:

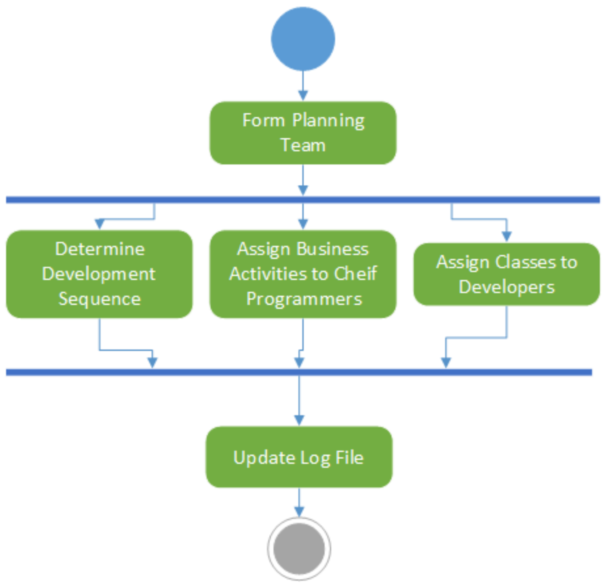
Form Planning Team
Determine Development Sequence
Assign Business Activities to Cheif Programmers
Assign Classes to Developers
Update Log File

Fig 4. Activity Diagram of Plan by Feature

3.4 Design and Build by Feature

EFDD combines the two distinct phases into one phase. The major changes are made in this phase. One of changes is to implement behavior driven development on EFDD. The chief programmer selects feature from feature list. Every feature is built within two weeks. The client and chief programmer define acceptance criteria for every feature. The feature team develops test for every feature and runs the test. If the test is failed, then define a step for test of feature and write failing code specification and refactor the code as shown Fig 5. The feature team made little change in code until feature or test get successful pass and run it again. The client and chief programmer analyze corresponding feature or acceptance test according to acceptance criteria. If the acceptance test fails for any feature, then repeat step that defines step for test of feature until it is passed successfully. If the acceptance test becomes successful for a feature then, the code should be built of corresponding feature. At the ending stage of this phase, we need to update a log file as shown Fig 5.

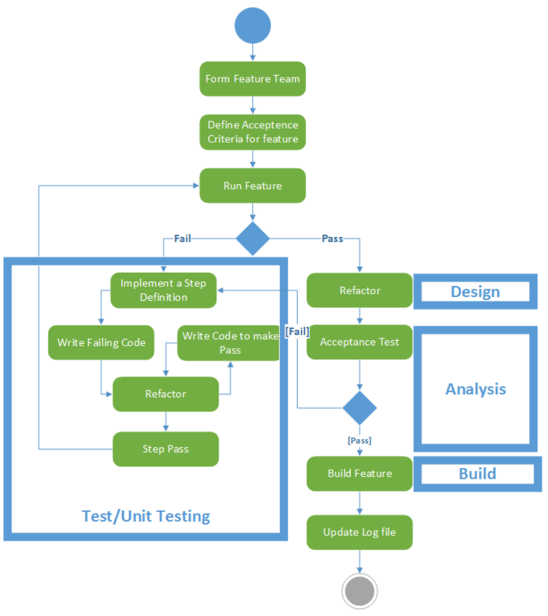

Fig.5. Activity Diagram for Design and Build by feature

3.5 Deployment

In this phase, all successfully built features are implemented into a system. Built features are deliver to client rapidly through deployment phase. EFDD supports continuous delivery to client. System implementation techniques may be used during the deployment phase. At the ending stage of this phase, we need to update a log file. We describe all activities of this phase with the help of following Fig 6:

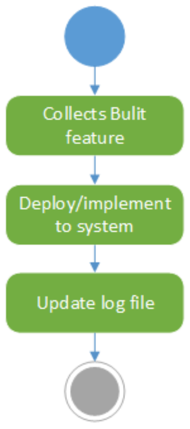

Fig.6. Activity Diagram for Deployment

3.6 Maintenance

The primary feature of EFDD is providing maintenance phase that handles change requirements after the deployment of system. This phase provides rapid responsiveness to change client requirements in efficient manner. If the client requirements change, it will need to convert requirements into features. These features transfer to design and build by feature phase. EFDD is an iterative model and iteration will continue when client has a wish to change requirements. EFDD provides quick response to change requirements through maintenance phase as shown Fig 7.

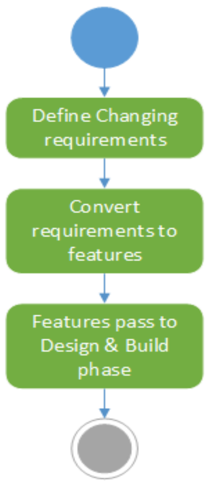
Define Changing requirements
Convert requirements to features
Features pass to Design & Build phase

Fig.7. Activity Diagram for Maintenance

4. Conclusion

Feature driven development is an agile methodology for software development according to client-valued functionality. FDD is used in large-scale organizations due to heavy modeling, inspections and other activities. These all activities reduce usability and agility of FDD. Moreover, FDD suffers from weak documentation, lack of communication control with client and within team members, extensive modeling activity, less iteration among the phases and stakeholders. FDD poorly handles change requirements and it could not allow rapid modifications. It will be time consuming process. Several papers are proposed for adaption of FDD in different scenarios and other papers are focused to improve and extend the FDD and offer case studies for different types of projects. However, these proposed solutions lack to offer rapid responsiveness to change client requirements, efficient test strategy, continuous delivery of project, increase strong communication with client and formal documentation. These proposed solutions also lack to offer FDD for small to medium size organizations because the original FDD model is specifically designed for large size organizations. In this paper, an extended model of FDD is proposed which is called Enhance Feature Driven Development (EFDD) to address the problems mentioned. This model is for small to medium size organizations and as well as large size organizations and it increases the agility. This paper gives clear steps on how to implement EFDD model. Thus, in our future we intend to evaluate EFDD with the help of different case studies on various organizational scales.

References

[1] Doshi, V.P. and Patil, V., 2016, February. Competitor driven development: Hybrid of extreme programming and feature driven reuse development. In Emerging Trends in Engineering, Technology and Science (ICETETS), International Conference on (pp. 1-6). IEEE.
[2] Mahdavi-Hezave, R. and Ramsin, R., 2015, April. Fdmd: Feature-driven methodology development. In Evaluation of Novel Approaches to Software Engineering (ENASE), 2015 International Conference on (pp. 229-237). IEEE.
[3] Firdaus, A., Ghani, I. and Jeong, S.R., 2014. Secure Feature Driven Development (SFDD) Model for Secure Software Development. Procedia-Social and Behavioral Sciences, 129, pp.546-553.
[4] Thakur, S. and Singh, H., 2014, May. FDRD: Feature driven reuse development process model. In Advanced Communication Control and Computing Technologies (ICACCCT), 2014 International Conference on (pp. 1593-1598). IEEE.
[5] Firdaus, A., Ghani, I. and Yasin, N.I.M., 2013. Developing secure websites using feature driven development (FDD): a case study. Journal of Clean Energy Technologies, 1(4), pp.322-326.
[6] Pathak, K. and Saha, A., 2013. Review of agile software development methodologies. International Journal, 3(2).
[7] Chowdhury, A.F. and Huda, M.N., 2011, December. Comparison between adaptive software development and feature driven development. In Computer Science and Network Technology (ICCSNT), 2011 International Conference on (Vol. 1, pp. 363-367). IEEE.
[8] Kumar, K., Gupta, P.K. and Upadhyay, D., 2011, April. Change-oriented adaptive software engineering by using agile methodology: CFDD. In Electronics Computer Technology (ICECT), 2011 3rd International Conference on (Vol. 5, pp. 11-14). IEEE.
[9] Cataldo, M. and Herbsleb, J.D., 2011, May. Factors leading to integration failures in global feature-oriented development: an empirical analysis. In Proceedings of the 33rd International Conference on Software Engineering (pp. 161-170). ACM.

[10] Kanwal, F., Junaid, K. and Fahiem, M.A., 2010, December. A hybrid software architecture evaluation method for fdd-an agile process model. In Computational Intelligence and Software Engineering (CiSE), 2010 International Conference on (pp. 1-5). IEEE.
[11] Rychlý, M. and Tichá, P., 2008. A tool for supporting feature-driven development. In Balancing Agility and Formalism in Software Engineering (pp. 196-207). Springer Berlin Heidelberg.
[12] Paetsch, F., Eberlein, A. and Maurer, F., 2003, June. Requirements Engineering and Agile Software Development. In WETICE (Vol. 3, p. 308).
[13] Coad, P., Lefebvre, E. & De Luca, J. (1999). Java Modeling In Color With UML: Enterprise Components and Process. Prentice Hall International. (ISBN 013011510X).
[14] Solis, C. and Wang, X., 2011, August. A study of the characteristics of behavior driven development. In Software Engineering and Advanced Applications (SEAA), 2011 37th EUROMICRO Conference on (pp. 383-387). IEEE.
[15] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, “Agile software development methods: Review and analysis.” 2002.
[16] S. R. Palmer and M. Felsing, “A practical guide to feature-driven development,” Pearson Education, 2001.
[17] S. Goyal, “Major seminar on feature driven development,” Jennifer Schiller Chair of Applied Software Engineering. 2008.
[18] Anwer, S. Aftab, U. Waheed and S. S. Muhammad, “Agile Software Development Models TDD, FDD, DSDM, and Crystal Methods: A Survey,” International Journal of Multidisciplinary Sciences and Engineering, vol. 8, no. 2, MARCH 2017.
[19] K. Pathak and A. Saha, “Review of agile software development methodologies,” International Journal, vol. 3, no. 2, 2013.
[20] M. Hayat and M. Qureshi, “Measuring the Effect of CMMI Quality Standard on Agile Scrum Model,” arXiv preprint arXiv: 1610.03180, 2016.
[21] M. R. J. Qureshi and J. S. Ikram, “Proposal of Enhanced Extreme Programming Model,” International Journal of Information Engineering and Electronic Business, vol. 7, no. 1, pp. 37, 2015.
[22] M. Fowler and J. Highsmith, “The agile manifesto,” Software Development, vol. 9, no. 8, pp. 28-35, 2001.
[23] D. Cohen, M. Lindvall, and P. Costa, “An introduction to agile methods,” ADVANCES IN COMPUTERS, vol. 62, 62, pp.1- 66, 2004.
[24] P. Coad, E. Lefebvre, and J. De Luca Java, “Modeling In Color With UML,” Enterprise Components and Process. Prentice Hall International, (ISBN 013011510X), 1999.
[25] B. Boehm, “A survey of agile development methodologies,” Laurie Williams, 2007.
[26] S. Khramtchenko, “Comparing eXtreme Programming and Feature Driven Development in academic and regulated environments,” *Feature Driven Development*, 2004.
[27] S. S. Tirumala, S. Ali and A. Babu, “A Hybrid Agile model using SCRUM and Feature Driven Development,” *International Journal of Computer Applications,* vol. 156, no. 5, pp. 1-5, December 2016.
[28] R. Mahdavi-Hezave and R. Ramsin, “Fdmd: Feature-driven methodology development,” in *Evaluation of Novel Approaches to Software Engineering* (ENASE), International Conference, pp. 229-237 IEEE. 2015
[29] Firdaus, A., Ghani, I. and Jeong, S.R., 2014. Secure Feature Driven Development (SFDD) Model for Secure Software Development. Procedia-Social and Behavioral Sciences, 129, pp.546-553.
[30] S. Thakur and H. Singh “FDRD: Feature driven reuse development process model,” in Advanced Communication Control and Computing Technologies, International Conference, pp. 1593-1598, IEEE.
[31] F. Siddiqui and M. A. Alam, “Ontology Based Feature Driven Development Life Cycle,” arXiv preprint arXiv:1307.4174, 2013.
[32] Firdaus, I. Ghani and N. I. M. Yasin, “Developing secure websites using feature driven development (FDD): a case study,” Journal of Clean Energy Technologies, vol. 1, no. 4, pp.322-326.
[33] K. Kumar, P. K. Gupta and D. Upadhyay, “Change-oriented adaptive software engineering by using agile methodology: CFDD,” in Electronics Computer Technology, 3rd International Conference vol. 5, pp. 11-14). IEEE, April 2011.
[34] F. Kanwal, K. Junaid and M. A. Fahiem, “A hybrid software architecture evaluation method for fdd-an agile process model,” in Computational Intelligence and Software Engineering International Conference pp. 1-5 IEEE. December 2010.
[35] Nawaz, Z., Aftab, S. and Anwer, F., 2017. Simplified FDD Process Model. International Journal of Modern Education and Computer Science, 9(9), p.5

Author’s Profile

Zahid Nawaz received MS(CS) degree with a Computer Science at the Virtual University of Pakistan, in 2018. Now, he serves as Head of Department of Computer Science at Punjab College Dunyapur Campus.

Mr. Zahid Nawaz has interest extensively in the area of Software Requirement Engineering, Software Process Models, Software Design and Software Quality Assurance.

How to cite this paper: Zahid Nawaz, " Proposal of Enhanced FDD Process Model", International Journal of Education and Management Engineering (IJEME), Vol.11, No.4, pp. 43-50, 2021. DOI: 10.5815/ijeme.2021.04.05