Simplified FDD Process Model

Article *in* International Journal of Modern Education and Computer Science · September 2017

DOI: 10.5815/ijmecs.2017.09.06

CITATIONS
5

READS
1,553

3 authors:

Zahid Nawaz
Virtual University of Pakistan
9 PUBLICATIONS **26** CITATIONS
SEE PROFILE

Shabib Aftab
Virtual University of Pakistan
26 PUBLICATIONS **201** CITATIONS
SEE PROFILE

Faiza Anwer
Virtual University of Pakistan
10 PUBLICATIONS **77** CITATIONS
SEE PROFILE

I.J. Modern Education and Computer Science, **2017, 9, 53-59**
Published Online September 2017 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2017.09.06

Simplified FDD Process Model

Zahid Nawaz, Shabib Aftab, Faiza Anwer
Department of Computer Science, Virtual University of Pakistan
Email: forzahid1@gmail.com, shabib.aftab@gmail.com, faiza.anwer28@gmail.com

Received: 09 June 2017; Accepted: 25 July 2017; Published: 08 September 2017

Abstract—Feature driven development (FDD) is a process oriented and client centric agile software development model which develops a software according to client valued features. Like other agile models it also has adaptive and incremental nature to implement required functionality in short iterations. FDD mainly focus on designing and building aspects of software development with more emphasis on quality. However less responsiveness to changing requirements, reliance on experienced staff and less appropriateness for small scale projects are the main problems. To overcome these problems a Simplified Feature Driven Development (SFDD) model is proposed in this paper. In SFDD we have modified the phases of classical FDD for small to medium scale projects that can handle changing requirements with small teams in efficient and effective manner.

Index Terms—Software Process Model, Agile Modeling, Customized Feature Driven Development Model, Tailored FDD, Modified FDD

I. Introduction

Agile software development is one of the most popular research areas now days. The agile models have become the need of the software developers due to their ability to incorporate change, rapid development and emphasis on quality [1]. These models have shifted the focus from "process" to "people" and gave importance to those aspects of software development that were neglected by traditional development models [4]. Extreme Programming (XP), Scrum, Test Driven Development (TDD), Dynamic System Development Method (DSDM), Feature Driven Development (FDD) and Crystal family are some of the well-known agile models [6] [29] [30].

These models provide the feature of agility in software development life cycle [5]. Alistair Cockburn, one of the initiators of the agile movement has defined agile software development as "agile implies being effective and maneuverable" [5]. The agile manifesto has defined twelve principles. These principles are highly focused around the involvement and satisfaction of the customer, incremental delivery of software and stakeholder's collaboration during development [7] [8] [29] [30]. These principles make agile process models more adaptive and efficient which obviously help to handle changing requirements effectively during software development [8] [9]. FDD is a process oriented agile development model that mainly focuses on design and building phases of software development [1] [2] [3]. The development process is completed in five sub processes that have clearly defined entry and exit criteria [2]. The five sub processes/phases are: Develop an Overall Model, Build a Features List, Plan by Feature, Design by Feature and Build by Feature. Every phase/process has different tasks [1] [3] [11]. These phases are based on well-known pattern called ETVX. FDD develops the software according to client valued functionality by using the iterative and incremental approach [10]. It uses eight best practices including domain object modeling, development by feature, individual class ownership, feature teams, inspection, configuration management, regular builds and progress reporting [1] [2].

First phase of FDD starts by developing overall model of the system after discussing the scope and context of the project in a walkthrough meeting [1] [2]. The modeling authority selects one best model for initiating further process [1]. Then different domain experts develop object models. In the second phase, feature team define a comprehensive list of features to be developed and grouped in feature sets [2] [3]. In plan by feature phase, priorities are assigned to every feature [1]. Features with higher priority are considered in early iterations. Every feature is checked against its business need after assigning a priority. It helps to check whether these features are according to the project's requirements. The development team identify feature dependency and measure the complexity of every feature. Feature ownership is assigned to every developer in the form of classes. In design by feature phase the development team design the sequence diagrams, write classes and refine overall model [1]. Moreover different design packages are produced for each class [1] [3].

These design packages are actually implemented and developed in build by feature phase. The activities of this phase include coding, code inspection, unit testing and integration testing [1] [2]. These activities are performed iteratively. FDD define six key roles, five supporting and three additional roles [1] [2]. Key roles include project manager, chief architect, development manager, chief programmer, class owner and domain experts. Supporting roles comprises of release manager, language guru, build engineer, tool smith and system administrator Whereas three additional roles of FDD are testers, deployers and technical writers.

No doubt, FDD strive for quality throughout the development process however there are some problems which are needed to be resolved to make FDD more flexible and applicable for small to medium scale projects.

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

II. Related Work

In the last decade, many researchers have discussed the FDD with different aspects. Here we listed some studies that customized FDD to reduce the identified problems or integrated with other software process models to combine the strengths and eliminate the weaknesses of both models.

Doshi and Patil [13] presented Competitor Driven Development (CDD). It is a hybrid process model that integrated the practices from Extreme Programming and Feature Driven Requirement Reuse Development (FDRD). FDRD is an enhanced version of FDD presented in [16]. Authors claimed that CDD is a self-realizing requirement generation model. This model keeps track of market trends as well as competitor's next product launch to extract requirements. It considers the market orientation of product to guess the product's success rate. However this model failed to provide any guidance about managing the changing requirements.

In [14] authors proposed a hybrid model SCR-FDD that integrated Scrum and FDD. This model taken schedule related aspects from Scrum and quality related aspects from FDD to cover the limitations of both agile models. This model resolved the problems regarding schedule, quality and deployment which were considered as the big hindrance during the development and release of software product. However the authors in this research did not provide clear and complete steps of implementation of SCR-FDD. Without detailed instructions for implementation it is hard to give feedback regarding success factors of this model.

Mahdavi, Hezave and Ramsin in [15] presented Feature-Driven Methodology Development (FDMD), an extension of Feature Driven Development. FDMD incorporated the features of object oriented approach with Situational Method Engineering (SME). In this model requirements are represented as features. These features are based on object oriented principles which are defined using action, result and object. This model tried to eliminate the issues of maintainability and reusability but remained silent about other issues of FDD.

Firdaus et al. [16] proposed an enhanced version of FDD called Secure Feature Driven Development (SFDD) for the purpose of secure software development. This model tried to cover security related issues of FDD by making some changes in classical FDD process model. Along with adding "In-phase Security " element in each phase, this model also incorporated two additional phases called " Build security by feature" and " Test security by feature". To ensure secure software development, proposed model also introduced a new role called security master. SFDD resolved security related problems but remained silent on other issues such as requirement gathering and non-suitability for small projects.

Thakur and Singh [17] proposed an enhanced version of FDD by introducing reusability in it. Feature Driven Reuse Development (FDRD) model considers re-useable feature sets along with new requirements. Although authors claimed that FDRD enhanced productivity and quality of product but it also required experienced staff to decide about re-useable components.

In [18] authors proposed an ontology based feature driven development model for semantic web application. This model used domain ontology concepts that are widely known in domain knowledge modeling. Each phase of this model has ontology as a basic building block. Ontology languages like RDF and OWL helped to overcome language ambiguity and inconsistency. This ontology based model can be evaluated using automated tools. However by adding domain ontology concepts in each phase the agility nature of FDD will be compromised.

In [19] authors conducted a case study to check the suitability of FDD for secure web site development. Authors found that by integrating more iterations, security practices and other helping tools can make the FDD suitable for secure software development. However there are no clear recommendations in the research for customizing FDD for secure software development.

Kumar et al. [20] proposed a framework to handle changing requirements efficiently. The proposed model is based on Adaptive Software Development and Cognizant Feature Driven Development (CFDD) that is a customized version of FDD. CFDD is not a complete development process. It is the collection of best practices which are mostly used during designing and development phases of process models. The proposed model is simple and easy to implement however it remained silent on other issues of FDD.

Kanwal et al. [21] have proposed a hybrid software architecture evaluation method (SAEM) by integrating Quality Attribute Workshop (QAW), Architecture Trade-off Analysis Method (ATAM) and Active Review for Intermediate Designs (ARID) with FDD. This model only deals with architecture evaluation issues and remained silent on other issues of FDD.

Rychlý and Tichá [22] presented a supporting tool for the implementation of FDD for software development. This tool allows the implementation in the form of sub processes in a multi-user web based environment. This tool has ability to track changes in requirements and map these modifications in design classes. This research also remained silent about other issues such as requirement of experience staff and non-suitability for small projects.

In [23] authors presented a customized model of FDD for aspect oriented development. Authors showed that FDD required a small refinement for aspect oriented software development. This model introduced separation of concerns that help in handling complexity and maintenance problems. Refinement in FDD could be

helpful in detecting inconsistencies among features and helped in smooth transition from one phase to other. This model does not provide complete solution and only tried to enhance designing phase of FDD.

Table 1. Summary of the Related Work

Title	*Summary*
Competitor Driven Development Hybrid Of Extreme Programming and Feature Driven Reuse Development [13]	Presented Competitor Driven Development (CDD), an Integration of XP and FDRD. CDD is a self-realizing requirement generation model that develop software product for mass targeted customers. Failed to properly address the issue of changing requirements.
A Hybrid Agile model using SCRUM and Feature Driven Development [14]	Proposed a SCR-FDD by integrating Scrum and FDD. SCR- FDD takes schedule related practices from Scrum and quality related practices from FDD. Failed to provide the clear strategy for implementation.
FDMD: Feature-Driven Methodology Development [15]	Presented Feature Driven Methodology Development (FDMD) an extension of FDD. FDMD used the concept of object oriented approach. The solution only focused on maintainability and reusability however did not address the remaining issues of FDD.
Secure Feature Driven Development (SFDD) Model for Secure Software Development [16]	Proposed an enhanced FDD for Secure Software Development. SFDD customized the FDD by adding two phases and a new role of security master. The proposed model only focused on security related issues and remained silent on other issued of FDD.
FDRD: Feature Driven Reuse Development Process Model [17]	Presented a customised FDD model called Feature Driven Reuse Development (FDRD). FDRD introduced the concept of reusability in FDD to enhance quality and productivity. Other issues of FDD are not addressed.
Ontology Based Feature Driven Development Life Cycle [18]	Proposed Ontology Based Feature Driven Development Model that used domain ontology as a basic building block in each phase of FDD. Ontology language OWL is used during requirement gathering and designing activities. With the proposed ontology feature the agility will be compromised.
Developing secure websites using feature driven development (FDD): a case study [19]	A case study was conducted to check the suitability of FDD for secure software web site development. No proper and clear guidance is provided to customize the FDD for secure development.
Change-Oriented Adaptive Software Engineering by Using Agile Methodology: CFDD [20]	Presented a customised FDD Model that integrates Adaptive Software Development with Cognizant Feature Driven Development to handle changing requirements efficiently. Other issues of FDD are not addressed.
A Hybrid Software Architecture Evaluation Method for FDD – An Agile Process Model [21]	Presented Single Software Architecture Evaluation Method (SAEM) for architecture evaluation of FDD. This model incorporate evaluation practices in two phases of FDD. . This model only deals with architecture evaluation issues and remained silent on other issues of FDD.
A Tool for Supporting Feature-Driven Development [22]	Presented a supporting tool for the implementation of FDD which will help to implement FDD in multi-user distributed environment. This study also remained silent about the other issues of FDD such as requirement of experienced staff and non-suitability for small projects.
Refining Feature Driven Development-A methodology for early aspects [23]	Proposed a refined model that incorporated Aspect Oriented Software Development in FDD. This helped in handling complexity and improved the maintainability. Refinement in FDD could be helpful in detecting inconsistencies among features and helped in smooth transition from one phase to other however it does not provide complete solution and only tried to enhance designing phase of FDD.
Comparison between adaptive software development and feature driven development [27]	FDD was compared with ASD on the basis of two knowledge areas software requirements and software constructions. Issues such as software quality and dealing with the small projects were not properly addressed.
Integrating security into agile development methods [32]	Security relevant features are introduced in FDD. Four step security strategy is followed and claimed by the authors that this can enable FDD for the development of security critical projects. The research lacks the proper empirical proof.
Comparing eXtreme Programming and Feature Driven Development in academic and regulated environments [33]	FDD was compared with XP on the basis of project management perspective. Many similar and contrasting features are identified and highlighted that both the models are suitable for different types of project. The study failed to provide the empirical evaluation.

In [27] the authors have compared the Feature Driven Development with Adaptive Software Development. The comparison mainly focused on two knowledge areas; software requirements and software construction. The basic purpose of the comparison is to evaluate the degree of agility of these two agile models. This comparison showed that there is no specific practices used for requirement elicitation and software construction in ASD however in FDD there are some predefined practices available for that purpose. Issues such as software quality and dealing with the small projects were not properly addressed.

In [32] the authors introduced security relevant features in Feature Driven Development. FDD was selected because of its concrete modeling techniques, used for software development. The authors have followed the four step security strategy in FDD. According to the authors, integrating security steps can enable FDD to be used for the development of security critical software however there is no empirical proof given to support the claim.

In [33] the authors compared Feature Driven Development with Extreme Programming. The comparison was based on different aspects however the special focus was on project management perspective. Results showed that both models have some similar and contrasting practices for software development which make them suitable for different project types. Although this paper presented a detailed comparison but failed to provide the empirical proof.

III. Problem Definition

Feature driven development is a process centric agile model that is used for adaptive software projects where requirements are based on client valued functions [4]. Five processes/phases of FDD provide a step by step procedure to produce quality working results [5]. It also adds more power by using best practices of software industry like inspection, class ownership, regular builds and progress tracking etc. [24]. FDD provide a better development approach as compared to other agile models. Other agile models (Like XP, Scrum, ASD) ignore the quality and design related aspects of software development for the sake of simplicity and agility [25] [26]. There are some limitations of FDD that restricts its benefits to some specific circumstances.

FDD process model does not provide guidance about whole development process rather it mainly focuses on design and building phases of software development [1] [2] [4] [20]. FDD needs supporting activities to complete development process successfully [24] [25]. For example in FDD only a walkthrough meeting is conducted for requirement elicitation [24]. Project tracking, maintainability, and extensibility are affected negatively if requirement elicitation activity is not performed accurately and precisely [27].

FDD lacks the ability to response rapidly changing requirements [12] that's why it performs well in large scale projects with stable and predefined requirements. It performs well in a situation where requirements have fewer tendencies to change [3] [12]. FDD greatly relies on staff's experience and capabilities for successful execution of the development process [2] [11]. Especially feature sets identification is a critical task performed by chief architect and chief programmer [12] [28]. If they do not have enough experience to extract required features, it puts a question mark on FDD's effectiveness.

FDD uses six key roles, five supporting roles and three additional roles [1] [2]. Although one team member can take responsibility of more than one role however these so many roles can fit in large teams. For small and medium scale projects, this practice will create problems.

Many researchers tried to resolve the problems by improving or customizing FDD process model [15] [16] [17] [18] [23]. Some researchers integrated FDD with other software development models to add the strengths of both models in one [13] [14]. However these proposed solutions did not perform effectively. All of the above discussion forces us to find the answer of following questions.

1) How to introduce an effective requirement elicitation technique in FDD which can handle rapid changing requirements?
2) How to customize the phases and roles of FDD to make it work effectively for small to medium scale projects?

IV. Proposed Model

In this research, Simplified Featured Driven Development Model (SFDD) is proposed to overcome the limitations of classical FDD. SFDD is designed for small to medium scale projects where requirements are more likely to change. Proposed model not only focuses on design and building phases but also concentrates on new requirement elicitation technique of story cards [29] [31]. This model is also intends to improve the software quality by introducing a testing phase within the iteration. SFDD also removed the constraint of trained staff which was one of the key limitations of classical FDD. All these features make us believe that this model can provide us a quality product, if all the activities in each phase are followed accurately and according to proposal. Below we are going to discuss the phases of SFDD in detail.

A. Develop an overall model

This is the first phase of SFDD in which project scope and requirements are identified. Chief Programmer and Domain Expert are the two active participant of this phase. They will decide the project scope initially. Chief Programmer is the focal person from the development team and the Domain Expert will represent the requirements detail from client side. This role can be represented by the client himself or can be represented by any person on behalf of client's company. Story cards are introduced as requirement gathering technique [29] [31]. Domain expert writes story card for each required feature in the system. These story cards effectively explain the required functionality without involving any technical detail. In each story card priority is given by the Domain expert for those features/requirements which should be completed in early release. After the completion of requirement gathering task, the chief programmer (with other team members) develops the use case diagrams as per requirements in story cards. Use case diagrams provide simple graphical view of requirements. After this task class diagram is developed by keeping in view the use case diagram. At the end of this phase four documents are generated: 1) Project Scope, 2) Functional & Non-Functional Requirements, 3) Use-case Diagrams and 4) class diagram.

B. Build feature list

This phase provides a foundation for upcoming phases. In this phase Chief Programmer extracts the features for the system.

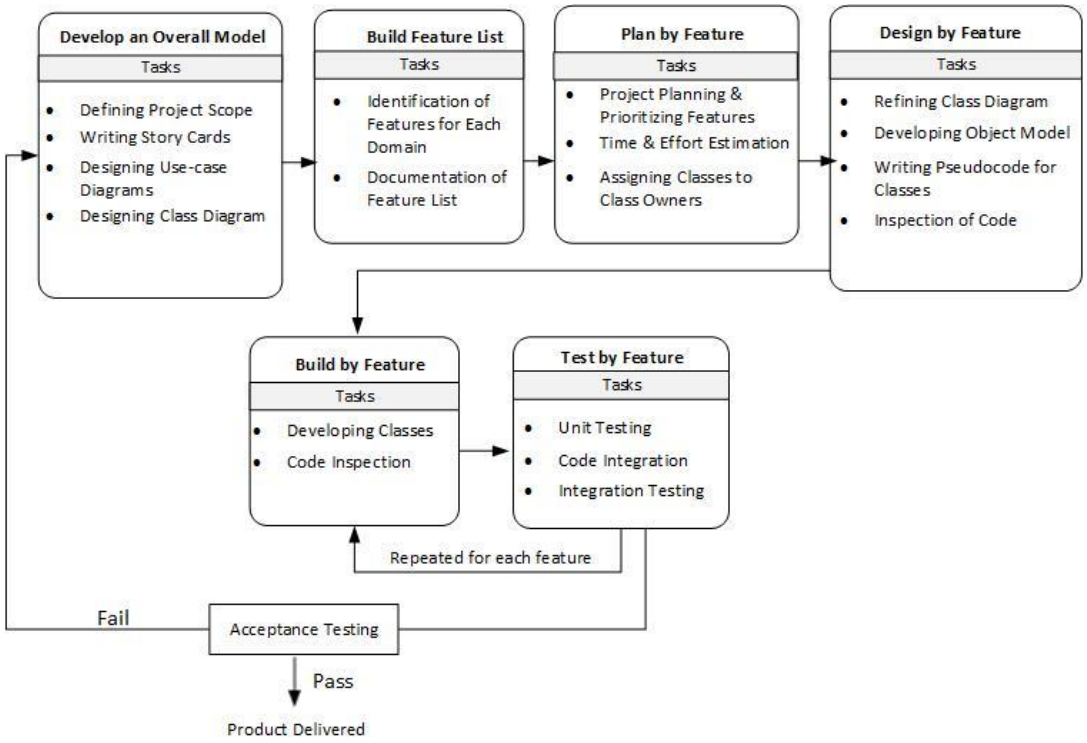

Fig.1. SFDD Process Model

He identifies the features for each domain of the system to be developed by using the documents produced in previous phase. Features belong to a specific domain are called a feature sets. Many interrelated requirements can be considered as a single feature set. List of features is documented and approved by the Domain Expert. At the end of this phase one document is generated named Feature List which includes the requirements associated to each feature.

C. Plan by feature

This phase consists of project planning activities and starts with the meeting between Domain Expert and Chief Programmer regarding budget and time frame. When they both agreed then Chief Programmer develops a Project Plan which is followed by the development team throughout the process. Keeping in view the total budget and time frame, Chief Programmer decides about the number of iterations which are needed to complete the project smoothly. Features are also documented which will be developed and included in iterations as per their priority. Resources such as hardware's and software's are also identified which are needed for the project. Chief programmer estimates the time and effort (resource persons) required to complete the each iteration. After taking these decisions chief programmer assign classes to class owners. At the end of this phase one document is generated named Project Plan which includes detail regarding iterations, resources and class owners.

D. Design by feature

This phase starts with the process of refining the class diagrams which were developed in the first phase of the model. After this process an object model is developed of the system by Chief Programmer and Class owners. Then Class owners write pseudo code for the assigned classes/modules. Complete design of the software is documented and inspected by the QA a manager. There is no doubt that a flexible and complete design is very important for a successful system that is why a role of QA manager is introduced in this phase which will thoroughly inspect all design related activities before taking it to the iteration phases of the model.

E. Build by feature

The iteration starts from this phase. The purpose of the iteration is to develop and deliver the project in small workable modules. After the release of first module every upcoming module is integrated with the previously released module(s) and this process goes-on until the project/software is completed. Iteration of this model consists of two phases Build by feature phase (this one) and the Test by feature phase (next one). In this particular phase, features are actually implemented by class owners. They write code for the classes. A formal code inspection session is conducted in the supervision of QA manager to assure that code is written according to the pseudo code and is working properly. This feature ensures the quality of the work in this phase. At the end of this phase small workable module will be ready to go in the next phase of the iteration. A document named Inspected Module is generated which will consist of the detail regarding developed module.

F. Test by feature

This is the second phase of iteration and the last phase of proposed SFDD. It deals with the testing activities to make sure that the software is bug free and according to the required features. This phase starts with the unit testing in which QA manager assured that the developed module is working properly as per required features. In case of successful unit testing the module is integrated with the previously developed and released module (s) by Chief Programmer. After integration, the integration testing is performed to check whether integrated module is working properly or not. After that, when both types of testing's are performed successfully then Domain expert final performs the acceptance testing to check whether the developed software is according to the requirements or not. At the end of this phase two documents are developed Testing Document and User's Manual. Testing Document contains the details regarding bugs (if any) noted during testing.

V. Conclusion

Feature Driven Development (FDD) is an agile software development process model. It provides a process oriented and iterative method with more focus on software quality. It is suitable for large scale projects with more focus on design and building activities of software development. There are a number of studies exist that tried to modify or improve classical FDD. Some of these tried to modify its development process by introducing new phases or new roles and other tried to integrate FDD with other models to add up the benefits of both models. However all of these studies failed to provide a comprehensive solution and remained silent for one or more of the following problems. FDD did not have an efficient requirement gathering technique which can handle changing requirements as well as it used to rely on experienced staff for successful projects. It is not a good choice for small scale projects due to its complex nature and large number of roles involved. This paper presents a customized form of FDD called Simplified Feature Driven Development (SFDD) Model. It provides the step by step solution of the identified problems without affecting its flexibility and agility.

References

[1] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, “Agile software development methods: Review and analysis.” 2002.

[2] S. R. Palmer and M. Felsing, “A practical guide to feature-driven development,” Pearson Education, 2001.

[3] S. Goyal, “Major seminar on feature driven development,” Jennifer Schiller Chair of Applied Software Engineering. 2008.

[4] F. 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.

[5] K. Pathak and A. Saha, “Review of agile software development methodologies,” International Journal, vol. 3, no. 2, 2013.

[6] M. Hayat and M. Qureshi, “Measuring the Effect of CMMI Quality Standard on Agile Scrum Model,” arXiv preprint arXiv: 1610.03180, 2016.

[7] 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.

[8] M. Fowler and J. Highsmith, “The agile manifesto,” Software
Development, vol. 9, no. 8, pp. 28-35, 2001.

[9] D. Cohen, M. Lindvall, and P. Costa, “An introduction to agile methods,” ADVANCES IN COMPUTERS, vol. 62, 62, pp.1- 66, 2004.

[10] P. Coad, E. Lefebvre, and J. De Luca Java, “Modeling In Color With UML,” Enterprise Components and Process. Prentice Hall International, (ISBN 013011510X), 1999.

[11] B. Boehm, “A survey of agile development methodologies,” Laurie Williams, 2007.

[12] S. Khramtchenko, “Comparing eXtreme Programming and Feature Driven Development in academic and regulated environments,” *Feature Driven Development*, 2004.

[13] V. P. Doshi and V. Patil, “Competitor driven development: Hybrid of extreme programming and feature driven reuse development,” in *Emerging Trends in Engineering, Technology and Science*, International Conference, pp. 1-6, IEEE. February, 2016.

[14] 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.

[15] 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

[16] 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.

[17] 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.

[18] F. Siddiqui and M. A. Alam, “Ontology Based Feature Driven Development Life Cycle,” arXiv preprint arXiv:1307.4174, 2013.

[19] A. 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.

[20] 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.

[21] 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.

[22] M. Rychlý and P. Tichá , “A tool for supporting feature-driven development,” in Balancing Agility and Formalism

in Software Engineering, Springer Berlin Heidelberg, pp. 196-207, 2008.
[23] J. Pang and L. Blair, “Refining Feature Driven Development-A methodology for early aspects. Early Aspects:,” Aspect-Oriented Requirements Engineering and Architecture Design, p.86. 2004.
[24] M. Umbreen, J. Abbas and S. M. Shaheed, “A Comparative Approach for SCRUM and FDD in Agile,” International Journal of Computer Science and Innovation, vol. 2, pp.79-87, 2015.
[25] E. Mnkandla and B. Dwolatzky, “Agile Software Methods: State-of-the-Art.” Agile Software Development Quality Assurance, 1, 2007.
[26] Dalalah, A., 2014. Extreme Programming: Strengths and Weaknesses. Computer Technology and Application, 5(1).
[27] A. F. Chowdhury and M. N. Huda, “Comparison between adaptive software development and feature driven development,” in Computer Science and Network Technology, International Conference, vol. 1, pp. 363-367, IEEE, December, 2011.
[28] U. Ismail, S. Qadri and M. Fahad, “Requirement Elicitation for Open Source Software By using SCRUM and Feature Driven Development,” International Journal of Natural & Engineering Sciences, vol. 9, no. 1, 2015.
[29] F. Anwer, S. Aftab, "SXP: Simplified Extreme Programing Process Model," International Journal of Modern Education and Computer Science (IJMECS), Vol.9, No.6, pp.25-31, 2017.
[30] S. Ashraf, S. Aftab, "Latest Transformations in Scrum: A State of the Art Review," International Journal of Modern Education and Computer Science (IJMECS), Vol.9, No.7, pp.12-22, 2017.
[31] F. Anwer, S. Aftab, SSM. Shah, U. Waheed, "Comparative Analysis of Two Popular Agile Process Models: Extreme Programming and Scrum," International Journal of Computer Science and Telecommunication, Vol 8, No 2, pp. 1-7, 2017
[32] M. Siponen, R. Baskerville, T. Kuivalainen, "Integrating security into agile development methods." System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on. IEEE, 2005.
[33] S. Khramtchenko, "Comparing eXtreme Programming and Feature Driven Development in academic and regulated environments." Feature Driven Development (2004). Final paper for CSCIE-275: Software Architecture and Engineering, Harvard University May 17, 2004

Authors’ Profiles

Shabib Aftab is working as a Lecturer in Computer Science Department at Virtual University of Pakistan. He got his MS degree in Computer Science from ‘COMSATS Institute of Information Technology’, Lahore. Previously he has completed M.Sc Information Technology from ‘Punjab University College of Information Technology’ (PUCIT), Lahore. His areas of research are Data Mining and Software Process Improvement.

Zahid Nawaz is student of MS Computer Science with the specialization of Software Engineering in Virtual University of Pakistan. His areas of interest are Software Process Improvement and Agile Development Models.

Faiza Anwer is student of MS Computer Science with the specialization of Software Engineering in Virtual University of Pakistan. Her areas of interest are Software Process Improvement and Agile Development Models.

How to cite this paper: Zahid Nawaz, Shabib Aftab, Faiza Anwer," Simplified FDD Process Model", International Journal of Modern Education and Computer Science(IJMECS), Vol.9, No.9, pp. 53-59, 2017.DOI: 10.5815/ijmecs.2017.09.06

