Using FDD for Small Project: An Empirical Case Study

Shabib Aftab[1], Zahid Nawaz[2], Faiza Anwer[3], Munir Ahmad[4], Ahmed Iqbal[5], Ashfaq Ahmad Jan[6], Muhammad Salman Bashir[7]
Department of Computer Science, Virtual University of Pakistan, Lahore, Pakistan

Abstract—**Empirical analysis evaluates the proposed system via practical experience and reveals its pros and cons. Such type of evaluation is one of the widely used validation approach in software engineering. Conventional software process models performed well till mid of 1990s, but then gradually replaced by agile methodologies. This happened due to the various features, the agile family offered, which the conventional models failed to provide. However besides the advantages, agile models lacked at some areas as well. To get the extreme benefits from any agile model, it is necessary to eliminate the weaknesses of that model by customizing its development structure. Feature Driven Development (FDD) is one of the widely used agile models in software industry particularly for large scale projects. This model has been criticized by many researchers due to its weaknesses such as explicit dependency on experienced staff, little or no guidance for requirement gathering, rigid nature to accommodate requirement changes and heavy development structure. All these weaknesses make the FDD model suitable only for large scale projects where the requirements are less likely to change. This paper deals with the empirical evaluation of FDD during the development of a small scale web project so that the areas and practices of this model can be identified with empirical proof, which made this model suitable only for large projects. For effective evaluation, the results of FDD case study are compared with a published case study of Extreme Programing (XP), which is widely used for the development of small scale projects.**

Keywords—Agile models; feature driven development; FDD; empirical evaluation; comparative analysis

I. Introduction

Today the agile methodologies have taken over the conventional models in software industry [13-15]. It happened due to the features agile family offers, which the conventional models failed to provide. The conventional models performed well till the mid 1999s but in last two decades, software industry faced various challenges which were ultimately resolved by the agile models [19-22]. The limitations of conventional models include long development duration, less user interaction, no adaptability, high cost, and no response to the frequently change in user requirements [13-16]. Agile models valued those factors which were neglected in traditional models and ultimately diverted the focus from process to people [23-26]. Several agile models are used in software industry now-a-days including Scrum, Extreme Programming (XP), Dynamic System Development Model (DSDM), Crystal Method, Test Driven Development (TDD), and Feature Driven Development (FDD) [13-18]. Each agile model contains its own development architecture and is suitable for particular type of projects (small, medium, large) [15-16]. However all the models work under one umbrella and follows the practices, values, and principles suggested by "Agile Manifesto" [24-26]. This manifesto is considered as a parent document of all the agile process models and consists of twelve basic rules of software development [14-15]. These principles include: frequent team communication, customer satisfaction, and managing frequent changing requirements [13]. The agile teams are self-organized, in which members work in a close collaboration. Moreover agile manifesto also focuses on simple design and timely delivery with reliable and qualitative product [14]. The agile models follow the iterative nature where each iteration brings a working module of the upcoming product, also known as partial working software [27-28]. Iterative development is very helpful to satisfy the customer as well as for the developers as it brings the customer feedback earlier which keeps the development team on track. FDD is one of the widely used agile development models by the software industry [13], [15], [19]. It is considered a process oriented and client centric development model, which mainly focuses on designing and building aspects of software development [15], [19], [31-32]. FDD follows the well-known pattern called ETVX and consists of five phases, also known as processes [19], [29-30]. The phases include: 1) Develop an Overall Model, 2) Build a Feature List, 3) Plan by Feature, 4) Design by Feature and 5) Build by Feature. Each phase further includes various activities and tasks. Besides the advantages, the FDD process model has always been criticized by many researchers due to its heavy structure. It is claimed that its explicit dependency on experienced staff and rigid nature to handle changing requirements make it only suitable for medium to large scale projects [13], [15], [19]. Due to these limitations, many researchers have proposed its customizations and integrations with other software models. This research deals with an empirical experience of using FDD for the development of small scale project. The empirical results are compared with a published research which used XP for the development of small scale project. Comparison is performed so that it can be evaluated that how FDD is not suitable for small scale project? Such empirical comparison can also guide the researchers towards an exact route for the modifications as well as for the integration of FDD with other models to achieve the maximum benefits. The empirical results presented in this research can be used as a baseline for further empirical comparisons.

II. Related Work

FDD has been modified by many researchers due to its limitations such as heavy structure and rigid nature to handle small scale projects. However no significant analysis with empirical results is found which could identify those factors that why FDD is not suitable for small project? However many researches are available which have either customized the FDD model or integrated it with other models to reduce its limitations and to improve the results. Some of the studies are discussed here. Researchers in [1] presented Competitor Driven Development (CDD) which is a hybrid process model that integrated the practices from Extreme Programming (XP) and Feature Driven Requirement Reuse Development (FDRD). According to Authors the proposed CDD is a self-realizing requirement generation model which keeps track of market trends as well as competitor's next product launch to extract requirements. This model considers the market orientation of product to guess the product's success rate. In [2], authors proposed SCR-FDD by integrating the Scrum and FDD. This model has eliminated the weaknesses of both models by taking schedule related activities from Scrum and quality related activities from FDD. The proposed solution has resolved the issues regarding schedule, quality and deployment, which were considered as the big obstacles during the development and release of software product. In [3], the authors presented Feature-Driven Methodology Development (FDMD), a modified version of Feature Driven Development. The proposed model incorporated the features of object oriented approach with Situational Method Engineering (SME). In FDMD requirements are represented as features which are based on object oriented principles and defined by using action, result and object. Researchers in [4] proposed a modified version of FDD called Secure Feature Driven Development (SFDD). The proposed solution tried to cover security related issues of FDD by making some changes in classical FDD process model. The model has added an activity in each phase called "In-phase Security". Moreover two additional phases are also incorporated 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. In [5], researchers introduced the feature of reusability in FDD and proposed Feature Driven Reuse Development (FDRD). This model considers re-useable feature sets along with the new requirements. In [6], authors introduced an ontology based feature driven development model for semantic web application. This model used domain ontology concepts which are widely known in domain knowledge modeling. Each phase of the proposed model has ontology as a basic building block. Ontology languages like RDF and OWL helped to overcome language ambiguity and inconsistency. In [7], a case study is conducted to check the suitability of FDD process model for secure web site development. According to authors, integration of more iterative activities along with security practices in FDD can make it a suitable candidate for secure software development. Authors in [8] presented a framework to handle changing requirements efficiently which is based on Adaptive Software Development and Cognizant Feature Driven Development (CFDD). The proposed model is simple and easy to implement however it remained silent on other issues of FDD. In [9], researchers have presented 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. In [10], researchers presented a supporting tool to implement FDD. This tool allows the implementation in a multi-user web based environment in the form of sub processes. The proposed tool has ability to track the changes in requirements and map these modifications in design classes. In [11], the authors have performed a comparative analysis between Feature Driven Development and Adaptive Software Development. The comparison mainly focused on two aspects; software requirements and software construction. The primarily purpose of this comparison was to evaluate the degree of agility in these two agile models. According to this study, no specific practices are used for requirement elicitation and software construction in ASD however in FDD some predefined practices are available for that purpose. In [12] the authors introduced the security relevant features in Feature Driven Development by following the four step security strategy in FDD. According to authors, after successful integration of these security steps, FDD can be used for the development of security critical software.

III. FDD Process Model

FDD (Fig. 1) is one of the widely used agile models, particularly for the development of large and complex projects [13], [15]. This model develops the software according to client valued features. It follows eight best practices including: domain object modeling, development by feature, individual class ownership, feature teams, inspection, configuration management, regular builds and progress reporting. FDD consists of following phases [19].

"Develop an Overall Model" is the first phase in which context and scope of the project is finalized, for this purpose a high level walk through meeting is conducted [15]. After this activity, multiple object models are developed for the project by different domain experts, then one model is selected after detailed and critical review, in some cases more than one model are selected but then merged into a single one.

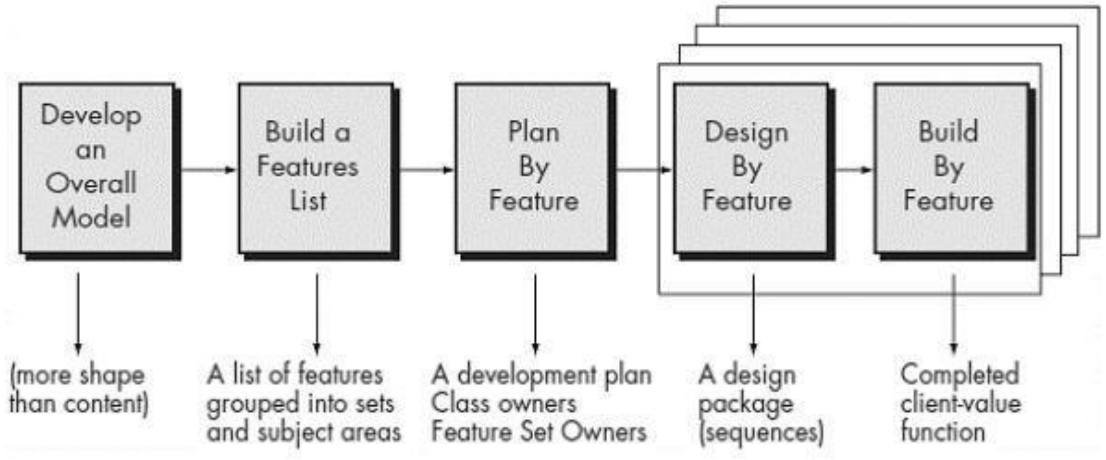
Develop an Overall Model
Build a Features List
Plan By Feature
Design By Feature
Build By Feature
(more shape than content)
A list of features grouped into sets and subject areas
A development plan Class owners Feature Set Owners
A design package (sequences)
Completed client-value function

Fig. 1. Feature Driven Development (FDD) Process Model.

The selected model is called domain model which can be further refined in later stages of development life cycle. “Build a Features List” is the second phase which deals with the development of feature list. As it name shows, FDD focuses on features, a feature is a valuable function which has some business value in software [19]. After the selection of domain object model in first phase, it is easier for the team to define a comprehensive list of features to be developed. These features are then grouped into a feature set, which is a collection of related features [19, 27]. Two weeks is the maximum time for the implementation of any feature, if it feels difficult to implement any particular feature within two weeks then it would be broken into two or more sub features. Finally all the documented features are approved by the customer. Plan by Feature is the third phase which deals with the planning to implement the features. The key activity of this phase is to assign priorities to features, so that the higher priority feature would be considered in early iterations. After priority assigning, each feature is checked against its business need which verifies that the features are according to the project’s requirements. In this phase followings things are also identified in features: dependencies, risk involved, complexity and team workload. Moreover features are assigned to developers which are known as Class Owners. Design by Feature is Fourth phase of the model and first phase of the iteration. An iteration can consists of one day to two weeks [15], [19]. This phase focuses on different activities such as: designing the sequence diagrams, writing the classes and refining the overall model Moreover different design packages are also produced against each class in this phase. Build by Feature is the second phase of iteration and last phase of FDD development life cycle. This phase deals with the actual implementation of features. After coding, code inspection and unit testing is performed. Classes are built in the sequence which was defined in plan by feature phase. When an iteration is completed successfully, the developed features are integrated with previously developed modules. FDD defines six key roles, five supporting roles and three additional roles. Key roles include: project manager, chief architect, development manager, chief programmer, class owner and domain experts. Supporting roles are: release manager, language guru, build engineer, tool smith and system administrator [15], [19]. Additional roles include: tester, deployer and technical writer. Document and artifacts produced during FDD life cycle are Features list, Design packages, Track by feature chart and Burn up chart.

IV. Empirical Evaluation

The purpose of this research is to evaluate the FDD process model with an empirical analysis during the development of small scale project. In order to effectively highlight the weaknesses of FDD, the empirical results are compared with the results of another published case study in which XP is used to develop small web based project. XP is one of the widely used agile models particularly for the development of small scale projects. This comparison would help us to pin point the issues which are limiting the use of FDD only for large projects. Characteristics of both the selected case studies are given in Table I.

TABLE I. Case Studies Detail

Characteristics	**FDD**	**XP**
Product Type	Human Resource Management	Real Estate Management
Size	Small	Small
Iterations	4	3
Programming Approach	Object Oriented	-
Language	C#, ASP.NET	PHP
Documentation	MS Office	MS Office
Testing	Browser Stack	-
Web Server	IIS	Apache Wamp Server
Project Type	Average	Average
Team Size	5 Member	3 Member
Feedback	Weekly	-
Development Environment	Visual Studio 2012	Macromedia Dream Viewer and Net Beans
Other Tools	MS Visio	MS Visio
Reports	Crystal Report	-

The FDD case study which is discussed in this research was the part of an academic research project, in which agile models were implemented to develop the client oriented projects for empirical analysis. That project was implemented in a software development company situated in Islamabad city, capital of Pakistan. The software company had experienced staff with higher degrees of computer science disciplines as well as with the dominating knowledge of software development. The development teams in that software company were using agile models for most of the projects. XP case study is taken from [26], in which one project is developed with three different models. FDD case study is implemented by the team which had significant experience of agile development as required by the model. On the other hand, XP case study was implemented by the computer science students of BS and MS programs, where the team had less or no experience of agile development however training session of 10 days was organized. The empirical results of both the case studies are presented in Table II.

Aggregated and Partial results of the developed project with FDD are already discussed in [15]. However this study reflects the complete experiment including detailed empirical results of all iterations by keeping in view the guidelines extracted from [13], [16], [33-35]. The reason of choosing the XP for comparison is it's widely acceptance by the software industry particularly for small projects. In this way it would be easy to identify the practices of FDD which need the customization or modification to effectively deal with small projects. In Table II, first column shows the numbers in series and second column represents the metrics/attributes which are observed and measured for both the models in each release. These metrics are used to analyze the developed product from various aspects including development time, cost, working, productivity, quality, effectiveness and efficiency [13], [16], [36-39]. The last column shows average/cumulative values of the attributes from all releases. The remaining columns (release 1 to release 4) reflect the values of attributes from column 2 in each release for both the models.

TABLE II. EMPIRICAL RESULTS

Sr. No	**Software Metric**	**Release 1**		**Release 2**		**Release 3**		**Release 4**	**Total**	
		FDD	**XP**	**FDD**	**XP**	**FDD**	**XP**	**FDD**	**FDD**	**XP**
1	Completion Time (weeks)	1	2	1	1	1	1	1	4	4
2	Number of Modules	2	2	1	1	2	1	1	6	4
3	No of User Stories	21	17	12	13	15	11	9	57	41
4	Budgeted Work Effort (h)	200	240	200	120	200	120	200	800	480
5	Actual Work Effort (h)	175	210	175	90	175	90	175	700	390
6	Number of User Interfaces	5	2	3	1	2	1	2	12	4
7	No of Classes	5	46	5	34	4	30	4	18	110
8	Lines of Code	4200	4500	3300	3200	2760	3300	2550	12810	11000
9	KLOC	4.2	4.5	3.3	3.2	2.7	3.3	2.5	12.8	11
11	No of Code Integrations	12	20	12	12	10	12	8	42	44
12	Post Release Defects	4	2	3	2	3	4	2	12	8
13	Post Release defects / KLOC	0.952	0.44	0.909	0.625	1.111	1.212	0.8	0.937	0.727
14	Productivity (= line of code/ actual time spent in hours)	24	21.4	18.9	35.6	15.8	36.7	14.6	18.3	28.2
16	No of Pre-release Change Requests	4	3	2	2	3	2	1	10	7
17	Total Change requests/KLOC	0.952	0.66	0.606	0.62	1.11	0.60	0.4	0.781	0.636
18	Time to Implement Changes (h)	5	4	4	3	3	1	2	14	8

V. Critical Analysis

The detailed empirical results are shown in Table II, which reflect the significant differences in some of the important software metrics. The size, nature and complexity level of both the projects were same however FDD model showed poor performance as compared to XP. KLOC of the application which is developed using FDD are 12.8 with the actual effort of 700 hours. However with XP model, 11 KLOC are produced in 390 hours (Fig. 2, 3). Actual effort in each release of both the models is also shown in Fig. 4.

There were five members in FDD project as compared to three in XP. Moreover the team members in FDD were experienced with agile development as that case study was implemented in a software house, however on the other hand members in XP project were hardly familiar with agile and got the training of ten days just before the development. This reflects the poor performance of FDD process model in small project. The KLOC produced in FDD project were higher than XP project with the difference of 1.8 but this does not justify the difference of 310 hours in actual effort which FDD team consumed even after having two more members than the XP team. These factors point out the heavy structure of FDD process model due to which it consumes more resources in small project.

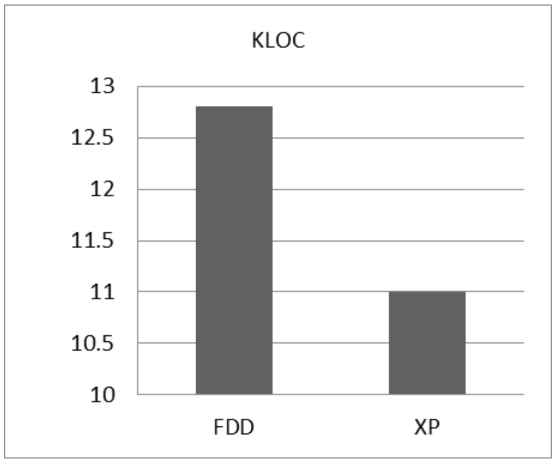

Fig. 2. KLOC.

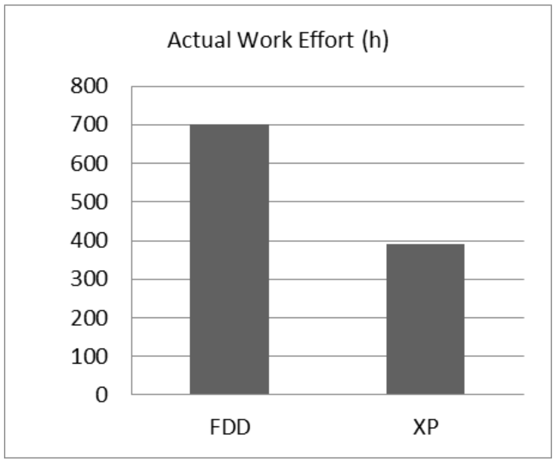

Fig. 3. Actual Work Effort.

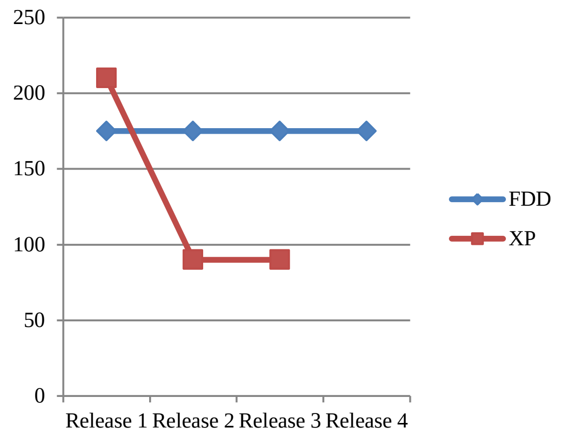

Fig. 4. Release wise Actual Work Effort.

The no of user stories (requirements) implemented in FDD project are 57 and with XP this no is 41 (Fig. 5). Moreover the no of code integrations in FDD are 42 whereas in XP this no is 44. There are more requirements in FDD project but less code integrations as compared to XP case study.

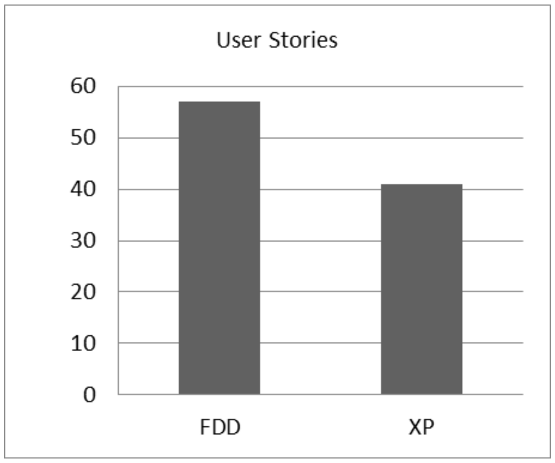

Fig. 5. No of user Stories.

The no of design classes shows the development approach adopted by the team particularly when viewed along with KLOC, implemented user stories, no of code integrations and designed interfaces. FDD team implemented 18 classes whereas XP team completed the development with 110 no of classes (Fig. 6). The no of user interfaces designed in FDD are 12 and in XP this no is 4. In comparison with XP, FDD completed the development with 16 more requirements, 8 more interfaces but with 92 no of less designed classes. This shows that the FDD team did not performed well in design by feature phase where classes are written as no of code integrations are almost same in both the projects.

The defects which appear at the client side after the release is also considered an important quality parameter which contributes to the ultimate satisfaction of the customer. The application developed with FDD showed 12 defects whereas with XP the no of defects are 8 (Fig. 7). This metric also raises the question on the quality aspect of FDD as it took much more time for development with experienced and large team as compared to XP.

Software productivity is an important metric which shows the whole team effort during the development period. It reflects that how much work the team has done in a particular time interval (actual effort) to achieve the desired goal. However to judge the efficiency and effectiveness of the model this single parameter is not enough. All the parameters included in Table II collectively contribute to reflect the quality of the model. FDD team showed the productivity of 18.3 and XP showed 28.2 (Fig. 8). The release wise productivity of both the models is shown in Fig. 9. FDD showed poor productivity because it has taken more time for development (Actual effort) as compared to XP.

The overall results show the poor performance of FDD as compared to XP. The projects implemented in both case studies have same complexity level, nature (web based) and size however environment, team size, coding language and development tool are different.

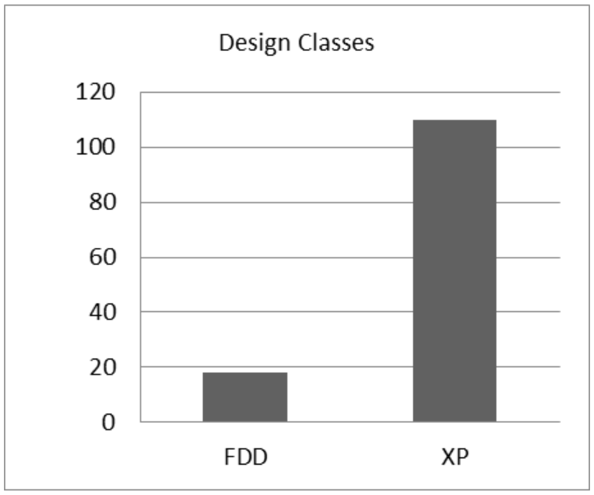

Fig. 6. No of Design Classes.

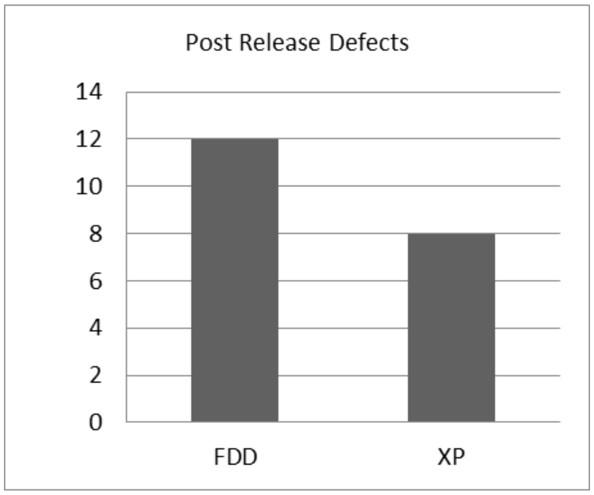

Fig. 7. No of Post Release Defects.

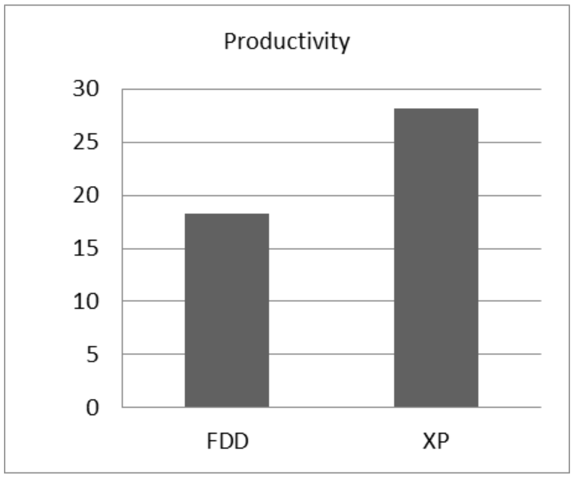

Fig. 8. Productivity.

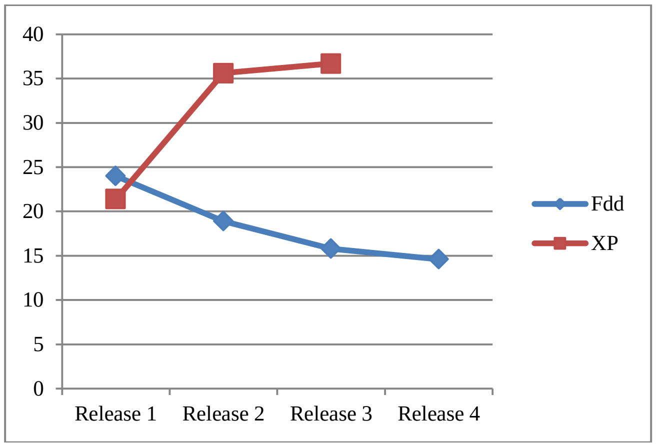

Fig. 9. Release wise Productivity.

There may be various causes of the poor performance of FDD process model. One is definitely the heavy structure due to which it could not perform well even with the more team members and with more time for development as compared to XP. The Complexity level of both the projects were same according to best of our knowledge however there may be a chance that FDD project was more difficult as it had more user interfaces but if its complexity level was higher than XP project even then the debatable thing is that the FDD team consisted of 5 experienced members with agile as compared to 3 non experienced members of XP. Moreover FDD project was developed in a software house with professional environment whereas XP project was implemented by graduate and undergraduate students in a lab. And finally after all the extra resources the FDD model consumed, the released product showed more defects as compared to XP. The communication issue or miss management can also be the reasons for poor performance in FDD as it has been seen that FDD team has designed very less classes as compared to XP and definitely the change management procedure, defect removal process and component based development as well as testing could be easy with appropriate no of classes.

VI. Conclusion and Future Work

FDD is one of the widely used agile models in software industry particularly for large scale projects. This model is criticized by many researchers due to its weakness such as: dependency on experienced staff due to its complex structure, rigid nature to accept requirement changes at later stages, little or no guidance for requirement extraction, and heavy development structure. All these limitations make this model only suitable for large scale projects where requirements are less likely to change. This paper evaluated the FDD process model on a small scale project through an empirical case study. The purpose of this study is to identify those areas and practices through empirical analysis which are limiting the use of FDD model to the large projects. In this study, the results of FDD case study is compared with a published case study of Extreme Programing (XP), which is a well-known agile model for the development of small projects, so that the performance of FDD can be evaluated effectively. According to the empirical analysis the performance of FDD is poor in almost every important quality metric as compared to XP. There could be many reasons of poor performance of FDD including heavy development structure, complex project, mismanagement of practices and communication issues among team members. However in our point of view the root cause of poor performance directly or indirectly is the heavy and rigid structure as well as the complex practices of FDD. This research can be used as a baseline for further empirical comparisons of FDD.

References

[1] V. P. Doshi and V. Patil, “Competitor driven development: Hybrid of extreme programming and feature driven reuse development,” 1st Int. Conf. Emerg. Trends Eng. Technol. Sci. ICETETS 2016 - Proc., pp. 1–6, 2016.

[2] S. S. Tirumala, S. Ali, and A. Babu, “A Hybrid Agile model using SCRUM and Feature Driven Development,” Int. J. Comput. Appl., vol. 156, no. 5, pp. 975–8887, 2016.

[3] R. Mahdavi-Hezave and R. Ramsin, “FDMD: Feature-Driven Methodology Development,” Proc. 10th Int. Conf. Eval. Nov. Approaches to Softw. Eng., pp. 229–237, 2015.

[4] A. Firdaus, I. Ghani, and S. R. Jeong, “Secure Feature Driven Development (SFDD) Model for Secure Software Development,” Procedia - Soc. Behav. Sci., vol. 129, pp. 546–553, 2014.

[5] S. Thakur and H. Singh, “FDRD: Feature driven reuse development process model,” Proc. 2014 IEEE Int. Conf. Adv. Commun. Control Comput. Technol. ICACCCT 2014, no. 978, pp. 1593–1598, 2015.

[6] F. Siddiqui and M. Afshar Alam, “Ontology based application model for feature driven development,” Proc. 5th Indian Int. Conf. Artif. Intell. IICAI 2011, pp. 1125–1137, 2011.

[7] A. Firdaus, I. Ghani, and N. I. M. Yasin, “Developing Secure Websites Using Feature Driven Development (FDD): A Case Study,” J. Clean Energy Technol., vol. 1, no. 4, pp. 322–326, 2013.

[8] K. Kumar, P. K. Gupta, and D. Upadhyay, “Change-oriented adaptive software engineering by using agile methodology: CFDD,” ICECT 2011 - 2011 3rd Int. Conf. Electron. Comput. Technol., vol. 5, pp. 11–14, 2011.

[9] F. Kanwal, K. Junaid, and M. A. Fahiem, “A {Hybrid} {Software} {Architecture} {Evaluation} {Method} for {FDD} - {An} {Agile} {Process} {Model},” 2010 {International} {Conference} {Computational} {Intelligence} {Software} {Engineering}, pp. 1–5, 2010.

[10] M. Rychlý and P. Tichá, “A tool for supporting feature-driven development,” Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 5082 LNCS, pp. 196–207, 2008.

[11] A. F. Chowdhury and M. N. Huda, “Comparison between adaptive software development and feature driven development,” Proc. 2011 Int. Conf. Comput. Sci. Netw. Technol. ICCSNT 2011, vol. 1, pp. 363–367, 2011.

[12] M. Siponen, R. Baskerville, and T. Kuivalainen, “Integrating Security into Agile Development Methods,” Proc. 38th Annu. Hawaii Int. Conf. Syst. Sci., vol. 00, no. C, p. 185a–185a, 2005.

[13] S. Aftab, Z. Nawaz, F. Anwer, M. Salman, M. Ahmad, and M. Anwar, “Empirical Evaluation of Modified Agile Models,” Int. J. Adv. Comput. Sci. Appl., vol. 9, no. 6, pp. 284–290, 2018.

[14] F. Anwer, S. Aftab, M. S. Bashir, Z. Nawaz, M. Anwar, and M. Ahmad, “Empirical Comparison of XP & SXP,” IJCSNS Int. J. Comput. Sci. Netw. Secur., vol. 18, no. 3, pp. 161–167, 2018.

[15] S. Aftab, Z. Nawaz, M. Anwar, F. Anwer, M. S. Bashir, and M. Ahmad, “Comparative Analysis of FDD and SFDD,” Int. J. Comput. Sci. Netw. Secur. (IJCSNS), vol. 18, no. 1, pp. 63–70, 2018.

[16] S. Ashraf and S. Aftab, “Pragmatic Evaluation of IScrum & Scrum,” I.J. Mod. Educ. Comput. Sci. Mod. Educ. Comput. Sci., vol. 1, no. 1, pp. 24–35, 2018.

[17] F. Anwer and S. Aftab, “Latest Customizations of XP: A Systematic Literature Review,” Int. J. Mod. Educ. Comput. Sci., vol. 9, no. 12, pp. 26–37, 2017.

[18] S. Ashraf, “Scrum with the Spices of Agile Family: A Systematic Mapping,” Int. J. Mod. Educ. Comput. Sci., vol. 9, no. 11, pp. 58–72, 2017.

[19] Z. Nawaz, S. Aftab, and F. Anwer, “Simplified FDD Process Model,” Int. J. Mod. Educ. Comput. Sci., vol. 9, no. 9, pp. 53–59, 2017.

[20] S. Ashraf, “IScrum: An Improved Scrum Process Model,” Int. J. Mod. Educ. Comput. Sci., vol. 9, no. 8, pp. 16–24, 2017.

[21] F. Anwer, S. Aftab, and I. Ali, “Proposal of Tailored Extreme Programming Model for Small Projects,” Int. J. Comput. Appl., vol. 171, no. 7, pp. 23–27, 2017.

[22] S. Ashraf and S. Aftab, “Latest Transformations in Scrum: A State of the Art Review,” Int. J. Mod. Educ. Comput. Sci., vol. 9, no. 7, pp. 12–22, 2017.

[23] F. Anwer and S. Aftab, “SXP: Simplified Extreme Programing Process Model,” Int. J. Mod. Educ. Comput. Sci., vol. 9, no. 6, pp. 25–31, 2017.

[24] F. Anwer, S. Aftab, S. Shah Muhammad, S. Shah Muhammad Shah, and U. Waheed, “Comparative Analysis of Two Popular Agile Process Models: Extreme Programming and Scrum” Int. J. Comput. Sci. Telecommun., vol. 8, no. 2, 2017.

[25] F. Anwer, S. Aftab, U. Waheed, and S. S. Muhammad, “Agile Software Development Models TDD , FDD , DSDM , and Crystal Methods : A Survey,” Int. J. Multidiscip. Sci. Eng., vol. 8, no. April, 2017.

[26] G. Rasool, S. Aftab, S. Hussain, and D. Streitferdt, “eXRUP: A Hybrid Software Development Model for Small to Medium Scale Projects,” J. Softw. Eng. Appl., vol. 06, no. 09, pp. 446–457, 2013.

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

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

[29] S. R. Palmer and M. Felsing, A Practical Guide to Feature Driven Development. 2002.

[30] B. Boehm, “A Survey of Agile Development Methodologies,” Laurie Williams, pp. 209–227, 2007.

[31] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, “Agile Software Development Methods: Review and Analysis,” 2017.

[32] D. Ph, “Major Seminar on Feature Driven Development Agile Techniques for Project Management Software Engineering By Sadhna Goyal Guide : Jennifer Schiller Chair of Applied Software Engineering,” p. 4, 2007.

[33] M. R. J. Qureshi, “Estimation of the New Agile XP Process Model for Medium-Scale Projects Using Industrial Case Studies,” Int. J. Mach. Learn. Comput., vol. 3, no. 5, pp. 393–395, 2013.

[34] S. U. Nisa and M. R. J. Qureshi, "Empirical Estimation of Hybrid Model: A Controlled Case Study," Int. J. Inf. Technol. Comput. Sci., vol. 1, no. July, p. 8, 2012.

[35] M. Qureshi, "Empirical Evaluation of the Proposed eXSCRUM Model: Results of a Case Study," Int. J. Comput. Sci. Issues., vol. 8, no. 3, pp. 150–157, 2012.

[36] N. E. Fenton, and S. L. Pfleeger, "Software Metrics: A Rigorous and Practical Approach: Brooks," 1998.

[37] S. H. Kan, Metrics and models in software quality engineering. Addison-Wesley Longman Publishing Co., Inc. 2002.

[38] C. Jones, "Applied Software Measurement", McGraw Hill, 1991.

[39] N. Fenton and J. Bieman, "Software Metrics: Roadmap," It Prof., vol. 2, pp. 38–42, 2014.