Evaluation for Feature Driven Development Paradigm in Context of Architecture Design Augmentation and Perspective Implications

Shahbaz Ahmed Khan Gahyyur[1], Abdul Razzaq[2]
Department of Computer Science and Software Engineering
International Islamic University Islamabad
Islamabad, Pakistan

Salman Ahmed[4]
Department of Computer Science and Software Engineering
International Islamic University Islamabad
Islamabad, Pakistan

Syed Zeeshan Hasan[3]
Department of Computer Science & Software Engineering
Faculty of Basic and Applied Sciences
International Islamic University
Islamabad, PAKISTAN

Rafi Ullah[5]
Department of Computer Science & Software Engineering
Faculty of Basic and Applied Sciences
International Islamic University
Islamabad, PAKISTAN

***Abstract*—Agile is a light weight software development methodology that is useful for rapid application development which is the need of current software industry. Since the focus of agile software development is the customer but it does not provide the detailed information about the application's architecture and documentation, so software architecture has its own benefits and use of it has many positive effects. The focus of this paper is to provide a systematic mapping of emerging issues in feature driven development that arises due to lack of architecture support in agile methodology and proposed solution's model. Results of this mapping provides a guideline for researcher to improve the agile methodology by achieving the benefits employed by having an architecture in place that is aligned with agile values and principles. Following research addresses to implement the SEI architecture centric methods in FDD methodology in an adapted form, such that the burden of architecture doesn't affect the agility of FDD. And the researcher found the de-motivators of agile which helps to understand the internal cycle and reduces the issues to implement the architecture. This study helps to understand the difference between architecture and FDD. This researcher mapping creates awareness about the process improvement with the combination of architecture and FDD.**

Keywords*—Software architecture; agile; architecture and agile; integration of architecture and agile; agile architecting practices

I. Introduction

Agile practices have gained popularity among various organizations due to its feature of reducing cost and encouraging change during the development cycle. In modern software development environment, changes to any software product are inevitable [39]. Agile methodology provides answer for this issue. Feature driven development lies under the umbrella of Agile. FDD is a process for assisting teams in producing features incrementally that are useful for the end user. It is extremely iterative and collaborative in nature [5]. The FDD process has extensive guidelines for identifying issues in the system. It also supports in providing builds to the end user on daily or weekly to add more features to the existing software. FDD process requires configuration management for its proper execution because features are being developed in parallel. In this way, integration of the features is made easy while executing the process. Feature Driven Development provides activity tracking support. Activities can include coding, design or testing. Details of this process are reflected in Fig. 1. Feature tracking is implemented by assigning the value ranging from 0 to 1 to the feature. 0 shows that this feature has not yet been developed and 1 depicts the completed feature [1].

Literature defines the software architecture as "The architecture of a software-intensive system is the structure or structures of the system, which comprises software elements, the externally visible properties of those elements, and the relationships among them" [3]. Software architecture defined by IEEE 1471 standard is "The fundamental organization of a system embodied in its components, their relationships to each other and to the environment, and the principles guiding its design and evolution" [7].

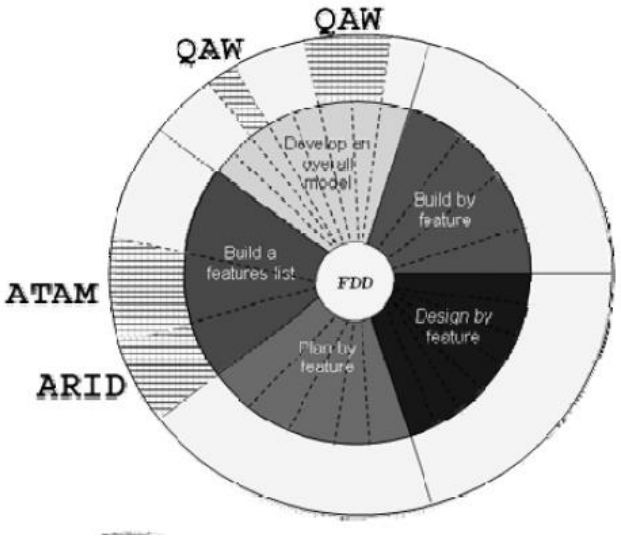

Fig. 1. Hybrid FDD with architecture evaluation methods [16].

A. FDD (Feature Driven Development)

Feature driven Development is a procedure for helping groups deliver visit, substantial working outcomes. It utilizes little squares of customer esteemed usefulness called highlights. It sorts out those little pieces into business-related capabilities. Fig. 1 demonstrates the half and half FDD with an engineering assessment. FDD centers engineers around creating working outcomes at regular intervals. FDD is better arranged to work with group where engineers' experience shifts. It offers advance following and announcing abilities. This solaces supervisors and makes it more alluring for enormous organizations [3].

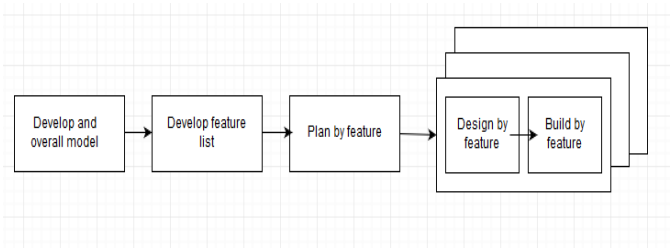

Fig. 2. Feature driven development [8].

- Process # 1: Develop an overall model

The first step of the FDD procedure is to make a detailed model of the system. The clue is for both field and progress members of the team to increase a worthy, shared understanding of the tricky domain. Fig. 2 shows all the phases flow of FDD.

- Process # 2: Build a feature list

The initial step of the FDD procedure is to manufacture an itemized model of the framework to be produced, which catches the partners' suspicions and necessities. The sign is for both field and advance individuals from the group to build a commendable, shared comprehension of the precarious space. Fig. 2 demonstrates every one of the stages stream of FDD [3].

- Process # 3: Plan by feature

Manager of project, Development Manager, and Chief Programmers design the request that the highlights are to be executed, in light of highlight conditions, stack over the improvement group, and the intricacy of the highlights to be actualized [3].

- Process # 4: Design by feature

Highlights different features are planned for improvement by doling out them to a Chief Programmer. Boss Programmer plans little gathering of high spot at once for enhancement. Such a gathering of highlights shapes a Chief Programmer's work Package. The Chief Programmer at that topic an element group by distinguishing the proprietors of the classes (designers) liable to be associated with the improvement of the chose feature(s). The central Programmer at that point refines the protest demonstrate in view of the substance of the succession diagram(s). The engineers compose class and technique prefaces [3].

- Process # 5: Build by feature

Working of the design plan bundle delivered amid the Design by Feature process, the class proprietors execute the things fundamental for their class to help the outline for the feature(s) in the work bundle. The code created is then unit tried and code examined, the request of which is controlled by the Chief Programmer. After an effective code assessment, the code is allowed to build [3].

B. Architecture – centric methods

IEEE 1471 standard [6] explains software architecture as "The fundamental organization of a system exemplified in its components [40], their relationships to each other and to the environment, and the principles managing its design and evolution".

The software architecture serves as the outline or skeleton of a software system to be built [8], [9]. The benefits of software architecture include a tool for stakeholder communication [7], designing decision documents, identifying design decision risks, reusing [10] scalability [2], allows to program, saving time, the cost of correction or reprocessing is recorded and above all, it helps to avoid software catastrophes [4].

C. Need of Systematic Mapping

In this paper, systematic mapping was explored to find problems that are faced during measurement of individual's performance. PSP quality principles were explored during systematic mapping which can be used for individual's performance measurement in agile team. Main purpose of study is to calculate architecture support provided in feature driven development that resides under the umbrella of agile [12], and how researcher can achieve benefits of architecture using agile methodologies without compromising the agile values [32]. This paper also describes how to cumulate the knowledge by performing systematic mapping study, there are few steps such as "planning", "conduct the research" and "selection of primary study".

II. Literature and Background Study

Architecture-centric approaches feature early suspicion, arranging and documentation of programming engineering. This infers a specific accentuation on quality traits and outline choices, exercises depend on across the board correspondence and joint effort among partners [13].

In Agile procedures clients or individuals are focal point of focus [35]. Touching delivery of working programming is need over weighted documentation and reports. Testing obliges the conveyance of each little working programming units and successive customer criticism on these product units enables keeping the product to extend on right track and lined up with objectives of the client. XP, Scrum, Feature Driven development are few examples of agile methodologies.

Since agile approaches have important influence on software development practices from industry perspective. However, there is also a prominent impact regarding issues that arises due to lack of SA, which is considered most important artifacts in traditional software development. Many industry professionals who are involved in using agile approaches view software architecture from the perspective of the plan- driven development paradigm [38]. According to them, software architecture design and evaluations requires too much effort which has very little impact to customer's needs from the system [37]. Hence, they view architectural activity as a part of highly formal processes. On the other hand, practitioners of software architecture believe that solid architectural practices cannot be followed using agile methods [36]. However, recently there is an increased appreciation related to the importance of incorporating architectural practices in agile methods. Hence, there is a growing interest and research in this perspective of integrating these two different cultures [11]. Researcher trust that a decent comprehension of current industry practices and issues identified with software architecture is a most important for building strategies to incorporate architecture in agile methodologies [31]. Literature has additionally highlighted a Hybrid Software Architecture Evaluation Method for FDD [33], [34]. Utility trees, affectability focuses and tradeoffs are the characteristic highlights of ATAM [18], [19].

III. Research Method

A. Rational

Researcher undertake the study to improve/evaluate the tailored feature driven development methodology by integrating software architecture support that was originally part of traditional software development so that organizations using FDD can also achieve benefits that are provided by Software architecture. Since software architecture is a very heavy activity which is against the agile core principles so a light weight version of software architecture has been proposed and evaluation will be made on this tailored FDD process as against with traditional FDD process. There is limited published research that validates and measures the impact of integrating architecture in FDD without compromising agile values, and this case study sought to contribute to the body of research in this area.

B. Objective of the Study

The objective of this study is to evaluate the impact of integrating architecture in FDD methodology with respect to reusability, cost, effort, requirement traceability and project failure risks due to unknown domain and untried solutions. Researcher proposed the solution model in proposed solution section.

C. Factors Analysis Method

Researcher used the Minitab static tool [49] for finding and analyzing the results of factors of Agile. Minitab tool helps to create the different types of graphs which help to understand the scenario of factors. Researcher provided the complete results of all factors in Appendix 'A' part and factor analysis result table. Moreover, Appendix 'B' section show result in different graphs [49].

D. Planing of Mapping

In this mapping, issues have been gathered that arises due to lack of architecture in agile methodology with reference to feature driven development (FDD). This mapping will help us to evaluate the benefits [49] that can be achieved if architecture support is provided in agile development.

E. Research Questions

RQ1. What are the problems that can be effectively resolved by integrating architecture in agile methodologies?

RQ2. What are the mapping drawbacks of agile with architecture?

Drawbacks of agile have been discussed in Table IV.

RQ3. What are the mapping limitations and benefits of agile with architecture?

Limitations and benefits of agile is discussed in Table V.

RQ4. What are the emerging challenges have been reported in literature about FDD?

Emerging challenges have been discussed in Table VI.

RQ5. What are the demotivators factors in agile have been reported in literature?

De-motivators have been discussed in Table VII.

F. Search Strategy

Computing databases become the basis for searching primary studies. Following search strings and keywords are used in these databases.

G. Keywaords

The following keywords are used for searching the studies: {architecture}, {architecture centric method}, {agile}, {Feature Driven development}, {FDD}, {integration}, {incorporation}, {combination}, {effect}, {influence}, {Values}, {principles}.

H. Search String(s)

1) {Architecture centric method} AND {agile} OR {Feature Driven development}.

2) {Integration} OR {incorporation} OR {combination} of {architecture} AND {agile} OR {Feature Driven development}.

3) {Agile issues} OR {software architecture benefits} OR {agile drawbacks} OR {agile problems}.

IV. Selection of Primary Study

A. Search Engine

Search strings are put in advanced search of following software engineering databases: IEEE, ACM, Science direct, Springer and Google Scholar. Fig. 3 shows all the digital libraries.

Fig. 3. Databases for paper selection.

B. Inclusion Criteria

Research papers are selected based on their titles and abstracts. Following criteria will be used to select the papers.

- Research papers discussing the integration of agile and architecture at any level.
- Research papers that highlights project failure using agile methodology.
- Research papers relevant to agile values will be included.
- Research papers that discusses the architecture impact on reusability, cost, effort and requirement traceability.

C. Exclusion Criteria

These papers were excluded.

- Books and slides, etc. were excluded.
- Papers other than primary and irrelevant studies.

V. Conducting Mapping

Search results from different digital libraries are mentioned in Table I. These digital libraries were selected because they were highly known for having empirical studies and literature surveys and are most relevant to software engineering field [27]. Digital libraries search was made to include all the papers that identify agile issues, architecture benefits, or any other paper that discusses integration of both of them. After this initial search, papers were selected from the digital libraries based on the inclusion and exclusion criteria mentioned in Section IV. With further investigation of selected papers, researcher has filtered studies that are most appropriate to the problem in hand. Table I shows all the found publications. These filtered papers are shown in Table II. Relevant studies are shown below in Table III.

TABLE I. Publication Count

Database	Publications count
IEEE	80
ACM	105
Springer	65
Science Direct	110
Scopus	149
Google Scholar	290

TABLE II. Primary Studies

No	Reference	Primary study
1	[5]	FDRD: Feature Driven Reuse Development Process Model
2	[38]	A Applied Example of Attribute-Driven Design (ADD)
3	[14]	FORM: A Feature-Oriented Reprocess Method with Domain-Specific Reference Architectures
4	[15]	Foremost Functional Development Session Agile Techniques for Project Management Engineering Software
5	[3]	Software Architecture as a Set of Architectural Design Decisions
6	[11]	An experimental study of architectural practices and challenges in term of used ofagile software development approaches
7	[16]	Agile techniques, organizational culture and agility: few insights
8	[17]	Reuse in large-scale agile software development and different factors of Communication for speed
9	[18]	Software architecture and ASD: clash of two cultures?
10	[19]	Flexible Working Architectures: Agile Architecting Using PPCs
11	[20]	A systematic mapping study on the combination of software architecture and agile development

TABLE III. Relevant Studies

No	Reference	Relevant study
1	[21]	ASD with CBSE
2	[22]	Effort approximation in Agile software development: A survey on practices
3	[23]	On the Responsibilities of Software Architects and Software Engineers in an Agile Environment: Who Should Do What?

4	[24]	Perceived Productivity Threats in Large Agile Development Projects
5	[25]	The combined OPN and UML method for developing an agile manufacturing control system
6	[26]	Building Software Solutions in an Industrial Information System: The 5+1 Software Architecture Model
7	[27]	The necessary nature of product traceability and its relation to Agile approaches
8	[1]	Agile software development methods review and analysis
9	[7]	IEEE Std 1471-2000, Recommended Practice for the Architectural Description of High Intensity Systems
10	[2]	Get ready for agile methods, with care
11	[21]	Agile software development for component based software engineering
12	[23]	On the Responsibilities of Software Architects and Software Engineers in an Agile Environment: Who Should Do What?

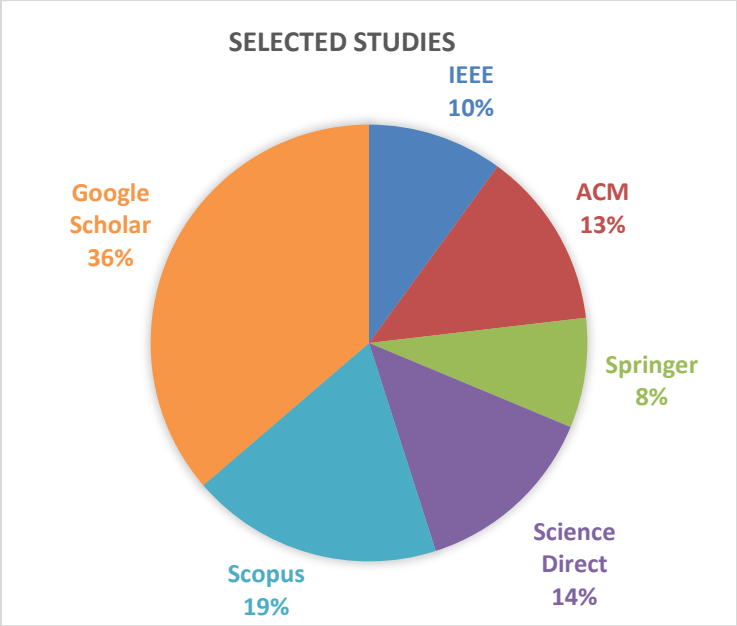

Fig. 4. Selected studies.

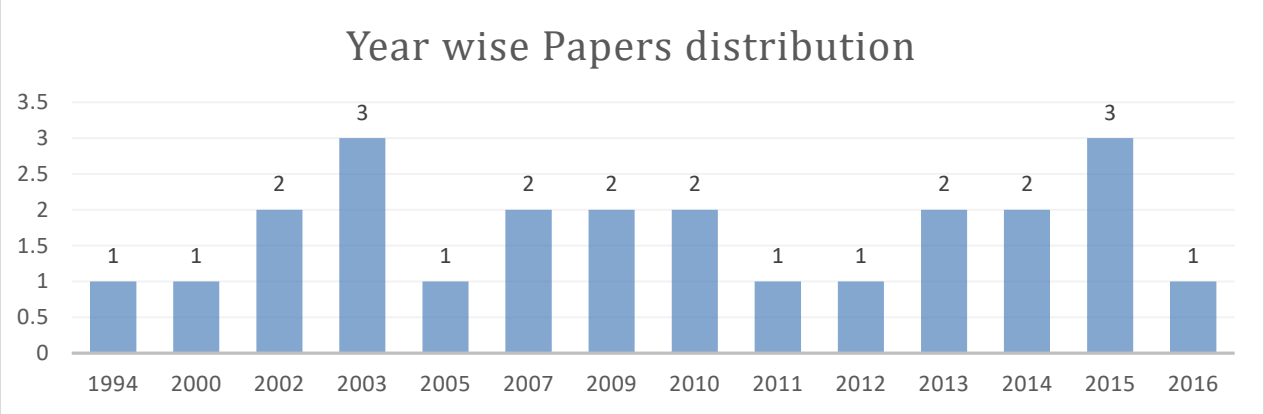

Fig. 5. Year wise paper distribution.

The total of primary and relevant study count that was included in this mapping is 23. These are the strong evidence that shows conflicting as in Fig. 4. Study source distribution on IEEE, Google Scholar, ACM, Springer, Science Direct and Scopus is displayed in the graph on the right side Fig. 3.

A. *Data Collection*

Data obtained from each study was:

- Source and full reference.
- Grouping of the study type (Agile architecture integration, Agile issues, Architecture benefits, Architecture agile conflict)
- Summary of each study that includes main research questions.

B. Data Analysis

The data was collected to show:

- Whether the study presents high level architecture support with evidence in feature driven development.
- Whether the study presents explained low level design support with evidence in feature driven development.
- Whether the study highlights any risks due to lack of architecture.
- Factors that are inherited by architecture but are against the agile values and vice versa.

VI. Mapping of Agile Drawbacks Related to Architecture

The issues are described in the below table (Table IV). By adding an architecture support in agile process, researcher can remove these drawbacks.

TABLE IV. Mapping of Agile Drawbacks Related to Architecture

Years in which issues are highlighted	Incapability to reuse components	Design erosion, knowledge and rationale vaporization	Risk of failure (or delayed feature distribution) in case of unknown domain	Highly experienced domain developers required for successful projects	Primary study references
2015	x				[5]
2013	x				[17]
2007		x			[15]
2014		x			[16]
2009			x	x	[11]

TABLE V. Mapping of Agile Limitations and Architecture Benefits

Sr. #	Agile limitations	Architecture benefits
1	Incapability to reuse components due to architecture discontinuities[5][17]	Documentation of Architecture is explicitly defined architecture discontinuities are limited and reusing component is made possible due to availability of documentation and design rationale[13] [14]
2	knowledge and foundation disappearance in Design destruction, as a result of ad-hoc design decisions documentations[15][16]	Design decisions' documentations addresses knowledge and design erosion, and rationale vaporization. [13][3]
3	In case of unidentified domain and novel solutions the Risk of failure [11]	[14] and [13] provides an unambiguous study of architectural decisions and a clear classification of user stories as quality scenarios and should decrease these risks.
4	Extremely skilled domain developers mandatory for successful projects[11]	[14] and [13] provides a step-by-step approach to architecture, which is also referred to as a plan-based approach, known for a person's exploration of the new domain.
5	Lack of requirements traceability[11]	[13] The step-by-step approach to architecture classifies the requirements according to their importance and documents them in the development of software

TABLE VI. Emerging Challenges of FDD

No.	Challenges	Ref.
1	Secure Development	[29]
2	Requirements gathering and managing	[29] [30]

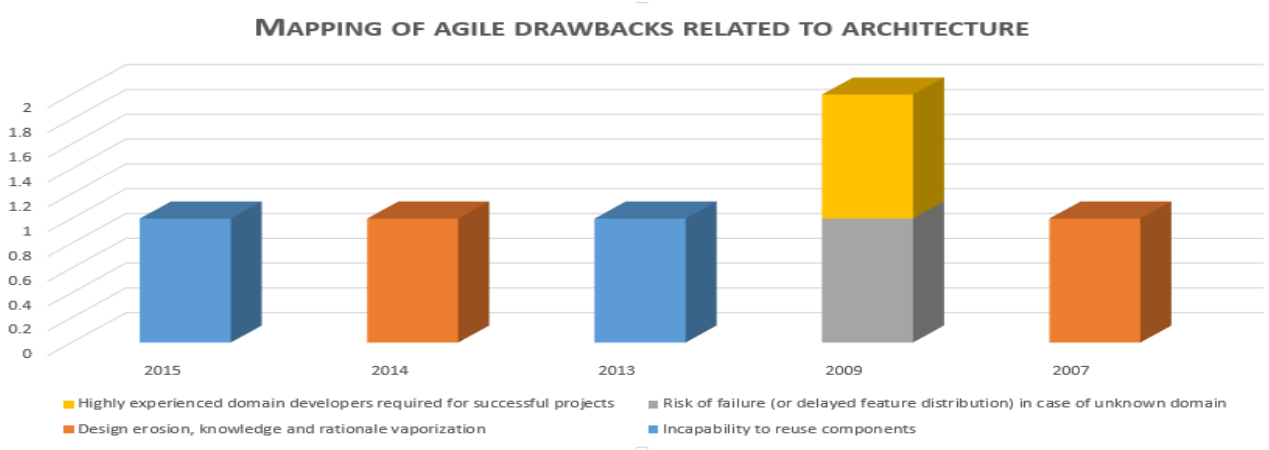

Fig. 6. Mapping of agile drawbacks.

TABLE VII. DE-MOTIVATORS OF AGILE FROM LITERATURE [50]

No	De-motivators factors	Ref.
1	Communication barrier	[22] [41][47] [48] [46]
2	Lack of relationship opportunities	[42] [43] [47] [45]
3	Unrealistic goals	[47] [45] [48]
4	Injustice in promotions	[47]
5	Poor quality software	[13] [47] [48]
6	Political environment	[44]
7	Uncompetitive pay	[47] [45] [48]
8	Unsupportive management	[45]
9	Lack of influence	[47] [45] [48]
10	Unfair reward system	[47] [45] [48]
11	Non-interesting work	[45]
12	personal preferences	[45] [48]
13	Risk	[11] [41] [47] [45] [48]
14	Stress	[45]

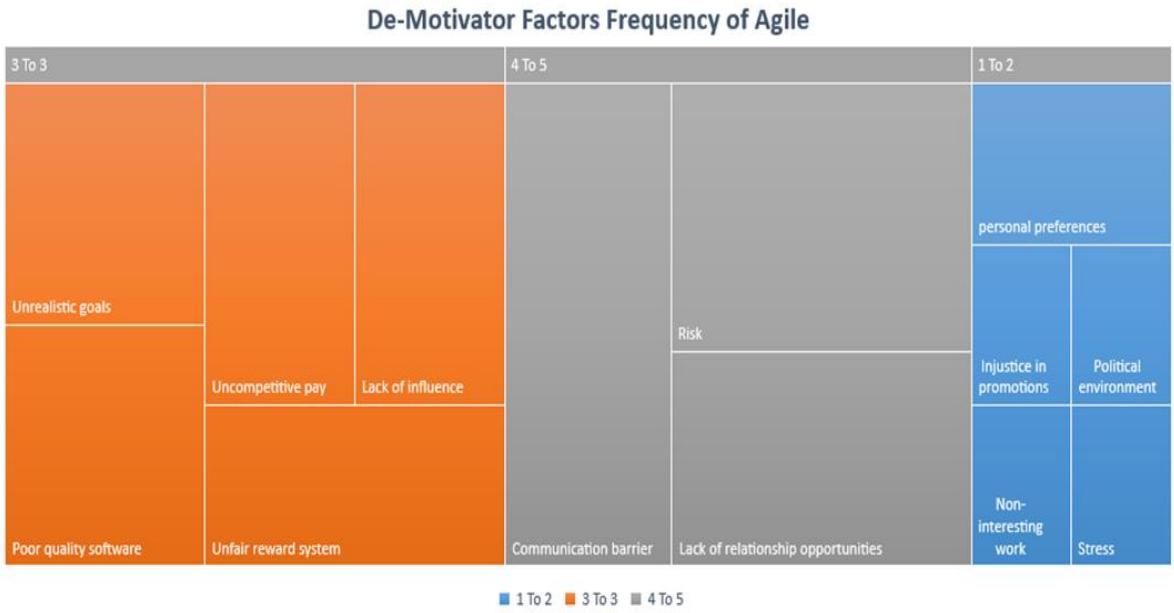

Fig. 7. Demotivator factors of agile.

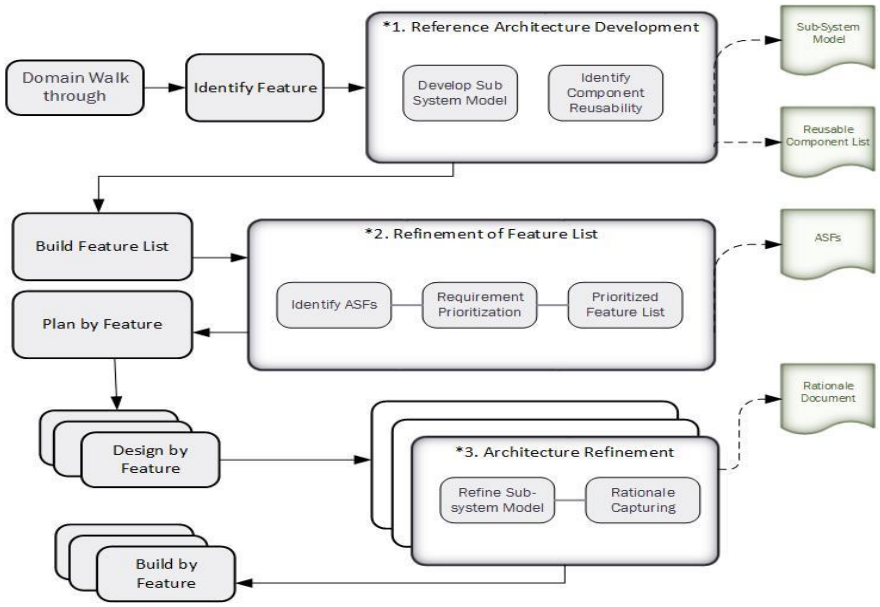

Fig. 8. Proposed solution.

VII. Proposed Solution Model

With the problem in hand, Researcher proposed the following model that suits the agility and embeds architecture support in FDD so that Researcher can achieve benefits of architecture without losing agility of Feature Driven development.

Following new artifacts have added in proposed process model:

Reference architecture development
1) Refinement of Feature List
2) Architecture refinement

Following new documents have produced in proposed model:

1) Sub-system model
2) Reusable component list
3) Architecturally significant Features (ASFs)
4) Rationale Document

Each sub process in the newly added artifacts is explained below.

A. Reference Architecture Development

1) Develop Sub-System Model

To develop sub-system model, engineering principles are used as an input to these models [28]. The engineering principles include design principles and general guidelines for subsystem design. Overall system structure is defined by grouping functions into subsystems, which are, then allocated to different hardware the model created for them is called subsystem model.

2) Identify component reusability

Reusability of the components and their fitness for large architecture is determined from subsystem model.

B. Refinement of Feature List

1) Identify ASFs

Indicators for architectural significance include:

- Extraordinary business value and/or technical risk.
- Important (influential, that is) stakeholder.
- budget overruns or client dissatisfaction

At the end of this activity, Researcher has a list of ASFs in hand to perform further processing based on this list.

2) Requirement Prioritization

Prioritization is done by ranking. Researcher gave each one a different numerical value based on its importance. For example, the number 1 can mean that the requirement is the most important and the number n can be assigned to the least important requirement, n being the total number of requirements based on the combined importance relevant to architecture and stakeholders. Researcher choose this method as it can be difficult to align different stakeholders' perspectives on what the priority of a requirement should be; taking an average can however, address the problem in this prioritization method.

3) Prioritized feature list

Prioritization done in the previous activity will listed down to form a Prioritized Feature list along the rationale of prioritization and get it approved from the concerned stakeholders.

C. Architecture Refinement

1) Refine sub-system model

Sub system model is refined in each iteration as the knowledge of stakeholder increases and issues they faced with the delivered iteration.

2) Rationale capturing

In refinement of sub-system model, every decision and change is documented in the specified template, so that back tracking is possible whenever needed.

VIII. Conclusion

In this paper, a systematic mapping of agile issues, the proposed model provides the detail to reduce these issues and architecture benefits have been presented so that researcher can addressed by integrating architecture in agile methodologies with reference to feature driven development. There are different types of architectural challenges reported in literature Table VI. Researcher has discussed in details about SA with FDD. Researcher discussed very clearly about drawbacks of agile related to architecture in Table IV. Researcher found the agile drawbacks and benefits of architecture in Table V. Fig. 5 shows the distribution of papers years wise. Fig. 6 shows the drawbacks in agile. Fig. 7 discusses the most important de-motivators of agile. Fig. 8 is the main thing that is proposed solution model. Minitab static tool is used to analyze the factors of agile which are mentioned in Fig. 7 and produce the result in the form of quantitative values and different views that graphs are Scree plot which is the major graph (Fig. 6) of this analysis. Other graphs and results are shown in Appendix section which are: Fig. 9, 10, 11, and 12. Research questions have been discussed through table for data. This mapping acts as a foundation for further research to incorporate architecture in agile methodologies in a way that is aligned with agile principles. In this systematic mapping, researcher described the concepts of Software Architecture with FDD.

IX. Future Work

Researcher will experiment based on the proposed model an adapted architecture that will be light weighted and can be integrated with feature driven development without harming the agility of this process. Researcher evaluated the proposed method and it proved to be useful in increasing reusability, traceability and also cost effective for middle sized projects. Moreover, this proposed process also puts positive effect on agile values and principles.

References

[1] 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.

[2] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, “Agile software development methods review and analysis,” VTT Publ., no. 478, pp. 3–107, 2002.

[3] 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.

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

[5] A. . Fallis, “No Title No Title,” J. Chem. Inf. Model., vol. 53, no. 9, pp. 1689–1699, 2013.

[6] I. A. W. Group, “IEEE Std 1471-2000, Recommended practice for architectural description of software-intensive systems,” p. i--23, 2000.

[7] J. Bosch, “Software Architecture : The Next Step,” Lect. Notes Comput. Sci., vol. 3047, pp. 194–199, 2004.

[8] “Len Bass, Paul Clements, Rick Kazman-Software Architecture in Practice-Addison-Wesley Professional (2003).” .

[9] B. Boehm, “Get ready for agile methods, with care,” Computer (Long. Beach. Calif)., vol. 35, no. 1, pp. 64–69, 2002.

[10] D. Garlan and M. Shaw, “An Introduction to Software Architecture,” Knowl. Creat. Diffus. Util., vol. 1, no. January, pp. 1–40, 1994.

[11] J. Melorose, R. Perroy, and S. Careas, “FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures,” Statew. Agric. L. Use Baseline 2015, vol. 1, pp. 1–28, 2015.

[12] P. O. Bengtsson and J. Bosch, “Scenario-Based Architecture Reengineering,” Proc. Fifth Int’l Conf. Softw. Reuse (ICSR 5), pp. 1–10, 1998.

[13] M. R. Barbacci, R. Ellison, A. J. Lattanze, J. a. Stafford, C. B. Weinstock, and W. G. Wood, “Quality Attribute Workshops, Third Edition,” Quality, no. August, p. 38, 2003.

[14] R. Kazman, L. Bass, and M. Klein, “The essential components of software architecture design and analysis,” J. Syst. Softw., vol. 79, no. 8, pp. 1207–1216, 2006.

[15] C. Hofmeister, P. Kruchten, R. L. Nord, H. Obbink, A. Ran, and P. America, “A general model of software architecture design derived from five industrial approaches,” J. Syst. Softw., vol. 80, no. 1, pp. 106–126, 2007.

[16] A. Tang, Y. Jin, and J. Han, “A rationale-based architecture model for design traceability and reasoning,” J. Syst. Softw., vol. 80, no. 6, pp. 918–934, 2007.

[17] X. Cui, Y. Sun, S. Xiao, and H. Mei, “Architecture design for the large-scale software-intensive systems: A decision-oriented approach and the experience,” Proc. IEEE Int. Conf. Eng. Complex Comput. Syst. ICECCS, pp. 30–39, 2009.

[18] M. a Babar, “An exploratory study of architectural practices and challenges in using agile software development approaches,” 2009 Jt. Work. IEEEIFIP Conf. Softw. Archit. Eur. Conf. Softw. Archit., pp. 81–90, 2009.

[19] W. G. Wood, “A Practical Example of Applying Attribute-Driven Design (ADD), Version 2 . 0,” Technology, vol. Version 2, no. February, p. 59, 2007.

[20] A. Jansen and J. Bosch, “Software Architecture as a Set of Architectural Design Decisions,” 5th Work. IEEE/IFIP Conf. Softw. Archit., pp. 109–120, 2005.

[21] L. Kompella, “Agile methods, organizational culture and agility: some insights,” Proc. 7th Int. Work. Coop. Hum. Asp. Softw. Eng. - CHASE 2014, pp. 40–47, 2014.

[22] A. Martini, L. Pareto, and J. Bosch, “Communication factors for speed and reuse in large-scale agile software development,” Proc. 17th Int. Softw. Prod. Line Conf. - SPLC ’13, p. 42, 2013.

[23] P. Kruchten, “Software architecture and agile software development: a clash of two cultures?,” 2010 ACM/IEEE 32nd Int. Conf. Softw. Eng., vol. 2, pp. 497–498, 2010.

[24] and P. P. A. Jennifer Pérez, Jessica Diaz, Juan Garbajosa, “Flexible Working Architectures: Agile Architecting Using PPCs,” October, vol. 2, pp. 1–20, 2003.

[25] C. Yang, P. Liang, and P. Avgeriou, “A systematic mapping study on the combination of software architecture and agile development,” J. Syst. Softw., vol. 111, pp. 157–184, 2016.

[26] W. Radinger and K. M. Goeschka, “Agile software development for component based software engineering,” Companion 18th Annu. ACM SIGPLAN Conf. Object-oriented Program. Syst. Lang. Appl., pp. 300–301, 2003.

[27] M. Usman, E. Mendes, and J. Börstler, “Effort estimation in Agile software development: A survey on the state of the practice,” ACM Int. Conf. Proceeding Ser., vol. 27–29–Apri, 2015.

[28] A. Tang, T. Gerrits, P. Nacken, and H. van Vliet, “On the Responsibilities of Software Architects and Software Engineers in an Agile Environment: Who Should Do What?,” SSE ’11 Proc. 4th Int. Work. Soc. Softw. Eng., pp. 11–18, 2011.

[29] J. E. Hannay and H. C. Benestad, “Perceived Productivity Threats in Large Agile Development Projects,” Proc. 2010 ACM-IEEE Int. Symp. Empir. Softw. Eng. Meas., no. 1325, pp. 1–10, 2010.

[30] M.-S. Lu and L.-K. Tseng, “The integrated OPN and UML approach for developing an agile manufacturing control system,” 2009 Int. Conf. Autom. Robot. Control Syst. ARCS 2009, pp. 24–31, 2009.

[31] S. B. A. Guetat and S. B. D. Dakhli, "Building Software Solutions in an Urbanized Information System: The 5+1 Software Architecture Model," Procedia Technol., vol. 5, no. 33, pp. 481–490, 2012.

[32] K. D. Palmer, "The essential nature of product traceability and its relation to Agile approaches," Procedia Comput. Sci., vol. 28, no. Cser, pp. 44–53, 2014.

[33] F. Kanwal, K. Junaid, and M. A. Fahiem, "A hybrid software architecture evaluation method for fdd-an agile process model," Comput. Intell. Softw. Eng. (CiSE), 2010 Int. Conf., pp. 1–5, 2010.

[34] R. L. Nord and J. E. Tomayko, "Software architecture-centric methods and agile development," IEEE Softw., vol. 23, no. 2, pp. 47–53, 2006.

[35] F. Breu, S. Guggenbichler, and J. Wollmann, The Agile Samurai. 2008.

[36] M. Fowler and J. Highsmith, "The agile manifesto," Softw. Dev., vol. 9, no. August, pp. 28–35, 2001.

[37] H. P. Breivold, D. Sundmark, P. Wallin, and S. Larsson, "What does research say about agile and architecture?," Proc. - 5th Int. Conf. Softw. Eng. Adv. ICSEA 2010, pp. 32–37, 2010.

[38] R. Wojcik and P. Clements, "Attribute-Driven Design (ADD), Version 2 . 0," Design, no. November, p. 55, 2006.

[39] A. en Claes Wohlin, Per Runeson, Martin Host, Magnus C.Ohlsson, Bjorn Regnell, Experimentation is Software Engineering. .

[40] D. Hristov, O. Hummel, M. Huq, and W. Janjic, "Structuring Software Reusability Metrics for Component-Based Software Development," no. c, pp. 421–429, 2012.

[41] Akhtar, M.J., Ahsan, A. and Sadiq, W.Z., 2010, October. Scrum adoption, acceptance and implementation (a case study of barriers in Pakistan's IT industry and mandatory improvements). In Industrial Engineering and Engineering Management (IE&EM), 2010 IEEE 17Th International Conference on (pp. 458-461). IEEE.

[42] Wagener, R.P., 2012. Investigating critical success factors in agile systems development projects (Doctoral dissertation, North-West University)..

[43] Chow, T. and Cao, D.B., 2008. A survey study of critical success factors in agile software projects. Journal of systems and software, 81(6), pp.961-971.

[44] Baddoo, N. and Hall, T., 2002. Motivators of Software Process Improvement: an analysis of practitioners' views. Journal of Systems and Software, 62(2), pp.85-96.

[45] Asghar, I. and Usman, M., 2013, December. Motivational and de-Motivational factors for software engineers: an empirical investigation. In Frontiers of Information Technology (FIT), 2013 11th International Conference on (pp. 66-71). IEEE.

[46] Beecham, S., Sharp, H., Baddoo, N., Hall, T. and Robinson, H., 2007, August. Does the XP environment meet the motivational needs of the software developer? An empirical study. In Agile Conference (AGILE), 2007 (pp. 37-49). IEEE.

[47] Beecham, S., Baddoo, N., Hall, T., Robinson, H. and Sharp, H., 2006. Protocol for a systematic literature review of motivation in software engineering. University of Hertfordshire.

[48] França, A.C.C., Gouveia, T.B., Santos, P.C., Santana, C.A. and da Silva, F.Q., 2011, April. Motivation in software engineering: A systematic review update. In Evaluation & Assessment in Software Engineering (EASE 2011), 15th Annual Conference on (pp. 154-163). IET.

[49] Shahbaz Ahmed, Abdul Razzaq, S. Ullah, S. Ahmed, Matrix Clustering Based Migration of System Application to Microservices Architecture, ijacsa, Jan, 2018.

[50] Ahmed, Shahbaz & Ahmed, Salman & Naseem, Adnan & Razzaq, Abdul. (2017). Motivators and Demotivators of Agile Software Development: Elicitation and Analysis. International Journal of Advanced Computer Science and Applications. 8. 10.14569/IJACSA.2017.081239.

FACTOR ANALYSIS RESULT REPORT

APPENDIX A

Factor: Communication barrier, personal preferences, Unrealistic goals, Poor quality software, Uncompetitive, Lack of influence, Unfair reward system, Uncompetitive pay, Lack of relationship opportunities, Risk, Political environment, Non-interesting work, Stress, Injustice in promotions

A. Principal Component Factor Analysis of the Correlation Matrix

1) Unrotated Factor Loadings and Communalities:

Variable	Factor1	Factor2	Factor3	Factor4	Factor5	Factor6
personal preferences	0.851	0.355	-0.007	0.068	-0.153	-0.028
Unrealistic goals	-0.102	0.279	0.579	0.496	0.273	-0.049
Poor quality software	-0.089	0.304	-0.517	-0.309	0.301	0.032
Uncompetitive pay	-0.140	0.552	-0.254	-0.408	0.335	-0.049
Lack of influence	-0.296	0.266	0.140	-0.214	-0.545	0.360
Unfair reward system	0.688	0.480	0.090	-0.308	-0.002	-0.072
Lack of relationship opportunit	-0.108	0.051	0.023	0.030	-0.665	-0.194
Communication barrier	0.152	-0.425	0.497	-0.495	0.170	-0.029
Risk	-0.235	0.430	0.471	0.192	0.332	0.262
Injustice in promotions	0.789	0.135	0.017	0.151	-0.009	0.002
Political environment	0.238	-0.412	0.408	-0.579	0.057	0.145
Unsupportive management	0.001	-0.296	-0.315	0.126	0.121	0.729
Non-interesting work	0.672	-0.135	-0.062	0.196	-0.034	0.418
Stress	-0.294	0.565	0.230	-0.226	-0.235	0.347
Variance	2.6294	1.8820	1.4870	1.3985	1.2537	1.0958
% Var	0.188	0.134	0.106	0.100	0.090	0.078

Variable	Factor7	Factor8	Factor9	Factor10	Factor11	Factor12
personal preferences	-0.011	0.063	-0.152	0.001	0.233	-0.098
Unrealistic goals	-0.045	0.117	-0.126	0.334	-0.275	-0.107
Poor quality software	-0.200	0.471	0.352	0.196	0.036	-0.065
Uncompetitive pay	-0.050	-0.427	-0.112	0.076	-0.305	-0.021
Lack of influence	0.285	0.428	-0.068	0.037	-0.216	-0.036
Unfair reward system	-0.125	0.125	-0.354	-0.041	0.060	-0.048
Lack of relationship opportunit	-0.659	-0.136	0.106	0.192	-0.047	0.014
Communication barrier	0.071	-0.033	0.092	0.418	0.239	-0.037
Risk	-0.348	0.109	0.173	-0.282	0.186	0.070
Injustice in promotions	0.126	0.018	0.271	0.116	-0.181	0.446
Political environment	-0.244	0.060	0.006	-0.264	-0.274	0.045
Unsupportive management	-0.248	-0.035	-0.351	0.204	0.059	0.141
Non-interesting work	0.006	-0.207	0.344	-0.028	-0.152	-0.349
Stress	0.198	-0.365	0.176	0.064	0.199	0.062
Variance	0.8782	0.8332	0.7019	0.5790	0.5417	0.3821
% Var	0.063	0.060	0.050	0.041	0.039	0.027

Variable	Factor13	Factor14	Communality
personal preferences	0.013	-0.172	1.000
Unrealistic goals	0.152	-0.014	1.000
Poor quality software	0.126	-0.008	1.000
Uncompetitive pay	-0.176	-0.051	1.000
Lack of influence	-0.175	-0.017	1.000
Unfair reward system	0.027	0.156	1.000
Lack of relationship opportunit	-0.037	0.004	1.000
Communication barrier	-0.167	0.002	1.000
Risk	-0.199	-0.003	1.000
Injustice in promotions	-0.046	0.013	1.000
Political environment	0.187	-0.057	1.000
Unsupportive management	0.014	-0.005	1.000
Non-interesting work	-0.057	0.046	1.000
Stress	0.251	0.012	1.000

Variance	0.2744	0.0630	14.0000
% Var	0.020	0.004	1.000

Factor Score Coefficients

Variable	Factor1	Factor2	Factor3	Factor4	Factor5	Factor6
personal preferences	0.324	0.189	-0.005	0.049	-0.122	-0.025
Unrealistic goals	-0.039	0.148	0.389	0.355	0.218	-0.045
Poor quality software	-0.034	0.161	-0.348	-0.221	0.240	0.029
Uncompetitive pay	-0.053	0.293	-0.171	-0.292	0.267	-0.045
Lack of influence	-0.112	0.142	0.094	-0.153	-0.435	0.329
Unfair reward system	0.261	0.255	0.060	-0.220	-0.002	-0.066
Lack of relationship opportunit	-0.041	0.027	0.016	0.021	-0.530	-0.177
Communication barrier	0.058	-0.226	0.334	-0.354	0.135	-0.027
Risk	-0.089	0.228	0.317	0.137	0.265	0.239
Injustice in promotions	0.300	0.072	0.011	0.108	-0.007	0.002
Political environment	0.091	-0.219	0.275	-0.414	0.045	0.132
Unsupportive management	0.001	-0.157	-0.212	0.090	0.097	0.665
Non-interesting work	0.256	-0.072	-0.041	0.140	-0.027	0.381
Stress	-0.112	0.300	0.155	-0.162	-0.188	0.317

Variable	Factor7	Factor8	Factor9	Factor10	Factor11	Factor12
personal preferences	-0.013	0.076	-0.217	0.001	0.429	-0.257
Unrealistic goals	-0.052	0.140	-0.179	0.578	-0.509	-0.280
Poor quality software	-0.228	0.565	0.502	0.338	0.066	-0.171
Uncompetitive pay	-0.056	-0.512	-0.160	0.131	-0.562	-0.054
Lack of influence	0.324	0.514	-0.097	0.063	-0.399	-0.095
Unfair reward system	-0.142	0.149	-0.505	-0.071	0.111	-0.125
Lack of relationship opportunit	-0.751	-0.164	0.152	0.331	-0.086	0.036
Communication barrier	0.081	-0.040	0.131	0.721	0.440	-0.097
Risk	-0.396	0.131	0.247	-0.488	0.343	0.183
Injustice in promotions	0.143	0.022	0.387	0.200	-0.333	1.167
Political environment	-0.278	0.072	0.008	-0.455	-0.506	0.116
Unsupportive management	-0.282	-0.043	-0.501	0.352	0.108	0.370
Non-interesting work	0.007	-0.248	0.490	-0.048	-0.281	-0.912
Stress	0.226	-0.438	0.250	0.111	0.367	0.162

Variable	Factor13	Factor14
personal preferences	0.046	-2.736
Unrealistic goals	0.554	-0.219
Poor quality software	0.460	-0.126
Uncompetitive pay	-0.640	-0.816
Lack of influence	-0.639	-0.271
Unfair reward system	0.097	2.479
Lack of relationship opportunit	-0.136	0.071
Communication barrier	-0.608	0.025
Risk	-0.727	-0.045
Injustice in promotions	-0.167	0.207
Political environment	0.682	-0.904
Unsupportive management	0.050	-0.085
Non-interesting work	-0.209	0.738
Stress	0.916	0.185

APPENDIX B

Fig. 9 shows the variation among all factors.

Fig. 10 provides the analysis of relationships amongst all the factors in the form of groups. This graph provides different relations in the context of positive and negative.

Fig. 11 tells the co-relationships in two ways amongst factors vertically and horizontally. These relationships are between two factors.

Fig. 12 biplot shows by using the both points.

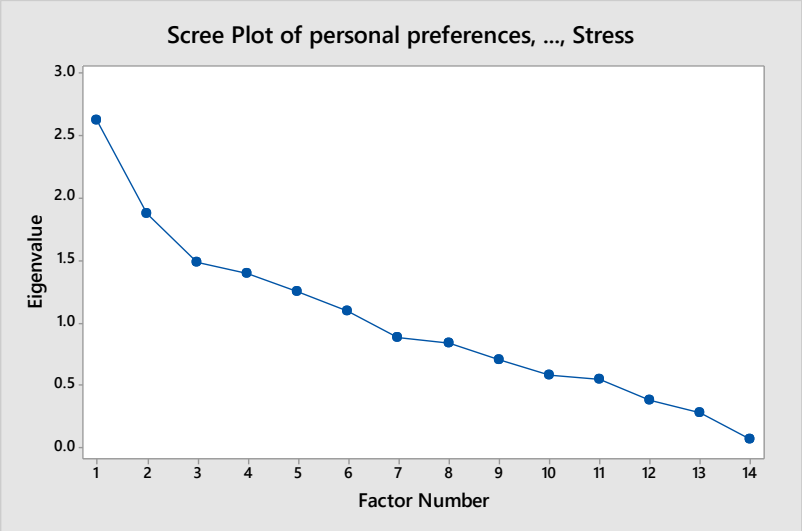

Fig. 9. Factors list analysis variation graph.

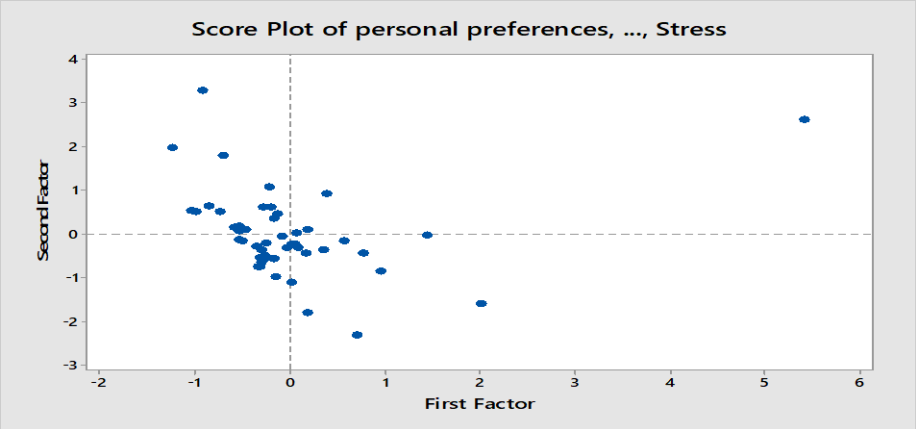

Fig. 10. Score plot.

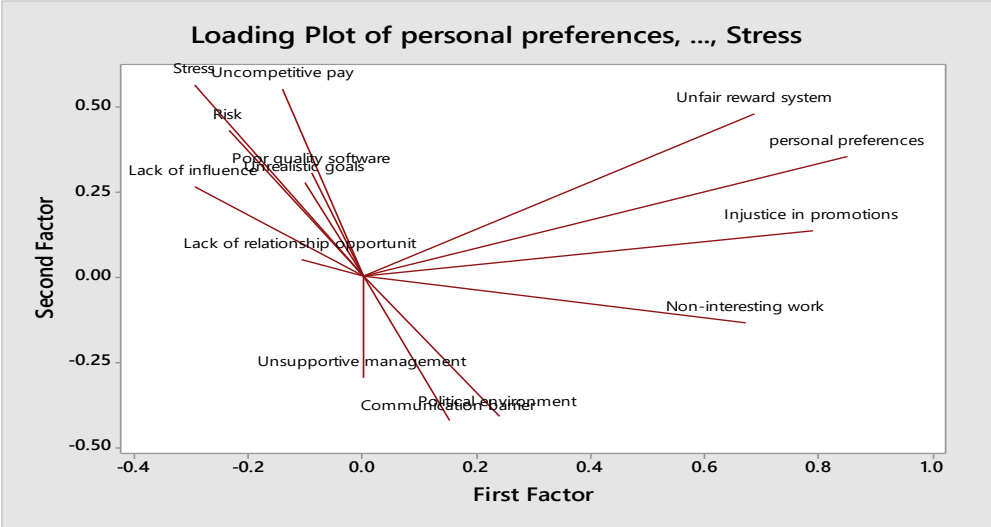

Fig. 11. Loading plot.

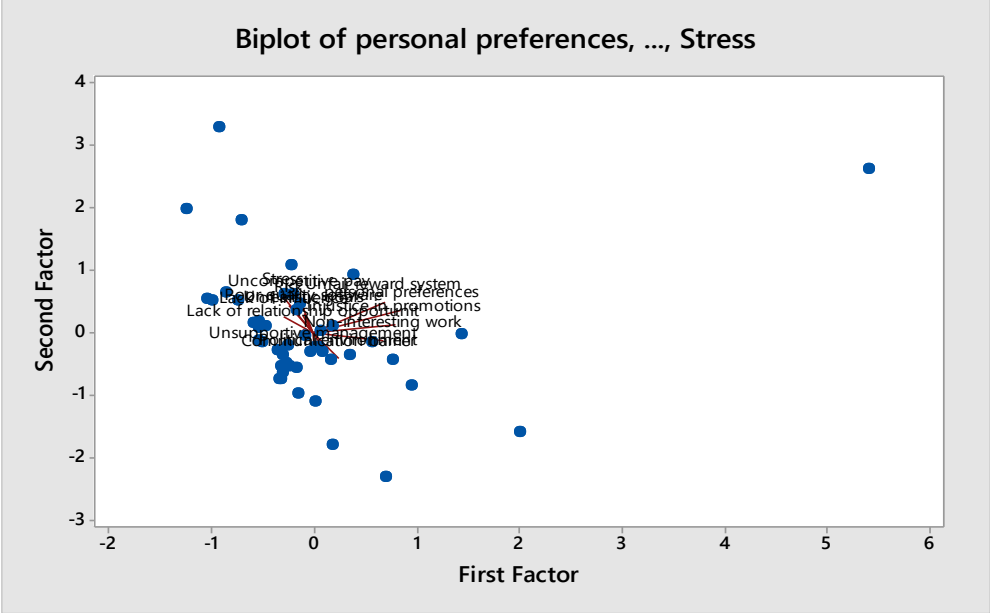

Fig. 12. Biplot.