

# Adaptation and Integration of Architecture in Feature-Driven Development



**Developed by:**

Syed Zeeshan Hasan

343-FBAS/MSSE/F12

**Supervised by:**

Mr. Shahbaz Ahmed Khan Ghayyur

Assistant Professor, DCS & SE, IIUI

**Department of Computer Science & Software Engineering Faculty of Basic and Applied  
Sciences International Islamic University, H-10 Islamabad**

2017



Accession No TH:18922 V/W



MS  
005.1  
HAA

Application software - Development .  
Computer " - "  
Agile methodology

# **Adaptation and Integration of Architecture in Feature-Driven Development**

**Syed Zeeshan Hasan**  
**Registration No. 343-FBAS/MSSE/F12**

Submitted in the partial fulfillment of the requirement for the

Master of Science in discipline of Software Engineering

At the faculty of Basic and Applied Sciences,

International Islamic University,

Islamabad

**IN THE NAME OF ALLAH, THE MOST MERCIFUL, AND BENEFICIENT**

**Department of Computer Science and Software Engineering  
International Islamic University, Islamabad**

Dated: 16-11-2017

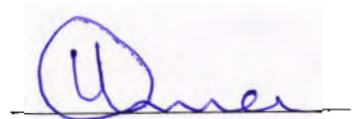
**Final Approval**

It is certified that we have read the thesis, title “**Adaptation and Integration of Architecture in Feature-Driven Development**” submitted by **Mr. Syed Zeeshan Hasan** Reg. No. **343-FBAS/MSSE/F12**. It is our judgment that the thesis is of sufficient standard to warrant its acceptance by the International Islamic University, Islamabad, for the degree of **Master of Science**.

**Examination Committee**

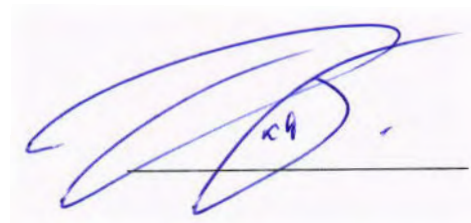
**External Examiner**

Dr. Muhammad Umer Khan  
Assistant Professor  
Dept. of Megatronics, Faculty of Engineering  
Air University, Islamabad



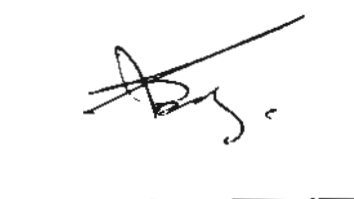
**Internal Examiner**

Dr. Qaisar Javaid  
Assistant Professor,  
DCS & SE, FBAS,  
International Islamic University Islamabad



**Supervisor**

Dr. Shahbaz Ahmed Khan Ghayyur  
Assistant Professor,  
DCS & SE, FBAS,  
International Islamic University Islamabad



## Table of Contents

<b>CHAPTER 1 INTRODUCTION</b>	<b>9</b>
<b>INTRODUCTION</b>	<b>10</b>
<b>1.1 FEATURE DRIVEN DEVELOPMENT</b>	<b>10</b>
<b>1.2 ARCHITECTURE – CENTRIC METHODS</b>	<b>12</b>
1.2.1 FEATURE-ORIENTED REUSE METHOD: (FORM)	12
1.2.2 SCENARIO BASED SOFTWARE ARCHITECTURE DESIGN: (SBAD)	15
1.2.3 QUALITY ATTRIBUTE WORKSHOP (QAW):	17
<b>1.3. ANALYTIC PRINCIPLES AND TOOLS FOR THE IMPROVEMENT OF ARCHITECTURE (APTIA):</b>	<b>19</b>
1.2.4 GENERAL MODEL FOR SOFTWARE ARCHITECTURE DESIGN (GMSAD):	20
1.2.5 ARCHITECTURE RATIONALE AND ELEMENTS LINKAGE (AREL):	22
1.2.6 ARCHITECTURE-BASED COMPONENT COMPOSITION/DECISION-ORIENTED DESIGN (ABC/DD):	24
<b>CHAPTER 2 LITERATURE REVIEW</b>	<b>29</b>
<b>LITERATURE REVIEW</b>	<b>30</b>
<b>2.1 SEARCH STRATEGY</b>	<b>30</b>
<b>2.2 KEYWORDS</b>	<b>30</b>
<b>2.3 SEARCH STRINGS</b>	<b>30</b>
<b>2.4 SEARCH ENGINE</b>	<b>30</b>
<b>2.5 INCLUSION AND EXCLUSION CRITERIA</b>	<b>30</b>
<b>2.6 SEARCH PROCESS</b>	<b>31</b>
<b>2.7 DATA COLLECTION</b>	<b>33</b>
<b>2.8 DATA ANALYSIS</b>	<b>33</b>
<b>2.9 PRIMARY STUDY TABLE</b>	<b>33</b>
<b>2.10 RELEVANT STUDY TABLE</b>	<b>34</b>
<b>2.11 QUALITY ASSESSMENT</b>	<b>35</b>
<b>2.12 QUALITY EVALUATION</b>	<b>35</b>
<b>2.13 SEARCH RESULTS</b>	<b>36</b>
<b>2.14 AGILE AND ARCHITECTURE</b>	<b>37</b>
<b>2.15 MAPPING OF AGILE DRAWBACKS AND ARCHITECTURE BENEFITS</b>	<b>38</b>
<b>CHAPTER 3 PROBLEM STATEMENT</b>	<b>40</b>
<b>PROBLEM STATEMENT</b>	<b>41</b>
<b>3.1 RESEARCH QUESTIONS</b>	<b>41</b>
<b>3.2 DISCUSSION WITH LITERATURE</b>	<b>42</b>

<b>CHAPTER 4 PROPOSED SOLUTION</b>	<b>44</b>
<b>PROPOSED SOLUTION</b>	<b>45</b>
<b>4.1 REFERENCE ARCHITECTURE DEVELOPMENT</b>	<b>46</b>
4.1.1 DEVELOP SUB-SYSTEM MODEL	46
4.1.2 IDENTIFY COMPONENT REUSABILITY	46
<b>4.2 REFINEMENT OF FEATURE LIST</b>	<b>46</b>
4.2.1 IDENTIFY ASFS	46
4.2.2 REQUIREMENT PRIORITIZATION	47
4.2.3 PRIORITIZED FEATURE LIST	47
<b>4.3 ARCHITECTURE REFINEMENT</b>	<b>47</b>
4.3.1 REFINE SUB-SYSTEM MODEL	47
4.3.2 RATIONALE CAPTURING	47
<b>4.4 DOCUMENT TEMPLATES:</b>	<b>48</b>
<b>CHAPTER 5 RESEARCH METHODOLOGY</b>	<b>52</b>
<b>RESEARCH METHODOLOGY</b>	<b>53</b>
<b>1.3 CASE STUDY</b>	<b>53</b>
<b>1.4 CASE STUDY DESIGN</b>	<b>53</b>
1.4.1 RATIONALE	53
1.4.2 OBJECTIVE OF THE STUDY	53
1.4.3 THEORETICAL FRAMEWORK	53
1.4.4 EXPLORATORY QUESTIONS	54
1.4.5 PROPOSITIONS AND HYPOTHESES	54
1.4.6 VARIABLES SELECTION	55
1.4.7 SELECTION OF SUBJECTS	56
1.4.8 METHODS OF DATA COLLECTION	56
1.4.9 INSTRUMENTATION	57
1.4.10 VALIDITY EVALUATION	57
<b>CHAPTER 6 ANALYSIS AND DISCUSSION</b>	<b>59</b>
<b>ANALYSIS AND DISCUSSION</b>	<b>60</b>
<b>6.1 PROCESS VS REUSABILITY</b>	<b>60</b>
<b>6.2 PROCESS VS TRACEABILITY</b>	<b>61</b>
<b>6.3 PROCESS VS COST AND EFFORT</b>	<b>62</b>
<b>6.4 EFFECT OF PROPOSED PROCESS VS TRADITIONAL PROCESS ON AGILE VALUES</b>	<b>63</b>
6.4.1 INDIVIDUALS AND INTERACTIONS	63
6.4.2 WORKING SOFTWARE	64
6.4.3 CUSTOMER COLLABORATION	65

6.4.4	RESPONDING TO CHANGE	66
6.5	EFFECT OF PROPOSED PROCESS VS TRADITIONAL PROCESS ON AGILE PRINCIPLES:	67
6.5.1	PEOPLE ORIENTATION	67
6.5.2	EMBRACE CHANGE	68
6.5.3	FREQUENT DELIVERY	69
6.5.4	COLLABORATION OF BUSINESS PEOPLE AND DEVELOPERS	70
6.5.5	MOTIVATION OF INDIVIDUALS	71
6.5.6	COMMUNICATION	72
6.5.7	WORKING SOFTWARE AS MEASURE OF SUCCESS	73
6.5.8	PROMOTE SUSTAINABLE DEVELOPMENT	74
6.5.9	KEEP ATTENTION TO TECHNICAL EXCELLENCE	75
6.5.10	SIMPLICITY	76
6.5.11	SELF-ORGANIZING TEAMS	77
6.5.12	IMPROVING EFFECTIVENESS	78
<b>CHAPTER 7 CONCLUSION AND FUTURE WORK</b>		<b>79</b>
<b>CONCLUSIONS</b>		<b>80</b>
<b>FUTURE WORK</b>		<b>80</b>



## List of Figures

Figure 1-1: Feature Driven Development [4] .....	11
Figure 2-1: Selected Studies .....	32
Figure 2-2: Year Wise Paper Distribution .....	32
Figure 2-3: Hybrid FDD with architecture evaluation methods [16].....	38
Figure 4-1: Proposed Solution .....	45
Figure 6-1: Process vs Reusability.....	60
Figure 6-2: Process vs Traceability.....	61
Figure 6-3: Process vs Cost and effort.....	62
Figure 6-4: Individuals and Interactions .....	63
Figure 6-5: Working Software .....	64
Figure 6-6: Customer Collaboration .....	65
Figure 6-7: Responding to Change .....	66
Figure 6-8: People Orientation.....	67
Figure 6-9: Embrace Change .....	68
Figure 6-10: Frequent Delivery.....	69
Figure 6-11: Collaboration.....	70
Figure 6-12: Motivation of Individuals.....	71
Figure 6-13: Communication .....	72
Figure 6-14: Working Software .....	73
Figure 6-15: Promote Sustainable Development .....	74
Figure 6-16: Keep attention to technical excellence.....	75
Figure 6-17: Simplicity .....	76
Figure 6-18: Self-organizing Teams .....	77
Figure 6-19: Improving Effectiveness .....	78

## List of Tables

<b>Table 2-1: Selected Studies</b> .....	<b>31</b>
<b>Table 2-2: Primary Studies</b> .....	<b>34</b>
<b>Table 2-3: Relevant Studies</b> .....	<b>35</b>
<b>Table 2-4: Quality Evaluation Criteria</b> .....	<b>36</b>
<b>Table 2-5: Search Results</b> .....	<b>36</b>
<b>Table 2-6: Mapping of Architecture benefits and Agile Issues</b> .....	<b>39</b>

**Abstract**

In recent years, merging software architecture with agile has received attention of many researchers. However, there exists no detailed research on the architecture-agility concepts. Since proportioned time and effort should be given to the architecture activity depending on varying size and complexity of the software to be built. Feature-Driven Development (FDD) being an agile methodology do not discuss design process for architecture in great detail and lacks steps that are needed to deliver a detailed architectural design. Following research addresses to implement the SEI architecture centric methods in FDD methodology in an adapted form, such that the burden of architecture does not affect the agility of FDD. The outcome is an architecture centric methodology for FDD that delivers significant quality, architecture benefits and FDD compliance gains when compared with traditional FDD.

## **CHAPTER 1 INTRODUCTION**

## 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. 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[1].

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. FDD process also provides support for tracking the activities like coding, design and testing so that end user is aware of the progress about implemented and upcoming features. 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. [2].

### 1.1 FEATURE DRIVEN DEVELOPMENT

FDD process is useful for assisting teams in building features incrementally that are useful for the end user. It is based on small set of functionalities named as features. FDD proposes to organize those business related little features in small groups named as feature sets. FDD is best suited for organizations where developer's experience varies from one another. FDD also offers progress tracking that makes it more suitable for big organizations where managers are interested in checking the progress of team members. The details of FDD are as follows:

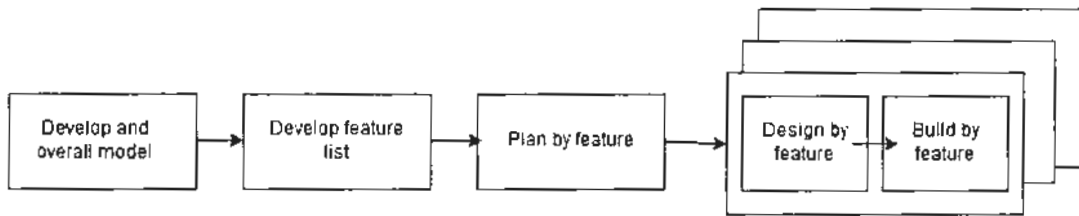


Figure 1-1: Feature Driven Development [4]

**Process # 1: Develop an overall model:**

The first step in Feature driven development is to perform development of overall model activity which is done by capturing stakeholder’s requirements and needs. The basic purpose of this activity is to perform better understanding of the problem in hand.

**Process # 2: Build a feature list:**

In this phase, a team is formed that has the responsibility to decompose the requirements. Domain experts of phase 1 has already partitioned the domain. Team that is formed in this phase breaks the requirements in to number of different areas which are named as feature sets. In each feature set, there are number of activities known as feature of the system. Feature also represents any valuable item for end user that has specific input and output. Results of this activity is the categorized feature list[3].

**Process # 3: Plan by feature:**

Higher management and architects of the organizations plan the features that needs to be implemented in which order depending on the feature dependencies and current available resources.[3]

**Process # 4: Design by feature:**

Features that are filtered out for the current iteration is further divided into small groups by chief programmer and is named as work package. This work package is assigned to developers who are expected to develop it. Developers then write the stubs for the assigned work packages[3].

**Process # 5: Build by feature:**

In this phase, developers build the necessary items to support the work package. Then this code is unit tested and inspected by the chief programmer. When there are no concerns on developed code and it supports the overall structure then it is allowed to build thoroughly [3].

**1.2 ARCHITECTURE – CENTRIC METHODS**

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”*. [6]

Software architecture acts as a outline or skeleton of a software system to be built [7]. Software architecture benefits include, a tool for stakeholders communication [8], design decisions documentations, risks identification of design decisions, basis for reuse [8], promotes scalability [9], enables scheduling which saves time, cost of correction or rework is saved and most importantly it helps avoiding software disasters [10].

Architecture centric approaches emphasize early anticipation, planning and documentation of software architecture. It involves explicit focus on quality attributes and design decisions, activities are based on extensive stakeholder communication and collaboration.

**1.2.1 FEATURE-ORIENTED REUSE METHOD: (FORM)**

It is an extension of Feature-Oriented Domain Analysis method which is named as Feature-Oriented Reuse Method (FORM). [11] Feature Oriented Domain Analysis (FODA) uses feature model for requirement engineering. However, FORM addresses the feature model for software design phase for development of domain architecture with reusable components. To design domain architecture with reusable components commonalities and differences across related software systems are discovered systematically. In FORM reusable architectures and components are

developed using process called *domain engineering*. In domain engineering the method analyzes the commonalities from the point of view of services, operating environment, operation and domain technologies. Commonalities that are identified are captured as features of product are arranged in the form of AND/OR graph the feature model is constructed in this way from this graph analysis. The domain architecture thus constructed from three viewpoints. Subsystem viewpoint is used to capture service features, Process viewpoint is used to capture operating environment features, and module viewpoint is used to capture feature associated with domain technology and implementation techniques. The FORM was applied on electronic two real life domains to investigate development of common domain language for developers and feature model for identifying reusable components. The feature model is build that helps in identifying commonalities and differences that is used to create domain artifacts for development of application. The feature model in this way can be used to produce feasible configurations of reusable architectures. These features of product are considered as first class objects for development. The FORM engineering processes involves domain and application engineering processes.

### **1-FORM Domain Engineering:**

Domain artifacts are developed for applications these artifacts increase the effectiveness of the reuse. In FORM domain engineering three main process are carried out which includes Domain Analysis, Reference Architecture Development and Reusable Component Development. The inputs and outputs of these processes are explained in detail in following section.

#### **Domain Analysis:**

Domain analysis involves identification of commonalties and differences in systems. Scope and intended use of the domain application is captured. The main processes involved in domain analysis are planning, feature analysis and validation activates. Planning process is concerned with the identification of family of products in a given domain. The commonalities and variabilities are identified based on the availability of standards and experts for that domain. Product features are identified in feature analysis process and validated with the help of validation activities. The product features are identified with the help of domain experts, design documents, or user manuals.



These features are then classified from the perspective of capabilities, domain and implementation techniques. To construct a feature model from these features a *graphical feature diagram* in AND/OR hierarchy is constructed and also *composition rules* are constructed that act as supporting document for feature diagram and finally the *issues, decisions, rational, and justifications* for selection of features are used. Feature dictionary is also the part of feature model. The feature model is then validated form application features of instantiated model. A validated *feature model* is the output artifact of domain analysis phase.

### **Reference Architecture Development:**

The development of Reference architecture in FORM consists of set of models which includes subsystem, process and module model. To develop these models *engineering principles* are used as an input to these models. The engineering principles include design principles and general guidelines for subsystem design, process design and module design. Categorizing similar functions to form a subsystem is done for defining overall system structure. These functions are then allocated to different hardware the model created for them is called *subsystem model*. The dynamic behavior of these subsystems is represented using *process model*. Finally, *module model* is created using different engineering principles such as modularity, data abstraction and information hiding.

### **Reusable Component Development:**

Reusability of the components and their fitness for large architecture is determined from subsystem process model. These reusable components are developed at module model level.

### **2-FORM Application Engineering:**

In this process application software is developed using domain knowledge which includes reference architecture and reusable components. The major process involved in application engineering are user requirement analysis, application architecture selection and application software development.

**User Requirements Analysis:**

User requirements analysis involves the feature selection of a product. The user requirements are matched with the feature model to trace the required feature for development.

**Application Architecture Selection:**

Reference architecture model is constructed through feature selection process. After that, subsequent step is to identify reusable modules with the help of specifications of modules. The application architecture is created from these reusable components and reference architecture.

**Application Software Development:**

Reusable components and application architecture is used to develop application software. The output of this process is the Application Software for the particular domain.

**1.2.2 SCENARIO BASED SOFTWARE ARCHITECTURE DESIGN: (SBAD)**

Scenarios Based Software Architecture Design (SBAD) [12] method is an iterative process for creating software architecture by applying design transformation. The method uses different techniques for evaluation of quality attributes of architecture such as scenarios, mathematical modeling, simulation, and reasoning. The method uses architectural transformations for iteratively assessing the achievement of quality attributes. Five different architectural transformations are applied in this method which includes architectural styles application, and application of architectural patterns and design patterns. The other architectural transformations that are applied are translating a quality requirement into functionality and dispensing quality requirement across different components or subsystems. The method is a rational design process that has the capabilities of matching quality requirements. The input of this method is requirement specifications document and output this method is architectural design. This architectural design is produced through application of various transformations which are applied iteratively. However, in first iteration only functional requirements are considered which results into an application architecture design. This architecture is evaluated and analyzed with respect to required quality attributes. This assessment may be quantitative or qualitative in nature. If the quality attribute requirements are up to the expectation the process is terminated architecture design is released. In

case quality requirements are not up to the expectation the architecture design process enters the second stage where architecture transformations are applied. The application of architectural transformation results into new version of architecture. This new version is then evaluated for required quality attribute requirements. If NFR not fulfill the loop is repeated otherwise the final architecture design version is released.

### 1- Functionality Based Architecture Design:

In this phase system architecture is designed by identifying the core abstractions of the system structure in top-down approach. These abstractions are modeled as objects using a creative design process which involves analysis of domain entities. These domain entities are then modeled as architecture entities and interactions between abstractions are identified.

### 2- Assessment of NFRs:

System NFRs are explicitly assessed using scenarios or experience based reasoning. The *scenario based* assessment of architecture is performed in three phases. The second approach recommended for assessment for quality attribute is *Simulation*. The third recommended approach is *Mathematical modeling* for quality attribute evaluation. The fourth approach recommended for assessing quality attribute is *experienced based reasoning* or *logical reasoning*.

### 3- Apply Architecture Transformation:

Architecture transformations are applied after the assessment of architecture properties. In this method five transformations are used which are:

- Architectural styles and patterns
- Design patterns
- Translation of quality requirements to working component
- Distribute requirements to subsystems.

Application of architectural styles results in software architecture reorganization. This reorganization result in certain quality attribute improvement but may lead to deprivation of another quality attribute. Second transformation that is applied is application of architectural patterns. This transformation affects the large part of the architecture by imposing certain rules. The third transformation is applied by utilizing different design patterns. However, this

transformation only affects small numbers of architectural components. In fifth transformation software quality requirements are concreted into functionality of the system. Finally, the distribute requirement transformation is applied by distributing quality requirements in number of components or subsystems.

#### 4- Evaluate Design:

Once the architecture transformations are applied on architecture it goes through assess quality attribute phase of the architecture design method. If all requirements meet the final version of the architecture is released for implementation.

### 1.2.3 QUALITY ATTRIBUTE WORKSHOP (QAW):

Quality Attribute Workshop [13] is an eight steps method for elicitation, identification and refinement of quality attribute of software intensive system. It is system-centric and its main focus is the system stakeholder for eliciting requirements regarding driving quality attributes of software architecture. The QAW engages systems stakeholders for better communication before the creation of the software architecture. The QAW has the ability to complement ATAM for analyzing architecture for tradeoffs points and sensitivity points. The method is not designed to provide absolute measure of quality. However, it provides a systematic way of elicitation, documentation, and prioritization of quality attributes. The system engineer uses these prioritized scenarios for analyzing architecture. It is their responsibility to prepare risk mitigation strategy and identify quality attribute concerns. QAW is a scenario based method and describes each scenario in term of stimulus, response and environment form. *Stimulus* is a factor that causes system to initiates, and reaction of this system is *response*. QAW is designed to address the challenges such as: precise meaning of quality attributes; elicit, identify, prioritize quality attributes; engaging system stakeholders in disciplined and repeatable process; processing and utilizing this information. The basic concern of QAW is system-level for identifying, prioritizing quality attributes. The QAW stakeholders group may range from five to thirty, during which they receive "*participant handbook*". The workshop requires focused and active participation of stakeholders.

#### Step-1: QAW Presentation

In this step facilitators and stakeholders have brief introduction about their role and responsibilities in the organization. Facilitator then presents standard slide presentation of QAW for the purpose of motivation, and to explain the steps involved in the method.

**Step-2: Present Business and/or Mission**

Facilitator in this step note the key quality attributes drivers during management presentation. In this step the management representative presents business concerns and/ or mission concerns of the system along with functional requirements, constraints and quality attribute in about an hour.

**Step-3: Present Architectural Plan**

In this step a technical stakeholder presents initial high-level system description and system context diagram. The technical representative presents architectural plan and strategies for meeting business and or mission requirements. While technical requirements and constraints of the system are presented facilitator captures information about architectural drivers.

**Step-4: Identify Architectural Drivers**

To identify architectural driver facilitator captures information regarding functional requirements, business concerns, mission concerns, goals, objectives, and system quality attributes. This information is transformed into list of key architectural drivers. The stakeholders then distilled the list of architectural drivers by some addition or deletion. This final list of distilled and key architectural drivers is used during subsequent brainstorming section.

**Step-5: Scenario Brainstorming Process**

Scenario brainstorming process is initiated by the facilitator. Facilitator reviews the generated scenarios by the stakeholders. During two round-robin passes of QAW at least two scenarios are contributed by the stakeholders. The facilitator ensures that collected scenarios are well formed and represented in the form of stimulus, response and environment.

**Step-6: Consolidate Scenario**



Similar scenarios are merged if facilitator finds that similar scenarios will not contribute anything. Therefore, consolidation prevents dilution of stakeholder's votes.

#### **Step-7: Prioritize Scenario**

Consolidated scenarios are prioritized by the system stakeholders. This prioritization of scenarios is based on voting activity which is done in round robin two passes. The number of votes determines the priority of the scenarios.

#### **Step-8: Scenario Refinement**

In scenario refinement process top five scenarios are refined and documented in the form of stimulus, its response, stimulus' origin and its surroundings, item which is stimulated and measurement of its response for that particular scenario. Additional information for these scenarios includes: business/mission goals.

### **1.3. ANALYTIC PRINCIPLES AND TOOLS FOR THE IMPROVEMENT OF ARCHITECTURE (APTIA):**

APTIA is a used [14] for improvement and design of software architecture which is based on reusable pre-existing components of ten techniques. The method has been applied on a commercial information system with real time requirements. This method is constructed on three principles of software architecture. The first principle is associated with abstraction of software architecture. The second principle is concerned with business and mission goals i.e. quality attribute requirements should be determined form business and mission goals. The third principle is concerned with design and analysis of architecture: The quality attribute needs directs the analysis and design of architecture. These principles and component techniques together are used in APTIA. These techniques are explained one by one in following section:

- 1- Unambiguous gathering of business goals.
- 2- Active participation of stakeholders.
- 3- Unambiguous gathering of architecture rationale
- 4- The realization of mission and business goals in 6-part quality attributes scenarios.

- 6- Use of architectural tactics.
- 7- Using templates to capture information.
- 8- Explicit elicitation of schedule, cost and benefit linked with architectural decisions.
- 9- The use of quality attributes models for architecture analysis and design decisions.
- 10- The use of design principles based on quality attribute models to identify alternatives for improvements.

The APTIA method provides more detailed analysis and design alternatives for improvement of architecture. The APTIA is created in such a way that it is split up into little steps with each step containing proven techniques. These techniques are considered as “component” which can be combined to create a new method. APTIA is one of such method that is created from these component techniques tailored for specific need. APTIA is has six phases. In its first phase ATAM is performed. In its second phase based on risk themes focus of analysis is determined. The third phase is concerned with use of quality attribute models for risk themes. In forth phase, model based analysis is used to suggest design alternatives. In fifth phase, design alternatives are being ranked based on cost and benefits. In final phase design decisions are made. These decisions are made based on costs/benefits associated with the architecture design using existing component techniques in an agile way. The APTIA therefore, provides a deeper analysis of architecture design and suggests design alternatives with new design principles. The consistency in design is achieved using two templates one template for documenting outputs of APTIA for analysis and second template for documenting outputs of APTIA for architectural alternatives.

#### **1.2.4 GENERAL MODEL FOR SOFTWARE ARCHITECTURE DESIGN (GMSAD):**

GMSAD [15] is based on the commonalities that can be found in five software architecture analysis and design methods. These methods are:

- Attribute driven design
- Rational software RUP 4+1

- Business architecture process and organization
- Siemens' 4 Views (S4V)
- Nokia Research's Architectural Separation of Concerns

The method provides a framework for comparison of strengths and weaknesses of these methods. The framework developed can be used for developing new methods. GMSAD is one such method that is developed using the framework which is based on the commonalities of the five industrial methods. The method presents three main and common activities in architectural design model which are:

**Architectural Analysis:** This is the first main activity whose inputs are Context and Concerns. The output of this activity is the architecturally significant requirements (ASRs). The activity is performed to articulate the ASRs. The architectural analysis activity is performed to filter the requirements that are not relevant to the architecture. Based on architectural concerns and system context a set of architecturally significant requirement are identified. The architectural problems are also analyzed in this activity.

**Architectural Synthesis:** Input to this method are ASRs and candidate architectural solutions is the result of this method. It is the core activity of architecture design. Based on architecturally significant requirements different solutions are proposed in this activity.

**Architectural Evaluation:** Candidate architectural solutions and ASRs are inputs to architectural evaluation. The Validated Architecture is the output of this activity.

In this phase the based on architecturally significant requirements, candidate architectural solutions are evaluated to validate that design decisions are correct. The evaluation is carried out repeatedly to ensure the validation of architecture. The artifacts and sub-activities involved in this model are:

**Architectural Concerns:** Architectural concerns reflect the interests of stakeholders. This can contain system consideration, mandated design decisions or regulatory requirements. All architectural concerns become the input to the architectural analysis activity.



**System Context:** IEEE 1471 standard's definition of context of system is used to determine situations of political or developmental/operational influences. The contexts and concerns together are the inputs to the architectural analysis activity.

**Architecturally Significant Requirements:** ASRs are extracted from system context or architectural concerns. The ASRs are those requirements that influence the software system architecture. Therefore, it is not necessary that all of the system requirements will be ASRs.

**Candidate Architectural Solutions:** Candidate architecture solutions are basically design decisions for software structure. Output of the architecture synthesis activity is the candidate architectural solutions which may be partial solution or alternative solution. These candidate architecture solutions include design rationale and its traceability.

**Validated Architecture:** Candidate solutions that are consistent with each other and are in line with ASRs together form a validated architecture. This validated architecture also includes design rationale.

Design information which is in the form of architectural styles or reference architecture or use of ADLs comes from the architect is an important input to design process. Analysis Knowledge in the form of analysis patterns and analysis models is another input to the design process. The other inputs that go to design process are the Knowledge of Evaluation Process and Realization Knowledge.

As the architecture design process progresses a backlog of issues or problems is built up. This backlog drives the architecture design process or workflow which is often non-linear. This backlog of need and issues is prioritized to resolve these problems. The architectural issues that are resolved by the architect are removed from backlog.

### 1.2.5 ARCHITECTURE RATIONALE AND ELEMENTS LINKAGE (AREL):

AREL [16] is developed to model architecture design rationale. This model is used to build linkage of Architecture rationale (AR) with Architecture elements (AE). AREL uses entity-relation diagram to model AR and AE entities. The modeling support is provided using UML. The AREL

provides an automated support for design reasoning traceability using Enterprise Architect. The traceability techniques in the AREL model are used for impact analysis and root-cause analysis. A traceable design rationale is used to trace relationships between the design objects. The rationale-based architecture model was designed to address issues such as conflicts, inconsistencies, and omissions in architecture design because of the absence of design rationale. The reasons behind architecture design decisions are captured using design rationale. The design rationale if capture and trace appropriately can help in understanding architecture design during verification and maintenance of large systems. Therefore, AREL was introduced to support architecture design rationale capture. The architecture design rationale is captured using quantitative design rationale and qualitative design rationale. Any arguments regarding design alternative whether in favor or against is captured using qualitative design rationale. However, for capturing the quantitative design rationale cost, benefit and risks associated with the design alternative are quantified.

The AREL model considers two forms of design reasoning one is motivational reasons and other is design rationale. Requirements, goals, constraints or design objects are considered as motivational reasons as they motivate in design. Main entities of conceptual model are Architecture Rationale; Motivational Reason; and Design Outcome. Architecture Rationale modeled as stereotype <<AR>>; Motivational Reason is modeled as <<AE>>; and Design Outcome is modeled as <<AE>>. <<AR>> encapsulates arguments, issues, questions or tradeoffs. Design Outcome <<AE>> is result of Decision made for Motivational Reason. The architecture element which is represented by AE, when participates in a decision in the form of input, it is called motivational reason, and AE when considered as outcome is called design outcome.

### **Architecture Elements: (AE)**

The architecture element AE is an artifact concerning business requirements, technical constraints or assumptions about architecture design. The architecture design elements are produced through architecture design process. These architecture elements are classified in a set of related concerns. These concerns are modeled using business viewpoints, data viewpoints, application viewpoints, and technology viewpoints in IEEE Standard 1471 format. Functional requirements, non-functional requirements, business and technology environment are considered in business

viewpoint. They act as Architecture Drivers of the architecture design. Data Models are contained in Data Viewpoint, Application Models contained in Application Viewpoints and Technology Models contained in Technology Viewpoint. Data viewpoint are used to represent data being used by an application. Application viewpoint are used for processing logic and structure. Technology viewpoint represents technology used to implement system. Requirements, Assumptions, Constraints, and Design Objects are motivational reasons and act as architecture elements.

### **Architecture Rationale: (AR)**

Three explanations are considered in Architecture Rationale which are named as Qualitative Rationale (QLR), Quantitative Rationale (QNR), and Alternative Architecture Rationale (AAR). The architecture rationale AR captured in AREL model simplified process and it only capture the explanations of the decisions. Architecture rationale captures the design rationale. The direct dependencies between decisions in a chain and architecture elements are used to understand design reasoning. Qualitative Rationale (QLR) contained information regarding decision issues, design assumptions, risks involved in design option, assessment and decisions, and finally supporting information. This information provides qualitative rationale for decision. However, quantitative rationales are not considered in AREL. Finally, the alternative architecture rationale AAR contains Alternative Behavior and Alternative Design. The AREL model is extended to adapt the evolution. The extended model that can capture the evolution history is eAREL. The model provides three types of traceabilities Forward Traceability, Backward Traceability and Evolution Traceability. The impact analysis of the design is provided through forward trace. The root cause analysis is performed using backward trace. Finally, the analysis of the evolution of a decision is provided by evolution trace. These traceabilities are automated with Enterprise Architect which is UML design tool.

### **1.2.6 ARCHITECTURE-BASED COMPONENT COMPOSITION/DECISION-ORIENTED DESIGN (ABC/DD):**

ABC/DD is an iterative process [17] for designing software architecture. In the first step architect elicit architecture design issues this phase is called Issue Elicitation Phase. The next stage is called Solution Exploiting Phase in which architect find solution to each issue based on design's reusable knowledge. The candidate architecture solution is produced automatically from this activity of

doing various issue solutions in Solution Synthesizing Phase. Architecture is evaluated in the Architecture Deciding Phase. The final phase is Ration Capturing Phase in which architect captures architecture rationale and issue rationale. The ABC/DD approach for software architecture design is based on software architecture principles of decision abstraction and issue breakdown. These values are used in solving complexity associated with the software architecture design. Decision abstraction principle is used to remove the inconsistencies between requirements and design, this principle consider architecture from the point of view of system level design decisions. Decision-abstraction provides necessary high level abstraction on problem space as well as on solution space for modeling of software architecture. The second principle “issue-decomposition” is used to consider the architecture design task as solving system wide problems. The design goals are decomposed into number of related design issues. These issues are used for making decisions. In ABC/DD an issue is referred to as Architecturally Design Issue (ADI). The solutions used for solving issues related to the architecture. This design decision is used for acceptance or rejection of any candidate architecture solution. Finally, a rationale is used to reason about an issue decision or architecture decisions. This method provides a way for making architecture decisions based on architecture level problems. These architectural significant problems are elicited in the architecture design phase and solution is provided for these problems. In this activity stakeholders and architects participates for considering solutions for each issue. The decisions made in this activity are recorded in a tool that automates this process. The main modules of the tools that are the part of subsystem Architecture Solution Synthesizer are Relation Analyzer; Solution Combiner; SA Model Synthesized; Deduction Engine. The modules which are the part of Visual Design Environment are Issue Edition; Issue Solution Edition; Architecture Decision and Rationale Edition; Knowledge Manager; and Component Diagram Editor. The data flows from Knowledge Manager to the Repository.

The ABC/DD method is successfully applied to Spaceflight Center Commanding System. This system is used to collect data of space automobiles with the help of telemetry and control network. Space vehicles current status, its orbit calculation and control commands are monitored using this network. The second application of this method is on Commanding Display Systems (CDSs) which is the Air Traffic Control system. The ABC/DD method is applied on this system during its re-architecting. Architect elicited functional and non-functional requirements. Architect then

elicited architecturally significant issues and solution for these issues. The instance model was then aerated for each issue solution. All these information was automated using tool and automated synthesis of architecture solution was produced. The architect and stakeholder use this solution for making architecture choices, tradeoff among quality attributes, and architecture evaluation for global considerations. These case studies provided the evidence that decision-oriented method which is stakeholder centric provides better architectural design approach then traditional artifact-oriented approach. The proposed approach provided them a systematic and rational design process. It also helped in reducing difficulties of software architecture design. The automation of the process provided them a solution synthesis of candidate architecture automatically and provides automatically elimination of unfeasible combinations of issue solutions. Multiple candidate architecture solutions are possible for a single issue with some advantages, disadvantages and tradeoffs. Therefore, global impact on the whole architecture need to be consider by the architecture. The proposed solution has to be evaluated and compare for the candidate architecture. The ABC/DD decision oriented approach clarified the usage of decision and rationale while doing design activity in a process as well as in architecture design models. Thereby, making process of capturing architecture decisions and rational more efficient and effective.

Apart from these above mentioned Architecture-centric methods, we have also review some other methods but these does not fit with agile manifesto. We have excluded these methods from our study, the details of these are given below:

**1. Rational Design Process: (RDP):**

Rational Design Process: (RDP) is not good as its focus is on to derive program systematically from precise requirements that is against the agile values as agile encourages change and changing requirements, so precise requirements approach is not applicable here.

**2. RMARTS:**

RMARTS is concerned with real time system engineering which is not in our scope of study so this design method will be excluded

**3. Scenario Based Software Architecture Reengineering (SBAR):**



SBAR is used to refine existing software architecture, since we do not have any architecture in place so this method does not apply in our case.

#### **4. The Attribute-Based Architectural Styles: ABAS:**

Since it provides reasoning framework for each of the quality attribute. The standard characterization of quality attribute is the prerequisite for execution of ABAS that will make the whole process too heavy (w.r.t documentation), so this method cannot be applied here

#### **5. The Architecture Based Design Method: ABD:**

ABD is designed for product line engineering, so its essence is different from the problem in hand. Moreover, Decomposition of function is the main foundation of Architecture Based Design Method. Considering these input parameters of method, our process already has this in the form of developing feature list, so this is unnecessary. Also, output of this method is concrete requirements but we as agile followers embrace change, so this method does not fit well with agile

#### **6. Quality-driven Architecture Design and Analysis: (QADA):**

QADA is designed to build quality oriented software architecture for product line field. So its main emphasis and context is very defined and limited. we are pursuing for providing overall architecture support that generically fulfills the purpose.

#### **7. Methodical Architectural Design: MAD:**

Methodical Architectural Design: MAD is an extension of ADD and both of these methods focuses on attribute driven model. Since our focus is FDD and FDD model is based on features, so this method does not in compliance with the essence of FDD.

#### **8. The Attribute-Driven Design: ADD:**

ADD focuses on attribute driven model. Since our focus is FDD and FDD model is based on features, so this method does not in compliance with the essence of FDD.

Since agile approaches have important influence on software development practices from industry perspective. However, there are many issue that arises due to lack of architecture in agile processes

which is considered one of the most important design artifacts in traditional software development practices. Many industry professionals who are involved in using agile approaches consider software architecture from the perspective of the plan oriented development strategy. According to their point of view, software architecture requires too much effort which have very little impact to customer's needs from the system. On contrary, practitioners of software architecture believe that solid architectural principles cannot be practiced with agile methods. Therefore, increased appreciation related to the importance of incorporating architectural practices in agile methods is under consideration recently. Hence, there is a growing interest and research in this perspective of integrating these two different cultures [18].

Main purpose of study is to calculate architecture support provided in feature driven development that resides under the umbrella of agile, and how we can achieve benefits of architecture using agile methodologies without compromising the agile values

We concentrate on articles describing the issues that arises by using agile methodology due to lack of architecture support, and articles that depict the benefits of using architecture, which is part of the traditional development.

## CHAPTER 2 LITERATURE REVIEW

TH:18922



## LITERATURE REVIEW

Our focus is on gathering the issues that arises due to lack of architecture in agile methodology with reference to feature driven development (FDD). So we identified the agile issue and map with the benefits of architecture. This mapping will help us to evaluate the benefits that can be achieved if architecture support is provided in agile development.

### 2.1 SEARCH STRATEGY

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

### 2.2 KEYWORDS

{architecture}, {architecture centric method}, {agile}, {Feature Driven development}, {FDD}, {integration}, {incorporation}, {combination}, {effect}, {influence}, {Values}, {principles}

### 2.3 SEARCH STRINGS

- {architecture centric method} AND {agile} OR {Feature Driven development}
- {integration} OR {incorporation} OR {combination} of {architecture} AND {agile} OR {Feature Driven development}
- {effect} OR {influence} of {architecture} on {agile} OR {Feature driven development} OR {Feature driven practice}
- {Values} OR {principles} of {agile} OR {Feature driven development} AND {architecture}
- {Agile issues} OR {software architecture benefits} OR {agile drawbacks} OR {agile problems}

### 2.4 SEARCH ENGINE

Search strings are put in advanced search of following software engineering databases: IEEE, ACM, Science direct, Springer and Google Scholar

### 2.5 INCLUSION AND EXCLUSION CRITERIA

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

- Research papers that discusses 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.
- Textbooks and web pages will be excluded as these are weak links.

## 2.6 SEARCH PROCESS

Search results from different digital libraries against all search strings are mentioned in Table 1. These digital libraries were considered due to the reason of their being heavily used for empirical studies and literature surveys. 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 3.5. With further investigation of selected papers, we have filtered studies that are most appropriate to the problem in hand. These filtered papers are shown in [Table 2]. Relevant studies are shown below [Table 3]

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

*Table 2-1: Selected Studies*

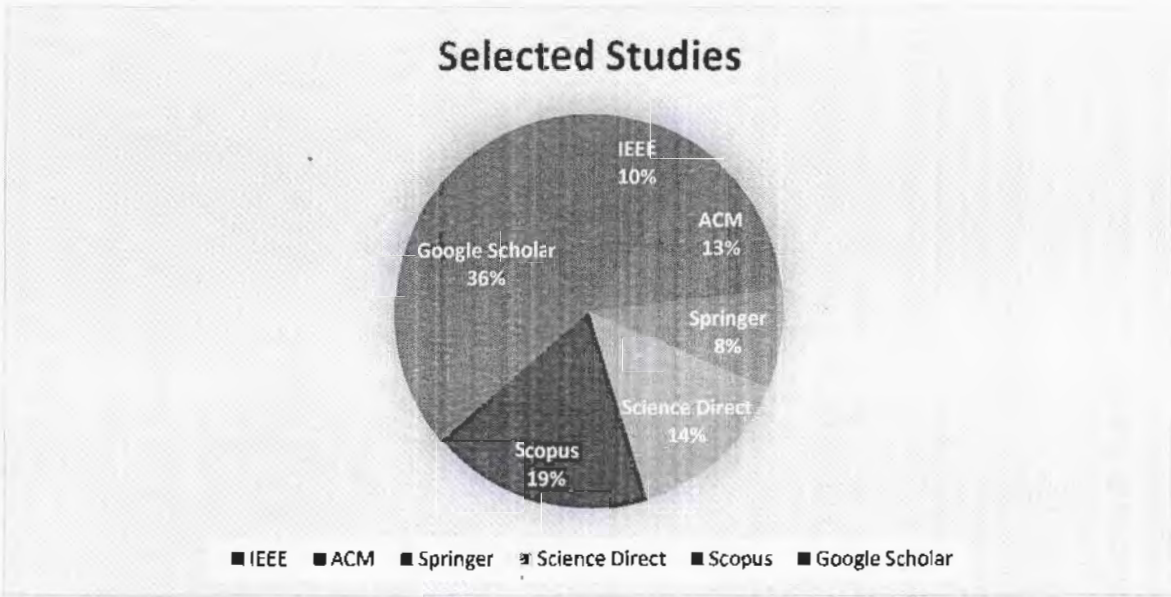


Figure 2-1: Selected Studies

Following table shows the year wise distribution in which these studies has been conducted

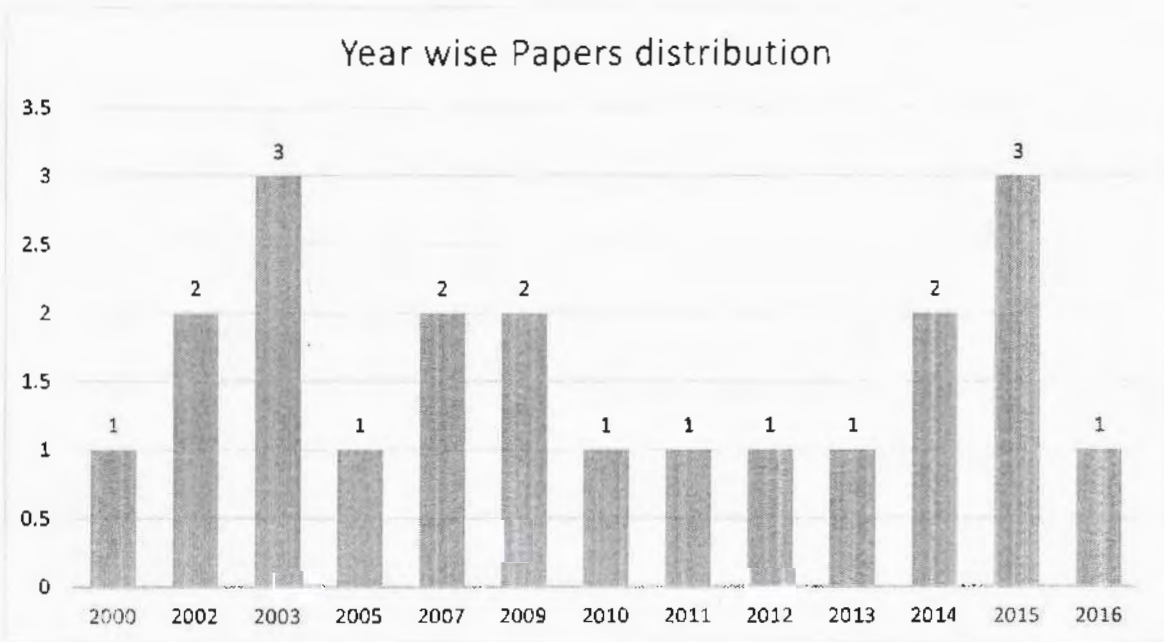


Figure 2-2: Year Wise Paper Distribution

## 2.7 DATA COLLECTION

Data obtained from selected studies was:

- Source of the study and its full reference.
- Grouping study type (Agile architecture integration, Agile issues, Architecture benefits, Architecture agile conflict)
- Summary of each study that includes main research questions.
- Quality assessment according to quality assessment criteria mentioned in section 3.4.

## 2.8 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
- Studies that describes the impact of architecture on reusability, requirement traceability, effort, cost in feature driven development/ agile methodology
- 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

## 2.9 PRIMARY STUDY TABLE

No	Reference	Primary study
1	[1]	FDRD: Feature Driven Reuse Development Process Model
2	[19]	A Practical Example of Applying Attribute-Driven Design ( ADD )
3	[11]	FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures
4	[3]	Major Seminar On Feature Driven Development Agile Techniques for Project Management Software Engineering
5	[20]	Software Architecture as a Set of Architectural Design Decisions

10	[9]	Get ready for agile methods, with care
----	-----	--

*Table 2-3: Relevant Studies*

## 2.11 QUALITY ASSESSMENT

Primary studies that are selected were assessed by using below mentioned quality assessment (QA) criteria.

Q1. Was the validity of current study properly assessed before selecting it?

Q2. Was the study described the problem and its solution properly and thoroughly?

The questions were scored as follows:

- QA1: Y, the authors have defined quality criteria i.e. empirical evidence in case of architecture and agile integration, impact of architecture on quality attributes with results, evidence of risks due to lack of high level architecture; P, the research discusses the issues in hand but not explicitly provided any evidence against the mentioned problem; N: no defined quality assessment described.

QA2: Y Required data is described clearly in study, P: only partial information about primary study is presented, N: results are not stated properly.

## 2.12 QUALITY EVALUATION

Quality assessment criteria has been mentioned above. Based on that criteria, following table shows the results of research questions mentioned in section 5 with respect to quality.

(RQ1) Architecture Integration Level in agile	(RQ2) Architecture Impact on Quality Attr.	(RQ3) Agile Architecture Conflicts
---	--	---

P, high level design described but not empirically validated	N, no quality attribute has been discussed in agile	P, discussed various agile architecture conflicts but not empirically validated with architecture support
--	---	---

Table 2-4: Quality Evaluation Criteria

### 2.13 SEARCH RESULTS

Ref.	Topic Area	(RQ1) Architecture Integration Level in agile	(RQ2) Architectur e Impact on Quality Attr.	(RQ3) Agile Architecture Conflicts
[22][1]	Reusability	N/A	N/A	Incapability to reuse components due to architecture discontinuities
[1][18]	Architecture related challenges	N/A	No attention paid to quality attributes	Incorrect prioritization of User Stories due to lack of architecture
[1][18]	Architecture related challenges	N/A	N/A	Lack of motivation and time for consideration of design choices
[18]	Requirement Traceability	N/A	N/A	Design erosion
[33][34]	Architecture integration	High level design available	N/A	N/A

Table 2-5: Search Results



## 2.14 AGILE AND ARCHITECTURE

In agile methodologies customers or people are center of focus[35]. Contiguous delivery of working software is priority over heavy documentation and reports. Testing goes along with the delivery of every small working software units and frequent client feedback on these software units helps keeping the software project on right track and aligned with goals of the customer. XP, Scrum, Feature Driven development are few examples of agile methodologies.

Agile methodologies are based on following four values that are described in agile manifesto.

1. “Individuals and interactions over processes and tools”
2. “Working software over comprehensive documentation”
3. “Customer collaboration over contract negotiation”
4. “Responding to change over following a plan”

Above mentioned four values is the basis for any agile methodology [36].

[23] Asserts that there is misunderstanding between agile methodologies and early focus on architectural design. Agile approaches lack empirical evidence for their claims. According to [23] agile literature is silent on how to go to about software architecture but also do not oppose it.

[18] Aims to address challenges and limitations of architecture practices in agile methods. According to [18], agile architecture have limitations such as lack of architecture analysis, lack of focus on quality attributes, architectural knowledge vaporization, lack of requirement traceability etc.

[18] Also states that there is little empirical evidence architectural practices followed by agile teams as a result little is known about the challenges an agile team faces during architectural activity. Currently large part of agile architecture related literature includes data solely from expert opinion. Lack of empirical studies also means that results of these studies cannot be generalized [37].

Literature has also reported Architecture Evaluation Method for Feature driven development[33]. Author suggested a hybrid of QAW, ATAM and ARID and FDD process

model. Proposed method is illustrated below:

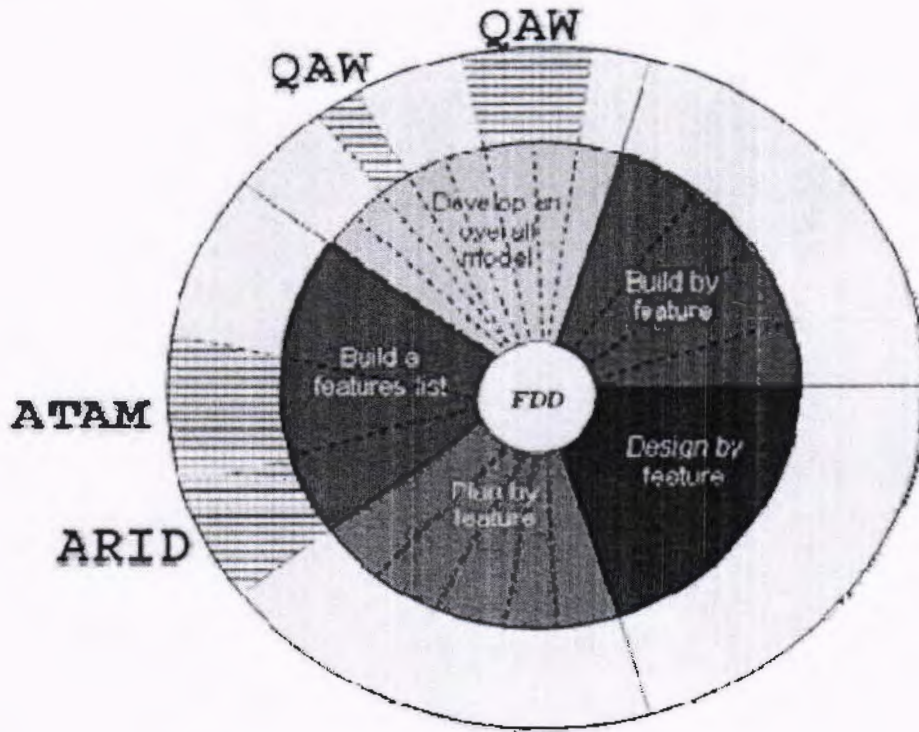


Figure 2-3: Hybrid FDD with architecture evaluation methods [16]

The problem with this proposed approach is that there is no empirical validation of this proposed process. Moreover, since these architecture-centric methods are very heavy weight that are against the agile values and we cannot incorporate them as is in the FDD process.

### 2.15 MAPPING OF AGILE DRAWBACKS AND ARCHITECTURE BENEFITS

Sr. #	Agile limitations	Architecture benefits
	Incapability to reuse components due to architecture discontinuities[1][22]	Architecture is explicitly defined and documented hence architecture discontinuities are limited and reusing component is made possible due to



		availability of documentation and design rationale[19] [11]
	Design erosion, knowledge and rationale vaporization as a result of ad-hoc design decisions documentations[3][21]	Documentations of design decisions addresses design erosion, knowledge and rationale vaporization. [19][20]
	Risk of failure (or delayed feature distribution) in case of unknown domain and untried solutions[18]	[11] and [19] provides for explicit exploration of architectural choices and clear definition of user stories as quality scenarios and these should minimize such risks.
	Highly experienced domain developers required for successful projects[18]	[11] and [19] provides step by step approach that deal with architecture also called plan-driven approaches that are known for exploration of new domain by a person.
	Lack of requirements traceability[18]	[19] step wise approach to architecture categorizes requirements according to their significance and document them throughout the software development

*Table 2-6: Mapping of Architecture benefits and Agile Issues*

## **CHAPTER 3 PROBLEM STATEMENT**

## PROBLEM STATEMENT

Current literature lacks any empirical study that validates or nullifies benefits achieved by employing architecture centric methodology with Feature-driven development that addresses limitations of agile and FDD architecture practices alike.

The difference between traditional architectural approaches and agile approach is that agile opposes the idea of extensive planning at the very beginning while architectural approaches promote the idea of early planning as some aspects of the project would not be easier to adapt later in the development cycles.

Agile architecture practices have limitations like incorrect prioritization of user stories, lack of architecture analysis, lack of quality attributes focus, highly experienced domain specific developers are required that steer a software project towards success, knowledge vaporization, lack of requirement traceability and some others [18]. FDD being an agile development methodology inherits the same limitations.

Architecture centric methodologies [11][38] provides for architectural as well as detailed design, explicit focus on quality, architectural analysis, clear documentation practices. So a hybrid or middle ground approach could be advocated to overcome agile-architecture challenges without defying practices and values of FDD.

So we need to integrate agile and architecture centric methodologies in such a way that we can achieve benefits of architecture without losing agility of Feature driven development.

### 3.1 RESEARCH QUESTIONS

Research questions that are focused by this study are:

- RQ1. At what level, architecture support has been provided in agile methodology in context of Feature Driven development.
- RQ2. What would be the impact of architecture on quality attributes in agile methodology with context of Feature driven development.
- RQ3. What would be the effect of Architecture on FDD values.

With respect to RQ1, we evaluated that whether the paper provides high level architecture support or low level design support has been provided.

With respect to RQ2, reusability, effort, cost and requirement traceability are the quality attributes that will be our focus for evaluation of architecture impact.

With respect to RQ3, we identify the factors that are not aligned with agile values but are inherited by the architecture, so that we can evaluate the effect of including architecture in Feature driven development methodology.

### 3.2 DISCUSSION WITH LITERATURE

Discussion of Research questions with respect to literature is explained below:

*RQ1. At what level, architecture support has been provided in agile methodology in context of Feature Driven development.*

We have seen that high level architecture support has been discussed in papers but there is no empirical evidence about the outcomes of integrating architecture in agile. As mentioned in [33], only integration of architecture in FDD has been discussed, so we need to further investigate the effect of integration of architecture with FDD with clear cut steps and validations

*RQ2. What would be the impact of architecture on quality attributes in agile methodology with context of Feature driven development.*

In agile, clients provide Users Stories to the project teams with the detailed design decisions. Most of the design decisions are evaluated on the features that is going to be delivered within the budget. So we need to cater that problem while architecture integration in agile

*RQ3. What would be the effect of Architecture on FDD values?*

One of the main problem that is against the architectural approach is that agile teams forced to limited number of solutions to achieve features within time and budget. Risk in this approach was architects might have missed better design choice by not doing full design upfront. In agile, developers have to justify everything, so they may be forced to skip alternative designs and implement known solution.

Another problem with following the agile values that we are unable to reuse components due to lack of architecture support.

Due to lack of time for proper documentation, design decisions remain with the individuals who took that. And requirement traceability would not be possible during maintenance phase of the project.

## CHAPTER 4 PROPOSED SOLUTION

## PROPOSED SOLUTION

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

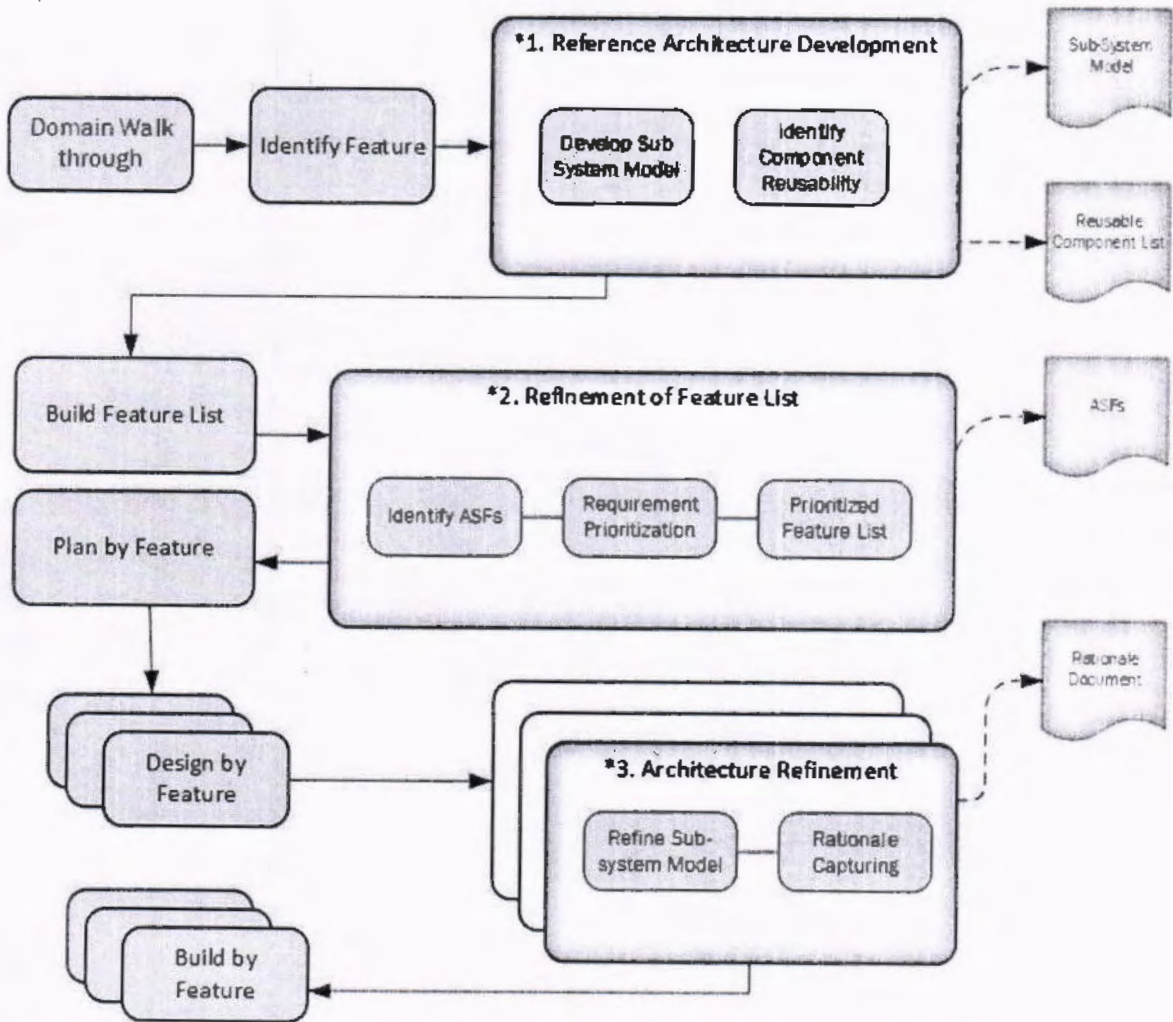


Figure 4-1: Proposed Solution

Following new artifacts have been added in proposed process model

1. Reference architecture development
2. Refinement of Feature List
3. Architecture refinement



Following new documents have been 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

## **4.1 REFERENCE ARCHITECTURE DEVELOPMENT**

### **4.1.1 DEVELOP SUB-SYSTEM MODEL**

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

### **4.1.2 IDENTIFY COMPONENT REUSABILITY**

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

## **4.2 REFINEMENT OF FEATURE LIST**

### **4.2.1 IDENTIFY ASFS**

A feature will be considered as architecturally significant that has broad effect, objectives trade-off points, supposition breaking, or difficult to achieve.

Architecturally significant features (ASFs) are extracted from system context or architectural concerns. The ASFs are those features that influence the software system architecture. Therefore, it is not necessary that all of the system requirements will be ASFs.

Indicators for architectural significance include:

- Feature is linked with high business value or technical risk.
- Feature is particularly important for any stakeholder.

- Feature is unique in its nature. Neither of the existing component addresses it in the system before.
- Feature has similar context in previous project and caused a major issue in terms of over budgeting or client dissatisfaction.

On close of this activity, we have a list of ASFs in hand to perform further processing based on this list

#### **4.2.2 REQUIREMENT PRIORITIZATION**

Prioritization is done by ranking. We 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. We 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

#### **4.2.3 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.

### **4.3 ARCHITECTURE REFINEMENT**

#### **4.3.1 REFINE SUB-SYSTEM MODEL**

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

#### **4.3.2 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.

#### 4.4 DOCUMENT TEMPLATES:

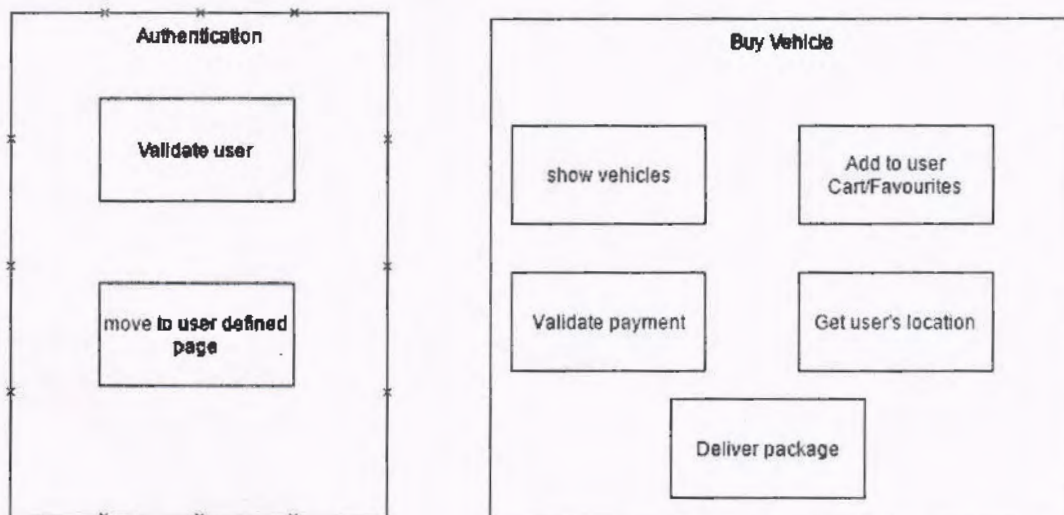
Newly produced documents in the proposed process will be developed based on following templates.

#### Sub System Model Template

##### Document Review Information

Review Date	Reviewer Name	Version	Reference / Evidence	Design version	Rationale

##### Sub System Model Sample



### Reusable Components Template

#### Document Review Information

Review Date	Reviewer Name	Version	Reference / Evidence

#### Reusability Index

Feature Name	Purpose	Input parameters	Output Parameters	Reusability (Amount of reuse, adaptability, maintainability, Quality, documentation)



**Architecture Significant Features Template**

**Document Review Information**

Review Date	Reviewer Name	Version	Reference / Evidence

**ASFs Index**

Feature Name	Architectural benefit	Architectural Risks	Priority level (1-n)

### Design Rationale Template

#### Document Review Information

Review Date	Reviewer Name	Version	Reference / Evidence

#### Design Rationale Index

Rationale ID	Rationale	Sub system Model Artifacts Added	Sub system Model Artifacts Removed	Sub-System Model version addressing the rationale	Stakeholders involved

## **CHAPTER 5 RESEARCH METHODOLOGY**



## **RESEARCH METHODOLOGY**

### **5.1 CASE STUDY**

Basic purpose of our case study is to assess architecture effect in Feature driven development which is a type of agile methodology. The focus of evaluation is on reusability, cost, effort and project failure risks that are due to lack of architecture.

We will use an action research strategy of case study and study will be of exploratory type because our focus is on FDD methodology enhancement and its effect and finding out what happens after integration of architecture in FDD

### **5.2 CASE STUDY DESIGN**

#### **5.2.1 RATIONALE**

We undertook 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.

#### **5.2.2 OBJECTIVE OF THE STUDY**

Studying the impact of integrating architecture in feature driven development methodology with respect to reusability, cost, effort, requirement traceability and project failure risks due to unknown domain and untried solutions is the main objective of our study.

#### **5.2.3 THEORETICAL FRAMEWORK**

Theoretical frame of reference is the literature that discusses agile and software architecture which is mentioned in section 3.3.

### 5.2.4 EXPLORATORY QUESTIONS

- How much components are reusable after having support of tailored software architecture in FDD
- What would be the impact of integrating architecture on cost and effort
- At what level, requirements are traceable with the involvement of architecture's documentation
- What would be the probability of mitigating project failure risks due to unknown domain and untried solutions

### 5.2.5 PROPOSITIONS AND HYPOTHESES

Three hypotheses have been formulated in this case study. Informally, they are:

1. FDD without having high level architecture support tend to have lack of reusability benefits that are achieved by having a software architecture in place. So it is expected that the newly defined FDD process with light weight architecture support is more helpful in achieving reusability as compared to traditional FDD process.
2. Since the introduction of new artifacts in the process requires more effort and cost than before to complete the project of either size (small or medium). So it is expected that cost and effort of the project increases proportionally with the size of the project.
3. We have an architecture rationale in place during the change process so it is easy to back track the design decision up to requirement and stakeholder needs, that is helpful in taking other decisions and conflicts of interest in requirements. So traceability will be increased by having the architecture support as compared to traditional FDD process.

On the basis of above mentioned informal statements, we can formally state them below along with measures that are required for assessing the hypotheses.

1. Null hypothesis, H<sub>0</sub>: There is no difference in reusability of Proposed FDD process(PP) and traditional process(TP).

H<sub>0</sub>: Reusability(PP) = Reusability (TP)

Alternative hypothesis, H<sub>1</sub>: Reusability(PP) > Reusability (TP)

Measures needed: projects done with both processes and reusability level achieved. Reusability level is measured by taking mean of following factors:

- Amount of reuse
- Adaptability
- Maintainability (adjustability to higher versions)
- Quality (in terms of no of bugs)
- Documentation (in terms of completeness)

2. Null hypothesis, H<sub>0</sub>: There is no difference in cost and effort(CF) of Proposed FDD process(PP) and traditional process(TP).

H<sub>0</sub>: CF(PP) = CF (TP)

Alternative hypothesis, H<sub>1</sub>: CF (PP) > CF (TP)

Measures needed: cost is measured in terms of time used to develop software. Time is measured in man hours

3. Null hypothesis, H<sub>0</sub>: There is no difference in traceability of Proposed FDD process(PP) and traditional process(TP).

H<sub>0</sub>: traceability (PP) = traceability (TP)

Alternative hypothesis, H<sub>1</sub>: traceability (PP) > traceability (TP)

Measures needed: number of features that are traceable back to actual requirement and stakeholders.

### 5.2.6 VARIABLES SELECTION

The independent variables are Proposed process model and Traditional process model of FDD. The dependent variables are reusability, cost, effort and traceability.

### 5.2.7 SELECTION OF SUBJECTS

Subjects are selected on the basis of project size and real life projects. For this, two small sized and two medium sized projects are taken to check the validity of the results. Project size is measured using function point analysis and are categorized based on the following metrics:

If the project's adjusted use case points count is less than 100, then it will be considered as small project. If the project's adjusted use case points count is greater than 150, then it will be considered as medium sized project.

### 5.2.8 METHODS OF DATA COLLECTION

Two sample projects PS (Small sized project) and PM (Medium sized project) were taken from the industry from a ABC company (hypothetical name) that is well known for out-sourced projects. At first both projects were completed by Team A by using traditional FDD process. Then the same projects were given to Team B and make them use the tailored FDD process.

The problem has been stated, and the independent and dependent variables have been chosen. Furthermore, the measurement scales have been decided for the variables. Thus, it is now possible to design the experiment. The first step is to address the general design principles:

#### **Randomization:**

Developers are selected randomly from among the available developers having experience of more than three years in professional environment.

#### **Blocking:**

There is no systematic blocking approach used in our study. All four projects are assessed rather than looking at each individual project. Therefore, impact of differences between individual projects can be blocked in this way.

#### **Balancing:**

It is better that we get a balanced data set for analysis, but our study uses industry projects for which domain knowledge of that particular field is important, and the current available developers in that company from which projects are taken and have required domain knowledge came from

different Universities and different professional experience with respect to companies in which they worked, and hence it is not feasible to nullify the effect of background of developers that makes us hard for us to get a balanced data set.

### **Standard Design Types:**

The available data was checked against standard design types mentioned in [39].

Design type is of “*one factor with two treatments*”. Factor was the Process model and treatments were proposed and traditional process models. Dependent variable was checked on a ratio scale, so, t-test was used.

### **5.2.9 INSTRUMENTATION**

Guidelines are needed to guide the participants in the experiment. Guidelines for our current study when performing treatment with the proposed methodology includes:

- Preparation of sub system model as per defined template
- Preparation of reusable component list as per defined template
- Preparation of ASFs as per defined template
- Documenting design rationale list as per defined template

The results of the experiment are measures using following checklists

- Reusability checklist as per defined template that is developed using [40]
- Traceability checklist as per defined template
- Cost and effort calculation on the basis of time consumed using both approaches

### **5.2.10 VALIDITY EVALUATION**

There are four levels of validity threats to consider[39]. Internal, external, conclusion and construct validity.

**Internal validity:**

Internal validity is probable as the factors of domain, project size may affect the outcome of the treatment, so in order to remove that threat, we decided to choose two samples projects of same size with slightly different nature so that size and domain threats does not affect the outcome

**External validity:**

As far as external threats are concerned, since a wide range regarding project nature exists, so we can generalize the results with respect to size.

**Conclusion validity:**

Quality of data is the main threat for conclusion validity because of developer's knowledge issue related to Process Model. Data inconsistencies are not related to any specific project or developer, so problem seems to be the same independent of any developer's background. Therefore, conclusion validity is not very critical in our study.

**Construct validity:**

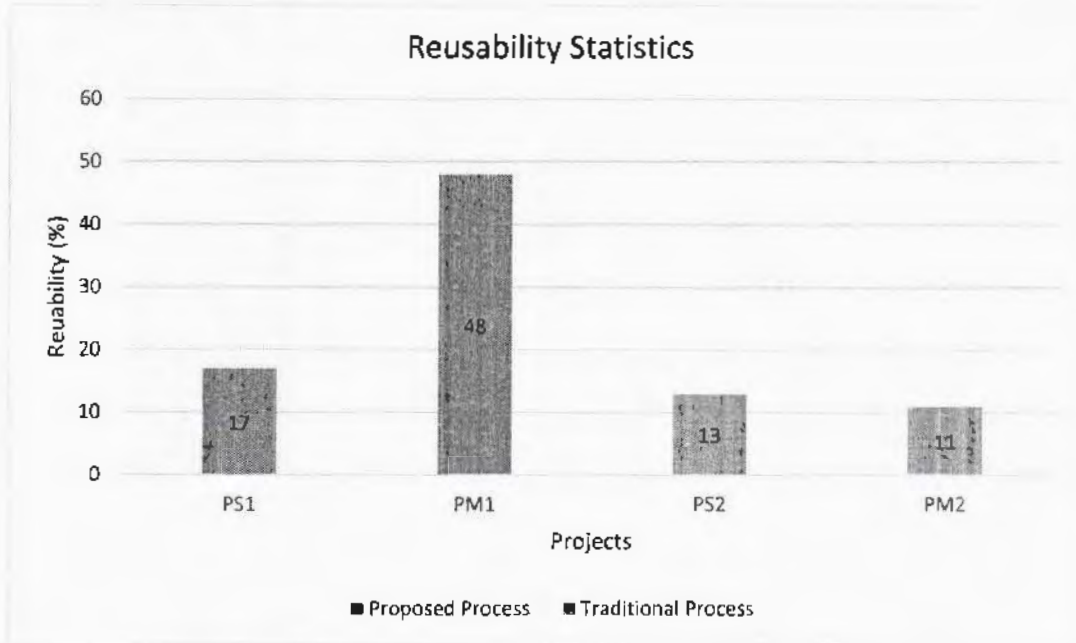
Construct validity seems not be a problem, as results are evaluated from pre-defined forms, problem statement is unseen and there is no room for biasness of the developer regarding the variables that we are evaluating during the course of developing the software and filling the template.

## CHAPTER 6 ANALYSIS AND DISCUSSION



## ANALYSIS AND DISCUSSION

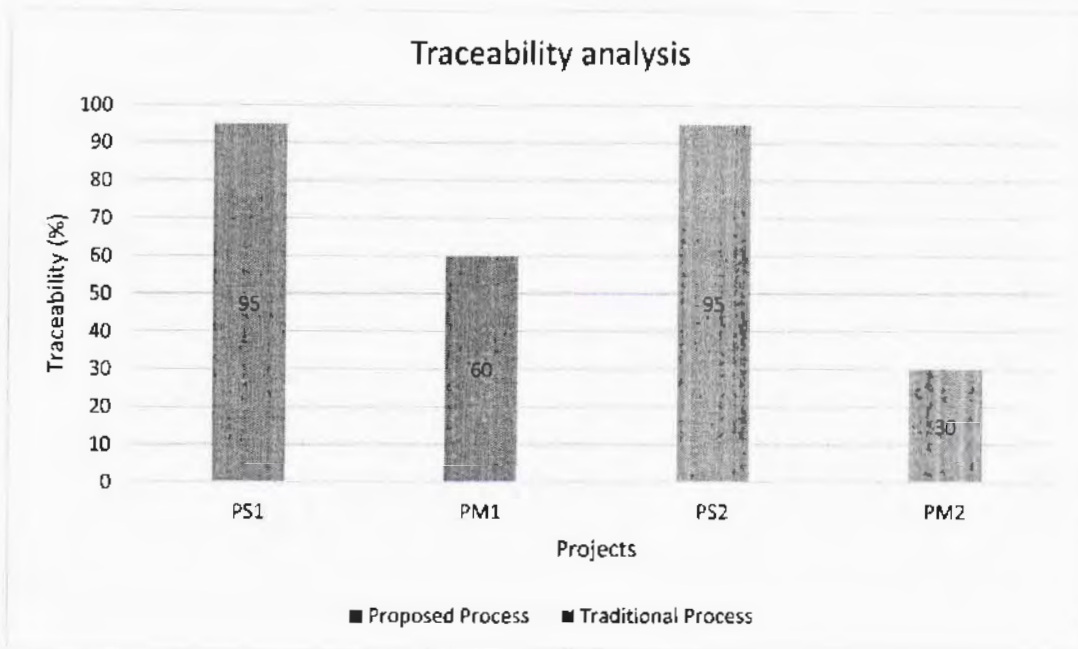
### 6.1 PROCESS VS REUSABILITY



*Figure 6-1: Process vs Reusability*

We have seen that reusability is increased when project is done with the proposed process as compared to the traditional process. This change is drastically increased as the project size increases. So it can be concluded that more benefits can be achieved when the project size is bigger and for small project, although there is an increase of reusability achieved but it is comparatively less than medium sized project.

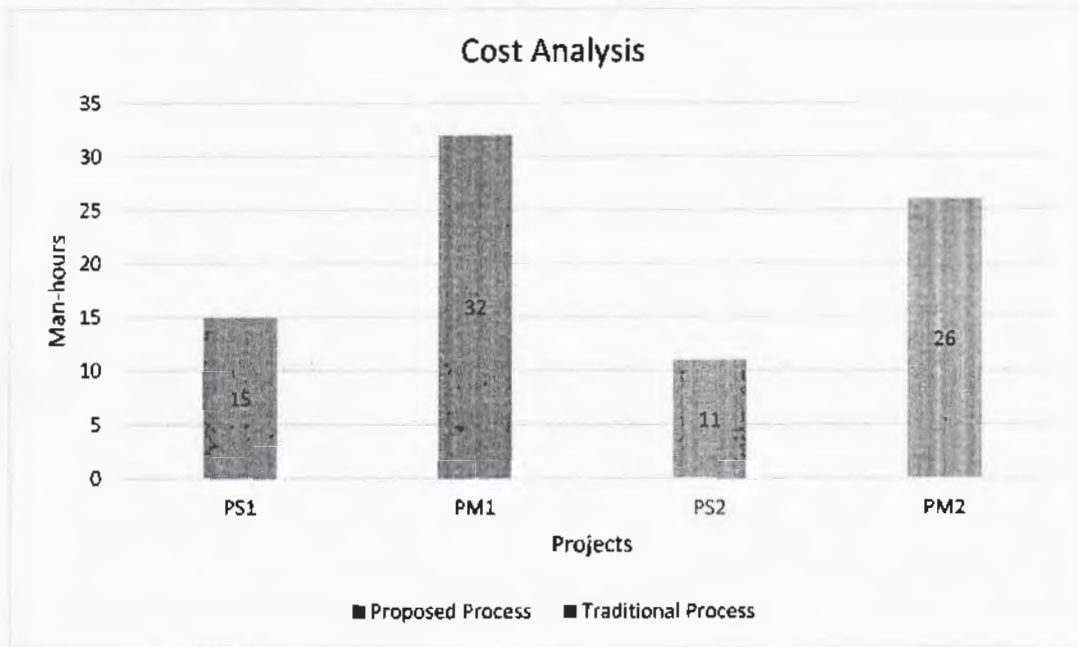
## 6.2 PROCESS VS TRACEABILITY



*Figure 6-2: Process vs Traceability*

We have seen that for small project, traceability is not effected when project is done with the proposed process as compared to the traditional process. This change is increased as the project size increases. So it can be concluded that more benefits can be achieved when the project size is bigger and for small project, there is no effect of proposed process on traceability.

### 6.3 PROCESS VS COST AND EFFORT



*Figure 6-3: Process vs Cost and effort*

We have seen that cost and effort is increased when project is done with the proposed process as compared to the traditional process as there is additional effort needed to produce the new documents. This change is increased as the project size increases a bit more than traditional process but this increase is much less than the benefits achieved by the incorporation of new artifacts.

## 6.4 EFFECT OF PROPOSED PROCESS VS TRADITIONAL PROCESS ON AGILE VALUES

Apart from the factors that are under consideration, we have also evaluated the effect of proposed process on agile values. Outcomes of comparison is listed below.

### 6.4.1 INDIVIDUALS AND INTERACTIONS

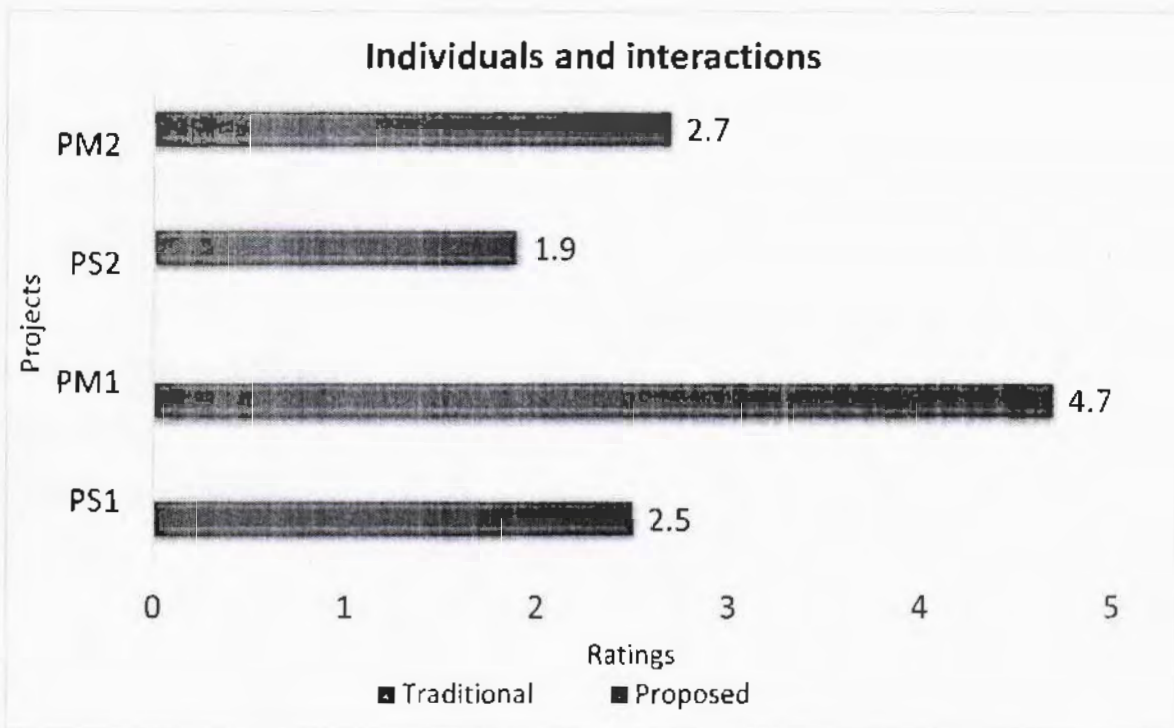


Figure 6-4: Individuals and Interactions

Results are collected qualitatively and we have seen positive effect on individuals and interactions over processes and tools as the little documentation provides more clear understanding for every individual and they can interact better with each other as compared to traditional process.

### 6.4.2 WORKING SOFTWARE

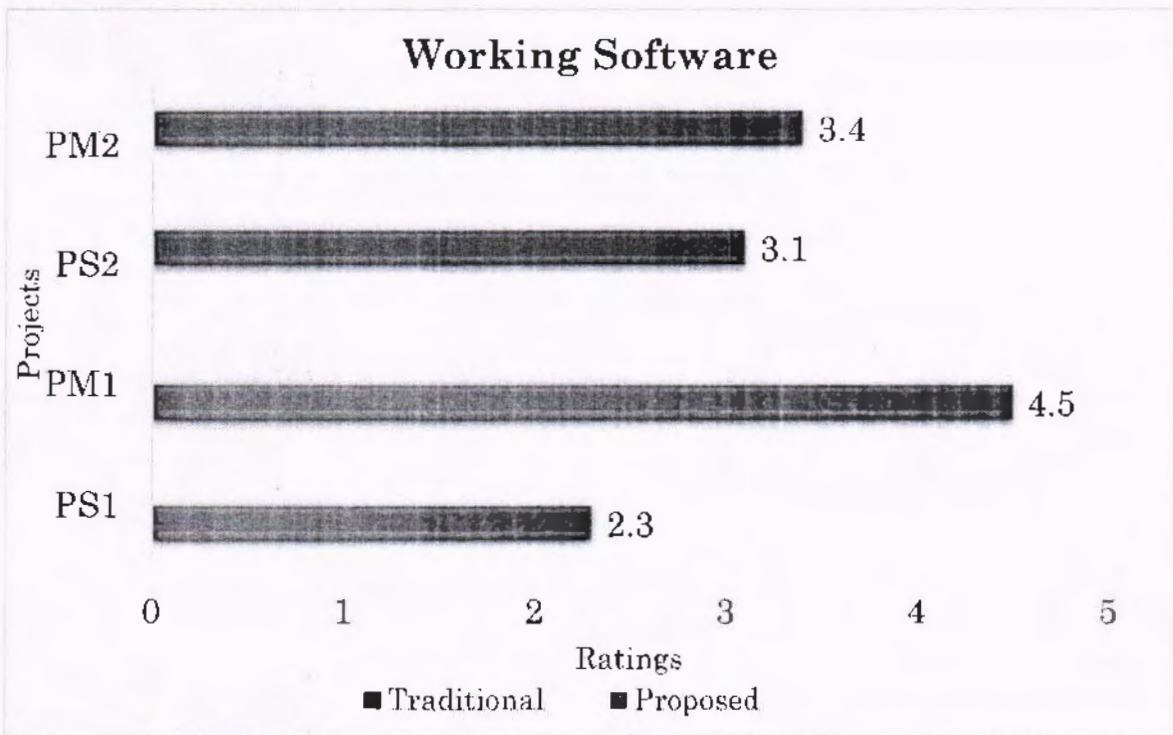
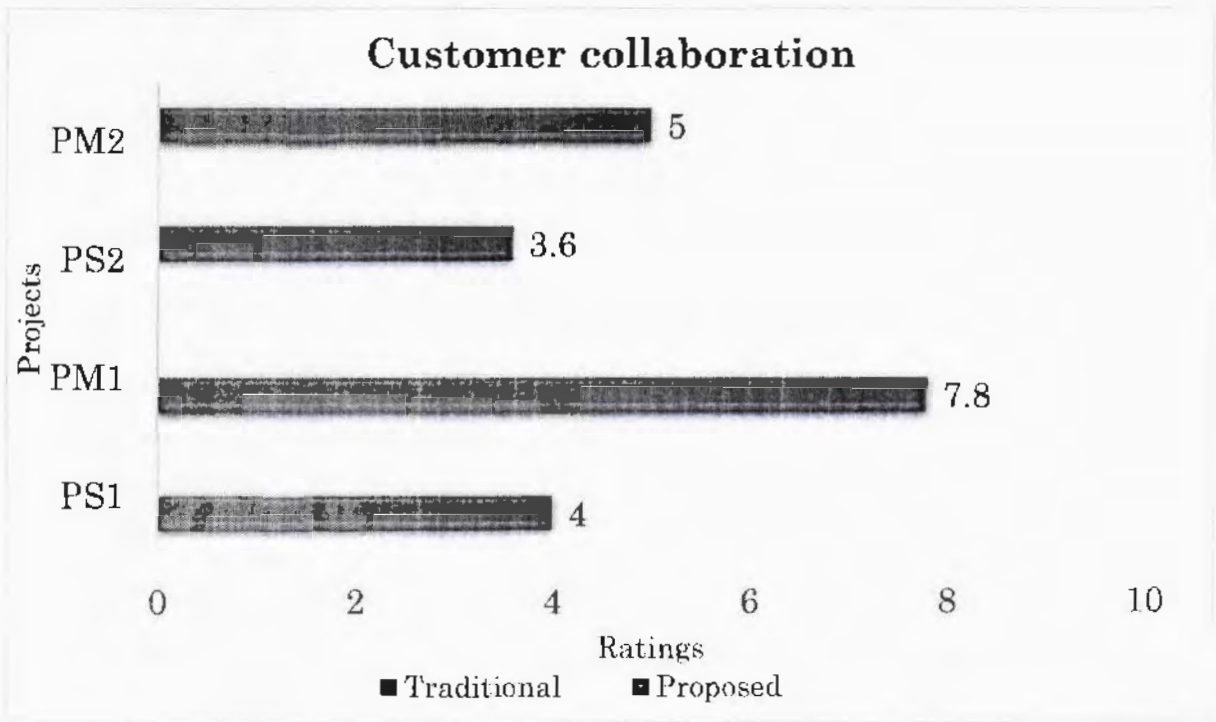


Figure 6-5: Working Software

Results are collected qualitatively and we have seen positive effect on working software for medium sized projects. For small projects, due to documentation, working software is comparatively less in case of proposed process but as the size increases, little documentation increases the understanding and ultimately productivity of the developers and they can produce more working software as compared to traditional process.

### 6.4.3 CUSTOMER COLLABORATION

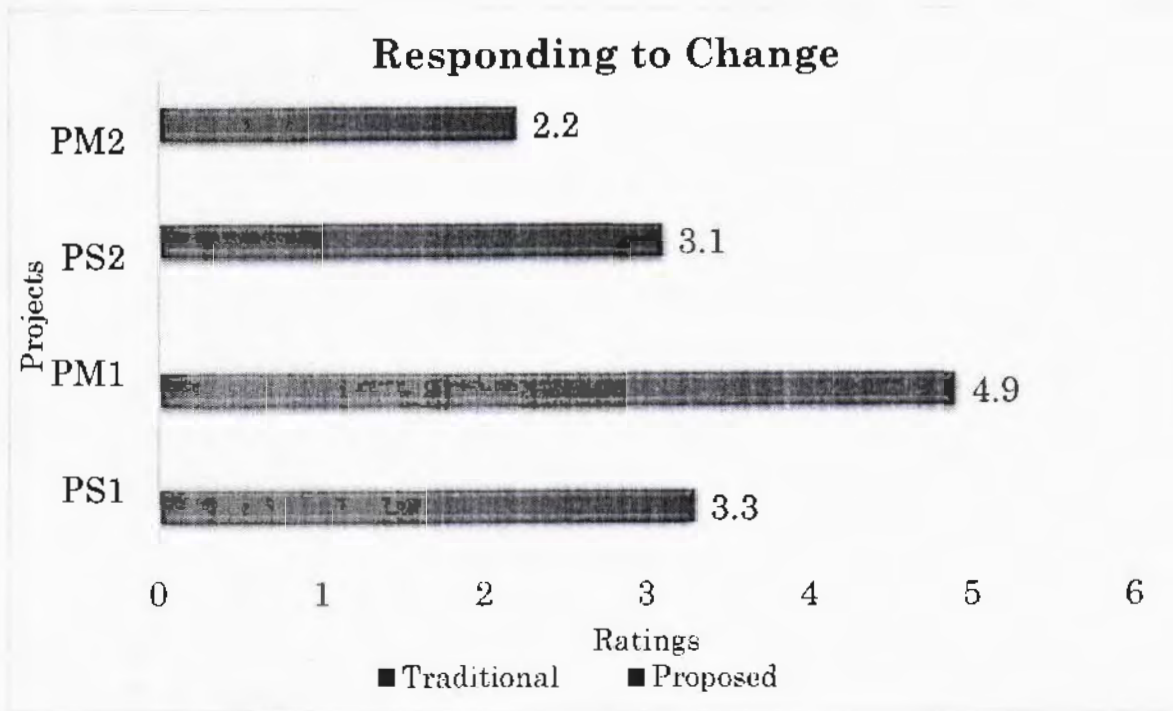


*Figure 6-6: Customer Collaboration*

Results are collected qualitatively and we have seen positive effect on customer collaboration. This is because of the reason that now we have more understanding of the project in hand and developers and other system stakeholders have better understanding of each other's perspective. This increases positive environment that ultimately increases customer collaboration.



#### 6.4.4 RESPONDING TO CHANGE



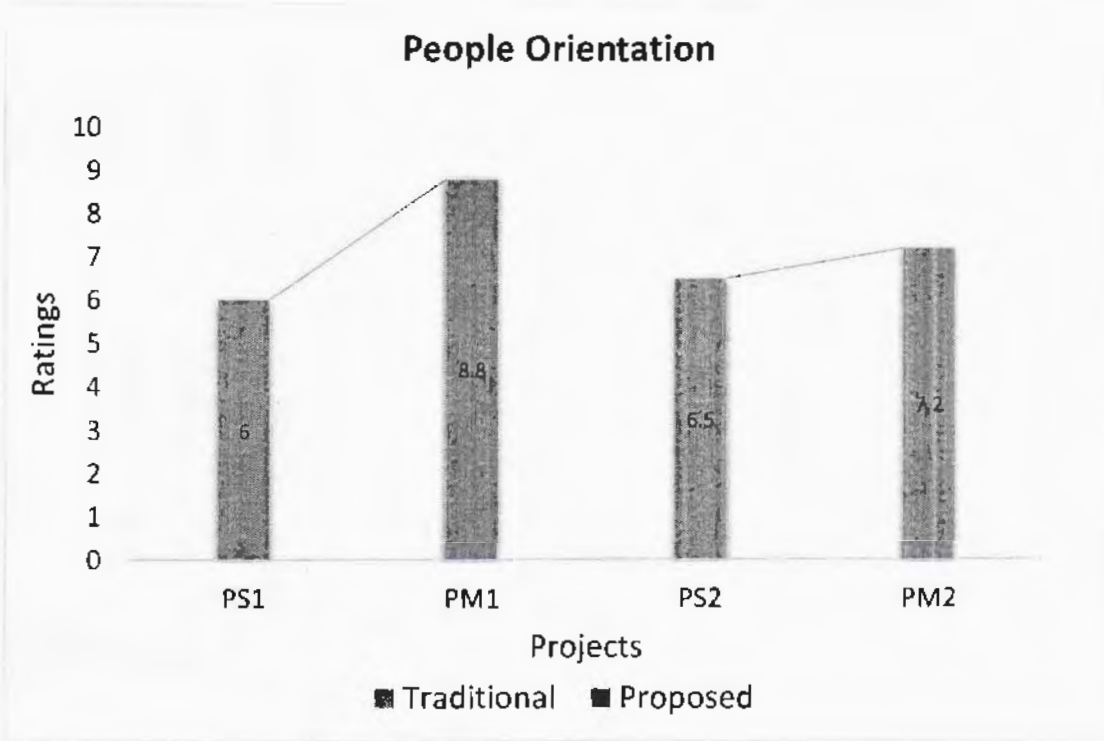
*Figure 6-7: Responding to Change*

Responding to change is difficult in traditional process when the project size increases as compared to the proposed process. In the proposed process we have clear definition of the system in hand so we can respond to change more easily and effectively as compared to the traditional process.



## 6.5 EFFECT OF PROPOSED PROCESS VS TRADITIONAL PROCESS ON AGILE PRINCIPLES:

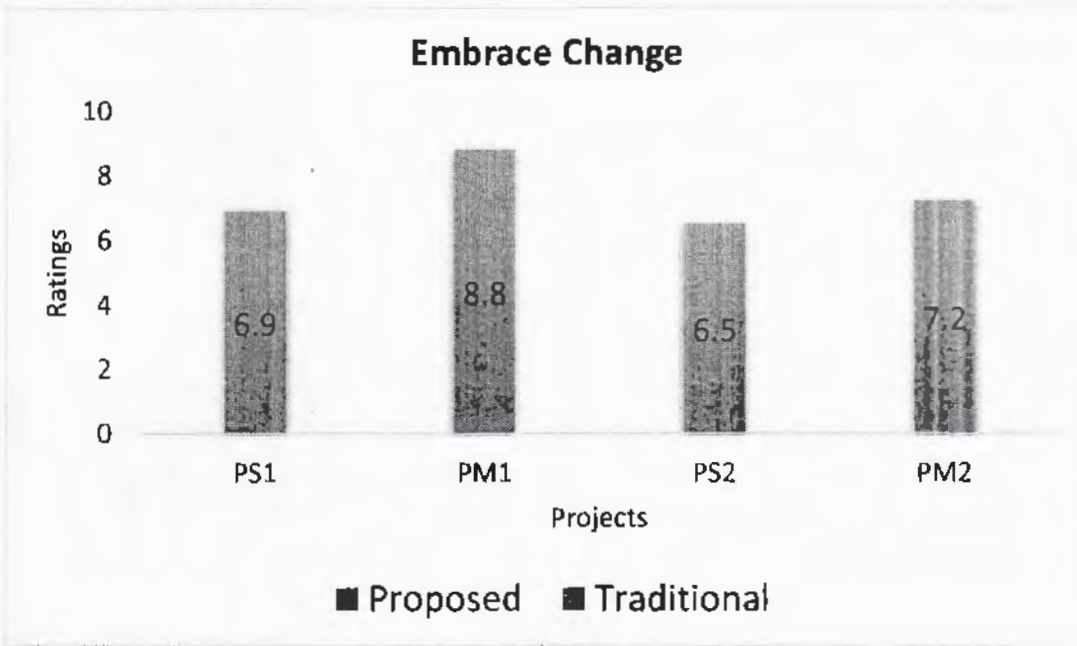
### 6.5.1 PEOPLE ORIENTATION



*Figure 6-8: People Orientation*

With the architecture in place, we have clear picture of our system and so we can satisfy customer needs more easily and effectively by providing valuable software. Only first iteration is a little bit slow as compared to traditional process but next iterations can be delivered more effectively and in timely manner.

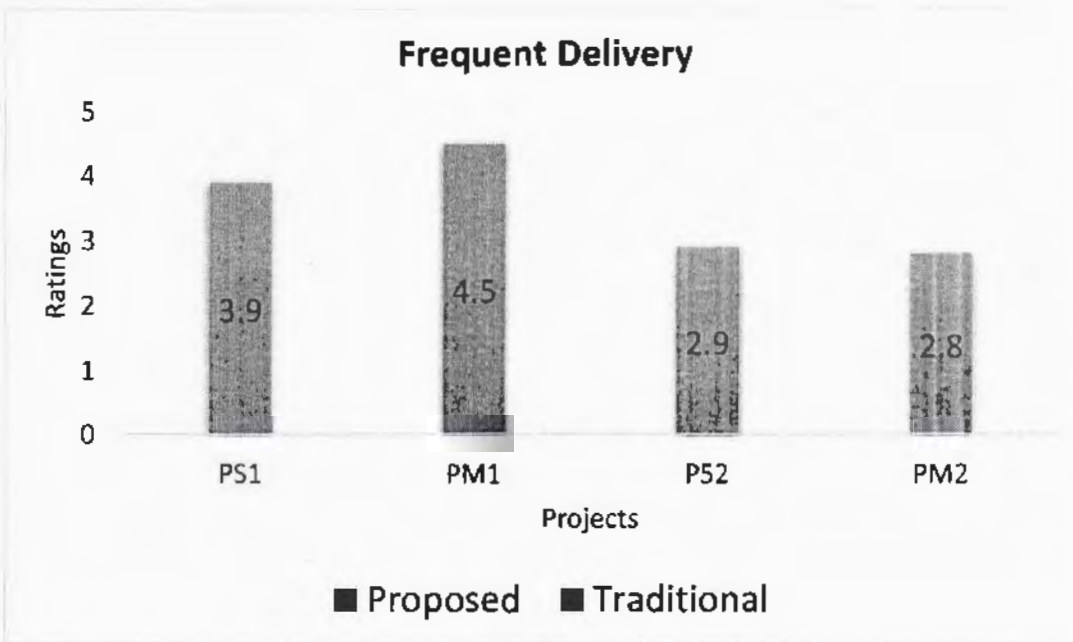
### 6.5.2 EMBRACE CHANGE



*Figure 6-9: Embrace Change*

With this newly proposed process model, we have an architecture in place. so we have clear picture of our system and any changes required by customer can be fulfilled easily in shorter time and with less effort as compared to traditional process. In this way, changes can be incorporated even late in the project to give customer's competitive advantage.

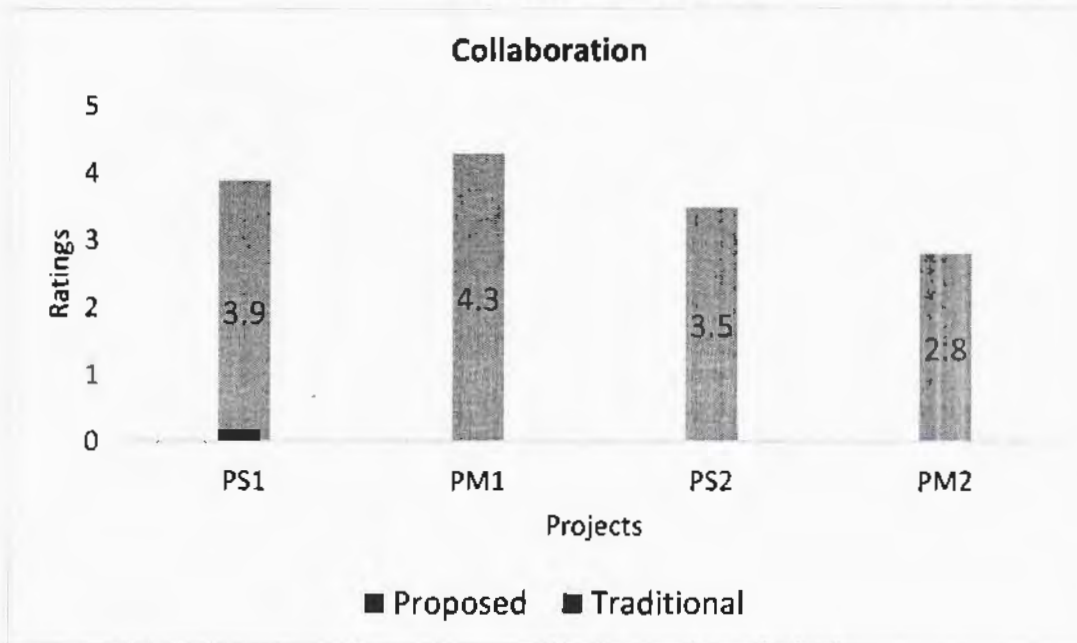
### 6.5.3 FREQUENT DELIVERY



*Figure 6-10: Frequent Delivery*

Due to ASFs in newly proposed process, software's backbone is delivered earlier, and there is a pre-defined approach of meeting the features in next phases, delivery time is reduced as compared to traditional process. This trend is significant as the project size increases.

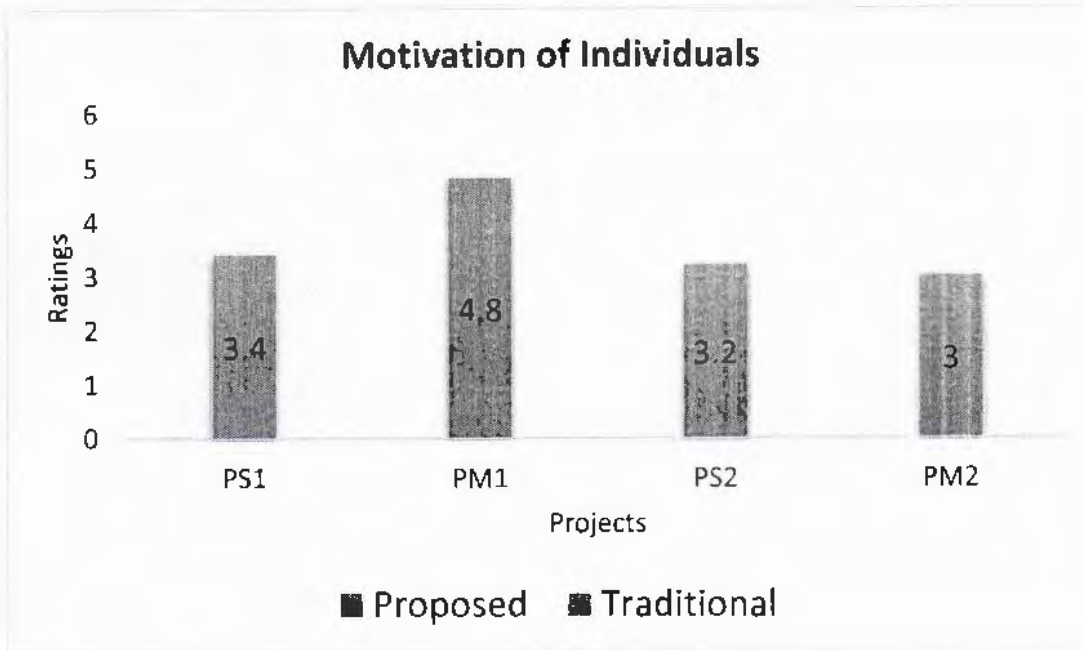
#### 6.5.4 COLLABORATION OF BUSINESS PEOPLE AND DEVELOPERS



*Figure 6-11: Collaboration*

Results are collected qualitatively and we have seen positive effect on collaboration of business people and developers. This is because of the reason that now we have more understanding of the project in hand and developers and other system stakeholders have better understanding of each other's perspective. This increases positive environment that ultimately have encouraging effect for business people and developers to collaborate.

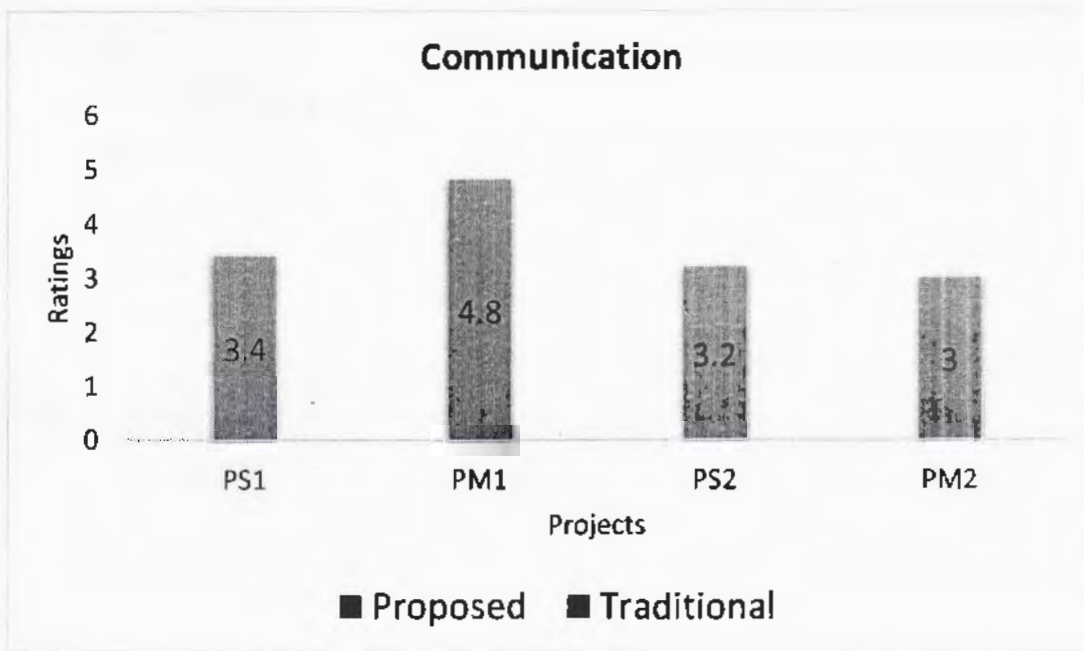
### 6.5.5 MOTIVATION OF INDIVIDUALS



*Figure 6-12: Motivation of Individuals*

Results are collected qualitatively and we have seen positive effect on motivation of individuals with newly proposed process. This is because of the reason that now we have more understanding of the project in hand and developers and other system stakeholders have better understanding of each other's perspective. Individuals know their role in project and what is expected from them. So a directed approach to meet the clearly defined objective increases their motivation.

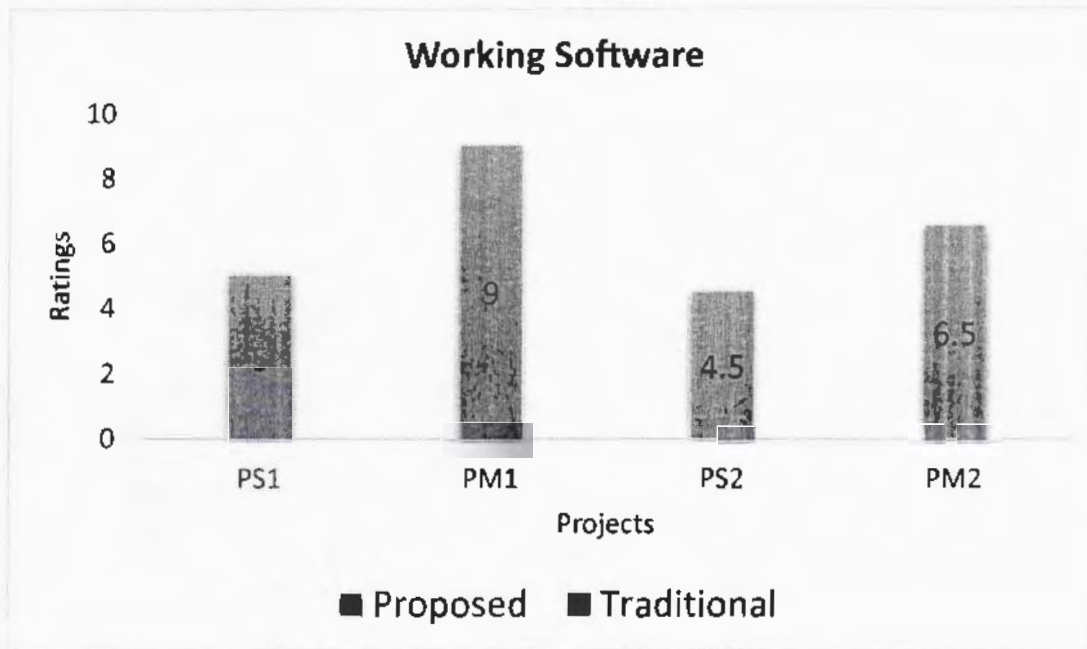
### 6.5.6 COMMUNICATION



*Figure 6-13: Communication*

Results are collected qualitatively and we have seen positive effect on communication in newly proposed process as compared to traditional. Due to architecture support, we have a clear picture of the system and developer and business people can have more clear and effective communication as compared to traditional process.

### 6.5.7 WORKING SOFTWARE AS MEASURE OF SUCCESS

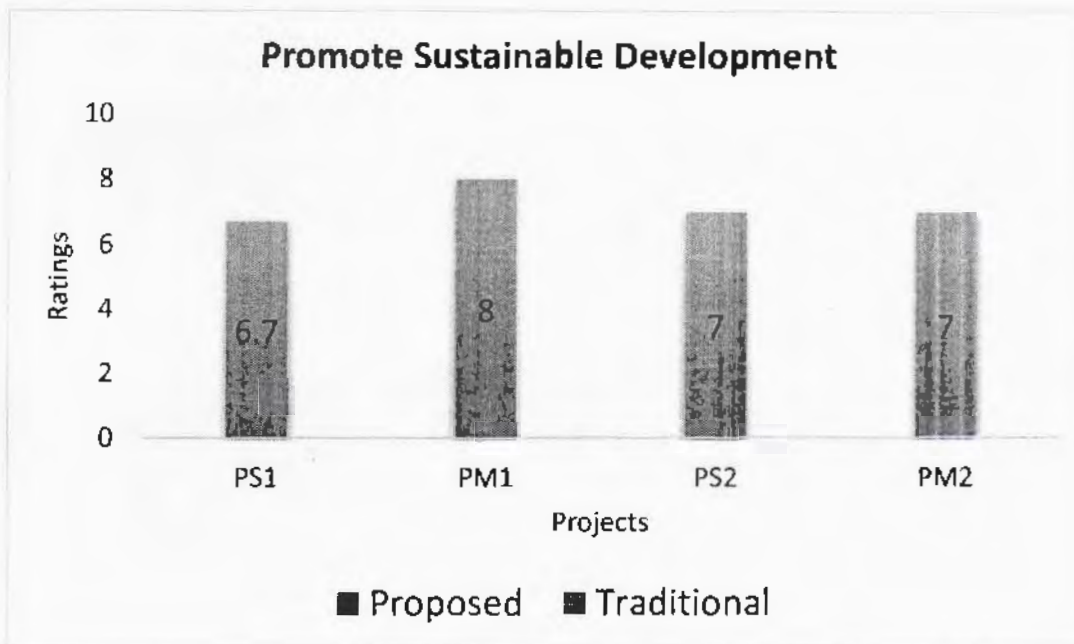


*Figure 6-14: Working Software*

Results are collected qualitatively and we have seen positive effect in delivering a working software. Architecture support resulted in clear understanding of the system to be developed. This leads to low bug ratio and increased customers' needs which resulted in providing more customer satisfaction and increasing success ratio.



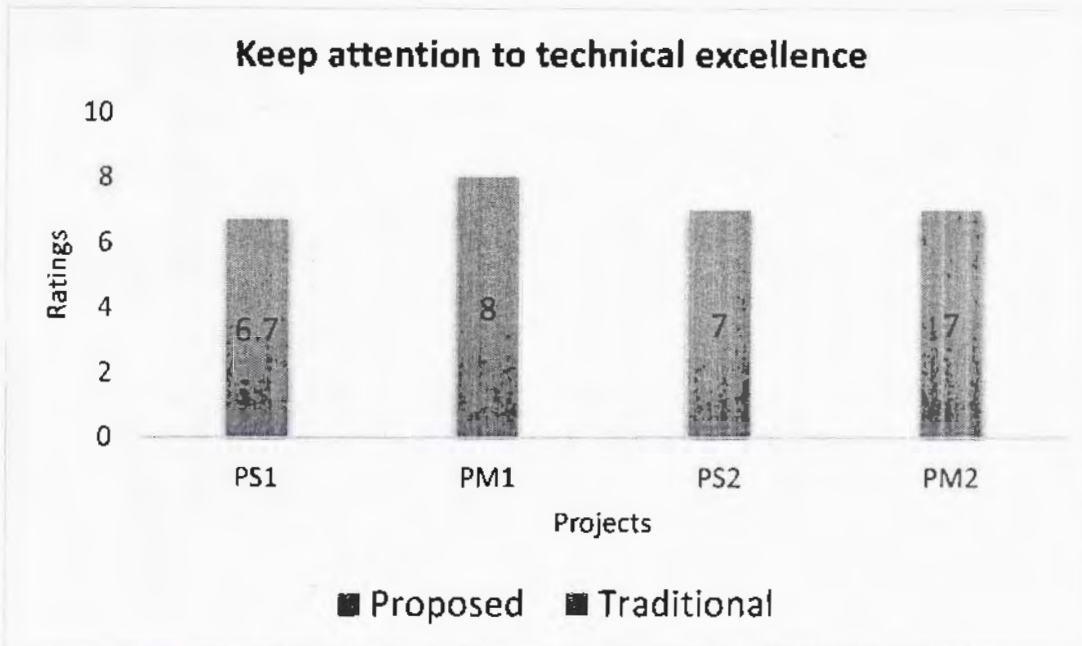
### 6.5.8 PROMOTE SUSTAINABLE DEVELOPMENT



*Figure 6-15: Promote Sustainable Development*

With newly proposed process model, we have clearly defined objectives, process and change requests with increase the constant pace of development as compared to traditional process. In traditional process, unforeseen domain and lack of customer satisfaction puts negative effect on maintaining the constant pace of delivery that is relatively reduced in newly proposed process, because of lightweight documentation and architecture support.

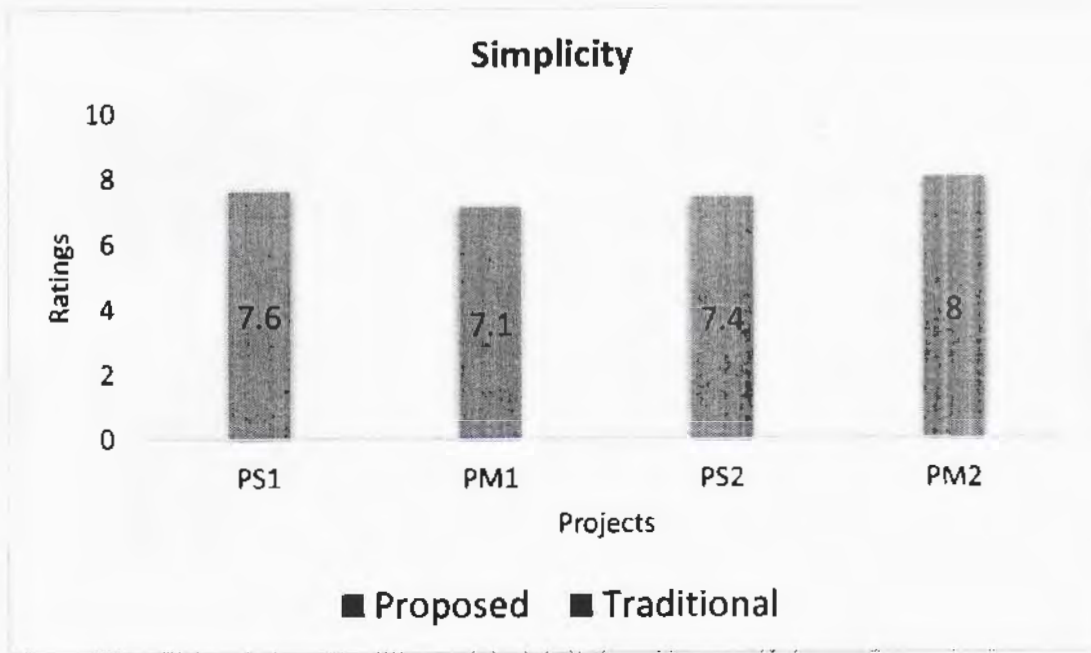
### 6.5.9 KEEP ATTENTION TO TECHNICAL EXCELLENCE



*Figure 6-16: Keep attention to technical excellence*

Results are collected qualitatively and we have seen positive effect in increasing technical excellence in new proposed process as compared to traditional process. This is because of the reason that now we have more understanding of the project in hand and developers and other system stakeholders have better understanding of each other's perspective and we have system model in place for project. This increases the analysis of the project's current technical status and helps the analyst perform better decisions.

### 6.5.10 SIMPLICITY



*Figure 6-17: Simplicity*

There is no comparative effect of proposed process and traditional process on simplicity.

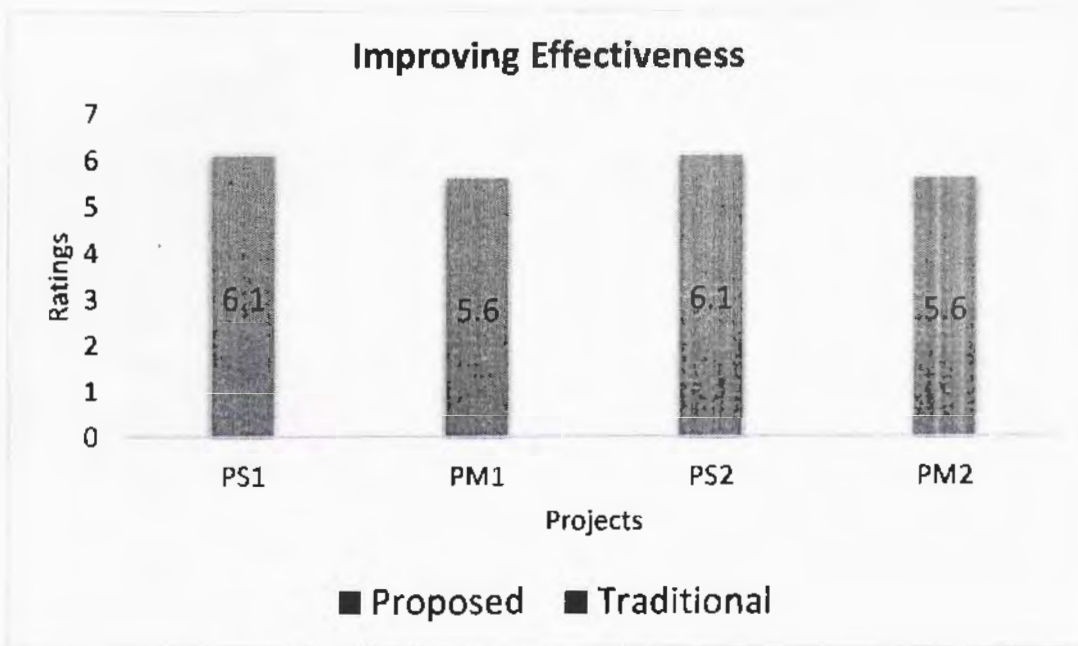
### 6.5.11 SELF-ORGANIZING TEAMS



*Figure 6-18: Self-organizing Teams*

With new proposed process, we have sub system model of the system to be developed, which helps in clear division of the system in multiple teams that facilitates the self-organization of the teams as compare to the traditional process.

### 6.5.12 IMPROVING EFFECTIVENESS



*Figure 6-19: Improving Effectiveness*

There is no considerable change on improving the effectiveness of the teams that how they can become more effective both in proposed and traditional process.

## **CHAPTER 7 CONCLUSION AND FUTURE WORK**

## **CONCLUSIONS**

Adapted architecture has been proposed in this research that is light weight and can be integrated with Feature driven development without harming the agility of this process. We 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.

This research also focuses on process adaptation that how architecture can be adapted in FDD process. Moreover, architecture evaluation is made with respect to its alignment with FDD process

## **FUTURE WORK**

In this research, architecture design methodologies were evaluated and adapted with FDD. Architecture evaluation techniques were not considered so more research needed to integrate and evaluate these methodologies too.

Moreover, adapted architecture is evaluated on small and medium sized projects with specific domain i.e. telecom project so the results cannot be generalized. Further research needed to in variable environments and variable sized projects so that results can be generalized.

Results also need to be verified on projects with different geo locations to verify their validity as currently these results confined in Pakistani context.



## 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. IEEE/IFIP 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.