Exploring the Agile Family: A Survey

Muhammad Ibrahim[1], Shabib Aftab[1,2], Birra Bakhtawar[1], Munir Ahmad[2], Ahmed Iqbal[1], Nauman Aziz[3], Muhammad Sheraz Javeid[2], Dr. Baha Najim Salman Ihnaini[4]

[1]Department of Computer Science, Virtual University of Pakistan, Lahore, Pakistan
[2]School of Computer Science, National College of Business Administration & Economics, Lahore, Pakistan
[3]Department of Computer Science, Superior University, Lahore, Pakistan
[4]Department of Computer Science, College of Science and Technology, Wenzhou Kean University, China

Summary
Selection of an appropriate software development process model is the key aspect, which leads to the development of high-quality product within scheduled time. The selection of development model depends upon various aspects, related to the project, such as: size, complexity, and scheduled time. Agile family has been satisfying the software industry since last two decades by providing various flavors of development models. Each model of the agile family consists of different practices and characteristics appropriate for specific projects. This paper provides a detail view about the work flow, structure, practices, principles, advantages and disadvantages of various famous and widely used agile models including: Test-driven Development, Extreme Programming, Scrum, Crystal Models, Feature-driven Development and Dynamic System Development Methodology.

Key words:
Agile Software Development, Agile Family, Test-driven Development, Extreme Programming, Scrum, Crystal Methodology, Feature-driven Development, Dynamic System Development.

1. Introduction

Agile models came into existence in 2001. These models provided the opportunity to develop the software application faster with iterative & incremental manner. Agile family got the customer satisfaction due to a unique feature which is: "to welcome changing requirements at any stage of development" [2],[4]. All the models under agile umbrella follow the practices mentioned in "Agile Manifesto", a parent document, contains the rules and guidelines for effective and efficient development. This document has twelve fundamental principles, which are primarily focus on following: frequent team-communication, customer satisfaction, welcome changing requirements at any stage of development lifecycle, and early delivery of working software (module) [3],[4]. The Agile Manifesto was originated in Feb 2001, by many researchers [5],[6],[10]. Various lightweight methodologies of software development were presented on behalf of agile manifesto to overcome the drawbacks of conventional models [5],[6]. According to the researchers in [6],[20],[83], four principles are the backbone of effective software development, which were neglected in conventional methodologies. These principles are about to value "Individuals and interactions over processes and tools", "Working software over comprehensive documentation", "Customer collaboration over contract negotiation", and "Responding to change over following a plan". The agile team is self-devoted, self-organized and works in close collaboration. They yield a working software in a certain frame of cycles [7]. The development team provides a bug free product as per customer requirements within scheduled time in defined iterations. Every iteration satisfies the client with completion of some requirements in the form of working software (module). Continuous and consistent feedback from the customer side is an important feature of agile models which helps the development team to get timely and prompt response from customer on the developing software. This practice reduces the chance of project failure and helps the development team to deliver the product with required functionality within scheduled time [8]. Many agile models are in use by the software industry. A particular model is selected by software industry due to its features and practices which satisfies the industry need. For example, if any industry deals in the development of large scale projects then their interest would be to choose that agile model which can handle such type of project and provide a qualitative product at the end. On the other hand an agile model with light development structure would be selected for small scale projects [9]. However, besides the advantages, the agile models also reflect some drawbacks as well. Agile methods usually focus on short team goals of an organization, and do not satisfy long-term business needs. Some of the models are more challenging and problematic in the development of large scale project [8],[11],[12]. As agile family consists of various development models so each model has its own set of values & techniques, benefits & limitations, and roles & responsibilities

Manuscript received October 5, 2020
Manuscript revised October 20, 2020
https://doi.org/10.22937/IJCSNS.2020.20.10.22

[13],[14],[25],[99]. This paper will thoroughly discuss six widely used agile process models including: Test-driven Development, Extreme Programming, Scrum, Crystal Models, Feature-driven Development and Dynamic System Development Methodology. During discussion various aspects will be focused including: development structure, phases, features, principles, practices, advantages and drawbacks.

2. Test-Driven Development (TDD)

Test-driven development is an art of 'first deciding what you want your code to do, create a failing test, then write the code to make that test pass'. This strategy is often associated with automated testing and unit testing. In addition, the principles of TDD could be used in manual testing as well [16]. It was first published in 2003 in the book "Test-Driven Development by Example" by Kent Beck. However, an early reference to its use is the mercury project developed by NASA in the early 1960s [4]. TTD methodology is an agile practice in which automated tests and passing code satisfy clients' requirements over multiple iterations. Unlike traditional methodologies, test driven development comprises of short and rapid iterations of "Create a Test, write the code to pass the test, and refactor". These short iterations generate fast feedback, and refactoring the test and code ensures simplicity and readability of evolving software [17].

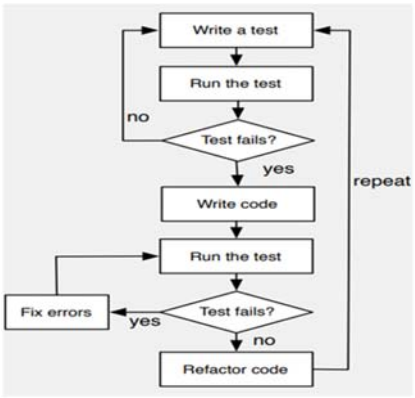

Fig. 1 Test-Driven Development Methodology [16]

2.1 Principles of TTD

Test-driven development lies on two simple and rapid development principles. These principles are about to formulating the failing test, and then write the productive code to pass the test and eliminate duplication [18].

2.2 Process Cycle of TDD

Lifecycle of TDD is comprised of six sequential steps as described below. A diagram of Test-driven development methodology is shown in Figure 1.

- Write small number of automated unit test cases
- Running new unit test cases to ensure that there is no code to pass them
- Write code which can pass implemented test cases
- Re-running new test cases to ensure the correctness of new code
- Re-Running all test cases to ensure that new code does not break any previously-test cases
- Refactor the code and Repeat the above five steps to forward next functionality of the software

Test-driven development uses a highly-iterative cycle to produce a piece of new system functionality that generally does not take more than a day. To push the next functionality in the system, all previously implemented test cases must be passed that provide some level of confidence of bug-free code up to the present time [15],[18].

2.3 Advantages of TDD

1. Unlike traditional methodologies, TTD technique is more appropriate to detect errors in the early-stage without forwarding the next functionality in the system. It is a Fault-avoidance, Fault- elimination and Fault-tolerance technique that catches error as soon as they occur in system [15].
2. Like regression testing, TTD ensures that the recent code implementation does not adversely affect existing base code [16].
3. Each iteration puts forward small functionality of the software that generally takes less than a day [15].
4. The Principles of TTD could be used in manual testing as well [19].
5. TTD is helpful to improve internal & external quality of the software [19].
6. In this development strategy, programmers break down the big system into smaller chunks that help to concentrate on each unit function, rather than the whole system at the same time and refactoring technique simplifies the logical and architectural design of the system periodically [18].

2.4 Disadvantages of TDD

1. Testing the software is the duty of tester but in TTD, it is performed by a programmer that needs extra skill to write unit test cases. In addition, it slows down the development processes [4].
2. TTD is not suitable for those projects in which software parts need synchronization [4].

3. This methodology works over less documentation, test cases are the overall system documents that are less-sufficient in the maintenance phase [4].
4. The TTD technique becomes more time consuming if more test cases are often failed [4].
5. TTD is an engineering practice and it lacks the project management aspects [4].

3. Extreme Programming (XP)

Extreme programming (XP) is one of the revolutionary & famous agile models invented by Kent Beck in 1999 [24]. It was designed to address the specific needs of software industry such as development of small project efficiently and effectively within defined time, and accepting the new or changing requirements in any stage of development, all of these features dramatically increased the software quality [54]. Extreme programming is the most widely used agile model as compared to others [89],[97],[98]. It is a lightweight and iterative methodology that delivers high-quality software with less documentation on a realistic schedule. It is a combination of best values, principles, and practices, which are collectively applied in a disciplined way to achieve a bug free, high quality software product. XP process model empowers the development team to confidently accept the new requirements and change requests even late in development cycle. This practice gains the immediate customer satisfaction [10],[22],[55].

3.1 Principles of XP

There are five fundamental principle values of XP suggested by "Kent Beck" which are extremely helpful to deliver high-quality software within defined time. The five principles are; communication, simplicity, feedback, respect, and courage. These principles guide the development team to focus on ; constant communication with the customer and fellow programmers, keeping the design simple and clean, get regular feedback from customer by testing software from the start of the project, deliver working software (module) as early as possible, and implement change-requests timely as they arise [21],[26].

3.2 Practices of XP

Extreme Programming was devised with twelve core engineering practices that can be expressed in three cycles of activities as shown in Figure 2 [6],[26],[93]. The outer cycle describes the project planning strategy which is carried out between the customers and the development team. This cycle includes the following practices; sit-together, planning games, acceptance tests and small releases practices. This cycle describes; programmers and the customer should sit-together and plan the iteration in which customer broadcast their business priorities and programmer technically estimates them which is known as "Planning games", the on-site customer provides feedback on evolving software and also helps to create acceptance test cases, XP prefer small release periodically which is the key factor in getting feedback on the actual evolvement [6],[23],[26]. The middle cycle describes teamwork and includes; coding standard, collective ownership, metaphor, continuous integration, and sustainable pace. These practices are all about team communication and coordination and about how they produce high-quality software using an effective teamwork. The cycle of these practices describe; use of the same coding standard which increases code readability and understandability, the collective ownership empowers all programmers to read, update and maintain the code, metaphor is a common system naming technique that provides the overall coherent theme to which both programmers and the customer relate to the system easily, XP team performs multiple integrations per day to keep the system fully integrated at all-time, this technique provides a constant progress report and take all programmers on the same page, XP suggests 40-hours per week work to keeps the team fresh and effective [6],[23],[26].

The innermost cycle represents core development activities including; simple design, test-first, pair programming and refactoring. These coding practices explain; use simplicity in design that increase readability & maintainability, create automated and unit test cases before writing productive code, always write code with a partner, and periodically improve internal design without affecting external behavior of the system [6],[23],[26].

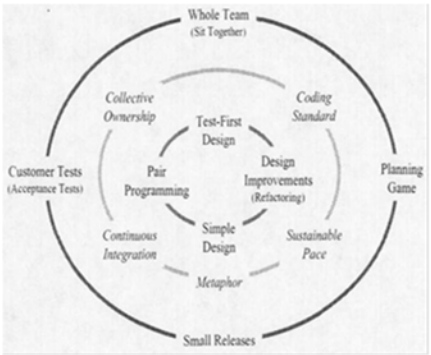

Fig. 2 XP Practices in cyclic form [6]

3.3 Team Roles

XP model does not tag the detail of each associated role. Here, some significant roles expressed by Kent Beck are as follow [39].

- **Programmers:** who design & code the software
- **Onsite Customer**: who are the clients and provide business priorities & constant feedback
- **Tester:** who test the software & assist the customer and programmer in writing test cases
- **Tracker:** Who monitors project progress, tracks goal estimates, & helps to improve process.
- **Coach:** who guides the team, and directs processes
- **Consultant:** The external guider, specifically accessed for technical and domain knowledge need
- **Manager:** A Big boss, who takes important decisions, tracks team progress, & resolve disputes in between team, and customers

3.4 Phases of XP

XP is a fine combination of ideal values and best engineering practices, these values and practices are applied in six stages which are also called phases named; exploration, planning, iteration-to-release, production, maintenance, and death as shown in Figure 3 [38]. These phases define the complete life cycle of a project in which the XP team performs four core activities including; coding, testing, listening, and designing [41].

1. In exploration phase: customer provides shortlist of business requirements and priorities. These are written down on particular story cards.
2. In Planning phase: user priorities are estimated to create a realistic plan.
3. In Iteration-to-release phase: development tasks are done in peer include; designing, coding, testing (unit testing) and integrations.
4. In production phase: customer provides feedback on rewarded work. And team restructure itself to move the project onward succeeding development (next iteration), and in maintenance phase.
5. In maintenance phase: suggested changes are made in software to surpass client expectations
6. In death phase: project transition is performed, containing final-software as per client desires [24],[36],[37],[38].

3.5 Advantages of XP

Major XP benefits include; incremental planning, respond to changing business requirements, constant feedback through small releases, good test coverage, and direct communication [55]. Core benefits of XP process model are discussed below:

- Set of twelve core practices and five values are helpful to emerge problem solutions quickly [34],[42].
- XP model promotes cost-effective development in multiple cycles through a small team [32],[24].
- XP model can accept changes at any stage of development [28].
- XP stresses on frequent delivery over unproductive documents creating [32].
- Automated tests provide frequent feedback on implemented codes that save time & cost [10],[95].
- Active user involvement keeps the project toward actual client business need [32],[35].

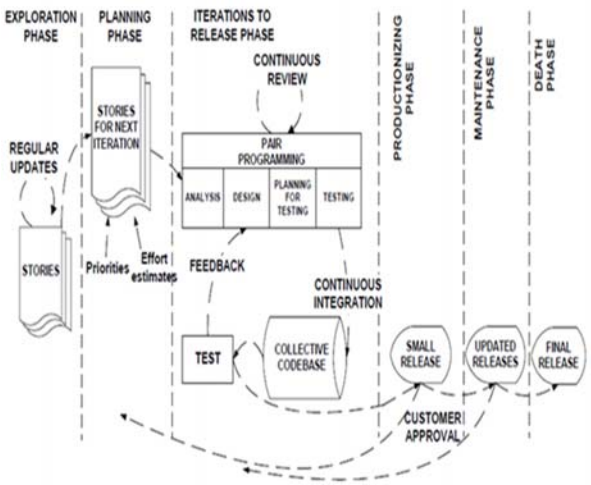

Fig. 3 Extreme Programming Project Cycle [36]

3.6 Disadvantages of XP

The Significant factors that resist the performance of XP include; week system design, less focus on paperwork, major part of the project cycle lies on oral communication, small team & close collaboration etc. [36].

In [27] researcher highlights five XP practices that are not appropriate in large scale projects with large team size. These practices are; pair-programming, planning game, collective ownership, metaphor, and oral coordination. Core limitations are discussed below:

- XP stresses on productive code writing over system designing [28],[30].
- XP methodology is iterative and non-incremental methodology [30].
- XP does not give good importance to Non-functional requirements, it only focuses on functional and productive work [40]
- Coding in pair is one of problematic situation in which both programmers need same skills, mutual understating and good coordination [27],[24]. And

double person-month are required to produce same features.
- XP process is hard to implement in a big organization having a large team size, it is only beneficial to low-risk projects with small team [29],[24].
- XP manifesto does not support geographically distributed projects and team [31],[32].
- XP lacks in project management practice and depends upon customer support that may become a risk of project failure [91],[92].
- XP is not suitable for medium and large scale projects development [96].

4. Scrum

From all agile models, Scrum is a well-planned methodology that was first explained by Ken Schwaber in 1996. Scrum process model relies on a set of meetings, tools, and roles and its ultimate goal is to ship the quality software faster. It is an iterative and incremental methodology that produces software in small sprint cycles. Often it is considered as a project management framework that helps to manage processes & structures the team [46].

Fig. 4 Scrum workflow [45]

4.1 Artifacts of Scrum

Scrum defines three artifacts that are significantly designed to maximize the transparency in work, enhance inspection, and adaptation to keep improvement in processes. These artifacts are;

- Product Backlog: this contains an ordered list of overall product requirements
- Sprint Backlog: it is the set of product backlog items selected for the sprint
- Increment: it is a sum of all product backlog items done in a sprint [7],[47],[90].

4.2 Ceremonies of Sprint

The four major events that occur inside the sprint are known as scrum ceremonies. These are described in the following.

1. Sprint Planning: it is the first scrum meeting (held between product owner, development team, and scrum master) in which the scrum team selects prioritized requirements for a sprint. The scrum master is one who organizes a sprint planning meeting. The product owner is responsible to declare important items & their priorities. And the development team is responsible to select enough & affordable work with respect to the limit of a sprint [39].
2. Daily scrum: it is a short and regular meeting between the scrum team that generally takes fifteen minutes. The goal of the daily scrum is to update the status of the project, in which the team shares the progress of the sprint, what they did yesterday, and what they are going today. And they also share impediments of the sprint [39].
3. Sprint Review: the product owner and development teams conduct the sprint review at the end of a sprint in which they validate the increment and share the progress with the customer. This meeting generally takes two to four hours [39].
4. Sprint Retrospective: the retrospective meeting is a short get-together conducted by the scrum team in which they focus on the improvement strategy of development process. In this meeting they disclose all effective and negative factors of the last sprint and create an effective plan to improve weaknesses in the successive sprint [39].

4.3 Team Roles

Scrum is comprised of three steady roles: product owner, scrum master, and development team. The team members of scrum are self-managed, self-organized, and cross-functional which performs constant communication, and interaction among the team [48],[50].

1. Product owner: he/she is responsible to describe product vision, return on investment, creates product backlog, set priorities with the customer, supports the scrum team to create sprint backlog, update product backlog after sprint review, and attends daily scrum & retrospective meetings [44].
2. Development team: the team of programmers and designers perform development tasks in the sprint like

coding, testing, and integrations. They are also responsible to take path in all meetings including sprint planning, daily scrum, sprint review, & sprint retrospective [44].

3. Scrum master: he/she works with the team for scrum success, shields the team from external interference to complete the sprint smoothly, resolve impediments, create daily scrum, and take parts in all other meetings [44].

4.4 Scrum workflow

Scum is an iterative and incremental methodology in which projects make progress in a series of sprints [6]. Each sprint delivers some amount of system features that are prioritized by the customer. The product owner gathers requirements from the customer using product backlog and defines the product vision to the team. After that, the team selects top user-requirements from a product backlog for a sprint and creates a sprint backlog. The sprint is a repeatable fix time-boxed cycle, it includes; development goals (coding, testing, and integrations), daily scrum, and product backlog refinement which are performed by developers and scrum master. At the end of each sprint, the team takes a sprint review meeting in which the team demonstrates evolving software or new system functionalities to the product owner and the interested customers provide feedback on sprint increment. After that, the team takes the sprint retrospective meeting before moving towards the next sprint [43][49][52][94]. The complete cycle is shown in Figure 4.

4.5 Advantages of Scrum

The key reasons that encourage industries to prefer Scrum on other agile models include; iterative & increment development with management facilitation, client involvement with proper channel, the shield provided by scrum master to the team from external interruptions, fix and work-friendly development cycles (sprints) to deliver work on time and process focuses on one goal at a particular time rather than whole project [33]. Below are some more significant points which also reflect the effectiveness of scrum.

- Scrum can deliver a large and complex system in small chunks of sprint easily [47],[54].
- Constant collaboration with the customer in each sprint keeps project on track [46].
- The mix and systematic strategy of ceremonies, artifacts and roles create self-management in an organization [55].
- Daily scrum meeting keeps track of task assignments of the scrum team constantly [39].
- Scrum process can also be used as a software maintenance tool [56].
- Scrum team does their full effort to reach the sprint goal within the time limit [39].
- Scrum allows the team to select affordable work that they can deliver under a time-limit calmly [57].
- Sprint retrospective activity eliminates weakness and promotes recommendations [52].

4.6 Disadvantages of Scrum

Scrum process is beneficial to the project done by a small & collocated team rather than a large or distributed team [45]. Key problems that arise in a distributed and large teams are; poor communication & coordination, lack of system architecture, integration constraints and repetition of work [58]. Few more ineffective points of scrum model based on research and experimental observation are listed below.

- Scrum gives priority to schedule the work, sprint and meeting over product quality [53]
- Scrum does not guide, how the team completes the development goals in a sprint or in a fixed interval of time [32]
- Scrum success depends on professionals and experts of scrum process [28]
- In sprint, team lacks the time to manage documents. The documentation is performed by different team which may increase bug rate in documents [59].
- The product owner and customer must have enough knowledge & experience of scrum processes, to keep scrum success.
- Once a sprint is defined then no changes can entail in it and defined work must be completed in the assigned time limit [51],[46].
- Scrum totally focuses on management rather than engineering and development activities [51]. Moreover, it does not describe the role of tester and programmers [10].
- Scrum does not offer globally distributed software development [58],[50].
- Daily scrum meeting (of fifteen minutes) is only a way to manage daily tasks [60].
- Scrum lacks the system test coverage [28].
- Late requirements may cause the integration issues in increments, which derail & distract the workflow of scrum [56].

5. Crystal Methodology

The crystal family is a set of agile and lightweight methodologies. It has different flavors that focus primarily on people's interaction and communication while working on a project over the tool or processes [10][66]. Moreover, organizations can tailor crystal methodologies to fit the

varying situations of distinct projects [61]. Crystal methodology was founded by Alistair Cockburn in the early 1990s while working in IBM on different projects. He stated that personnel skills, interaction, and constant communication give a positive impression on the development. People are important, so processes should be mold to meet the requirements of them [62].

5.1 Crystal Flavors

Crystal methodology is characterized in different color fashion, according to the number of team members being coordinated:

Clear method is designed for small projects and small team up to eight members

- Yellow is for team size 10-20 peoples
- orange is for 20-50 peoples
- Red is for 50 to 100 peoples
- And Maroon, Blue, & violet are for larger teams' size [62].

Selection of appropriate crystal method and size of the team relies on four important factors shown in Figure 5. The horizontal axis identifies the number of people being coordinated with respect to defining methodologies. The 2nd axis represents potential losses caused by an undetected defect [63],[65].

1. Loss of comfort (C)
2. Loss of discretionary money (D)
3. Loss of essential money (E)
4. Loss of Life (L)

These four factors formulate a suitable methodology for an individual project. The programmers estimate the project life (L) by seeing the team size, available money for the project (E), required money (D) & comfort of the resources to work (C) [63].

	Flavers	Clear	yellow	Orange	Red
Criticality of the project	Life (L)	L6	L20	L40	L80
	Essential money (E)	L6	E20	E40	E80
	Discretionary Money (D)	D6	D20	D40	D80
	Comfort (C)	C6	C20	C40	C80
	Team Size	upto 8	10 to 20	20 to 50	50 to 100

Fig. 5 Crystal's coverage of different project types

According to Alistair Cockburn, the small team can only handle small projects, if the project's size and complexity increases, it is necessary to add more people to complete the project within the time limit [64]. The initial three crystal methods were constructed by Cockburn; Clear, orange, and orange web. However, only the first two have been practically evaluated [39]. Clear methodology is for a small group of peoples, who want to build their own, personal, strong, and effective method to deliver software product faster. It is designed for small projects with collocated-team where less documentations is used. It generally has six developers in a team and one team works on one project [61].

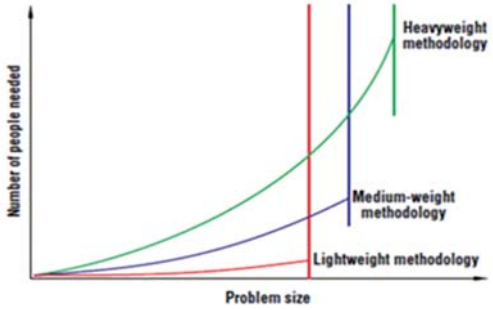

Fig. 6 How problem size and methodology affect staff numbers [64].

Crystal orange is for medium scale projects having a team size of 10-40 members. In this method, several groups of team work on the same project, while each group of people is collocated, and cross-functional just like a Holistic Diversity strategy. Orange keeps constant communication among the groups while having a heavy project team structure. Still, it does not support distributed development. It emphasizes short deliveries to reduce maintenance costs, incremental development, and accept changes in business requirements [39].

5.2 Policy Standards

In [39], the author defines some crystal policy standards that need to be applied during the development process, these policies and practices are described below.

- Deliver product incrementally on regular basis
- Deliveries are the basic milestone to track project progress
- Active user involvement
- Use automated test coverage strategies
- Take two user review on each release
- Maintain product as well as methodology in beginning and in the middle of increments

5.3 Seven Properties of Crystals

Cockburn defines seven effective properties of the crystal, which were set up by the best team interviews. The first three properties help to create a better team and the last four keep the teams & projects in a safe zone. These properties are applied to all sizes of projects except osmotic communication. Osmotic is always lying inside the small group of collocated peoples [62]. The seven properties are discussed below;

1. Rapid delivery
2. Reflective improvement & make necessary changes
3. Osmotic communication
4. Personal safety, means speak when something bothering you
5. Focus on work with peace of mind & time
6. Easy access to the expert user
7. Technical Environment with Automated Tests, Configuration Management & Constant Integration [62].

5.4 Strategies & Techniques of Crystals

Crystals are a human-oriented methodology that does not need any specific strategy and technique. However, Cockburn suggested a few strategies and techniques that are used by modern agile development teams [62]. Strategies include: Exploratory 360°, Early Victory, Walking Skeleton, Incremental Re-architecture, and Information Radiators. Techniques include: Methodology Shaping, Reflection Workshop, Blitz Planning, Delphi-Estimation, Daily Stand-up, Essential Interaction design, Process Miniature, Side-by-Side Programming and Burn Chart.

5.5 Team Roles

Crystal methodology has eight specific roles in which: executive sponsor, ambassador user, lead designer, designer-programmer are important. Other four roles are additional including; coordinator, Business expert, Technical Writer, and Business tester. The important roles are described below [39].

1. Executive Sponsor
   - Assign money for the project
   - Creates project visibility
   - Takes business-level decision
   - It decides when and whether to continue or stop the project and how to trim the remaining function of the system to recover business value
2. Ambassador User
   - He/she is a person who is well aware of the operation procedures & system use
   - It clears system requirements
3. Lead Designer
   - It is a level 3 designer
   - Creates system design
   - Having experienced of software development
   - Checks that the project is on track or not, and tells how to get back on track
   - Methodology shaping workshop
   - It works as coordinator as well as architecture, adviser, & programmer
   - It handles difficult programming tasks
4. Designer-programmer
   - The combine words of designer and programmer reflect both roles performed by a same person, it designs and programs the system

5.6 Crystal Cyclic Processes

The workflow of crystal methodology is formulated in cyclic processes of various lengths shown in Figure 6 & 7. The length of each cycle is different for each individual project, it depends upon the size of the team and criticality of the project [62].

1. Project Cycle: The project cycle contains a charting activity, two or more delivery cycles, and project wrap-up [62].
2. Delivery Cycle: The delivery cycle has four parts: recalibration of the release plan, one or more iterations, deliveries, and a completion ritual [62].
3. Iteration Cycle: The Iteration cycle has three parts: iteration planning, base code integration cycles, reflection workshop & calibration. The length of iteration is depending upon the size of the team, it may be varying from one-week to two-month [62].
4. Integration cycle: The integration cycle can run from half an hour to several days, it totally depends on the team's practices. In this cycle, the team integrates few design episodes in once a day or three times a week. However, some developments use continuous integration mechanisms. Some significant properties applied to this cycle include; technical environment, automated test, configuration management, and continuous integration [62].
5. The week & the day: The week and the day have their own rhythms; group activities occur on a weekly basis including Monday morning department meetings, regular progress reporting, a weekly seminar, or a Friday afternoon refreshment party [62].
6. Development Episode: The development episode is a unit part of actual development. A programmer picks a small design assignment in each episode and completes one or more episodes in a day. Each development should be checked under a configuration management system that may take a few minutes to a longer time, depending on the developer. It is suggested to use a

short period for each episode, ideally less than one day in length [62].

5.7 Advantages of Crystal methodology

Crystal methodology was distilled by different successful project's team interviews. It is a collection of a simple set of rules that mainly focuses on people's communication and interaction [4]. Some positive points of crystal methodology are listed in the following.

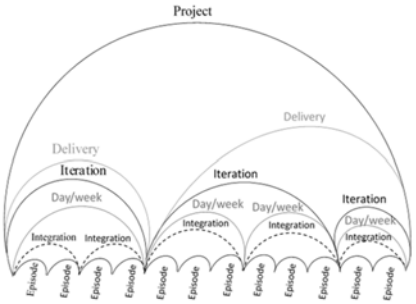

Fig. 6 Cyclic Processes of Crystal methodology.

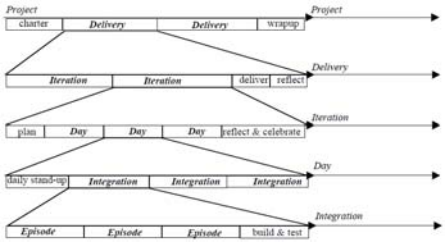

Fig. 7 Expanded Diagram of Crystal processes to show specific activities in each cycle [62].

- The crystal method is the most flexible and adjustable approach, according to the project type, team size, and project criticality [4],[10].
- Crystal methodology provides effective communication among the team [68].
- Crystal has various team communication styles for distinct projects [62].
- Osmotic communication is a good technique to help & learn with other experts [62].
- Crystal methods include; configuration management tool and technical practices for successive project development [39].
- The crystal method promotes short deliveries [62].
- Add more peoples, as project criticality increases [64].
- Every project can be developed with minimum team members [62].
- Crystal clear typically has a fixed price contract, it helps to estimate team size and expected time for a project [62].
- By using crystal methodology, Projects can be classified clearly [68]

5.8 Disadvantages of Crystal methodology

Crystal clear, crystal orange and orange-web are only the versions that are defined by Cockburn. However, other darker versions are not defined completely. Moreover, few significant limitations of crystal methodology are found in several papers that are discussed below.

- Crystal methodology lacks design and code verification system [68].
- The crystal team's do less effort on documentation, same as extreme programming [69].
- Crystal methods have no business enterprise guidance [4].
- Crystal was only implemented with a small team and small projects which were not life-critical projects [10].
- Principles of each color vary as the team size and project size changes [63].
- Crystal methodology is not suited for globally distributed projects because it needs continuous communication among the team, it is not possible when organizations have distributed departments or globally distributed team [32].
- The crystal clear and crystal orange methodologies lack the description of practices and techniques to be used by the crystal team [39].
- The project plan and the project development rely on the size of the team, rather than requirements [62].

6. Feature Driven Development (FDD)

Feature-driven development (FDD) was invented by the contribution of three experts, their names are Peter Coad, Jeff De Luca, and Stephen Palmer [1]. It was developed for a bank project that needed a proper progress reporting framework that could deliver the project iteratively & incrementally and could handle a large team [66]. FDD was first published in 1999 by Peter Coad & Stephen Palmer [70]. It uses core industry-recognized best practices that are driven from a client-valued feature perspective [6]. The list of priority features is implemented in short-iterations where a feature is a unit functionality of the system. Moreover, FDD model is mainly focusing on system architecture and building phase instead of covering the entire process [67].

6.1 Practices of FDD

FDD process model has several best practices that are derived from software engineering. These include; Domain-object modeling, Development by feature, Individual class ownership, Feature teams, Inspection, Configuration management, Regular builds, and Progress reporting [78].

1. Domain-object Modeling helps to explore and explain the overall system architecture & design, a large problem is presented in class diagrams with classes, relationships, methods, and attributes where the implementation of each class or object solves a small problem [6],[72],[74].
2. Developing by feature teaches to decompose large and complex functions of the system into sub-problems in which one or more can be implemented within two weeks. Each unit of sub-problem is called a feature of the system, this technique helps to solve big-problems with agility [6],[72],[74].
3. Individual code ownership means, assign a distinct piece of code to individual programmers, each class of code can only be implemented by their class owner. Each programmer is responsible for their assigned classes as well as consistency, performance, and conceptual integrity of the class [6],[72],[74].
4. A feature team is a small group of programmers, who works on a small part of the software, having distinct class ownership of the same functionality. Thus, the feature owner is supposed to be a team leader, who coordinates with all class owners [6],[72],[74].
5. Inspection is a process to ensure the quality of code and design, primarily to detect errors and bugs [6],[72],[74].
6. Configuration management is used to identify the latest implemented source code files and to provide historical tracking of classes as the feature team updated them [6],[72],[74].
7. Regular builds ensure the up to date and fully integrated system that can be demonstrated to the stakeholders and customers. This practice helps to uncover integration errors in base code to the feature team in the early stages of development [6],[72],[74].
8. Progress reporting is used to demonstrate the feature-team progress and project results based on complete work, it helps to make all development and process transparent inside and outside of the project [6],[72],[74].

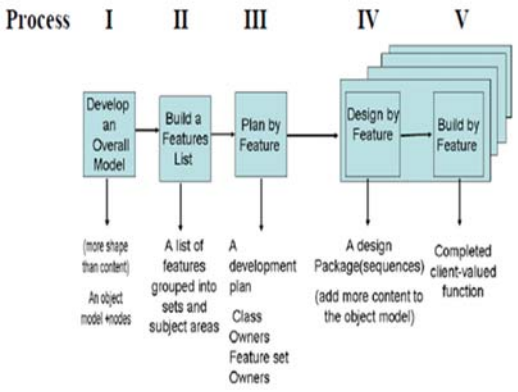

Fig. 8 Five Process of FDD Model with their outputs [72].

6.2 FDD Team Roles

Software projects consist of people, processes, and technologies where people and their roles are important. FDD defines six essential roles and eight supporting roles, these are described below [72],[78].

1. Key Roles in FDD Process
   - The project manager is the admin of the project, he is responsible for progress reporting, managing budget, handling project overheads, and managing space, equipment, & resources [72].
   - Chief Architect is the one who is responsible for the overall architecture of the system, it runs the workshop design sessions in which the team collaborates in system designing. Chief architecture should have good facilitation, technical, and modeling skills [72].
   - The development manager has the responsibility of leading daily wages development activities, resolving conflicts for resources, perform coordination in the absence of chief programmer. This role required good facilitating and technical skills [72].
   - Chief programmers are core developers, who are responsible for high-level requirement analysis and design, and leads the small team in low-level analysis, design, and development of new features [72].
   - Class Owner is a programmer, it is a member of the feature team. The responsibility of class owners includes: design, code, test, and document the features which are assigned to them [72].
   - The Domain experts may be users, sponsors, business analysts, etc. They should have good verbal, written, presentation skills. Their knowledge and participation in development are important for the vital success of the project [72].

2. Supporting Roles in FDD Process
   - The Release Manager tracks the progress of the project, collects weekly progress report from the chief programmer and then report directly to the project manager [72].
   - A Language Guru is one who helps in knowing a programming language or a specific technology if the firm is going to use the first time [72].
   - The Build Engineer is responsible for maintaining, adjusting, and running the regular build process [72].
   - The TOOlsmith is a unique role in FDD that helps the development-team, test-team, and data-conversion-team by creating small development tools [72].
   - The system administrator configures, manage, and troubleshoot the problems of servers, network, and workstations for the specific project team [72].

3. Additional Roles in FDD Process
   - The tester verifies the system independently that developed system's function meets the user requirements [72].
   - Deployer transforms the system data to the required format, it works on the physical deployment of new releases [72].
   - The Technical Writer is responsible to create user manual and user guide documents [72].

6.3 Phases of FDD

FDD process is consists of five sequential phases as shown in Figure 8. The first three phases are static in which team creates overall UML model of the desired system, create a list of system features where each feature could be implemented within two weeks, and in the third phase team creates a prioritized list of feature & an implementation plan. The successive last two phases are iterative in which actual system development occurs. Each release produces a deliverable product that can be ship to the customer. As system features are implemented, the feature list is reprioritized to highly satisfy business needs [66][71][73][75][76]. The detailed activities in each phase are described in the following.

1. In the 1st phase, the domain experts and development team create an overall system model with the help of Chief architecture. Domain experts provide a high-level walkthrough and detailed walkthrough of each area of the problem domain. After each walkthrough developers work in a small group to create an object model. Ones all problems and sub-problems are modeled then these all are merged to form the overall model which is further updated incrementally with content by design-by-feature phase [72].
2. In the 2nd phase, the chief programmers decompose the domain problems and sub-problems into major-function sets which are further breaks into small features sets to form feature list [72].
3. The third phase is project planning. In this planning phase, the project manager, development manager, and chief programmers create an implementation plan based on feature priorities and feature dependencies [72].
4. In the 4th phase, the list of selected features is scheduled for development by assigning them to chief programmers. Then chief programmers select features for the development and identify the class owners to form a feature team. The class writers refine and inspect the design of the object model based on the content of the sequence diagrams [72].
5. In the last phase, the class owners create the classes that support the design of the feature. The unit test and code inspection are performed after that code is permitted by the chief programmer to build [72].

6.4 Advantages of FDD

FDD process is one of the simple agile methods that is being used to convey large and complex projects iteratively with agility [79],[80]. Some major benefits of FDD include:

- FDD process can produce a full functional solution in the first phase. Moreover, each-increment produces a running and tangible set of features that are ready to use or deliver to the client [53].
- FDD gives more importance to designing and modeling aspects of the model and focuses on quality throughout the development lifecycle. Moreover, Simple steps of process help ramp-up new members easily [4].
- Unit testing and refactoring activities are considered as important activities [76].
- Domain experts provide better system knowledge and understanding to develop a correct system [39].
- The project cycle of the FDD process is similar to the ETVX best-known pattern [77][82].

6.5 Disadvantages of FDD Process

FDD model does not fit in all types and sizes of projects. Several limitations are maintained in the following.

- FDD model does not lead to some important development activities include; requirement gathering, requirement analysis, and risk management as these are essential activities in the development process [4].
- FDD model does not support teamwork or collective ownership in codes [56].

- System design and diagrams are only the working documents in the FDD process [39],[79].
- The overall process of FDD rely on Chief programmer, his role includes; coordinator, lead designer, and mentor [39].
- The workflow of the FDD model has a lack of requirement change management system. Moreover, the structure of the system is heavy that is explicitly dependent on experienced staff [79].
- FDD model does not guide to deal with a multi-view if there are multiple models [85].
- FDD process does not define any security role and security elements inside the system [81].

7. Dynamic System Development Method (DSDM)

Dynamic System Development Method (DSDM) was invented by the DSDM Consortium organization in 1994. DSDM is based on rapid application development principles that emphases on the people's interactions, constant user involvement, develop project iteratively, deliver system incrementally within scheduled time and budget, and adjust change requirements along with the development process. The Process of the DSDM model truly relies on people, not tools. DSDM develops high-quality business solutions that can be delivered faster at low-cost [86],[87].

7.1 Principle & Practices of DSDM

DSDM process is fundamentally based on nine principles that are common to other agile approaches. These principles are actually the techniques that are important to follow during project cycle [39],[84].

1. Active user involvement during the project cycle
2. Empower the team to make decision
3. Prefer as short and frequent product delivery as possible
4. Develop software according to business needs for acceptance criteria
5. For correct development, deliver product in iterative and incremental way
6. All changes are reversible, during development
7. The baseline of system needs is at a high level
8. High test coverage of the system throughout the project life-cycle
9. Use better collaboration and cooperation technique among team & stakeholders [88].

7.2 Roles of Users & Programmers in DSDM

DSDM model defines fifteen different roles for users and programmers. In [39], some important roles are defined that are listed in the following.

1. Developers and senior developers
   - Developers perform all development tasks including analysis, design, code, etc.
   - A senior developer would lead the team, and must have good experience in development and should be able to perform all development tasks with other developers.
2. Technical Coordinator
   - He defines the system architecture and is responsible for technical quality & technical controls in the project such as the use of software configuration management.
3. Ambassador User
   - He is from the user community that eventually uses the software or system.
   - He is responsible to define all point-of-views of the user community.
   - Disseminate the progress of the project to the other users.
   - He ensures that an adequate amount of feedback is received.
4. Advisor User
   - He is any user, who represents an important point of view from the project.
   - He can be from IT staff, financial auditor, etc.
5. Visionary User
   - He has the most accurate perception of the objective of the project.
   - He ensures that the important requirements are found at the early stage of the project that keeps project on the right track.
6. Executive Sponsor
   - He is a project sponsor form the side of user organization.
   - He has financial authority and responsibility for making decisions.

7.3 Phases of DSDM Model

The DSDM model consists of three sequential phases; the pre-project phase, the Project-cycle phase, and the post-project phase. The first phase is focused to identify the project, estimates the fund required for the project and project commitment. The post phase is about to ensure the system operating effectively and efficiently [6]. The project cycle phase comprises of five stages, in which the first two are static and are performed once in the entire project cycle whereas rest of the stages are performed iteratively and

produce software incrementally. DSDM model uses a time boxing technique to deliver a project on time, typically time box duration is taken from a few days to a few weeks [39],[87].

The five stages of the DSDM process are described as following.

1. Feasibility study Stage: in this process stage, the project is decided to develop with the DSDM framework after knowing all related aspects of the project like project feasibility, organizational & people issues, required resources, technical possibilities, and project risk. This does not take longer than a few weeks. After concerning all these, two work products are made -a feasibility report, and an outline plan for development [39],[87].
2. The Business Study: In this stage important characteristics of the business and technology are analyzed. Customer's experts are accessed to consider and prioritize all needed functions in the system. The two outputs of this phase include high-level system architecture definition and an outline prototyping plan. The high-level system architecture is a sketch of the system that is allowed to change during the project cycle. The prototyping plan defines the prototyping strategy of the next three iterative stages and a plan for configuration management [39],[87].
3. Functional Model Iteration: it is an iteration phase in which the content and approach for the iteration are planned. The analysis, code, and tests are done to produce a functional model. The four primary outputs of this phase include; Prioritized functions (a prioritized list of functions), Functional prototype review documents (user comments about current iteration), Non-functional Requirements (a list of NFR to deal with in the next stage), Risk analysis of further development (an important document in the functional model iteration phase) [39],[87].
4. The Design and Build iteration stage: in this stage, the system is mainly built. The output of this stage is a fully tested system that fulfills an adequate amount of customer requirements. Further development relies on the users' comments [39],[87].
5. The Implementation stage: it is the final stage in which the system is delivered to the customer along with user training, manual, and a project review report [39],[87].

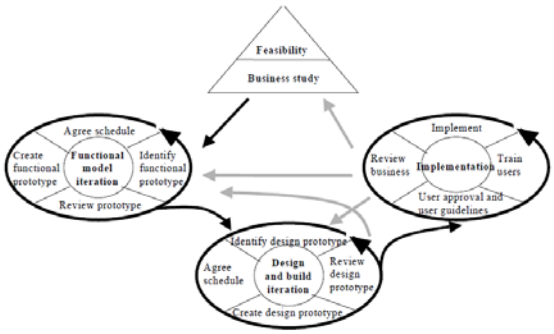

Fig. 9 DSDM Phases and Sub-Phases [39].

7.4 Advantage of DSDM Model

DSDM is a non-commercial and generic framework that relies on nine principles and practices. It provides proper development, project management, and tracking risk control mechanisms in one framework [4]. Some more significant benefits of DSDM are listed below.

- DSDM is based on Rapid Application Development Framework principles [69].
- DSDM can handle variable requirements and can maintain delivery deadline over scheduled time and available resources [30].
- Time boxing technique is the key feature of DSDM model [88].
- DSDM uses MOSCOW prioritization technique [39].
- Active user involvement is a key practice and a primary principle of DSDM model [87].
- It teaches how people from different disciplines can work together as a team [39].

7.5 Disadvantages of DSDM Model

The DSDM model all possess some drawback that limits its usefulness. Some few are listed in the following.

- It is not suitable for life-critical projects [4].
- DSDM does not define the selection of team size [4].
- It cannot deal with the average and complex project [32].
- DSDM does not support the management of sizeable teams [32].
- DSDM is not effective to use for the development of scientific or engineering applications [32].
- DSDM is more suitable for the development of small scale projects only [32].

8. Conclusion

Many software development organizations have adopted the agile process models in last two decades. The concept of agility from the agile family is quickly and widely accepted by the industry. The agile manifesto provides the solutions of various real time development challenges which helped the organization to build qualitative software as per user demand. Common benefits of agile methods include; improved collaboration among the team and stakeholders, periodically improved development processes and increase of lifecycle transparency. Moreover, all agile approaches share the same goals to deliver working software in small life cycles. This research work aimed to provide deep knowledge of agile methods including: Test-driven Development, Extreme Programming, Scrum, Crystal Models, Feature-driven Development and Dynamic System Development Methodology. These models are discussed in detail by focusing their roles, lifecycles, benefits, and drawbacks. This paper provides a solid foundation to the researchers and guides them to remove the barriers of limitation that resist the performance of agile models.

References

[1] Palmer, S. R., & Felsing, M. (2001). A practical guide to feature-driven development. Pearson Education.

[2] Wang, X. (2011, August). The combination of agile and lean in software development: An experience report analysis. In 2011 Agile Conference (pp. 1-9). IEEE.

[3] Qureshi, M. R. J. (2012). Agile software development methodology for medium and large projects. IET software, 6(4), 358-363.

[4] Anwer, F., Aftab, S., Waheed, U., & Muhammad, S. S. (2017). Agile software development models tdd, fdd, dsdm, and crystal methods: A survey. International journal of multidisciplinary sciences and engineering, 8(2), 1-10.

[5] Williams, L. (2010). Agile software development methodologies and practices. In Advances in computers (Vol. 80, pp. 1-44). Elsevier.

[6] Kavitha, C. R., & Thomas, S. M. (2011). Requirement gathering for small projects using agile methods. IJCA Special Issue on Computational Science-New Dimensions & Perspectives, NCCSE.

[7] Ashraf, S., & Aftab, S. (2017). Latest transformations in scrum: a state of the art review. International Journal of Modern Education and Computer Science, 9(7), 12.

[8] Qureshi, M. R. J., & Kashif, M. (2009, October). Seamless long term learning in agile teams for sustainable leadership. In 2009 International Conference on Emerging Technologies (pp. 389-394). IEEE.

[9] Oza, N. (2010). Lean Enterprise Software and Systems. Springer Berlin Heidelberg.

[10] Kumar, G., & Bhatia, P. K. (2012). Impact of agile methodology on software development process. International Journal of Computer Technology and Electronics Engineering (IJCTEE), 2(4), 46-50.

[11] Qureshi, M., & Abass, Z. (2017). Long Term Learning of Agile Teams. International Journal of Software Engineering & Applications (IJSEA), 8(6).

[12] Qureshi, M. R. J., & Kashif, M. (2017). Adaptive framework to manage multiple teams using agile methodologies. International Journal of Modern Education and Computer Science, 9(1), 52.

[13] Anwer, F., Aftab, S., Shah, S. M., & Waheed, U. (2017). Comparative Analysis of Two Popular Agile Process Models: Extreme Programming and Scrum. International Journal of Computer Science and Telecommunications, 8(2), 1-7.

[14] Rasool, G., Aftab, S., Hussain, S., & Streitferdt, D. (2013). eXRUP: A Hybrid Software Development Model for Small to Medium Scale Projects.

[15] Williams, L., Maximilien, E. M., & Vouk, M. (2003, November). Test-driven development as a defect-reduction practice. In 14th International Symposium on Software Reliability Engineering, 2003. ISSRE 2003. (pp. 34-45). IEEE.

[16] Munir, H., Moayyed, M., & Petersen, K. (2014). Considering rigor and relevance when evaluating test driven development: A systematic review. Information and Software Technology, 56(4), 375-394.

[17] Hayes, J. H., Dekhtyar, A., & Janzen, D. S. (2009, May). Towards traceable test-driven development. In 2009 ICSE Workshop on Traceability in Emerging Forms of Software Engineering (pp. 26-30). IEEE.

[18] Beck, K. (2003). Test-driven development: by example. Addison-Wesley Professional.

[19] Munir, H., Wnuk, K., Petersen, K., & Moayyed, M. (2014, May). An experimental evaluation of test driven development vs. test-last development with industry professionals. In proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering (pp. 1-10).

[20] Maher, P. (2009, April). Weaving agile software development techniques into a traditional computer science curriculum. In 2009 Sixth International Conference on Information Technology: New Generations (pp. 1687-1688). IEEE.

[21] Nagalambika, S., Majunath, R., & Praveen, K. S. (2016). Component Based Software Architecture Refinement and Refactoring Method in Extreme Programming. International Journal of Advanced Research in Computer and Communication Engineering, 5(12).

[22] Paulk, M. C. (2001). Extreme programming from a CMM perspective. IEEE software, 18(6), 19-26.

[23] Newkirk, J. (2002, May). Introduction to agile processes and extreme programming. In Proceedings of the 24th International Conference on Software Engineering. ICSE 2002 (pp. 695-696). IEEE.

[24] Anwer, F., Aftab, S., Bashir, M. S., Nawaz, Z., Anwar, M., & Ahmad, M. (2018). Empirical comparison of XP & SXP. IJCSNS, 18(3), 161.

[25] Anwer, F., & Aftab, S. (2017). Latest customizations of XP: a systematic literature review. International Journal of Modern Education and Computer Science, 9(12), 26.

[26] Mahajan, E. R., & Kaur, E. P. (2010). Extreme Programming: Newly Acclaimed Agile System Development Process. International Journal of Information Technology, 3(2), 699-705.

[27] Crocker, R. (2001). The 5 reasons XP can't scale and what to do about them. Proceedings of XP.
[28] Qureshi, M. (2012). Empirical evaluation of the proposed exscrum model: Results of a case study. arXiv preprint arXiv:1202.2513.
[29] Qureshi, M. (2014). Estimation of the new agile XP process model for medium-scale projects using industrial case studies. arXiv preprint arXiv:1408.6228.
[30] Balasupramanian, N., Lakshminarayanan, R., & Balaji, R. D. (2013). Software engineering framework using agile dynamic system development method for efficient mobile application development. International Journal of Computer Science and Information Security, 11(9), 126.
[31] Qureshi, M. R. J., & Hussain, S. A. (2008). An Improved XP Software Development Process Model. SCIENCE INTERNATIONAL-LAHORE-, 20(1), 5.
[32] Qureshi, M. R. J., & Bajaber, F. COMPARISON OF AGILE PROCESS MODELS TO CONCLUDE THE EFFECTIVENESS FOR INDUSTRIAL SOFTWARE PROJECTS. Cell, 966, 536474921.
[33] Qureshi, R. J., Alassafi, M. O., & Shahzad, H. M. (2019). Lean Agile Integration for the Development of Large Size Projects. International Journal of Modern Education and Computer Science, 11(5), 24.
[34] Builes, J. A. J., Bedoya, D. L. R., & Bedoya, J. W. B. (2019). Metodología de desarrollo de software para plataformas educativas robóticas usando ROS-XP. Revista Politécnica, 15(30), 55-69.
[35] Nisa, S. U., & Qureshi, M. R. J. (2012). Empirical estimation of hybrid model: A controlled case study. IJ Information Technology and Computer Science, 4(8), 43-50.
[36] Saeed, T., Muhammad, S. S., Fahiem, M. A., Ahamd, S., Pervez, M. T., & Dogar, A. B. (2014). Mapping Formal Methods to Extreme Programming (XP)–A Futuristic Approach. International Journal of Natural and Engineering Sciences, 8(3), 35-42.
[37] Ahmad, G., Rahim Soomro, T., & Raza Naqvi, S. M. (2016). An overview: merits of agile project management over traditional project management in software development. Journal of Information & Communication Technology, 10(1), 105-120.
[38] Anwer, F., Aftab, S., & Ali, I. (2017). Proposal of Tailored Extreme Programming Model for Small Projects. International Journal of Computer Applications, 171(7), 23-27.
[39] Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2017). Agile software development methods: Review and analysis. arXiv preprint arXiv:1709.08439.
[40] Anwer, F., & Aftab, S. (2017). SXP: Simplified Extreme Programing Process Model. International Journal of Modern Education and Computer Science, 9(6), 25.
[41] Juric, R. (2000, June). Extreme programming and its development practices. In ITI 2000. Proceedings of the 22nd International Conference on Information Technology Interfaces (Cat. No. 00EX411) (pp. 97-104). IEEE.
[42] Beck, K. (2000). Extreme programming explained: embrace change. addison-wesley professional.
[43] Schwaber, K. (2004). Agile project management with Scrum. Microsoft press.
[44] Sutherland, J., & Schwaber, K. (2007). The Scrum Papers. Nuts, Bolts and Origins of an Agile Process.
[45] Hassanein, E. E., & Hassanien, S. A. (2020). Cost Efficient Scrum Process Methodology to Improve Agile Software Development. International Journal of Computer Science and Information Security (IJCSIS), 18(4).
[46] Schwaber, K. (1997). Scrum development process. In Business object design and implementation (pp. 117-134). Springer, London.
[47] del Nuevo, E., Piattini, M., & Pino, F. J. (2011, August). Scrum-based methodology for distributed software development. In 2011 IEEE Sixth International Conference on Global Software Engineering (pp. 66-74). IEEE.
[48] Cho, J. (2009). A hybrid software development method for large-scale projects: rational unified process with scrum. Issues in Information Systems, 10(2), 340-348.
[49] Almutairi, A., & Qureshi, M. R. J. (2015). The proposal of scaling the roles in scrum of scrums for distributed large projects. Journal of Information Technology and Computer Science (IJITCS), 7(8), 68-74.
[50] Hossain, E., Babar, M. A., & Paik, H. Y. (2009, July). Using scrum in global software development: a systematic literature review. In 2009 Fourth IEEE International Conference on Global Software Engineering (pp. 175-184). Ieee.
[51] Kumar, A., & Goel, B. (2012). Factors influencing agile practices: A survey. International Journal of Engineering Research and Applications, 2(4), 1347-1352.
[52] Albarqi, A. A., & Qureshi, R. (2018). The proposed L-Scrumban methodology to improve the efficiency of agile software development. International Journal of Information Engineering and Electronic Business, 10(3), 23.
[53] Tirumala, S., Ali, S., & Babu, A. (2016). A Hybrid Agile model using SCRUM and Feature Driven Development. International Journal of Computer Applications, 156(5), 1-5.
[54] Ahmed, A., Ahmad, S., Ehsan, N., Mirza, E., & Sarwar, S. Z. (2010, June). Agile software development: Impact on productivity and quality. In 2010 IEEE International Conference on Management of Innovation & Technology (pp. 287-291). IEEE.
[55] Bashir, M. S., & Qureshi, M. R. J. (2012). Hybrid software development approach for small to medium scale projects: RUP, XP & Scrum. Cell, 966, 536474921.
[56] Hayat, M., & Qureshi, M. (2016). Measuring the effect of cmmi quality standard on agile scrum model. arXiv preprint arXiv:1610.03180.
[57] Schwaber, K., & Beedle, M. (2002). Agile software development with Scrum (Vol. 1). Upper Saddle River: Prentice Hall.
[58] Qureshi, R., Basheri, M., & Alzahrani, A. A. (2018). Novel Framework to Improve Communication and Coordination among Distributed Agile Teams. International Journal of Information Engineering & Electronic Business, 10(4).
[59] Ashraf, S., & Aftab, S. (2017). IScrum: An improved scrum process model. International Journal of Modern Education and Computer Science, 9(8), 16.
[60] Nikitina, N., Kajko-Mattsson, M., & Stråle, M. (2012, June). From scrum to scrumban: A case study of a process transition. In 2012 International Conference on Software and System Process (ICSSP) (pp. 140-149). IEEE.
[61] Chang, M. (2010). Agile and Crystal Clear with Library IT Innovations. In VALA2010 Conference.

[62] Cockburn, A. (2004). Crystal clear: A human-powered methodology for small teams: A human-powered methodology for small teams. Pearson Education.
[63] Grey, J. (2011). The development of a hybrid agile project management methodology (Doctoral dissertation, North-West University).
[64] Cockburn, A. (2000). Selecting a project's methodology. IEEE software, 17(4), 64-71.
[65] Saini, N. (2020). Role Of Agile Methodologies In Improvement Of Health Sector. Studies in Indian Place Names, 40(40), 1870-1876.
[66] Livermore, J. A. (2008). Factors that Significantly Impact the Implementation of an Agile Software Development Methodology. JSW, 3(4), 31-36.
[67] Paetsch, F., Eberlein, A., & Maurer, F. (2003, June). Requirements engineering and agile software development. In WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003. (pp. 308-313). IEEE.
[68] Mirza, M. S., & Datta, S. (2019). Strengths and Weakness of Traditional and Agile Processes-A Systematic Review. JSW, 14(5), 209-219.
[69] Strode, D. E. (2006). Agile methods: a comparative analysis. In Proceedings of the 19th annual conference of the national advisory committee on computing qualifications, NACCQ (Vol. 6, pp. 257-264).
[70] Coad, P., & Palmer, S. (1999). Feature-driven development. Java Modeling in Color with UML, 182-203.
[71] Pang, J., & Blair, L. (2004). Refining Feature Driven Development-A methodology for early aspects. Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design, 86.
[72] Goyal, S. (2008). Major seminar on feature driven development. Jennifer Schiller Chair of Applied Software Engineering.
[73] Siddiqui, F., & Alam, M. A. (2013). Ontology based feature driven development life cycle. arXiv preprint arXiv:1307.4174.
[74] Ozercan, S. (2010). Adapting feature-driven software development methodology to design and develop educational games in 3-D virtual worlds (Doctoral dissertation, Ohio University).
[75] Nebić, Z. (2016). Agilni razvoj programske opreme v plansko vodenih organizacijah (Doctoral dissertation, Univerza v Ljubljani).
[76] Chowdhury, A. F., & Huda, M. N. (2011, December). Comparison between adaptive software development and feature driven development. In Proceedings of 2011 International Conference on Computer Science and Network Technology (Vol. 1, pp. 363-367). IEEE.
[77] Nawaz, Z., Aftab, S., & Anwer, F. (2017). Simplified FDD Process Model. International Journal of Modern Education & Computer Science, 9(9).
[78] Aftab, S., Nawaz, Z., Anwar, M., Anwer, F., Bashir, M. S., & Ahmad, M. (2018). Comparative Analysis of FDD and SFDD. International Journal of Computer Science and Network Security, 18(1), 63-70.
[79] Aftab, S., Nawaz, Z., Anwer, F., Bashir, M. S., Ahmad, M., & Anwar, M. (2018). Empirical Evaluation of Modified Agile Models. INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS, 9(6), 284-290.
[80] Aftab, S., Nawaz, Z., Anwer, F., Ahmad, M., Iqbal, A., Jan, A. A., & Bashir, M. S. (2019). Using FDD for small project: An empirical case study. International Journal of Advanced Computer Science and Applications, 10(3), 151-158.
[81] Firdaus, A., Ghani, I., & Jeong, S. R. (2014). Secure feature driven development (SFDD) model for secure software development. Procedia-Social and Behavioral Sciences, 129, 546-553.
[82] Boehm, B. (2007). A survey of agile development methodologies. Laurie Williams, 45, 119.
[83] Rao, K. N., Naidu, G. K., & Chakka, P. (2011). A study of the Agile software development methods, applicability and implications in industry. International Journal of Software Engineering and its applications, 5(2), 35-45.
[84] Stoica, M., Mircea, M., & Ghilic-Micu, B. (2013). Software Development: Agile vs. Traditional. Informatica Economica, 17(4).
[85] Pang, J., & Blair, L. (2004). Refining Feature Driven Development-A methodology for early aspects. Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design, 86.
[86] Stapleton, J. (1999, June). DSDM: Dynamic systems development method. In Proceedings Technology of Object-Oriented Languages and Systems. TOOLS 29 (Cat. No. PR00275) (pp. 406-406). IEEE.
[87] Stapleton, J. (1997). DSDM, dynamic systems development method: the method in practice. Cambridge University Press.
[88] Sani, A., Firdaus, A., Jeong, S. R., & Ghani, I. (2013). A review on software development security engineering using dynamic system method (DSDM). International Journal of Computer Applications, 69(25).
[89] Mnkandla, E., & Dwolatzky, B. (2007). Agile software methods: state-of-the-art. In Agile software development quality assurance (pp. 1-22). Igi Global.
[90] Ashraf, S., & Aftab, S. (2018). Pragmatic Evaluation of IScrum & Scrum. International Journal of Modern Education and Computer Science, 12(1), 24.
[91] Mushtaq, Z., & Qureshi, M. R. J. (2012). Novel hybrid model: Integrating scrum and XP. International Journal of Information Technology and Computer Science (IJITCS), 4(6), 39.
[92] Qureshi, M. R. J., & Hussain, S. A. (2008). An Improved XP Software Development Process Model. SCIENCE INTERNATIONAL-LAHORE-, 20(1), 5.
[93] Kazi, S., Bashir, M. S., Iqbal, M. M., Saleem, Y., Qureshi, M. R. J., & Bashir, S. R. (2014). Requirement change management in agile offshore development (RCMAOD). Science International, 26(1).
[94] Qureshi, M. (2017). Evaluating the Quality of Proposed Agile XScrum Model. International Journal of Modern Education & Computer Science, 9(11).
[95] Khan, A. I., Qureshi, M., & Khan, U. A. (2012). A Comprehensive Study of Commonly Practiced Heavy & Light Weight Software Methodologies. arXiv preprint arXiv:1202.2514.
[96] Qureshi, M. R. J., & Barnawi, A. (2014). Kinect Based Electronic Assisting System to Facilitate People with Disabilities Using KXPRUM Agile Model. Life Science Journal, 11(10).

[97] Qureshi, M. R. J., & Hussain, S. A. (2008). A reusable software component-based development process model. Advances in engineering software, 39(2), 88-94.
[98] Qureshi, M. R. J., & Ikram, J. S. (2015). Proposal of Enhanced Extreme Programming Model. International Journal of Information Engineering and Electronic Business, 7(1), 37.
[99] Ashraf, S., & Aftab, S. (2017). Scrum with the Spices of Agile Family: A Systematic Mapping. Modern education and computer science (MECS), 9(11).