Received: December 14, 2021. Revised: September 5, 2022.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Design of a New Agile Methodology : Eclectic – An Approach to Overcome Existing Challenges

Arnika Jain[1]* **Rabins Porwal[2]** **Vineet Kansal[3]**

*[1]Department of Computer Science and Engineering, Faculty of Engineering and Technology,
SRM Institute of Science and Technology, NCR Campus, Modinagar, Ghaziabad, U.P., – 201204, India*
*[2]Department of Information Technology,
Lal Bahadur Shastri Institute of Management, New Delhi, Delhi – 110 075, India*
[3]Dr. APJ Abdul Kalam Technical University, Lucknow, U.P. - 226 031, India
* Corresponding author's Email: jain.arnika2009@gmail.com

Abstract: Agile development has proven its benefits over traditional software life cycle development models for more than a decade. Through offering agility as responsiveness to change, its practitioners maintain flexibility to accommodate change even at later stage of development process and simultaneously depict progress through iterative development. This scope creep and rapidly emerging market needs is placing more stringent constraints on agile projects in terms of meeting timelines, containing cost and adhering to quality. Though methodologies offered by agile have spawned several practices that are claimed to be efficacious, but still empirical validation and evolvement of them is required before adopting them as is for any project. This paper proposes and provides the design of a new methodology named as "Eclectic Agile Methodology". The key concept of eclectic methodology is to provide numerous advantages to mid and large scale organizations through enabling them to utilize novel and efficient ways of processes, tools and validations to assess the vast benefits before choosing any methodology upfront. The inspiration for designing this methodology is to address and overcome various untouched challenges in the current existing agile methodologies like but not limited to tightly coupling with upstream and downstream applications, sprint encountering frequent unplanned severity 1 issues which need immediate attention, shared hosted environments along with schema, dependencies outside the project, cross track impacts and distributed team.

Keywords: Agility, Distributed team, Empirical validation, Iterative development, Severity 1 issues.

1. Introduction

To develop a product, there exists a number of Software development methodologies. In early 1970's, various traditional software development processes like Waterfall model [1] and Spiral model [2] were used. However Agile development methodologies [3-6] grew out of the real-life experiences of software professionals who were tired of the challenges and limitations of the traditional waterfall methodology. The agile approach is promoted by a direct response to the issues associated with traditional software development – both in terms of overall philosophy as well as specific processes [7-8]. The prime focus of the agile is welcoming changing requirements of the customer and in building product incrementally [9-12]. Although there are many benefits of an Agile model, there are also a number of common challenges that prevent many teams from successfully scaling agile processes out to the enterprise level. As of today, many software development agile methodologies [13] exist in IT industry like Scrum[14], extreme programming, dynamic software development method, crystal methodology etc. Although these methodologies are considered the most proven and innovative methodologies today, but unfortunately they also have a number of limitations[15-16]. As there is little to no guidance on how to handle various emerging challenges like but not limited to shared resources between teams, common hosting environment,

Received: December 14, 2021. Revised: September 5, 2022.

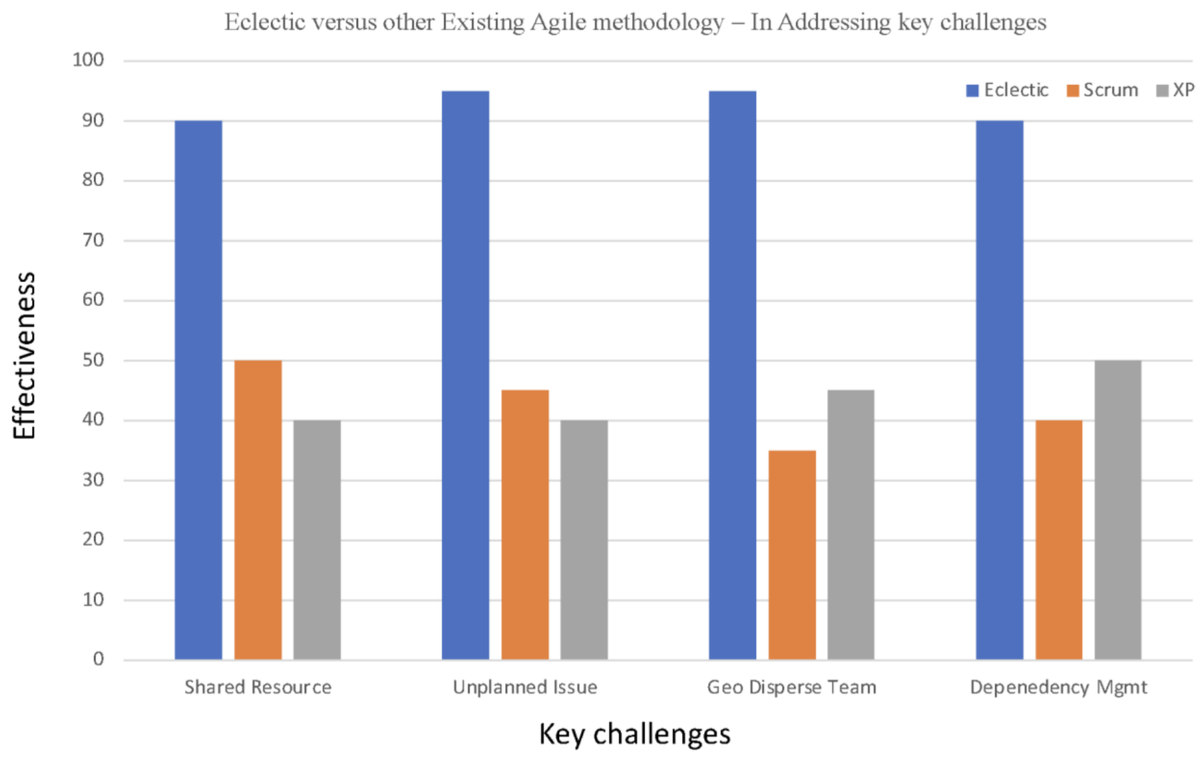


Figure. 1 Effectiveness of eclectic versus other existing agile methodologies while addressing key challenges

unplanned severity 1 issues between sprints, distributed teams in multiple time zones, minimum to no documentation, cross track impacts, dependency on upstream / downstream applications, dependency management and so on.

Fig. 1 shows the comparison of effectiveness of eclectic and other existing agile methodologies while addressing key challenges like shared resources, unplanned issues, geographically dispersed teams and dependency management. To overcome the various limitations that many software development agile methodologies [17-19] have, a new methodology has been proposed and designed in this paper named as eclectic agile methodology for executing agile projects successfully in between of various challenges. A methodology is meant for software development which is considered as a structure which is used for planning [20], executing and controlling the project that will eventually provide enormous benefits to customers who are working in these challenges [21-22].

2. Agile software development

Agile software development refers to "Software development methodologies focused over the concept of iterative development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams" [23-24]. Agile processes or agile methods generally enhance a project management process which is disciplined enough to encourage adaptation to frequently changing customer requirements and frequent inspection.

Agile development is a development process that adheres to the concepts of agile manifesto. The agile software development focuses over delivering values faster, with greater quality and predictability, and greater adaptability to respond to changed customer requirements [25-27]. Agile software development focusses over customer involvement at a higher level and also includes reviews from the project development team and the customer at regular intervals.

3. The four values of agile manifesto

The agile manifesto comprises of four foundational values and twelve supporting principles leading the agile development approach to software development. The four foundational values are applied through agile methodology in several ways guiding the process of development as well as the delivery of working software which is having high-quality. These four foundational values are given as:

1. "Individuals and Interactions Over Processes and Tools":

   The foremost important agile manifesto is "Individuals and interactions over processes and tools". The people should be given more value than processes or tools since the people give response to business needs and drive the development process. If the processes or the involved tools drive the development process then the team will be less adaptable to change and less likely to satisfy the demands of customers.

2. "Working Software Over Comprehensive Documentation":

   It has been observed from earlier traditional software development that documentation of product development consumes enormous amounts of development time. Technical specifications, technical requirements, technical prospectus, interface design documents, test plans, documentation plans, and approvals are required for each process. The list was extensive and was a cause for the long delays in development. This documentation can be reduced by agile development process by streamlining it in a way that gives the information to the developer that what is needed to do the work without getting so much into documentation. Agile documents represent customer requirements as user stories, which are then processed by a software developer to begin the task of building a new function. This agile manifesto values working software more over documentation.

3. "Customer Collaboration Over Contract Negotiation":

   The customer and the product manager negotiate the delivery details of the software in such a way that these details can be renegotiated and the customer collaboration includes the collaboration of the customer in the development of the software. In traditional software development process model like waterfall, customers define and negotiate the requirements for the product in detail prior to the commencement of the development process. In such models, the customer was involved in the development process before development process begins and after this process got completed, but not during the development process. This agile manifesto collaborates the customer throughout the development process, making the development far easier to meet the customer needs effectively. During agile development, the customers can also be included in the development process through their interactions during periodic demos.

4. "Responding to Change Over Following a Plan":

   During traditional software development, making a change in the product during its development according to customer requirements was very expensive. The focus was to develop detailed and elaborative plans, with a defined set of features and to follow these plans on a priority basis and with a large number of many dependencies on delivering in a certain order so that the team can work on the next piece of the puzzle.

4. The twelve agile manifesto principles

The agile software development methodology is focused on twelve principles designed by a software development team. These agile principles help in streamlining the development cycles thus resulting in achieving the desired goals. A working product increment is delivered to customer at regular intervals so that valuable feedback can be received timely to better enhance the adaptability. Agile principles are applicable to projects of any team size, building a stronger working connection while placing faith in individuals to do their tasks.

There is a series of sprints / iterations in agile development which break the process into multiple small processes. The duration of a sprint is 2 to 4 weeks but it can be changed depending on the product to be developed. The sprint goals are discussed with the team members and the emphasis is given over collaboration among the team and the customer.

These agile manifestos are integrated with the development process to improve the product quality efficiently according to the customer needs.

The 12 agile principles are given as:

1. "Our highest priority is to satisfy the customer through early and continuous delivery of valuable software."

   The satisfaction of the customer is very crucial for the early and ongoing success of a product. A working software is delivered to customers at regular intervals iteratively to get the feedback which is further used to improve the product quality.

2. "Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage."

   The customer views and feedback are considered timely within each sprint to better understand the requirements of the customer and to change the product to satisfy customer needs.

3. "Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale."

   Work on completing minor milestones that will eventually contribute to the overall completion of the product.

4. "Business people and developers must work together daily throughout the project."

Agile principles bring together disparate divisions by emphasizing regular cooperation and communication as a means of sharing information and resources.

5. "Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done."

   Agile considers the involvement of right people with specified skill set and assign the right roles to achieve the desired output. The team members are expected to work in a team with full support to one another with dedication.
6. "The most efficient and effective method of conveying information to and within a development team is face-to-face conversation."

   Proper daily meetings, sprint planning, sprint retrospective, product demos etc. are to be included for effective communication among the members of the team.
7. "Working software is the primary measure of progress."

   Developing something with limited or minimum features comparative to delivering whole functionalities in a single go leads to timely delivery of working product increment to the customer which leads to capturing the feedback and the faster development of the final product.
8. "Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely."

   During sprints, it's essential for product teams to have realistic targets and manageable expectations. This aids morale and prevents staff from becoming burned out.
9. "Continuous attention to technical excellence and good design enhances agility."

   After each iteration, products should be evaluated to ensure that meaningful progress is being made.
10. "Simplicity—the art of maximizing the amount of work not done—is essential."

    Agile is all about simplifying processes and streamlining entire cycle, and the agile principles help keep that on track. Even the most minor distractions or unnecessary tasks can slow progress. Embrace automation tools whenever possible.
11. "The best architectures, requirements, and designs emerge from self-organizing teams."

    Teams should be autonomous and capable of responding quickly without requiring approval for every task.
12. "At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly."

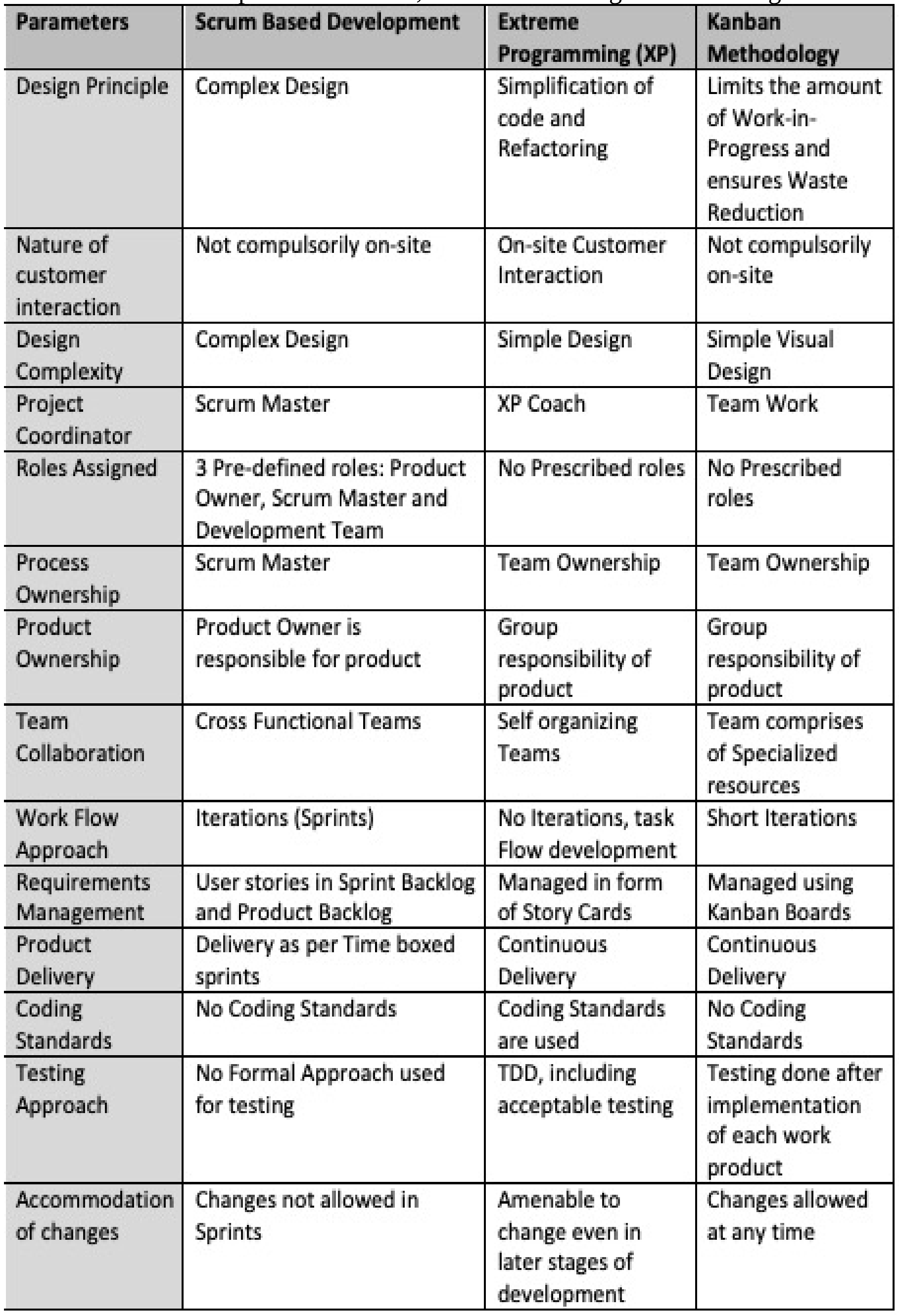
    Rather than pushing forward blindly, teams should be encouraged to assess their work and make improvements to their products.

5. Existing agile software development methodologies

Agile software development is an umbrella for a set of frameworks and practices based on the values and principles defined in agile manifesto. There are many existing agile software development methodologies and a few of them are scrum, feature-driven development (FDD), extreme programming (XP), crystal and dynamic systems development method (DSDM)[28].

1. Extreme programming (XP): Extreme programming is also known as XP. It is a "software-development-centric agile method which is intended to improve quality and responsiveness to evolving customer requirements". XP basically focuses on software development good practices. Simplicity, feedback, respect, communication, and courage are the basic values of XP, and these values are represented in the practises used throughout the XP life cycle.
2. Feature-driven development (FDD): FDD is a "simple-to-understand yet powerful approach to building products or solutions". Following the FDD approach, a project team will create an overarching model for the product, a feature list, and a work plan. The team then develops the features through design and build iterations. Develop overall model, prepare feature list, plan by feature, design by feature, and build by feature are the five essential activities in FDD.
3. Adaptive software development (ASD): By evolving their products with lightweight planning and continuous learning, adaptive system development focuses on enabling teams to rapidly and efficiently adapt to changing requirements or market needs. ASD encourages teams to develop according to a three-phase process: speculate, collaborate, and learn.
4. Dynamic systems development method (DSDM): Among various earlier agile methods, DSDM is one of them and it started out quite prescriptive and detailed. Its project life cycle coverage is vast, encompassing all aspects of an agile project from feasibility and business case to implementation. This agile project delivery methodology is used for software and non-IT solutions development. It addresses frequent IT

Table 1. Comparison of scrum, XP and kanban agile methodologies



Parameters	Scrum Based Development	Extreme Programming (XP)	Kanban Methodology
Design Principle	Complex Design	Simplification of code and Refactoring	Limits the amount of Work-in-Progress and ensures Waste Reduction
Nature of customer interaction	Not compulsorily on-site	On-site Customer Interaction	Not compulsorily on-site
Design Complexity	Complex Design	Simple Design	Simple Visual Design
Project Coordinator	Scrum Master	XP Coach	Team Work
Roles Assigned	3 Pre-defined roles: Product Owner, Scrum Master and Development Team	No Prescribed roles	No Prescribed roles
Process Ownership	Scrum Master	Team Ownership	Team Ownership
Product Ownership	Product Owner is responsible for product	Group responsibility of product	Group responsibility of product
Team Collaboration	Cross Functional Teams	Self organizing Teams	Team comprises of Specialized resources
Work Flow Approach	Iterations (Sprints)	No Iterations, task Flow development	Short Iterations
Requirements Management	User stories in Sprint Backlog and Product Backlog	Managed in form of Story Cards	Managed using Kanban Boards
Product Delivery	Delivery as per Time boxed sprints	Continuous Delivery	Continuous Delivery
Coding Standards	No Coding Standards	Coding Standards are used	No Coding Standards
Testing Approach	No Formal Approach used for testing	TDD, including acceptable testing	Testing done after implementation of each work product
Accommodation of changes	Changes not allowed in Sprints	Amenable to change even in later stages of development	Changes allowed at any time

project failures such as running over budget, missing deadlines, and a lack of user participation. DSDM is built on the following eight core concepts: Focus on the business requirement, deliver product on time, collaborate, never sacrifice product quality, build incrementally from solid foundations, develop iteratively, communicate constantly and explicitly, and demonstrate control.

5. Kanban: The kanban method is a "method for designing, managing, and improving flow systems for knowledge work". Organizations can use the approach to start with their current workflow and drive evolutionary change. They can do this by visualizing their flow of work, limit work in progress (WIP) and stop starting and start finishing.
6. Crystal: Crystal is a "Family of methodologies designed for projects ranging from those runs by small teams developing low- criticality systems (Crystal Clear) to those run by large teams building high-criticality systems (Crystal Magenta)". The crystal framework provides a great example of how a method is tailored to match a project's and an organization's characteristics.

Received: December 14, 2021. Revised: September 5, 2022.

7. Scrum: Scrum is a light-weighted and the most popular agile model and it is very easy to understand. Transparency, inspection and adaptation are three important pillars of scrum [29]. The methodology documented in the scrum framework is a set of team guidance practices, events, roles, artifacts, and rules to execute projects by.

The mostly used existing agile methodologies are scrum, extreme programming (XP) and kanban [30]. A comparison of these agile methodologies is shown in Table 1.

6. Need of proposed eclectic methodology

The determination of the best software development practice with distributed teams combined with agile principles is the logic of the proposed methodology. In the simple terms, the design for agile adoption defines a set of processes and strategies to make agile projects successful in a distributed environment. The design for agile adoption ensures that every project executed in the proposed methodology delivers low cost quality output with higher customer and team satisfaction. This methodology replaces the practices of current agile methodologies defined for a collocated team and brings in the essence of distributed development. This sets a new standard in the industry for executing agile projects in a distributed environment [31-32].

The design for proposed agile methodology solves the challenges of geographically distributed team with a lightweight process definition, along with several good practices for distributed development. The processes introduced are successful and proven practices for distributed agile development. Combining agile principles with distributed development provides every organization with the ability to manage their money effectively and efficiently. Apart from reduced cost, several other advantages, such as achieving faster time to market and higher return on investment; reduced rework and improved quality; improved responsiveness and reliability; increased accuracy and decreased risk with higher collaboration and transparency are achieved. Hence, the concept of distributed agile principles has gained a great deal of momentum in the industry[33].

7. Result

Design of eclectic methodology defines a set of values, processes, tools and strategies to make agile projects successful in the challenges and limitations which have no guidance in existing agile methodologies. Once the eclectic methodology tools and processes become ingredients of organization project execution, design of eclectic methodology ensures that every project developed under this will reap higher customer satisfaction through meeting schedule and cost effectiveness. Design of eclectic methodology adoption does not replace or ignore any of the agile principles. It defines and highlights several guidelines, techniques and tools which can help eclectic methodology project to get completed successfully. Eclectic methodology is all about harvesting the mindset of planning the right things at the right time in a right way. By following the tools, guidelines and principles of eclectic methodology, the practitioner can achieve the following goals:

1. Achieve zero to minimal slippage of schedule through managing Severity 1 during sprints.
2. Achieve within budget delivery through addressing and managing dependencies, cross track impacts etc. during the release planning.
3. Achieve right set of documentation so that product can be maintained in the long term.
4. Achieve better collaboration and communication on distributed teams.

The key building blocks of eclectic methodology revolves around the process, people and product as shown in Fig. 2. Keep optimizing the process for the people so that they can work efficiently on the product. The more we keep people motivated and collaborated, the more productive they become.

1. Have the right people: When preparing to implement the eclectic methodology, it is impediment to have positive and right set of people working on the project. Those responsible for delivery should embody a persona of service delivery mindset —get the work done, no matter whatever the obstacles are on the way. People possessing these traits will drive the eclectic methodology adoption forward. This can be achieved by putting those practitioners who understand and give value to the eclectic methodology.
2. Remember, process makes perfect: People make mistakes and these mistakes hamper the schedule and cost of the project. If everyone run in their own direction, chances of rework and conflicts become too high at the time of integration. In simple words, processes outlined in eclectic methodology are developed from several good practices. These effective processes provide consistent approach on the projects which enable our

Received: December 14, 2021. Revised: September 5, 2022.

most precious resource, our people, to work their magic!

3. Selecting and adopting the right tools: The third element of a successful implementation is the selection and use of the right tools. Eclectic methodology has defined multiple tools like upstream / downstream dependency identification and estimations and dependency management tool etc. along with the phases we have to use them.

The working of the proposed eclectic methodology is based on the concept of four segments implemented one after the other. This eclectic methodology has been shown in Fig. 3 and the four segments are given as:

1. Value and fitment assessment
2. Estimation
3. Planning
4. Implementation

7.1 Phase 1: Value and fitment assessment

The value and fitment segment is the base of the eclectic agile methodology. It assesses and demonstrates the benefits a project should achieve by adopting it[34]. To develop a project according to the needs of customer changing requirements, the right methodology should be deployed based on the characteristics and needs of the project. The fundamental concept to choose the methodology is based on the appraisal segment.

The novel criteria to select the best development methodology comprises of value and fitment assessment. The functioning of the criteria is two-fold and shown in Fig. 4. The value and fitment assessment phase of eclectic methodology starts with its first step named as value analysis. This value analysis step defines few assessment parameters that

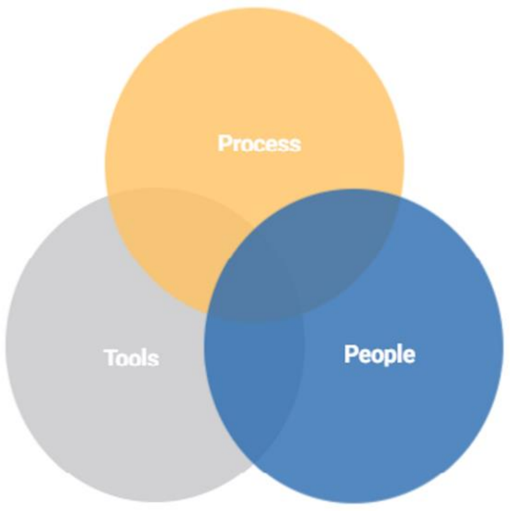


Figure. 2 Key building blocks of eclectic methodology

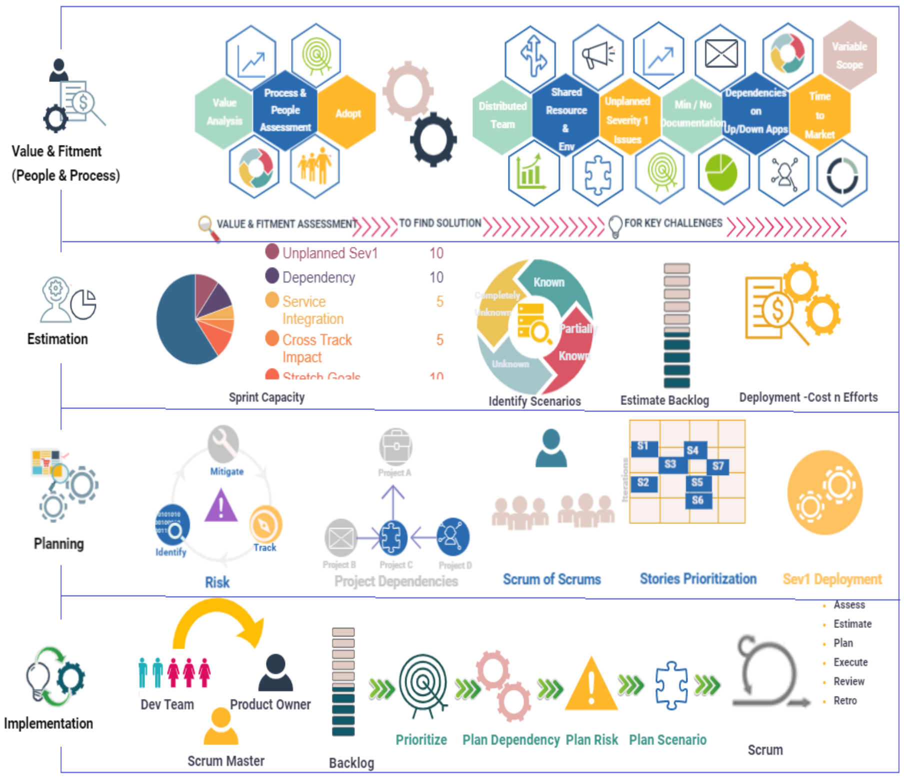


Figure. 3 Eclectic agile software development methodology

Received: December 14, 2021. Revised: September 5, 2022.

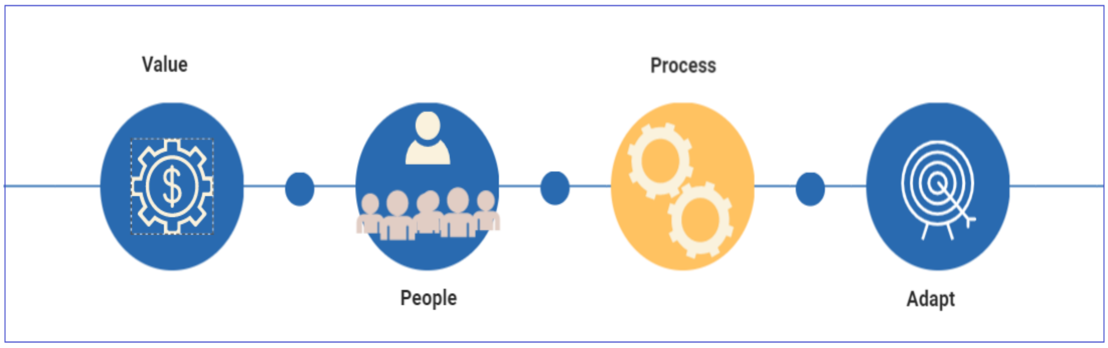


Figure. 4 Assessment chart

Table 2. Upstream / downstream dependency identification and estimation tool

“Col A”	“Col B”	“Col C”	“Col D”	“Col E”	“Col F”	“Col G”	“Col H”	“Col I”
Upstream/ Downstream dependency identification and estimations								
User Story #	User Stories	Do you completely understand the Functional Dependency of the User Story?	Do you completely understand the Technical Dependency of the User Story?	No. of Dependency on Upstream Applications (UA)	No. of Dependency for which Functional and Technical Dependency is known	No. of Dependency on Downstream Applications (DA)	No. of Dependency for which Functional and Technical Dependency is known	User Story Type
1	User Story A	Completely	Completely	10	10	10	10	KD
2	User Story B	Completely	Partially	10	9	10	9	PKD7
3	User Story C	Partially	Completely	10	7	10	7	PKD5
4	User Story D	Partially	Partially	10	5	10	5	PKD2
5	User Story E	Partially	Partially	10	2	10	2	UD

find out the expected benefits on implementing eclectic methodology in the project. Following the benefits justification, the methodology determines if the people and project capabilities are sufficient to meet the requirements.

The following are the seven value assessment parameters: “Shared resources / Hosting Environment”, “Unplanned Severity 1 issues”, “Minimum to no Documentation”, “Dependency on Upstream / Downstream applications”, “Time to market” and “Variable Scope”.

After the value analysis phase, the process and people assessment phase is used to evaluate the process and people characteristics against the methodology's standards.

7.2 Phase 2: Estimation

In agile projects, the entire team participates in the estimation process during the scrum ceremony named as sprint planning meeting. Estimations in story points is a comparative analysis to estimate the product backlog items roughly with relative sizing. Estimation in story points is highly acceptable and became a standard in Industry.

The objective of the estimation is to estimate the high priority user stories for the sprint and pick them in the sprint which could be delivered during the time box of the same sprint. Product owner ensures that the prioritized user stories have been groomed and can be picked by team for estimation.

As the scrum team in total is responsible for the delivery of the product increment, care would be taken to select the user stories for the sprint based on the size of the product increment and the effort required for the same. User story points are used to estimate the size of the product increment. Once the size is determined, the effort is estimated by means of the past data, i.e. effort per user story point called productivity.

Sprint capacity planning: Capacity planning also called commitment planning for sprint, is based on the team’s available capacity (in hours) for the sprint and tries to fill that capacity effectively without overburdening and under committing the team members. However scrum master should consider past trend to reserve some velocity for unplanned and unknown things. Stretch goals can be used as magic wand to fill the space if unplanned things don’t come across.

Eclectic methodology provides various tool to estimate the unknown challenges and ways to consider them during our team capacity and velocity planning.

7.2.1. Upstream and downstream applications

Agile teams are supposed to be atomic, cross-functional and capable of delivering a product independently to the end user completely. However, as an organization project scales and the product becomes more complex, it is quite often that team deliverables handshake with multiple upstream and downstream applications. Because agile teams are set up for self-drive and independent, having them rely on one another can cause friction, which ultimately delays projects.

The applications feeding data (inputs) to us are the downstream applications and the applications we feeding data (output) into are the upstream applications. In release planning, if there are user stories which have to do handshake with upstream and downstream applications then these dependencies should be identified very early in the project.

Eclectic methodology provides the below mentioned tool to identify and estimate the upstream and downstream applications dependencies. Let us consider the user stories which are having dependency with several upstream and downstream applications. A user story can be composed of multiple handshakes with upstream and downstream applications. Based on the Table 2, every user story / change should be identified with number of dependencies on the upstream and downstream applications and values should be put in column E and column G. There may be cases where we have identified the number of dependencies on upstream and downstream application, but do not have a complete understanding on the working of these applications or the handshake of them with our application. Column F and column H depicts the area where the user enters the total number of dependencies that are completely understood for a user story for upstream applications and downstream applications respectively.

Once all of the values have been entered into “Col A”, “Col B”, “Col C”, “Col D”, “Col E”, “Col F”, “Col G”, “Col H”, and “Col I”; scrum master should track and manage these dependencies with the scrum master of upstream and downstream projects through regular scrum of scrum and estimate the effort needed to meet these dependencies in early stage.

7.2.2. Severity 1 issues

After sprint planning, we are all set with our sprint goal, however no one knows if an unknown severity 1 issue is waiting for us. If it comes then the team focus will shift left from sprint goal to resolving severity 1 issue first which could defeat the sprint commitment. As no one wants this to happen so to overcome this challenge we need to park aside certain portion in our velocity to address these unknown P1 issues which cannot wait till next sprint. However how much velocity has to park for it needs certain analysis of past sprint behaviour. As a scrum master we need to collect past data for at least 4-6 sprints to identify trend and how much velocity has been consumed in development and in severity 1 support.

Table 3. Sample data respective to past performance of 4 Sprints

Sprints	Capacity SP	Development Effort	Severity 1 Support	Severity 1 effort in %
1	50	40	10	20%
2	50	38	12	24%
3	50	42	8	16%
4	50	38	12	24%

Table 4. Cross track impact tool

Effort Estimate	Team	SM	Tech Lead	Feature ID	High-Level Description of technical changes, potential impact	Sprint						Impacted Projects/ Functionality	Cross Track Impact (Code, Integration or None)	Comments
						1	2	3	4	5	6			
2 SP	A	SM1	TL1	F1	Description	-	-	X	-	-	-	Core-API Design-API	Code	Common method name m1 & m2 of core-API are getting changed
3SP	B	SM2	TL2	F2	Description	-	-	-	-	X	-	Design-API	Integration	Functional changes on UI
4SP	C	SM3	TL3	F3	Description	-	X	-	-	-	-	Core-API	Code	

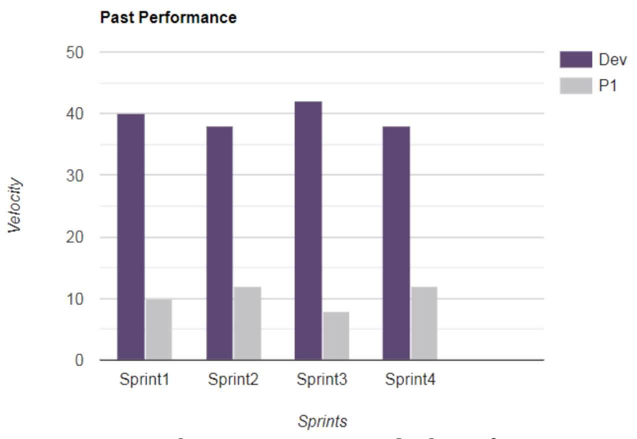


Figure. 5 Graph respective to sample data of 4 sprints

As shown in Table 3, the past performance of 4 sprints has been analysed respective to development effort as well as severity 1 issue support. As it can be seen that sprint 1 has used 80 % of its velocity for development and rest 20 % has been used for fixing the severity 1 issues. After analysing the past performance of 4 sprints, the average severity 1 effort is calculated as 21 %.

Average severity 1 effort = (20% + 24% + 16% + 24%) / 4 = 84% / 4 = 21%. An equivalent graph has also been shown for this sample data in Fig. 5.

As it can be seen from average severity 1 effort that while planning we should keep aside 21 % of our sprint capacity for addressing severity 1 issues and 79 % left for development. The solution for the same is to stretch goals in sprint. However still to mitigate risk and to abide by sprint commitment, scrum master should consider 10 % - 15 % of their development goals as stretch goals i.e. 10 % - 15 % of the sprint development goals should be set optional to achieve.

7.2.3. Cross track impact

When multiple teams are working on a common release, it is highly likely that their changes impact each other. So are the negative impact on estimates if these cross track changes left unidentified during the estimation of the release. Eclectic methodology offers cross track impact tool to identify them early in the planning and accommodate ample out of estimate for them and keeping committed items delivered at the end of sprint.

Technical discussions with-in the teams are required to determine code or testing collision due to all of the teams working in the same domain. The goal is to identify and mitigate potential collisions before they happen or to quickly resolve if/when they do to minimize impact to all. Once identified, we need to estimate the effort went into mitigating it.

The cross track impact tool has been described in Table 4 for the same and the following steps are to be followed:

1. Each team has to first validate / update the details of their team in the cross track impact tool. Then, as needed enter the high level technical changes related to your work in this PSI / release.
2. Under the sprint columns, indicate whether you will be doing just development, testing or development and testing. By testing we refer to major BA/QA testing and anything that we would want to track to assess potential impact to other teams (i.e. disabling or enabling configuration for a requirement, etc. that another team may also need at that time).
3. Enter Java projects being modified along with high level details if changes happening to any core API or common function.
4. Comments and cross track impact columns will be completed during or after our cross track discussion.

Team A mention about the impact that they feel going to happen because of the code changes they are doing in Core-API common methods, the tool is capturing the sprint in which the changes are going to merge in the common branch. Once entries are made by the producer in this table, it become senior scrum master / RTE responsibility to convey these changes to all the teams using this APIs, Then the scrum master of the impacted teams can estimate their effort to mitigate the impact. The same way other entries respective to other teams are made in the above tool, so that none of the cross functional and common code related impact left unnoticed.

7.2.4. Identifying scenarios

Known – unknown scenario: Every project will have a few scenarios that are completely clear on what has to be achieved, while some will be partially clear, and others may be completely unknown. Based on this concept, the eclectic methodology categorizes scenarios as one of the three types to define / help in the estimate. These scenarios are defined as:

1. Known scenario: These are the scenarios in a project whose requirements are completely clear on what has to be achieved. These will be referred to as KS. There is no problem in estimating them. Eclectic methodology expects that there should be zero deviations for this type of scenarios.
2. Partially known scenario: These are the scenarios in a project where you know some of the requirement, but there is more to be learned and clarified. Based on the clarity of the scenario, you can estimate the percentage of the ‘known factor’. These kinds of scenarios are expected to variance between +25 % to -25 %.
3. Unknown scenario: These are the scenarios where one has very minimal knowledge of what needs to be developed. Basis their category, estimate will have a variance of +40 % to -40 %.

7.3 Phase 3: Planning

Delivering a good quality software with required standards without any flaws is the prime focus of any project. As the development phase is considered an important step in a project, planning also plays a vital role in it. During agile software development, proper and right planning should be done along with the development process to avoid any schedule miss and over budget. Eclectic methodology recommends the following sets of tools and multiple core focus areas which have been elaborated below:

7.3.1. Dependency management

Dependencies are the stepping stones that need to happen in order for an agile team to deliver an increment, but that cannot be achieved by the team alone. It is always sensible to anticipate ahead of time dependencies that are likely to be arisen and ensure dependencies are clearly articulated, estimated and

Table 5. Dependency management tool

Feature ID	Dependency raised by	Team with Dependency	Accepted By	Delivered On	Projected Delivery Sprint	Estimate	Notes
F45078	TeamC	TeamB	SM-B	Sprint 3	Sprint 3	3 SP	-
F39876	TeamA	TeamC	SM-C	Sprint 5	Sprint 5	5 SP	-

commonly understood. Cross - team dependencies can be a source of immense friction and enough chaos thus resulting in all kind of project disfunctions. Ensure that dependency delivery timelines and scope are agreed by both parties. Leftover unmanaged dependencies can result in three times of original estimate.

Eclectic methodology offers a simple instrument, called dependency management tool which should be used to estimate and resolve the cross - team dependencies. This dependency management tool is shown below in Table 5. For the complete development of a particular feature / requirement, team A identified a dependency on team B without which, feature cannot be delivered. So for timely completion of the identified dependency, team A will fill the dependency matrix tool and approach team C for accepting their dependency. Scrum master of team C will review and discuss the dependency requirement with team A and his team; after proper understanding, he will update the “Accepted By” field of the dependency management tool.

Just because the entries have been logged in the matrix, don't assume that the other team knows about it. Reach out to them and follow the below steps. Estimate all the dependencies raised, on or by your team and keep aside the equal amount of velocity in targeting sprints to address the dependencies raised, by or on, your team.

1. Scrum master for team A logs a dependency against team B in the dependency matrix tool.
2. SM for team A reaches out to SM for team B and sets up a meeting to review if required.
3. If accepting the dependency, then SM for team B fills out the 'Accepted By' field in the matrix.
4. If rejecting the dependency, then SM for team B leaves 'Accepted By' blank and adds a brief explanation to the notes field.
5. If two teams cannot work out an agreement on a dependency by the deadline, they should go to the release train engineers for help.
6. Both the team should count the effort spent / anticipated in their estimation.

7.3.2. Scrum of scrum (SOS)

Planning and conducting SOS is a cross-team synchronization tool whose focus is to enhance the team productivity while discussing the risks that can be caused during the project execution and to help in collaborating and coordinating the work among various teams. It is a problem solving and decision-making tool which plays a vital role in any project. Well planned SOS should be conducted by scrum master / agile coach of the project with all the scrum master of other downstream and upstream projects on which our project is having dependency or dependent on as shown in Fig. 6. The focus of SOS is to find out the level of dependency between our project and the various other downstream and upstream projects. A scrum of scrums tool has been defined with 8 columns as shown in Table 6. There are some assessment parameters that are listed column wise starting from column A to column H as:

1. Project name
2. Any outstanding dependency on upstream/ downstream applications
3. Project name
4. Any outstanding dependency on upstream/ downstream applications
5. Any risks require escalation
6. Any functional spike
7. Any technical spike
8. Are all functional requirements understood
9. Any specific test data required
10. Any cross track impact

“Col A” is representing the name of the project like project A, project B and so on. These projects have been checked for any upstream / downstream applications they are dependent on or having any type of dependency. These dependencies can affect the timely and correct implementation of a project. So, once such dependencies are known, the various risks are discussed which can be the result of such dependencies. “Col B” is marked as yes if the project

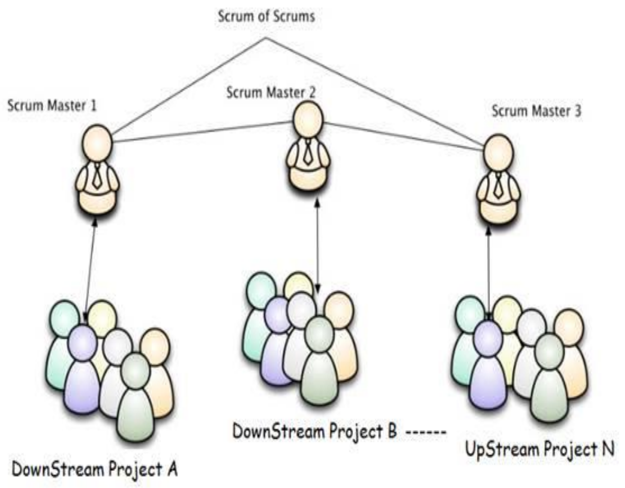


Figure. 6 Scrum of scrums tool

Table 6. Scrum of scrums tool

"Col A"	"Col B"	"Col C"	"Col D"	"Col E"	"Col F"	"Col G"	"Col H"
Scrum Of Scrums							
Project Name	Any outstanding Dependency on Upstream/ Downstream Applications	Any Risks require escalation	Any functional Spike	Any Technical Spike	Are all Functional Requirements Understood	Any specific Test data required	Any Cross Track Impact
Project A	Yes	No	-	-	-	-	-
Project B	No	Yes	-	Yes	Yes	-	-
Project C	No	Yes	Yes	-	-	Yes	-
Project D	No	-	-	-	-	-	-

is having any such dependency otherwise No. risk is a potential negative condition which can affect the project output and it should be identified and quantified. Agile teams are able to prioritise work by identifying and quantifying risk. It necessitates the development of a strategy to offset any unfavourable effects on the project.

We also need to know what we can accomplish with the resources we have, as well as be honest about how costly risk mitigation may be. For identified risks, a mitigation strategy should be devised and implemented. Since dependencies are a form of risk, it stands to reason that they should be included in any mitigation plan. Them, establish a mitigation plan for (at least) high-priority risks. A discussion has to be done to find out that whether any risks require escalation? All the possible discussion is to be made to mitigate risks and mentioned in "Col C".

Further, the various upstream/ downstream applications have been asked for any functional and technical spike and the corresponding output is to be put in "Col D" for functional spike and in "Col E" for technical spike. Then, it is checked whether the functional and technical requirements are well understood by them. And if not then how much time it will take to understand the functional and technical requirements? Do all the functional and technical requirements are well understood? This is indicated in

"Col F" of the SOS tool. Once the requirements have been discussed, they have been asked for any specific test data requirements. If it is needed then it should be provided on time to successfully mitigate risks that can be caused (in "Col G"). Then the entire things have been checked for any cross track impact? There should not be any type of delay during the release of the project. All the things should get executed in a smooth manner with desired output.

Through these scrum of scrums, scrum master should outline the target sprint of each of these dependencies, along with Target sprint validation of contract / output date, which should be shared between interdependent projects to avoid any last moment flaws during integration phase.

7.3.3. Communication plan

Communication is a key ingredient for a successful agile team, especially when the project has been executed in onshore / offshore model. Team needs to be able to collaborate and exchange information quickly. An agile communications plan's goal is to evaluate our present communication channels, prioritise them, and structure them in a way that assures effective communication and data exchange while shortening decision time as shown in Table 7. The communication plan shown here contains 4 columns represented as the person / ceremony initiating the communication, the role or the title that the person is performing, the frequency with which the person is communicating, along with the format used for the communication. Few entries have been made in the communication plan as shown.

The ground rules for the eclectic methodology communication plan are mentioned below. A proper communication plan should be thought through with the team and need to be published in beginning of the project execution which includes the following:

1. Promote collaborative culture: Ensure that the team collaborate more frequently to resolve issues through all possible channels. A phone call should be a preferred way over emails to resolve or to find a quick solution of an issue. Dedicated phone line has to be available for all practitioners.
2. Real time collaboration: Frequent real time collaboration should happen with team, all the agile ceremonies need to be done through video conferencing.
3. Information radiators: Workplaces need to be equipped with information radiators to keep everyone informed on the tasks status - red / yellow / green.

7.3.4. Planning and monitoring of schedule and cost variance

The prime focus of eclectic methodology is to control schedule and cost variance after each sprint, to better monitor the cost and schedule. Scrum master monitors the projected velocity with the actual velocity and actual spend with the projected spend. An eclectic methodology schedule and cost monitoring tool has been shown in Table 7. It has 7 columns representing the sprint number, project velocity, actual velocity, schedule deviation, projected spend in $, actual spend, and cost deviation. Suppose there is a sprint represented with sprint 1 having the project velocity as 50 SP as well the actual velocity as 50 SP. So, there is no schedule deviation

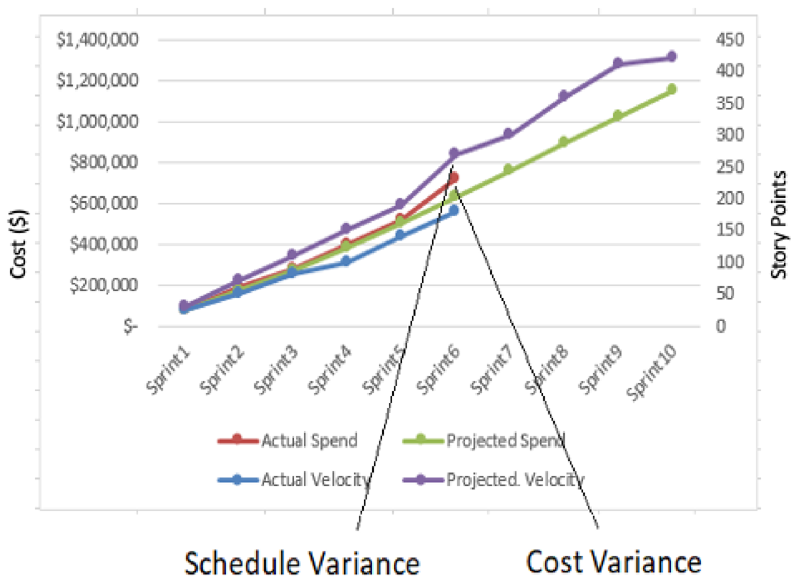


Figure. 7 Measure schedule and cost variance graph

Table 7. Communication plan

Communication Plan			
Person / Ceremony	Role / Title	Communication Frequency	Format / Channel
A	Tech Lead	Daily , weekly	-Facilitates daily with Team (Conference call) -Weekly Tech call to discuss tech debt (Video)
Sprint Planning	Scrum Master	Every Sprint	Scrum Master should facilitates this with whole team (Video)
Backlog Grooming	Product Owner	Weekly	Product should facilitates this with whole team (Video)
Sprint Retro	Scrum Master	Every Sprint	Scrum Master should facilitates this with whole team (Video)
B	QA	Daily	Reports on testing status and bug squashing (Emails & should reflect on information radiators)

Table 8. Eclectic methodology schedule and cost monitoring tool

Sprint#	Project Velocity	Actual velocity	Schedule Deviation	Projected Spend in $	Actual Spend	Cost Deviation
Eclectic Methodology Schedule and Cost monitoring						
Sprint 1	50 SP	50 SP	0%	$ 50K	$ 50K	0%
Sprint 2	50 SP	50 SP	0%	$ 45K	$ 45K	0%

in this sprint. Suppose the projected spend is $50 and the actual spend is also $50, leading to 0 % cost deviation. Similarly two more sprint data is shown in this table. Once these parameters have been measured, these values have been plotted on a graph as shown in Fig. 7.

Spend wise, we can make some estimates; we have achieved 180SP out of 420SP which is 42 % (180 / 420 *100 = 42) of the overall project. We have spent $720K of the $1153K till Sprint 6 which is 62 % (720/1153 *100 = 62) of the budget.

Having spent 62 % of your budget and only achieved 42 % of your projected progress indicates that you are spending faster than anticipated.

So to avoid all the above variances, the eclectic methodology aims to remove the blockers through its early phases of estimation and identifying challenges before they can break the momentum. Eclectic methodology schedule and cost monitoring is shown in Table 8.

KPA of the schedule management during the planning phase expect that every project executed through eclectic methodology perform continuous monitoring and do analysis with the past data

(Traditional Methodology project Variance) to compare the variances. A project is considered compliant to eclectic methodology if it achieves almost 0-2 % deviation in actual schedule for every sprint when compared to the projected estimate. In case of any deviation come across, a root cause analysis has to be done during retro and through separate meetings if required, to determine the failure to adherence with eclectic methodology guidelines.

8. Conclusion

I want to conclude this paper through jotting down the few key points. There are still unexplained complications, issues, and constraints, despite the availability of various agile techniques. In this paper, we have highlighted the issues and challenges that do exist in the existing agile methodologies and designed a new proposed methodology to overcome all those challenges with a blend of processes and tools using which any team can manage a project development successfully through adopting it. For effective and timely product delivery, selecting the appropriate approach for the intended product is critical. This paper focuses on the eclectic methodology, which is one of several methodologies available for software development.

Adopting and embracing a new methodology means getting into a new culture. Organization and its people should be willing and ready to adopt a new set of process, tools, techniques and a mindset shift. We have put our best efforts to provide solution to the major issues and challenges faced by organization while adopting the existing agile methodologies on the project. Our methodology relies on four major blocks:

1. Assessment and fitment analysis
2. Estimation
3. Planning
4. Implementation

Evolution of the eclectic methodology is based on the principles -

1. Do a fitment analysis before adopting the methodology on the project right away.
2. Manage and monitor the dependencies and cross track impact which can't be resolved within the team.
3. Meeting schedule and delivering project with the budget.
4. Value people and process, shave consistent focus on improving team efficiencies.

This study is a follow-up to our previous publications on agile methodology challenges and agile estimations, the various ways of project adaptability assessment for adopting eclectic agile methodology etc. Future research interest will include more tools and process on methodology implementation and in many fold steps involved in adopting eclectic agile methodology.

Conflicts of interest

The authors declare no conflict of interest.

Author contributions

The paper conceptualization, methodology, software, validation, formal analysis, investigation, resources, data curation, writing—original draft preparation, writing—review and editing, visualization, have been done by Arnika. The supervision and project administration have been done by Rabins Porwal and Vineet Kansal.

References

[1] Alshamrani and A. Bahattab, "A comparison between three sdlc models waterfall model, spiral model, and incremental/iterative model", *International Journal of Computer Science Issues*, Vol. 12, No. 1, pp. 106-111, 2015.

[2] W. Boehm, "A spiral model of software development and enhancement", *Computer*, Vol. 21, No. 5, pp. 61–72, 1988.

[3] Aldave, J. M. Vara, D. Granada, and E. Marcos, "Leveraging creativity in requirements elicitation within agile software development: A systematic literature review", *Journal of Systems and Software*, Vol. 157, No. 1, pp. 110396-110427, 2019.

[4] S. Campanelli and F. S. Parreiras, "Agile methods tailoring–a systematic literature review", *Journal of Systems and Software*, Vol. 110, No. 1, pp. 85–100, 2015.

[5] L. Williams, "Agile software development methodologies and practices", In: *Proc. of Advances in Computers Elsevier*, Vol. 80, pp. 1–44, 2010.

[6] T. Dingsøyr, S. Nerur, V. G. Balijepally, and N. B. Moe, "A decade of agile methodologies: Towards explaining agile software development", *Journal of Systems and Software*, Vol. 85, No. 6, pp. 1213-1221, 2012.

[7] T. Dingsøyr and N. B. Moe, "Towards principles of large-scale agile development", In: *Proc. of International Conference on Agile Software Development Springer, Cham.*, pp. 1-8, 2014.

[8] M. Paasivaara and C. Lassenius, “Scaling scrum in a large globally distributed organization: A case study”, In: *Proc. of IEEE 11th International Conference on Global Software Engineering*, pp. 74-83, 2016.
[9] P. Abrahamsson, J. Warsta, M. T. Siponen, and J. Ronkainen, “New directions on agile methods: a comparative analysis”, In: *Proc. of 25th International Conference on Software Engineering IEEE*, pp. 244–254, 2003.
[10] B. Schatz and I. Abdelshafi, “Primavera gets agile: a successful transition to agile development”, *IEEE Software*, Vol. 22, No. 3, pp. 36–42, 2005.
[11] C. C. Silva and A. Goldman, “Agile methods adoption on software development: a pilot review”, In: *Proc. of Agile Conference IEEE*, pp. 64–65, 2014.
[12] S. Zhong, C. Liping, and C. Tian-en, “Agile planning and development methods”, In: *Proc. of 3rd International Conference on Computer Research and Development IEEE*, pp. 488–491, 2011.
[13] Arnika and R. Porwal, “An overview of implementation methodologies in agile”, In: *Proc. of 11th National Conference on Next Generation Technologies for e-Business, e-Education & e-Society*, 2016.
[14] B. Nejmeh and D. S. Weaver, “Leveraging scrum principles in collaborative, inter-disciplinary service-learning project courses”, In: *Proc. of IEEE Frontiers in Education Conference (FIE)*, pp. 1–6, 2014.
[15] K. Conboy and N. Carroll, “Implementing large-scale agile frameworks: challenges and recommendations”, *IEEE Software*, Vol. 36, No. 2, pp. 44–50, 2019.
[16] T. Dyba and T. Dingsoyr, “What do we know about agile software development?”, *IEEE Software*, Vol. 26, No. 5, pp. 6–9, 2009.
[17] K. Kuusinen, “Overcoming challenges in agile user experience work: cross-case analysis of two large software organizations”, In: *Proc. of 41st Euromicro Conference on Software Engineering and Advanced Applications*, pp. 454-458, 2015.
[18] B. Boehm and R. Turner, “Management challenges to implementing agile processes in traditional development organizations”, *IEEE Software*, Vol. 22, No. 5, pp. 30–39, 2005.
[19] P. S. Shama and A. Shivamanth, “A review of agile software development methodologies”, *International Journal of Advanced Studies in Computers, Science and Engineering*, Vol. 4, No. 11, pp. 1–7, 2015.
[20] Arnika and R. Porwal, “Planning and estimation techniques in agile methodologies”, *International Journal of Pure and Applied Mathematics*, Vol. 118, No. 20, pp. 1611–1617, 2018.
[21] K. Korhonen, “Evaluating the impact of an agile transformation: a longitudinal case study in a distributed context”, *Software Quality Journal*, Vol. 21, No. 4, pp. 599–624, 2012.
[22] A. E. D. Hamouda, “Using agile story points as an estimation technique in cmmi organizations”, In: *Proc. of Agile Conference*, pp. 16–23, 2014.
[23] P. Maher, “Weaving agile software development techniques into a traditional computer science curriculum”, In: *Proc. of Sixth International Conference on Information Technology: New Generations*, pp. 1687–1688, 2009.
[24] S. Milenkovic and V. Devedži´c, “Teaching agile software development: A case study”, *IEEE Transactions on Education*, Vol. 54, No. 2, pp. 273–278, 2011.
[25] M. Kohlbacher, E. Stelzmann, and S. Maierhofer, “Do agile software development practices increase customer satisfaction in systems engineering projects”, In: *Proc. of IEEE International Systems Conference*, pp. 168–172, 2011.
[26] R. Imreh and M. S. Raisinghani, “Impact of agile software development on quality within information technology organizations”, *Journal of Emerging Trends in Computing and Information Sciences*, Vol. 2, No. 10, pp. 460–475, 2011.
[27] J. A. Livermore, “Factors that Significantly Impact the Implementation of an Agile Software Development Methodology”, *Journal of Software*, Vol. 3, No. 4, pp. 31–36, 2008.
[28] M. Paasivaara and C. Lassenius, “Scaling scrum in a large distributed project”, In: *Proc. of International Symposium on Empirical Software Engineering and Measurement*, pp. 363-367, 2011.
[29] M. Marchesi, K. Mannaro, S. Uras, and M. Locci, “Distributed Scrum in research project management”, In: *Proc. of International Conference on Extreme Programming and Agile Processes in Software Engineering*, pp. 240-244, 2007.
[30] A. M. M. Hamed and H. Abushama, “Popular agile approaches in software development: Review and analysis”, In: *Proc. of International Conference on Computing, Electrical and Electronic Engineering*, pp. 160-166, 2013.
[31] L. Cao, K. Mohan, P. Xu, and B. Ramesh, “A framework for adapting agile development methodologies”, *European Journal of*

Information Systems, Vol. 18, No. 4, pp. 332–343, 2009.

[32] S. Nerur, R. Mahapatra, and G. Mangalaraj, "Challenges of migrating to agile methodologies", *Communications of the ACM*, Vol. 48, No. 5, pp. 72–78, May, 2005.

[33] S. W. Ambler and M. Lines, "Disciplined agile delivery: A practitioner's guide to agile software delivery in the enterprise", US: IBM press. (2012). [Online]. Available: https://books.google.com/books?id=CwvBEKsCY2gC&lpg=PP1&pg=PP1#v=onepage&q&f=false

[34] Arnika, R. Porwal, and V. Kansal, "Project adaptability assessment for adopting new eclectic agile methodology", *Turkish Journal of Computer and Mathematics Education*, Vol. 12, No. 11, pp. 2763–2771, 2021.