IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 24 October 2023, accepted 7 November 2023, date of publication 9 November 2023,
date of current version 27 November 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3331759

TOPICAL REVIEW

Practices of De-Motivators in Adopting Agile Software Development Methods at Large Scale Development Teams From Management Perspective

FARMAN ALI[1], MUHAMMAD USMAN[2], (Senior Member, IEEE), MUHAMMAD FAISAL ABRAR[2], SHAMS UR RAHMAN[2], INAYAT KHAN[2], AND BADAM NIAZI[3]

[1]Department of Computer Software Engineering, University of Engineering and Technology, Peshawar, Peshawar, Khyber Pakhtunkhwa 25000, Pakistan
[2]Department of Computer Science, University of Engineering and Technology, Mardan, Mardan, Khyber Pakhtunkhwa 23200, Pakistan
[3]Faculty of Computer Science, Nangarhar University, Jalalabad 2601, Afghanistan

Corresponding author: Badam Niazi (badam@nu.edu.af)

ABSTRACT During the last few decades, agile methodologies have become the most widely used software development methods. Agile methodologies provide great customer satisfaction and fast product delivery at the low cost. The low cost and customer satisfaction make agile methods highly attractive. Agile methods were designed and used, in its early days, for small-scale projects. There are many challenges in applying agile methods for large-scale projects as determined by some previous studies. The objective of this research is to identify the practices of De-motivators in adopting agile projects at a large scale from management point of view by employing a systematic literature review (SLR).In the review process, a total of 72 practices were discovered. De-motivator factors have been identified and ranked, through the analytical hierarchical process, through SLR in our previous study (Abrar et al., 2020). Analytical hierarchical processing (AHP) is a well-known method used for multi-criteria decisions implemented by Satty (2023). In this study AHP is used to prioritize de-motivators and their categories based on their relative importance. The prioritized de-motivators and their categories provide a strong background to the software project manager and practitioners for adopting Agile Software Development in large-scale projects.

INDEX TERMS Agile software development, agile transformation, adopting agile methodologies, de-motivators, large-scale agile, practices, solutions, systematic literature review.

I. INTRODUCTION

Software project managers face several issues with software development projects in terms of quality, cost and time. Many of these issues can be overcome by deploying Agile Software Development Methods (ASDM) [1]. ASDM is a set of methods for fast and iterative software development [2]. The Agile Manifesto prioritizes individuals, interactions, working software, customer collaboration, and adaptability over rigid processes and tools, comprehensive documentation, contracts, and fixed plans. It provides a set of four core values and 12 guiding principles to promote agility and customer-centricity in software development. Agile methods provide an abstract structure for undertaking a worldwide distributed and co-located software project teams. Agile methods depend on team member's expertise and informal communication rather than formal and comprehensive paperwork. Therefore, agile methods pursue to avoid overwhelming processes, having little contribution to actual software development. Such approaches are strongly associated with progressive and incremental development [3]. Timely and persistent delivery of quality software, flexibility

The associate editor coordinating the review of this manuscript and approving it for publication was Porfirio Tramontana.

towards change, simple and polymorphic design, promotion of sustainable development, good contact, and working together between the customers and agile team participants are the main focus of these methods [4]. Agile methodologies include Dynamic System Development Method, Scrum, eXtreme Programming (XP), Crystal, Adaptive Software Development, and Feature-Driven Development.

Agile methods were originally developed for onsite collaborating teams working on software development projects. The Agile Manifesto and many of the early Agile methodologies, such as Scrum and Extreme Programming (XP), were conceived with the idea of co-located teams in mind. These methodologies encouraged face-to-face communication and close collaboration among team members, stakeholders, and customers, which were more practical in traditional office settings [1]. However, over time, Agile methodologies have evolved and adapted to accommodate distributed and remote teams by incorporating tools and practices that support effective communication and collaboration, even when team members are not physically located in the same place. This flexibility has allowed Agile principles to remain relevant in today's diverse work environments. Therefore, Scrum and XP have been restructured as Distributed Scrum and Distributed XP [5]. To meet market demands while faced with ever-increasing competition, large-scale development firms are adopting agile methods because these methods enable software developers to deliver quality software in a short period of time. Agile methods were developed to enhance the development phases by eliminating hurdles and to make development phases to accepting business requirement changes. With agile methods, it is not essential to constrain design details and business requirements during the development of software [5]. Agile methods can be adapted for large-scale software development projects, but they require adjustments and the use of scaling frameworks like SAFe or LeSS. These adaptations address complexities such as team coordination, backlog management, communication, dependencies, testing, compliance, and cultural shifts. Successful large-scale Agile implementation hinges on tailoring Agile principles and practices to suit the organization's unique needs while fostering a culture of agility and continuous improvement. Large-scale Agile adoption is necessary for organizations that employ extensive teams to tackle complex challenges, harness symmetrical employment delivery, foster collaboration with external partners, adhere to industry-specific laws and regulations, and navigate the intricacies of cultural, environmental, and technological landscapes. [1].

II. LARGE SCALE AGILE PROJECTS

Early agile practices were designed for small-size teams and projects [8]. Though because of their utility, these methodologies can be practical to LSAD teams and projects. As large-scale software projects require more coordination than small-scale projects, so it is highly challenging to adopt agile for large-scale projects and teams [9]. In addition to the development team, LSAD projects involve contributions of other organizational elements such as which are project management, Human Resources (HR), and marketing [10].Furthermore, there are challenges in aligning different organizational cultures, managing resistance to change, reallocating resources effectively, promoting cross-functional collaboration, adapting performance metrics and processes, and addressing fixed marketing campaigns and talent management practices. Overcoming these challenges often requires investment in training, change management, and fostering open communication to ensure a successful integration of Agile principles across departments [7], [13], [11]. The size can be counted in terms of individuals or number of staff, code size, and cost estimates of the project.

There are a range of criteria proposed by various researchers. For example, according to [14], a project comprised of 7 teams and forty people can be characterized as a large-scale project, whereas according to [15], a project with costs of more than 10,000 Great Britain Pound (GBP) and a team comprised of fifty peoples may be characterized as a large-scale project. Moreover, [18] argues that a project that contains more than five million LOC will be considered a large-scale project; the authors of [16] argue that a project would be characterized as a large-scale project if it is with a range of 60 to 80 qualities/ characteristics within two years of spell time; the authors of [17] argue that the scope of a project can be quantified by the amount of team coordination and collaboration. The study implies that a project containing 2 to 9 cooperating teams would be considered a large-scale project whereas a software product with more than ten cooperating groups will be labeled as large size development project. Furthermore, in the above-mentioned studies, various researchers also discussed LSAD projects. Many of these studies claim that the number of personnel is the most important factor while characterizing the size of a project. Moreover [20] suggests that agile projects comprising about 50 personnel will be termed small projects while projects having 50 to 100 personnel will be termed large projects [21]. Whereas the previous study concludes that, we can express large scale to point out Software Development Organizations (SDOs) comprises of fifty persons or more than 50 persons or contains a minimum of 6 development teams. Furthermore, all employees are to be developers besides that they should belong to that SDO. Moreover, all employees shall be employed for the same project or working on the same product which means they will have collaborate such as, software architects and scrum controllers. In addition, agile methodologies also deal with functions associated with business and management. Movement in the direction of the iterative and feature-centric model is the most difficult task for an organization as that would mean abandoning life-cycle models [12] and a transformation in attitude. The emphasis has to be shifted from long-term organizational planning to ephemeral project-level planning [22]. Agile methodologies advocate short-term planning [23], although commercial client relationship policies often encourage enduring organizational strategies because such policies consider efficiency

in planning a matter of serious concern. ASDM have been both condemned and supported. Research has manifested that accepting change may be a part and parcel of failure and success [24]. It has been concluded by the research community that agile methods have raised the expectation of both clients and salespersons. It is worth mentioning that some studies argue that ASDM might not be suitable for large development teams and projects [9]. So far by the mean of secondary studies (SLR), the practices of Demotivatiors of LSAD from a software project manager's point of view have not been discovered. This study aims to fill the gap with a handout that can be resultant in SLR for LSAD teams and projects from a software project manager's point of view. Moreover, the significance of ASDM in the background of LSAD teams from the software project manager's point of view, there has not been carried out SLR towards ASDM on projects on a large scale from the software project manager's point of view in common and on the finding the practices that have a notable influence on vendor organization in particular terms.

III. SCOPE AND SIGNIFICANCE OF THE STUDY

Adopting ASDM methodologies for large-scale projects requires overcoming many challenges issues from a project manager's perspective [1]. Scaling agile methods for large-scale development projects various de-motivators were found in our previous study. Practices of de-motivators will assists projects managers in scaling agile at large size from software project manager point of view. So a detailed Systematic Literature Review (SLR) was conducted to discover practices of those de-motivators for adoption of agile at large-scale from the point of view of management team. A total of 15 de-motivators factors were identified through SLR in our previous study, in the current study AHP is used to prioritize and explore the de-motivators for scaling agile methodologies in large-scale projects. Analytical hierarchical processing (AHP) is a well-known method used for multi-criteria decisions [68]. The AHP method was used to prioritize de-motivators and their categories based on their relative importance. The de-motivators and their categories provide a strong background to the software project manager to adopt agile at large-scale projects.

However, for the identification of practices of de-motivators, SLR on agile software development has not been undertaken with a view to large-scale development from a view of management which has a significant impact on the vendor organization in particular, irrespective of the significance of agile projects from a software project manager's point of view. The subsequent Research Question are propose to identify these factors.

RQ1. What are the practices / solutions De-motivators in agile large-scale software development from software project manager point of view through Systematic literature review?

RQ2. How could the identified De-motivators be prioritized for successful management of Large-Scale Software Development Project using the AHP approach?

IV. RESEARCH METHODOLOGY

The research approach consists of a systematic review of literature (SLR) approach and an Analytical hierarchical process (AHP).

A. SYSTEMATIC LITERATURE REVIEW

SLR is a secondary study and is used to extract, explore and evaluate appropriate data from primary studies. SLR is a comprehensive and unbiased process for collecting; interpreting and evaluating all published literature related to a particular topic, research question, or phenomenon of interest [6]. Systematic literature reviews are different from ordinary literature surveys because SLRs possess different scientific, technical, and methodological values. The second phase consists of an empirical study to validate the SLR findings and to identify practices for the identified de-motivators. The research methodology was demonstrated in Figure 1. To go a more in-depth analysis of the existing literature, we use an additional snowballing process.

The practices of de-motivators to adopt ASDM at a large scale were determined in our previous study [1]. To the best of our knowledge, no other studies have conducted SLR on the adoption of ASDM and the use of practices of De-motivators implementation in this context on a large scale. This study aims at applying AHP at the discovered practices of De-motivators and rank them based on their relative priority and importance. Therefore, the findings of this study will enable project managers, team members, and developer to successfully adopt agile methodologies. Below we describe the stages used for conducting SLR [25]. These stages are also depicted in Figure 1.

1) SEARCH STRATEGY

Search strings were developed and used to conduct manual search on various digital libraries, such as, Springerlink, IEEExplore, ACM, Google Scholar, and ScienceDirect.

2) SEARCH STRING

A manual search was conducted using the developed search strings on different digital libraries. These libraries have some constraints on the search strings. Some libraries accept large strings while others require concise strings. Therefore, several trial search strings were developed, the details of which are following.

3) TRIAL SEARCH STRING

The following trial search string was used, as a baseline, forsearchingi the Google Scholar Library to identify whether the results are accurate. The search string was (''Agile Method'') AND (''Large Scale Development Team'') AND (''de-motivators'' OR ''Risk Factor'' OR ''barriers''). If the trials search produced the intended results, then it will be to search other libraries also.

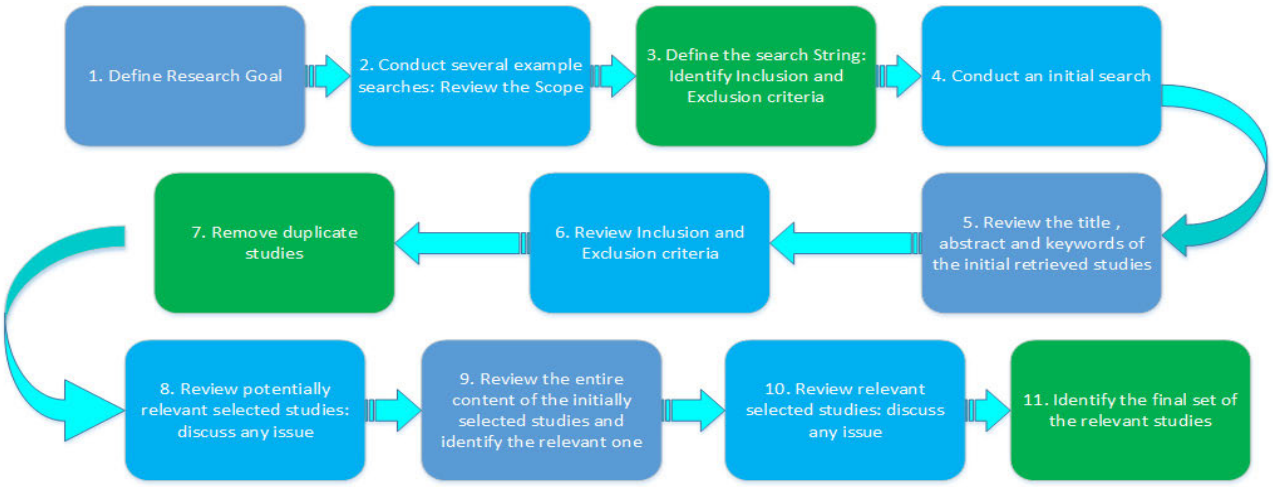

FIGURE 1. Steps involved in SLR methodology.

4) LENGTHY SEARCH STRING

To reduce the possibility of missing any relevant literature, synonyms of the keywords and major terms of the search string were identified and a combination those was constructed using Boolean operators (AND, OR). Digital libraries and databases were searched using lengthy search strings constructed by a combination of keywords and their synonyms. If the search with a lengthy string succeeds then the obtained results are processed further otherwise the lengthy string is broken down into smaller sub-strings.

5) SMALLER SUBSTRINGS

As mentioned earlier, some of the digital libraries do not accept lengthy strings. Therefore, for such libraries, the lengthy search strings were broken down into smaller sub-strings and a combination of those strings was constructed using Boolean Operators (AND, OR). The process for building search string is discussed below.

a: DERIVATION OF MAJOR TERMS

In this step, key terms or major terms or keywords from the research questions and terms related to the research area were extracted.

b: IDENTIFICATION OF ALTERNATIVE SPELLING AND SYNONYMS

The synonyms of the extracted words and related terminologies to these major terms were identified.

c: VERIFICATION OF KEYWORDS

The major terms were validated relevant literature.

d: USE OF BOOLEAN OPERATORS FOR CONJUNCTION

To combine the major terms and their synonyms related to the research area, Boolean operators (AND/ OR) were used. The AND operator is used to combine Major terms whereas the OR operator is used to combine the synonyms of the keywords related to each other.

The lengthy search string thus derived is given below:

(“Agile Method” OR “Agile Software Development” OR “Agile Method” OR “Agile Development”) AND (“Large Development Team” OR “Large Scale Development Team” OR “Large Development Team”) AND (Disincentives or “de-motivators” OR “Risk Factor” OR “Negatively affects” OR “barriers” OR “challenges” OR “critical risk factors”)

As mentioned above, some of the digital libraries do not accepts lengthy strings, therefore the lengthy string was broken down into smaller sub-strings given below:

String 1: (“Agile Method” OR “Agile Development”) AND (“Large Development Team” OR “Large Scale Development Team” OR “Large Development Team”) AND (“Disincentives” OR “de-motivators”)

String 2: ((“Metadata”: “Agile Method” OR “Agile Software Development” OR “Agile Development”) AND (“Large Development Team” OR “Large Scale Development Team” OR “Large Development Team”) AND (“Disincentives” OR “de-motivators” OR “Risk Factor”))

String 3: (“Agile Method” OR “Agile Software Development”) AND (“Large Scale Development Team” OR “Large Development Team”) AND (“De-motivators” OR “Risk Factor”)

6) PUBLICATION SELECTION

a: INCLUSION CRITERIA AND EXCLUSION CRITERIA

Table 1 illustrates the inclusion and exclusion criteria.

7) SELECTING PRIMARY STUDIES

The Five-Phase approach called the Tollgate approach [26] was used for the primary selection and refinement of the most relevant articles. The tollgate approach is described in Figure 2 and its steps are given below:

Phase 1: Finding relevant articles based on the search terms.

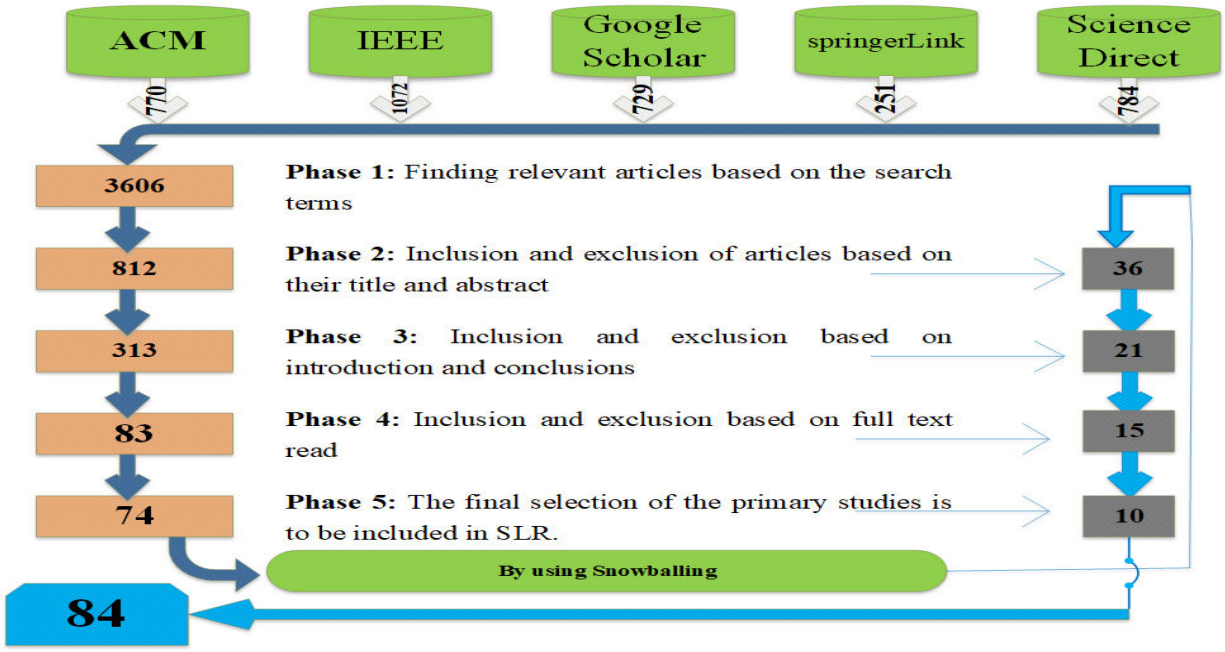

FIGURE 2. Tollgate approach for paper selection.

TABLE 1. Inclusion / exclusion criteria.

Inclusion Criteria	Exclusion Criteria
Articles and papers that deal with the De-motivators Risk factors or Challenges in adopting ASDM at large-scale development teams from the management point of view	Articles that do not explicitly discuss ASDM in the context of the De-motivators and risk factors
Articles that are available in full text and are available in English only	Articles that do not obey the inclusion criteria

Phase 2: Inclusion and exclusion of articles based on their title and abstract.

Phase 3: Inclusion and exclusion based on introduction and conclusions.

Phase 4: Inclusion and exclusion based on full text read

Phase 5: Final selection of the primary study is to be included in SLR.

In the first phase, a primary study search was conducted at various digital libraries and 3606 results were obtained. In the second phase, the inclusion and exclusion criteria was applied based on the titles and abstracts of the obtained papers; as a result of which, 812 papers were extracted. In the third stage, the introduction and conclusion sections of the extracted articles were read and using the tollgate approach, more articles were filtered thereby reducing the number of selected studies to 313 based on the inclusion and exclusion criteria. In the fourth phase, full contents of the articles selected in phase-3 were read as a result of which a total of 83 studies were chosen for the last phase. In the last phase duplicated papers were removed, and which reduced the total number of articles to 74.

8) SNOWBALLING PROCESS

We have considered the cited references in the paper and the references in which a selected paper is cited by performing the backward and forward snowballing technique. A total of 36 papers were extracted using the snowballing technique. As shown in Figure 2, in the final selection process, 10 papers were considered from snowballing technique by applying the tollgate technique. Finally, 84 studies were selected for the data extraction process illustrated in Figure 2. The studies selected by the snowballing process are labeled as "SS" to indicate their use in the paper.

9) PUBLICATION QUALITY ASSESSMENT

Results obtained from the final selection were checked for the quality assessment test and 74 papers were selected according to quality standards. Applying similar steps, we extracted 10 more papers using the snowballing process. The following questions are included in the quality assessment checklist.

- Is it clear in what way the solution or practices for the De-motivators in the adoption of ASDM at a large scale were identified?
- Is adequate data available to support the results?
- Is the researcher giving the impression to report optimistic results larger than adverse results?
- Are the objectives of the research clearly defined?
- Are the outcomes of the research connected to the objective of the research?
- Has the ASDM context been discussed clearly?

For every paper, these answer to these questions were recorded as YES, NO, or NA (no applicable) and then the papers were graded based on the responses to these questions. Every question was worth one point and a minimum score of

50% was required to pass the quality assessment. The compulsory question must be answered in the data extraction form to discover the practices of De-motivators. If the compulsory question is not responded to and the other questions are correctly answered, the publication will still fail the quality assessment. All 84 articles that were chosen received the minimum score, and the data was extracted properly.

10) DATA EXTRACTION

The data was extracted from published articles that include review date, article title, and authors of the article, references, libraries, practices, research location, country, target population, and methodology of the article (survey or reports, case study, interview, SLR). No disagreements were found after the quality assessment test after data extraction.

11) DATA ANALYSIS

Frequency analysis methods were used on the gathered data. The determined practices of De-motivators in agile at large-scale development were counted manually and the percentage of each practice was calculated from their corresponding frequencies. A table of practices is given in Appendix B. The relative importance of a practice was found by its comparison with the other reaming agile practices. Figure 4 describes the frequencies of the methodologies used in the selected articles. Furthermore, the analysis represents the detail of the final selected articles as well as their research methods.

Continent-wise categorization was carried out so as to analyze the agile practices at large-scale teams in each continent. Therefore, the data were extracted based on the country author or location of the case study data. Figure 4 describes the frequencies of the methodologies used in the selected articles. Figure 3 describes the frequencies of the selected articles continent-wise. The final selected papers were classified based on the continent. The final selected papers were categorized based on research methodologies: survey, case study, SLR, etc. Search strategies were used to research related articles and data was extracted from the most relevant articles.

12) CLASSIFICATION OF PRACTICES

For the successful adoption of agile for large-scale development teams, we developed a list of the practices of De-motivators through SLR. Further, we identified critical practices in the list of practices. The criteria for categorization of a practice as a critical De-motivator practice was that its frequency percentage should not be less than a certain threshold [27]. We considered different threshold values such as 10%, 15%, and 20% sometimes 25%. This threshold acts as a filter [27] keeping only those practices whose frequencies satisfy the threshold. In this regard, we used various thresholds to present the significant practices to the project manager to easily adopt agile in large-scale development teams.

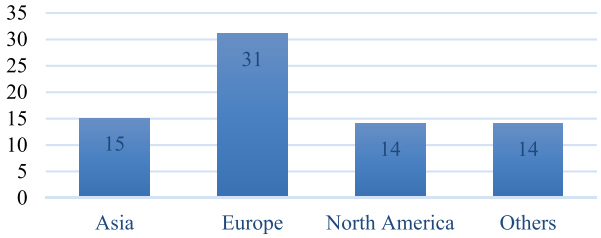

FIGURE 3. Final selected publications Continent wise.

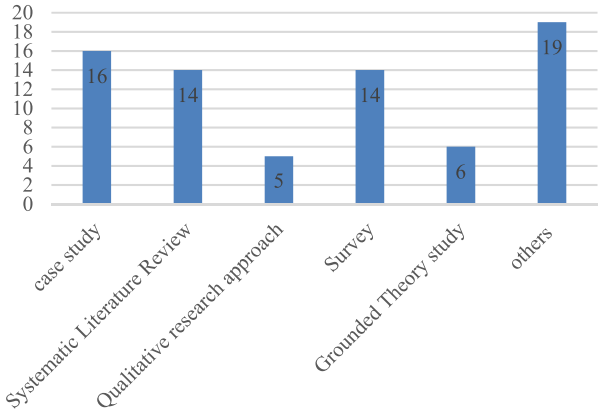

FIGURE 4. Research Methodologies used in selected publications.

B. ANALYTICAL HIERARCHICAL PROCESSING

Analytical hierarchical processing (AHP) is a well-known method used for multi-criteria decisions developed by Saaty [68]. To answer RQ2, we used AHP. A total of 15 de-motivator factors were identified through SLR in our previous study [1], and in the current study AHP was used to prioritize and rank those de-motivators for scaling agile methodologies in large-scale projects. The AHP method was used to prioritize de-motivators and their categories based on their relative importance. The de-motivators and their categories provide a strong background to the software project manager to adopt agile at large-scale projects. The aim of using AHP in this study to rank the most critical De-motivator factors from project manager perspectives. It will help the project managers to focus on the most critical factors while adopting agile methods at large scale.

The AHP method involves seven steps. These seven steps are illustrated in figure 5.

Step 1: Decompose the problem into its hierarchical structure. In this step we identify goals, and categories and rank the challenging factors.

Step 2: Construct a pairwise matrix of sub-factors to find the priority weight/vector of de-motivator factors. A pairwise matrix was constructed with the help of experts including three participants. Priority weight of each category was determined using a nine-point intensity scale, as shown in Table 18, and then using Equations (1) and (2) the consistency of the pairwise matrix was checked: Level 1 defines goals to prioritize the attributes. In Level 2 makes categories of

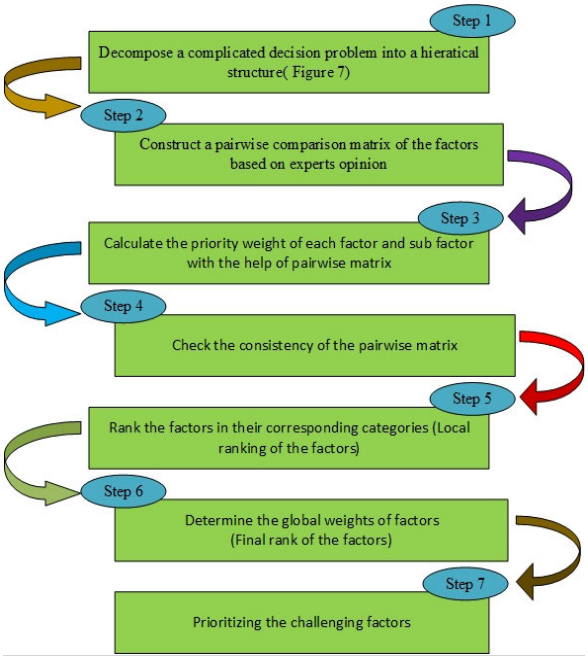

FIGURE 5. AHP Stages.

our attributes. In Level 3 sums all the attributes into their respective category. Figure 6 illustrates these levels in detail.

Step 3: Calculate the Priority Weight of Each de-motivator. To calculate the motivators' priority weights, the categories, and the de-motivators pairwise comparison was performed [40]. Based on their relative significance and the criteria specified, de-motivators are compared at every level. The pairwise comparison matrices are used to calculate the priority weight as follows:

(i) Matrix: a pairwise comparison matrix of the de-motivators.

(ii) Normalizing the matrix: divide each value in each column by the sum of that column.

(iii) Priority weight: calculate an average of each row of a matrix for normalization.

Step 4: Perform consistency check. To calculate pairwise matrix consistency, consistency ratio (CR) and consistency index (CI) are used in AHP [69], [70], [71]. We use Equations (1) and (2) for consistency check.

$$\mathrm{CI} = (\lambda max - n)/(n-1) \tag{1}$$

Where CI represents the consistency index, λ max is the eigenvalue of the matrix, and n represents the size of the matrix or the number of de-motivators factors in the matrix. We find the consistency ratio by using Equation 2.

$$CR = CI/RI2 \tag{2}$$

Where CR represents the consistency ratio, CI is used for the consistency index and RI is the random consistency index illustrated in Table 19 which has constant values.

Priority factors are only acceptable when the consistency ratio value is less than 0.1, whereas consistency ratio values up to 0.1 are acceptable. To enhance the consistency of the pairwise table, the procedure is repeated if the CR values are not within the recommended range.

Step 5: Calculate the Local Weight (LW) of attributes. The local weight of attributes is the priority weight assigned to each attribute inside its respective category.

Step 6: Calculate the Global Weight (GW) attributes. The value of the local weight inside each category multiplied by the value of the local weight of the corresponding category produces the value of the global weight of each attribute.

Step 7: Identify and create the overall priority ranking. The final list of attributes, based on each attribute global weight, is created in this step. Attributes are considered as highly ranked if they have a greater global weight value across all categories. The steps of AHP are illustrated in figure 5.

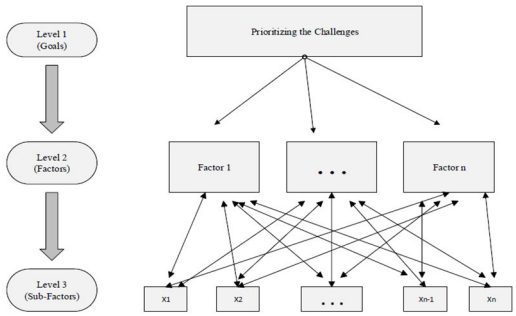

FIGURE 6. AHP Levels.

V. RESULTS AND DISCUSSION

A. SLR RESULTS

This section illustrates the results of the research obtained using SLR in the form of tables, charts, graphs, and bullet points. Table 2 illustrates a list of De-motivators along with their frequencies of the identified practice and the Paper ID from which practices were extracted. The percentage was calculated according to the frequencies of identified practices of each factor out of total selected studies. The list of the final selected article is given in appendix A. Table 3-17 describes each of the identified practices.

We identified 76 practices through SLR for the solution of the De-motivators which can enable a project manager to adopt agile for large scale development teams. Moreover, these practices will also be beneficial to the software industry. Passing the threshold for practices will have a percentage greater than 10%, 15%, and 20% while in some cases greater than 25% and 30%. Our research will help the project manager to remove barriers in the adoption of ASDM on a large scale.

1) LACK OF EFFECTIVE COMMUNICATION

Instead of formal and comprehensive documentation, agile methods team emphasizes members' abilities and informal communication [1]. As the team size increases communication overheads arise. Moreover, geographical distribution

TABLE 2. List of de-motivators with identified practices frequencies and paper id's.

Sr.no	De-Motivators / Risk factors	Frequency out of 74	Percentage	Paper ID
1.	Lack of effective communication	42	57%	P1, P2, P3, P4, P5, P6, P8, P12, P13, P14, P15, P16, P20, P23, P24, P25, P26, P27, P28, P29, P30, P33, P31, P32, P35, P36, P37, P38, P48, P49, P51, P52, P53, P54, P56, P58, P59, P62, P63, P64, P69, P74, S4, S13, S14, S19
2.	Reduced productivity due to delay	37	50%	P3, P5, P7, P12, P13, P16, P17, P18, P19, P21, P22, P24, P34, P35, P39, P42, P43, P44, P45, P46, P47, P48, P53, P54, P59, P60, P61, P62, P63 ,P64, P65, P66, P65, P67, P69, P70, P72, S17, S19, S21
3.	Traditional organizational culture	25	34%	P7, P9, P12, P16, P17, P21, P27, P26, P27, P31, P39, P40, P41, P43, P44, P50, P51, P52, P57, P64, P66, P69, P71, P72, P73, S7, S9, S12, S13, S14
4.	Problem of team feedback	20	27%	P7, P13, P21, P24, P26, P28, P33, P39, P43, P44, P46, P47, P49, P53, P57, P58, P60, P63, P70, P7, S1, S4, S14, S19, S21
5.	Lack of management and commitment support	19	26%	P1, P12, P13, P14, P15, P17, P18, P21, P26, P29, P38, P41, P43, P46, P53, P57, P64, P66, P68, S1, S4, S7, S9, S13, S14, S17
6.	Lack of team orientation	19	26%	P5, P7, P9, P12, P15, P21, P27, P28, P30, P34, P35, P36, P46, P47, P50, P54, P56, P61, P65, S14, S21
7.	Problem in requirement elicitation	14	19%	P3, P5, P8, P9, P17, P19, P20, P21, P22, P29, P32, P47, P49, P61, S1, S12, S19
8.	Continuous testing and integration	12	16%	P3, P5, P15, P18, P22, P30, P31, P32, P43, P56, P72, P74, S7, S12, S17
9.	Bad customer relationship	11	15%	P13, P15, P21, P28, P31, P42, P43, P51, P56, P57, P70
10.	Exhaustive Pair Programming	11	15%	P8, P29, P30, P31, P43, P46, P51, P52, P56, P58, P60 S7, S9, S12, S13, S14
11.	Lack of team training	10	14%	P2, P9, P21, P24, P26, P27, P15, P44, P53, P66, S4, S7, S14, S21
12.	Lack of Customer Presence	09	12%	P2, P3, P7, P19, P26, P31, P35, P47, P50, S4, S12, S19
13.	Lack of customer knowledge	08	11%	P8, P17, P31, P42, P46, P52, P66, P73
14.	Reluctance/ reluctant to adopt	07	9%	P9, P12, P13, P27, P42, P44, P69, S7, S13
15.	Lack of agile experts	06	8%	P9, P12, P23, P44, P59, P66, S4

TABLE 3. Practices of lack of effective communication identified through SLR.

Sr.no	Lack of effective communication	Percentage
1.	Frequent meetings on daily biases enhance communication among team members and improve collective problem-solving. Who has the right information should be communicated.	31%
2.	Agile collaboration tools enhance the exchange of information among team members and promote communication as well for the project as a sharing vision. For effective communication, appropriate communication tools are important for team members	27%
3.	For better communication, the teams should be divided into sub-teams. Communication will be successful but consistency and redundancy of data may occur that there is another practice team members rotation	16%
4.	Team members should be collocated to promote frequent interactions between team members for effective communication.	15%
5.	To reduce challenges in larger-scale distributed agile software development various communication channels are available. I.e. telephones, video conferencing calls, etc.	20%
6.	Face-to-face communication is possible in collocated agile software development with team members to exchange knowledge.	18%
7.	Improve large-scale agile development team collaboration by maintaining valuable documentation. Documentation helps in communicating requirements changes to project team members or customers at a large scale at distributed locations. Teamwork enables using performance models enabling coordination, collaboration, and communication among agile software development teams.	14%
8.	The project manager should ensure that all obtained information were organized in an effective way for the team. Changes in requirements were always organized and regular feedback was provided on how the team is performing. The information that should be communicated to the entire team should be presented in a continuous manner	8%
9.	Good relationships between team members can also reduce language barriers. An effective relationship between developer and customer can enhance communication and Customer collaboration	18%
10.	Informal communication helps to develop a strong team	9%
11.	The project backlog is shared between distributed development teams to share project requirements and knowledge.	5%
12.	Sharing a common and open working space for all team members can make communication effortless and quick.	15%
13.	Give training to a team members to improve their communication skills. For language barriers set English as an official language and give training to a team members to overcome language problems	20%

impacts communication negatively [30]. The main goal of agile methods is to overcome communication gaps and reduce documentation overheads [40]. Several studies suggest ensuring regular agile meetings with the customer, encouraging periodic and face-to-face meetings, making use of agile practices such as information hub and product owner,

also making use of different available communication tools such as phone, instant messing, and asynchronous tools like email, if agile teams are globally distributed [30], [32], [44], [48], [64]. Agile methods suggest dividing large teams into smaller teams to reduce communication overheads. Video conferencing tools are alternative for face-to-face communication for globally distributed large agile teams [28], [44], [62], [63], [64]. The selection of appropriate communication media can help minimize ambiguity in communication. To reduce ambiguities in communication and to successfully implement agile, every organization should apply a combination of communication mediums [41]. Research suggested that centralization in decision-making has vital importance in large-scale projects. Centralization in decision-making structure has a positive impact on project performance in large agile projects. A centralized decision-making unit is responsible for coordinating key decisions among teams as well as resolving disagreements and inconsistencies [35]. Frequent talking and face-to-face communication can solve conflicts and problems [35]. Informal communication can also overcome communication problems. Informal communication includes face-to-face conversation, casual communication through video-conferencing or telephone, written communication through email, online chat, and short message service (SMS) [35], [41]. Table 3 demonstrates the list of practices related to the lack of effective communication in agile software development in large-scale teams from the management point of view. We have identified thirteen practices of lack of effective communication as shown in Table 3. In the practices, the following nine practices with percentage greater than 15 are considered critical practice.

2) REDUCED PRODUCTIVITY DUE TO DELAY

According to the conclusion of Abrar et al. [1], reduced productivity due to delay is also a critical de-motivator factor in transforming to agile methods at a large scale from the management point of view. Project delays can be caused by poor technological infrastructure [48]. Ignoring teams or their proper management them has a high impact on team effectiveness and productivity. Lack of participation and low engagement of stakeholders has been identified one of the obstacles to productivity [45]. Teamwork productivity can be influenced by the cultural differences between offshore organizations [55]. Adopting agile methods enable productivity gains at a large scale, enhances product quality, and decreases time to market [29], [46], [47], [49]. Productivity is related to the effectiveness and efficiency of teamwork [51], [55]. Communication and coordination can also impact productivity [42], [50], [51]. It is concluded in [51] that agile teams affect productivity positively while external factors and factors depending on the organization more than half affect productivity negatively. Professional training also enhances productivity. Agile accepts change at any step and thus enhances productivity. Reduced documentation, active contact with the customer, and continuous involvement of stakeholder impacts productivity positively. Keeping the code simple, technically advanced developers, and lessening documentation enhance productivity [36], [53], [54]. Pair programming has a positive impact on productivity and quality [52]. New team members may bring new ideas, solutions, and energy to make improvements and establish agile practices. Proper task assignment, capabilities, and skills of teams are the most important factors in team productivity. Experienced and flexible people contribute more to productivity [51]. Table 4 demonstrates the list of practices related to reduced productivity due to delays in agile software development at large-scale teams from the management point of view. We have identified eleven practices of reduced productivity due to delay as shown in Table 4. In the following eight practices, a practice is considered critical if its percentage score greater than 10.

3) TRADITIONAL ORGANIZATIONAL CULTURE

Abrar et al. [1] identified traditional organizational culture as a critical De-Motivation in the implementation of ASDM in large development teams from the management point of view. Organizational culture should be flexible and supportive to adopt agile. Sometimes it is essential to change the entire culture of the organization to adopt agile methods. Organizational culture is a critical factor when adopting agile methods. So, it is important to give more attention to organizational culture while adopting agile methods [28], [29]. Promote a corporate organizational culture that supports rapid communication, trust between agile stakeholders, and quick customer feedback [1], [30]. Table 5 demonstrates the practices/solutions of the De-motivator, that is, Traditional Organizational Culture in [1]. We have identified five practices for traditional organizational culture as shown in Table 5. In the following five practices, with a percentage score greater than 15 is considered critical practice.

4) PROBLEM OF TEAM FEEDBACK

According to Abrar et al. [1], the problem with team feedback is the eighth-most critical de-motivator factor when adopting agile methods for large-scale development teams. Since teams are large and geographically distributed, it is difficult for management teams to get timely feedback. Khan et al. [27] have suggested that team management should meet with team member to encourage them to share their feedback and provide which approaches are working and which are not. To achieve high moral and positive feedback, coaching should be provided to the motivated and performing team members. Dialogue of feedback should be created with every team member to inform and encourage each other [27]. Early feedback about development dramatically increases motivation among team members. Frequent feedback from development teams and management provides a better understanding of the work environment [46]. Participation in knowledge-sharing activities can motivate providing feedback about the project. Presentations among team members help team members to provide project feedback in time [32]. Short iterations of the project improve team feedback as well as customer feedback [47]. Larger teams should be divided into smaller ones

TABLE 4. Practices of reduced productivity due to delay identified through SLR.

Sr.no	Reduced productivity due to delay	Percentage
1.	The strict control of procedure leads to productivity. A motivating workplace provides the needed opportunity it also leads to productivity	11%
2.	Developing software in short iterations can provide delivery of a small part of the project to the customer before the deadline.	14%
3.	The management should act as a facilitator to the teams so that they will focus on the delivery of the product	11%
4.	Overcoming communication gaps enhances the productivity	11%
5.	Experienced and flexible peoples contribute more to productivity. The new team member may bring new ideas, solutions, and energy to make improvements and establish agile practices. Professional training is also the most significant factor in productivity.	8%
6.	Pair programming has a positive impact on productivity and best quality	11%
7.	Developers keep code as straightforward, simple, and technically advanced as possible, and lessening documentation can also enhance productivity.	8%
8.	Agile accepts changes at the implementation step and it can also increase productivity	12%
9.	Automation can enhance productivity	4%
10.	Productivity is related to effectiveness and efficiency. And also depends on agile teamwork.	12%
11.	Close collaboration with the customer and customer representatives produces quality software within a period and enhances customer relationship	11%

TABLE 5. Practices of traditional organizational culture identified through SLR.

Sr.no	Traditional Organizational Culture	Percentage
1.	The team behavior and organizational culture should be aligned to agile values. Organizational culture should be flexible to adopt agile methods.	20%
2.	Reduce the organizational complexity to adopt agile at a large scale. Adopting the Agile method is difficult due to resistance to change. Effective approaches should be used to make changes in the organization	24%
3.	Support and involvement from management on regular basis have a vital impact on the adaptation of agile methods	20%
4.	An organization should provide an environment that facilitates communication among customers and team members.	16%
5.	These resources should be utilized in team training and team member memberships to prepare them to use ASDM.	10%

because small teams enable effective communication and early feedback [33]. Personal communication can improve the morale of the team which includes feedback and Pair Programming. To ensure effective feedback, personal communication among customers and team members was highlighted as crucial [40]. Table 6 provides a list of practices related to Problem of team feedback in agile software development at large-scale teams from the management point of view. We have identified three practices for the problem of team feedback as shown in Table 6. Practices with a percentage greater than 10 are considered critical practices.

5) LACK OF MANAGEMENT AND COMMITMENT SUPPORT

According to Abrar et al. [1], management and commitment support is the 3rd most critical risk factor in adopting agile methods in large-scale development teams. Owing to its significant impact, it is highly important that the management supports change adaptation. Therefore, Top-level management should indicate their participation clearly to adopt agile software development. Top management and Cooperation from every team member were important [31], [45]. Timely support from management leads to successful agile adaptation. To successfully adopt agile, management should provide 100% support and should understand the value of Agile. Management support in Agile has a significant role in terms of facilitating knowledge-sharing, trust-building, and improving communication [32], [33]. Table 7 lists the practices of management and commitment support with their frequency percentages. We have identified four practices for lack of management and commitment support as shown in table 7. Practices with a percentage greater than 10 are considered critical practice.

6) LACK OF TEAM ORIENTATION

The responsibilities assigned to the team and the attitudes that team members have toward one another are referred to as team orientation [66]. It indicates a degree of acceptance of team rules, team cohesiveness, and the significance of team collaboration, *e.g.* assigning a high priority to the team goals and voluntarily collaborating in all areas of the team [65]. Inclusion or exclusion of new members to the team may affect the orientation of the team that has already been established [1]. It is stated that high team orientation improves overall team performance in self-managing teams, encourages team members to collaborate more, and improves individual actions. To achieve a better and more consistent understanding of agile, open events and training are best practices to deliver the same message to all team members [29]. Interpersonal attraction and mutual support of team members can overcome team orientation problems [67]. Knowledge-sharing sessions provide opportunities for team members to learn new ideas from each other [40], [54], [58]. Cross-training is an instructional strategy in which each team member is trained in the duties of his or her teammates [58]. High team orientation enhances team understanding of the goal, information sharing, and task involvement. Table 8

TABLE 6. Practices of problem of team feedback identified through SLR.

Sr.no	Problem of team feedback	Percentage
1.	Customer involvement on regular bases impacts the feedback of the development team. Small teams enable effective communication and fast feedback	16%
2.	The teams were instructed to change their work practice so that it involves a short feedback cycle. Short iterations improve team feedback as well as customers feedback	15%
3.	The demonstration is the best tool to get feedback	5%

TABLE 7. Practices of lack of management and commitment support identified through SLR.

Sr.no	Lack of management and commitment support	Percentage
1.	Top-level management's participation in Agile Coaching sessions was a successful strategy for increasing their knowledge and demonstrating their commitment to the Agile transformation	15%
2.	Domain knowledge and technology aspects should be balanced by teams to avoid ambiguities and organizational processes, management practices requirements, and design confusion.	12%
3.	Management support in Agile has a significant role in terms of facilitating knowledge sharing, building trust, and improving communication.	16%
4.	Another study concluded that engineers were motivated by agile principles and project managers need motivation and training in agile methods	12%

TABLE 8. Practices of lack of team orientation identified through SLR.

Sr.no	Lack of team orientation	Percentage
1.	By widening the knowledge of the other components to reduce team orientation. Team members should work in pairs. Pair programing facilitates learning.	15%
2.	Inclusion or exclusion of new members to the team may cause the orientation of the team already been established.	7%
3.	Delivering presentations team members share their experiences and expertise can lead to removing team orientation problems. Mutual support of team members can overcome team orientation	12%
4.	Team support each other when they communicate frequently and share knowledge	9%
5.	Cross-training is an instructional strategy in which each team member is trained in the duties of his or her teammates	12%

demonstrates the list of practices related to the lack of team orientation in agile software development at large-scale teams from the management point of view. We have identified five practices of lack of team orientation as shown in Table 8. Among these practices, three practices with a percentage greater than 10 are considered critical practices.

7) PROBLEM IN REQUIREMENT ELICITATION

Another critical De-motivator factor identified in the previous study is the Problem in requirement elicitation while transforming to agile methods at large-scale development [1]. Inconsistent scope of definition and ambiguous problem statements results in unstable user requirements and leads to an unstable development environment for application [41]. Inadequate customer involvement causes issues in requirements gathering and clarifying [43]. Therefore, to gather requirements correctly, it is important to depute experienced personnel with adequate skills to interact with the customer for requirements elicitation. Sometimes there are individuals with domain knowledge and technical skills [14]. To respond frequently and quickly to changing and ambiguous requirements, ASD suggests the collocation of development teams. The process of requirements understanding, and feedback incorporation occurs many times until the fulfillment of the user's requirements to the application. The result of the study shows that ASD requires rich communication at the start of the project to elicit requirements in an accurate way [41]. Face-to-face communication between stakeholders and developers can lead to better understanding of requirements and reduced productivity delays [41], [42]. The organization should have the right culture such as supporting rapid communication, dynamicity in requirements changes, trusting people, and obtaining fast feedback from customers [33]. Instant messaging tools provide a chat tool where requirements can be discussed through the exchange of text with customers [44]. Table 9 demonstrates the list of practices related to a problem in requirement elicitation in agile software development at large-scale teams from the management point of view.

8) CONTINUOUS TESTING AND INTEGRATION

The criticality of continuous testing and integration is paramount according to the review literature. The process of continuous testing and integration leads to automation testing [1]. Previous studies concluded a lack of automated testing as well as integration of various parts of the projects [1]. Keshta and Morgan [50] has suggested conducting regular conferences for the integration of various parts of the project on the specific milestones. Rashid and Khan [3] has identified that iterative development and integration help find out possible bugs at early stages and resolve them on time. Rashid and Khan has further discussed that timely unit testing and integration testing can provide rapid feedback to developers [3], [42]. Modifications to the code should be integrated into

TABLE 9. Practices of problem elicitation identified through SLR.

Sr. no	Problem in requirement elicitation	Percentage
1.	Iterative development and retroactive meetings with the client make the team respond to the customer's requirements.	7%
2.	The developer can select the best method for the customer if they did not have a way of working requirements. A good experience of interaction with a customer should be required for elicitation requirements.	8%
3.	Improved communication between stakeholders and developers, instant releases, and enhanced flexibility can lead to enhance better understanding of requirements.	5%
4.	One of the key aspect of ASD is the collocation of the development teams. To react quickly to frequently changing and ambiguous requirements.	7%
	User stories are used to force customers to provide clear requirements with clear business drives	7%
5.	The instant messaging tool was integrated with backlog tool where user can reference requirements in backlog via a direct link	7%

TABLE 10. Practices of continuous testing and integration identified through SLR.

Sr.no	Continuous testing and integration	Percentage
1.	Conduct regular conferences for the integration of various parts of the project on specific milestones. Modifications to the code should be integrated into the code base regularly and examined for integration issues. Frequent check-ins are performed.	11%
2.	Practices for testing an automated test run, test-driven development, refactoring, and unit testing.	9%
3.	Introducing the practice of early and continuous testing because it takes time and effort to introduce this properly	8%

TABLE 11. Practices of bad customer relationships identified through SLR.

Sr.no	Bad customer relationship	Percentage
1.	Customer involvement impacts development team feedback on regular bases.	8%
2.	Frequent communication with the customer also leads to effective communication and good customer relationship	11%
3.	Team members should develop relationships with customers based on trust.	5%
4.	Increasing the ability to meet customer requirements enhance customer relationships	7%

TABLE 12. Practices of exhaustive pair programming identified through SLR.

Sr.no	Exhaustive Pair Programming	Percentage
1.	Using pair programming there is no need for a distributed team environment	4%
2.	Pair programming facilitates learning if partners are exchanged regularly in this way to achieve optimal learning and increase collective code ownership.	11%
3.	Pair programming in GDAD can be achieved using screen sharing and audio calls	7%
4.	The arrangement of the Development team cell in which resources are seated in a circle facilitates that team members can use these resources mutually and can work together. Open workspace allows sharing of resource	7%

the code base regularly and examined for integration issues. Dybå and Dingsøyr [47] have suggested introducing the practice of early and continuous testing and integration because it takes time and effort to introduce this properly. Table 10 demonstrates the list of practices related to continuous testing and integration in agile software development at large-scale teams from the management point of view. Among these, practices, with a percentage score greater than 10, are considered critical practices.

9) BAD CUSTOMER RELATIONSHIP

Bad customer relationship is another critical De-motivator factor identified by Abrar et al. [1]. According to the survey [37] conducted on the eXtreme Programming (XP) project, the continuous presence of the customer in the XP project is most important. Taking high priority on achieving early customer satisfaction and frequent delivery of valuable software is one of the ASD principles. It is important that customers are not only present on-site with software development teams but should also be highly active and motivated and consider themselves liable members of the project [38]. Therefore, customer commitment and relationship are important success factors for agile methods. The authors of [39] shares their experiences from many projects show, in their study, the significance of customer and development relationships as a success factor. Customer involvement, commitment and collaboration has also been emphasized by [33] for the adoption of ASDM. Table 11 provides a list of practices identified through SLR for bad customer relationships. Practices with a percentage score greater than 10 are considered critical.

10) EXHAUSTIVE PAIR PROGRAMMING

Pair programming is a method of software development in which two programmers collaborate on a single computer. Some research suggests that pair programming is not always productive [59]. Pair programming has been considered difficult due to different skills and a large number of pair members [9]. Frequent partner changes are suggested as a way to

achieve optimal learning and increase collective code ownership [47]. Pair programming facilitates learning if partners are exchanged regularly [47], [52]. Table 12 demonstrates practices of exhaustive pair programing in agile software development at large-scale teams from the management point of view. Practices with percentage greater than 10 is considered critical practice.

11) LACK OF TEAM TRAINING

One of the reasons for not repeating the pilot teams' experience when implementing agile is a lack of training [19]. Khan et al. [60] have correctly identified this issue by claiming that the trainers were not properly trained, which raised the possibility of incorrect agile practices being taught. To obtain new knowledge, experts' conferences and training sessions are useful for team members. Team should be enrolled in online training sessions for learning new knowledge [32]. Training provides motivation and improves chances of success and also help team members to become more positively inclined toward the new way of working [29]. A person on a team that gets training, can educate or train other team members. Training sessions on the basics in the business knowledge and company-specific areas are highly useful [61]. Table 13 demonstrates the list of practices related to the lack of team training in agile software development at large-scale teams from the management point of view.

12) LACK OF CUSTOMER PRESENCE

Lack of customer presence is identified as a critical de-motivator factor in SLR [1]. Rashid and Khan [56] identified a lack of customer presence in 62% of their results as a de-motivator factor in software development. Software development takes a long time to complete and has higher chances of failure in the absence of the customer. Regular collaboration of customers with the team allows for a shared goal between them [48]. The most significant practice of agile methodologies is continuing presence of the customer. This decreases the development efforts to build the software within the estimated and defined time [56]. Earlier XP demanded on-site customer practice as a requirement for the customer's full-time presence with the development team in a shared workspace. This practice enables frequent feedback and effective communication, but has been found problematic [57]. The Onsite Customer has been substituted by the Real Customer Involvement method, which proposes that customers can engage in weekly and quarterly meetings but does not demand full-time customer participation. A full-time customer presence is not required by the development team. Customer should be involved in the development process when the contents of the iterations are discussed and analyzed, as well as during the release when the customer can provide feedback on the iterations' results [57]. Customer presence is an important factor because they have knowledge about the business domain and can decide on the scope, schedule, and resources, and can clarify project complexity [28]. Software developers value onsite customers and prefer to be available to onsite customers [28], [58]. Table 14 demonstrates the list of practices related to lack of customer presence to delay in agile software development at large-scale teams from the management point of view. Practices with percentage greater than 10 are considered critical practices.

13) LACK OF CUSTOMER KNOWLEDGE

Lack of customer knowledge is also identified as a critical de-motivator through SLR in a previous study by Abrar et al. [1] and [45]. Top management's lack of knowledge about their employees is another barrier to agile adaptation [45]. Customers' lack of knowledge about agile methods is another challenging task requires convincing them to adopt agile methods [31]. Engaging customers in the agile process is an effective strategy to improve their knowledge and with the passage of time customers are get experience and know-how about agile methods [45]. It has also been noted that a lack of knowledge among team members leads to the Product Owner's intrusion into the team's organization. Product Owners face similar problems with stakeholders or senior management representatives when there is lack of knowledge. An effective strategy to enhance the knowledge of customers to arrange coaching sessions. Arrangement of coaching sessions also demonstrates management's commitment to adopting agile methods [45]. Table 15 demonstrates the list of practices related to the lack of customer knowledge in agile software development at large-scale teams from the management point of view.

14) RELUCTANCE TO ADOPT

Among the 15 De-motivator identified by [1] 'reluctance to adopt' is the sixth most critical in adopting agile software development at a large scale from the management point of view. Project managers find it difficult to adopt agile methods because of fear of change and of them being accustomed to old technologies. It is highly important that the management should be willing for change and adaptation. Top-level management should indicate their participation in changes being made for the adoption of agile software development. Organizations with traditional cultures face issues while adopting agile methods as there is resistance to change. Sometimes employees of the organizations and software development teams resist change because of fear of new technologies. Another problem may arise in productivity due to changes because of the reluctance of the team's members to new technology [5]. Employees get hesitant because of the increase in new roles and responsibilities [14]. Adaptation of agile methods in practice and experience in applying these methods will have a positive effect on agile adopters' resistance [36]. Another de-motivator factor of their resistance is working in groups instead of individual offices [21]. To adopt agile software development at large scale organizations [14] suggests customizing the agile approaches. The research study finds out practices for these factors. Table 16 demonstrates the list of practices related to Reluctance to adopt agile at a large scale from the management point of view.

TABLE 13. Practices of lack of team training identified through SLR.

Sr.no	Lack of team training	Percentage
1.	The agile team should be trained on formal specifications of components interfaces for the purpose to enhance the communication among the team members when they are separated from developers.	4%
2.	To remove the stress from the team freedom should be given to the team and the organization in findings between autonomy and control.	5%
3.	Training sessions should be arranged on the basics knowledge of Agile.	4%
4.	The training in agile practice should give to development teams because of the implementation of agile practices at the stage of development teams.	3%
5.	To organize overall strategy, the Agile team must be continuously guided by strategic inputs to take optimum decisions	7%
6.	One person on the team who got training will train other team members.	3%

TABLE 14. Practices of lack of customer presence identified through SLR.

Sr.no	Lack of customer presence	Percentage
1.	Software developers value onsite customers and prefer to be available to onsite customers. The Regular collaboration of customers with the team allows for a shared goal between them	11%
2.	The incorrect arrangements could provide miscommunication and conflict among stakeholders that can result in ineffective customer presence	4%
3.	To uncover problems facing customer presence electronic collaboration is the best factor for large customers. Onsite customer is also agile best practice.	7%

TABLE 15. Practices of lack of customer knowledge identified through SLR.

Sr. no	Lack of customer knowledge	Percentage
1.	Close customer collaboration can lead to closely aligned end users so that they can freely provide valuable knowledge	10%
2.	Knowledge can be transferred in a better way due to frequent feedback at each iteration and good communication	5%

TABLE 16. Practices of reluctance to adopt identified through SLR.

Sr. no	Reluctance to adopt	Percentage
1.	Adaptation of agile methods in practice and experience in applying these methods will have a positive effect on agile adopters' resistance.	4%
2.	The change should be perceived as easy enough and understandable to the team with good reasons to adopt it.	5%
3.	Management is unwilling to adopt change because of a lack of understanding and lack of involvement in agile methods.	4%

TABLE 17. Practices of lack of agile experts identified through SLR.

Sr.no	Lack of agile experts	Percentage
1.	Training and coaching sessions to education personal to make them agile experts	8%
2.	Provide training on agile methods and coach teams as they learn by doing	4%
3.	Training in the methodology can produce experts in the methodology	5%

15) LACK OF AGILE EXPERTS

According to Abrar et al. [1], the most critical De-motivators is the Lack of agile experts when ASDM is adopted at large scale from the management perspective. To adopt agile methods the management and teams expert must have optimum knowledge of the emerging technology. Proper training requires getting the best expert in agile software development. If there is not enough expertise with human resources in agile software development, it may lead to serious problems. The reluctance of the management to provide the funds to their human resources for organizing workshops and trainings inhibits the development expertise thereby making the adoption of the agile methods difficult [34]. Moreover, conducting proper training is tricky because the real work environment significantly different from the training environment [13]. Management should provide support and on-the-job training to deal with development team's uneasiness with ASD, particularly those who lack previous ASD experience. Training and coaching sessions educate personnel and with sufficient trainings they can become agile experts [1], [28], [29], [35]. Table 17 describes the practices with their frequency percentage for the lack of agile experts.

B. ANALYTICAL HIERARCHICAL PROCESS

The Analytical Hierarchical Process (AHP) is used for ranking of the demotivators factors. The ranking of the demotivators helps project managers to scale agile development methods at large scale developments teams. All the demotivator's priority weights relative to their categories are calculated and listed in this stage.

TABLE 18. Point of scale table.

Terms	Importance
Equally important	1
Moderately important	3
Strongly more important	5
Very strongly more important	7
Extremely more important	9
Intermediate values	2, 4, 6, 8

TABLE 19. Size of matrix table.

Size of matrix	1	2	3	4	5	6	7	8	9	10
RI	0	0	0.58	0.9	1.12	1.32	1.41	1.45	1.45	1.49

TABLE 20. List of de-motivators factors.

Sr.no	De-Motivators
DM1	Traditional organizational culture
DM2	Lack of management and commitment to support
DM3	Lack of agile experts
DM4	Reluctance to adopt
DM5	Bad customer relationship
DM6	Problem in requirement elicitation
DM7	Lack of customer knowledge
DM8	Problem of team feedback
DM9	Reduced productivity due to delay
DM10	Lack of Customer Presence
DM11	Exhaustive Pair Programming
DM12	Lack of team training
DM13	Lack of effective communication
DM14	Lack of team orientation
DM15	Continuous testing and integration

Our approach: As discussed in the section IV-B of the proposed methodology, we proceed using the below steps:

Step 1: We defined our goals, categories, and de-motivators that encourage agile adoption from a management perspective in large-scale agile development shown in Table 20.

Step 2: We decomposed the problem into a hierarchal structure as in following levels:

In Level 1, we defined our goals to prioritize the de-motivator. In Level 2, we made categories of our attributes/ de-motivators. In Level 3, we summed all the attributes into their respective category. Figure 6 illustrates these levels in detail. Figure 7 sums up these levels according to current context of the De-motivators.

Step 3: In this step we made pairwise matrices using a 9-point scale intensity table 18. An online questionnaire survey was conducted through which we have collected responses from 54 respondents. Data of 54 respondents were analyzed using geometric mean of Demotivators of the respective pairwise matrix, which were the input to each cell of the pairwise matrix.

Steps 4 to step 5 were performed on the pairwise matrices accordingly. We calculated the consistency index and consistency ratio of each pairwise matrix of each category. Tables 21-30 illustrate the pairwise matrices as well as their normalized pairwise matrices.

Steps 6 and 7 were performed using normalized pairwise matrices. We calculated local rankings and global rankings of the Demotivators and ranked Demotivators from highest priority to lowest priority as shown in Table 32.

The priority vector/weight shows the local weight of each de-motivator after analysis, while the total shows the sum of values of the respective column.

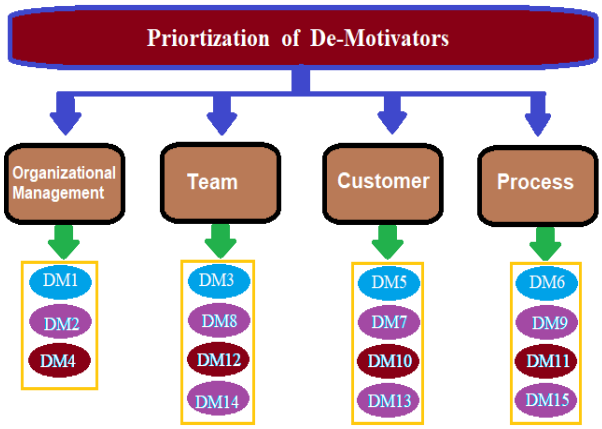

FIGURE 7. Mapping and categorization of De-motivators.

TABLE 21. Pair wise matrix for the category of "organization management".

S.no	DM1	DM2	DM4
DM1	1.00	3.00	0.20
DM2	0.33	1.00	0.14
DM4	5.00	7.00	1.00
Total	6.33	11.00	1.34

TABLE 22. Normalized matrix for the category of "organization management".

S.no	DM1	DM2	DM4	Priority weight
DM1	0.16	0.27	0.15	0.19
DM2	0.05	0.09	0.11	0.08
DM4	0.79	0.64	0.74	0.72

λ Max= 3.11,CI = 0.06, CR = 0.096< <0.1(Consistency ok).

Table 31 illustrates the final prioritized list of de-motivators based on their local and global weights. The final prioritization is based on the global weight of each de-motivator. De-motivators having a higher global value in all categories are classified as a top priority.

Table 32 illustrates the results of the prioritization of the Demotivators. DM11 is categorized as a high-priority risk factor while DM14 is categorized as the least priority risk

TABLE 23. Pairwise matrix for the category of "team.".

S.no	DM3	DM8	DM12	DM14
DM3	1.00	5.00	3.00	7.00
DM8	0.20	1.00	0.20	3.00
DM12	0.33	5.00	1.00	0.20
Dm14	0.14	0.33	0.20	1.00
Total	1.68	11.33	4.40	11.20

TABLE 24. Normalized pairwise matrix for the category of "team".

S.no	DM3	DM8	DM12	DM14	Priority weight
DM3	0.60	0.44	0.68	0.63	0.59
DM8	0.12	0.09	0.05	0.27	0.13
DM12	0.20	0.44	0.23	0.02	0.22
Dm14	0.09	0.03	0.05	0.09	0.06

λ Max =4.130, CI = 0.043, CR= 0.048<0.1(consistency ok)

TABLE 25. Pairwise matrix for category of "customer".

S.no	DM5	DM7	DM10	DM13
DM5	1.00	0.20	0.33	5.00
DM7	5.00	1.00	2.00	7.00
DM10	3.00	0.50	1.00	7.00
DM13	0.20	0.14	0.14	1.00
Total	9.20	1.84	3.48	20.00

TABLE 26. Normalized pairwise matrix for the category of "customer".

S.no	DM5	DM7	DM10	DM13	Priority weight
DM5	0.11	0.11	0.10	0.25	0.14
DM7	0.54	0.57	0.58	0.35	0.50
DM10	0.33	0.27	0.29	0.35	0.31
DM13	0.02	0.08	0.04	0.05	0.05

λ Max=4.247, CI= 0.082, CR =0.091 <0.1(consistency ok)

TABLE 27. Pairwise matrix for the category of "process.".

S.no	DM6	DM9	DM11	DM15
DM6	1.00	3.00	0.33	0.33
DM9	0.33	1.00	0.14	0.20
DM11	3.00	7.00	1.00	3.00
DM15	3.00	5.00	0.33	1.00
Total	7.33	16.00	1.81	4.53

factor for the adoption of agile software development at a large scale from the management perspective.

VI. STUDY LIMITATIONS

This section describes the validity threats related to SLR. Using the SLR approach, we identified solutions/practices for each De-motivator while adopting agile methods at a large scale from the management point of view, but how valid are our findings? The first author collected most of the data during the SLR procedure steps. However, we have attempted to mitigate this risk by observing any ambiguous topics that were discussed in a group setting, but there is a greater threat that a single researcher may be biased and extract incorrect data regularly. Nevertheless, to validate the results with the interrelated test the authors made their best efforts, *i.e.,* every author checked the results independently and concluded which practices should be included and which excluded. Although, checking each result by every author was time-consuming but this exercise enhanced accuracy of our results. The SLR's scope has been limited to the five digital databases, namely IEEE, SpringerLink, ScienceDirect, Google Scholar, and ACM library; and is also limited to search strings. It may have resulted in the omission of a large body of related literature about agile development at a large scale from the management perspective. However, we substantiated our search strategy with prior SLR studies [1], [6], [27], [29] and continued to monitor our search technique throughout all SLR studies. Finally, the authors concluded that the databases and search strings they selected were sufficient for locating contemporary research on success criteria for identifying the practice of De-motivator factors for agile development at a large scale from the management point of view.

TABLE 28. Normalized pairwise matrix for the category of "process".

S.no	DM6	DM9	DM11	DM15	Priority weight
DM6	0.14	0.19	0.18	0.07	0.15
DM9	0.05	0.06	0.08	0.04	0.06
DM11	0.41	0.44	0.55	0.66	0.52
DM15	0.41	0.31	0.18	0.22	0.28

Λ Max=4.199, CI =0.066, CR =0.074 <0.1 (consistency ok).

TABLE 29. Pairwise matrix for overall categories.

S.no	Organizational Management	Team	Customer	Process
Organizational Management	1.00	3.00	0.33	0.14
Team	0.33	1.00	0.20	0.14
Customer	3.00	5.00	1.00	0.33
Process	7.00	7.00	3.00	1.00
Total	11.33	16.00	4.53	1.62

TABLE 30. Normalized pairwise matrix for overall categories.

S.no	Organizational Management	Team	Customer	Process	Priority weight
Organizational Management	0.09	0.19	0.07	0.09	0.11
Team	0.03	0.06	0.04	0.09	0.06
Customer	0.26	0.31	0.22	0.21	0.25
Process	0.62	0.44	0.66	0.62	0.58

λ Max = 4.219 , CI = 0.073 , CR = 0.081<0.1(consistency ok).

TABLE 31. Final list of prioritized de-motivators.

Categories	Categories Weight	De-Motivators	Local Weights	Local Ranking	Global Weights	Final Priority
Organizational Management	0.11	DM1	0.19	2	0.0203	10
		DM2	0.08	3	0.0087	13
		DM4	0.72	1	0.0760	5
Team	0.06	DM3	0.59	1	0.0352	8
		DM8	0.13	3	0.0078	14
		DM12	0.22	2	0.0133	11
		DM14	0.06	4	0.0037	15
Customer	0.25	DM5	0.14	3	0.0385	7
		DM7	0.50	1	0.1263	3
		DM10	0.31	2	0.0725	6
		DM13	0.05	4	0.0093	12
Process	0.58	DM6	0.15	3	0.0974	4
		DM9	0.06	4	0.0296	9
		DM11	0.52	1	0.2929	1
		DM15	0.28	2	0.1589	2

TABLE 32. Rankings of demotivators.

Sr. No	De-Motivators	Final Priority
DM11	Exhaustive Pair Programming	1
DM15	Continuous testing and integration	2
DM7	Lack of customer knowledge	3
DM6	Problem in requirement elicitation	4
DM4	Reluctance to adopt	5
DM10	Lack of Customer Presence	6
DM5	Bad customer relationship	7
DM3	Lack of agile experts	8
DM9	Reduced productivity due to delay	9
DM1	Traditional organizational culture	10
DM12	Lack of team training	11
DM13	Lack of effective communication	12
DM2	Lack of management and commitment to support	13
DM8	Problem of team feedback	14
DM14	Lack of team orientation	15

VII. THREATS AND VALIDITY

The limitations of the study have been discussed in the previous section, considering the use of SLR validity paradigms, as recommended by previous studies [69]. We now explain the credibility, transferability, dependability, and conformability for the SLR.

A. CREDIBILITY

We conducted the SLR study in the 2021 in which we identified 76 practices for the De-motivators factors as identified in [1]. Practices identified in the study will help project manager at large scale agile management teams.

B. TRANSFERABILITY

We thoroughly studied the field of the study in literature review. Extracting different practices factors from different research papers is considered a valuable practice for the enhancement of transferability. We explicitly refer to the practices identified through SLR for each de-motivator factors.

C. DEPENDABILITY

The research study was performed by one scholar, therefore it last for up to two years. Where continuous feedback from the research supervisor helped to understand the research methodology and limiting interpretation biases.

D. CONFORMABILITY

We have attempted to mitigate this risks by observing any ambiguous topics that were discussed in a group setting, but there is a greater threat that a single researcher may be biased and extract incorrect data regularly. Nevertheless, to validate the results with the interrelated test the authors made their best efforts, i.e., every author checked the results independently and concluded which practices should be included and which excluded. Although, checking each result by every author was time-consuming but this exercise enhanced accuracy of our results.

Prioritizing the Demotivators using AHP methods is one possible threat. The prioritization of the Demotivators is grounded in the expert's opinion collected by questionnaire survey. Hence, a hasty approach may completely impair the effectiveness of the study, although the CR was calculated for each pairwise comparison matrix, indicating an acceptable internal validity in the priority of the challenges.

VIII. CONCLUSION AND FUTURE WORK

Our study identified 76 practices for De-motivators through SLR to adopt Agile Methods from a large-scale management point of view. Our results reveal that to adopt agile methodologies these practices / solutions can be helpful to project managers. Apart from the above-mentioned limitations, we are sure that our research will be useful in both academic and industrial situations.

AHP is used to prioritize the challenges using the responses collected form experts through questionnaire surveys and interviews and based on a pairwise comparison matrices of the challenges. A total of 15 Demotivators were extracted from the [1] and their classification were made into four

TABLE 33. Titles of final selected papers.

TITLES OF FINAL SELECTED PAPERS		Authors / Date of Review
Sr.no	**SpringerLink**	
1.	Exploring software development at the very large scale: a revelatory case study and research agenda for agile method adaptation	Torgeir Dingsøyr & Nils Brede Moe & Tor Erlend Fægri & Eva Amdahl Seim (2008)
2.	Component-Oriented Agile Software Development	Zoran Stojanovic Et Al (2003)
3.	Exploring knowledge management in agile software development organizations	Carine Khalil & Sabine Khalil (2019)
4.	Divide After You Conquer: An Agile Software Development Practice for Large Projects	Ahmed Elshamy and Amr Elssamadisy (2006)
5.	The Impact of Agile Software Development Approach on Software Developers' Responsibilities	Anne-Maarit Majanoja, Petri Avikainen, and Ville Leppänen (2014)
6.	Applying Agile to Large Projects: New Agile Software Development Practices for Large Projects	Ahmed Elshamy and Amr Elssamadisy (2007)
7.	Organisational Culture in Agile Software Development	Peter Wendorff (2002)
8.	How product owner teams scale agile methods to large distributed enterprises	Julian M. Bass (2014)
9.	Large-scale agile transformation at Ericsson: a case study	Maria Paasivaara Et Al (2008)
10.	Perspectives on Productivity and Delays in Large-Scale Agile Projects	Deepika Badampudi, Samuel A. Fricker, and Ana M. Moreno(2013)
11.	Scaling up the Planning Game: Collaboration Challenges in Large-Scale Agile Product Development	Felix Evbota, Eric Knauss and Anna Sandberg (2016)
	GoogleScholar	
12.	De-motivators for the adoption of agile methodologies for large-scale software development teams: An SLR from management perspective	Muhammad Faisal Abrar ET ALL (2020)
13.	Prioritizing challenges of agile process in distributed software development environment using analytic hierarchy process	Mohammad Shameem Et All (2018)
14.	Preference in using Agile development with larger team size	Ahmed Zia et al (2018)
15.	Agile Large-Scale Software Development Success Factors, Challenges and Solutions	Amro Al-Said Ahmad (2014)
16.	people factors in agile software development and project management	Vikash Lalsing, Somveer Kishnah and Sameerchand Pudaruth (2012)
17.	Challenges in adopting Agile practices: Perceptions of Software Practitioners in Indonesia	Emyr Al Kautsar and Norsaremah Salleh, Et all (2013)
18.	Comparison between Traditional Plan-based and Agile Software Processes According to Team Size & Project Domain	Nesma Keshta, Yasser Morgan (2017)
19.	Factors Influencing Agile Practices: A Survey	Akhil Kumar (2012)
20.	Communication and Quality in Distributed Agile Development: An Empirical Case Study	R. Green, T. Mazzuchi, and S. Sarkani (2010)
21.	Transition of organizational roles in Agile Transformation Process: A Grounded Theory approach	Milos Jovanovic , Antonia Mas , Antoni Mesquida , Bojan Lalic (2017)
22.	Motivations and Measurements in an Agile Case Study	Lucas Layman et all (2004)
23.	Coordination In Large Agile Projects	Peng Xu (2009)
24.	Communication Factors for Speed and Reuse in Large- Scale Agile Software Development	Antonio Martini et all (2016)
25.	Effective Communication in Distributed Agile Software Development Teams	Siva Dorairaj, James Noble, and Petra Malik (2011)
26.	Agile Software Development with Distributed Teams: Senior Management Support	Siva Dorairaj et all (2013)
27.	The Adoption of Agile Processes in Large Web Development Enterprises: A Survey in Jordan	Omaima Al-Allaf (2010)
28.	Agile Software Development with Distributed Teams: Agility, Distribution and Trust	Siva Dorairaj, James Noble (2013)
29.	Working with Agile in a Distributed Environment	Marek Majchrzak Et Al (2014)
30.	The Adoption and Evolution of Agile Practices	Marco Sampietro (2016)
31.	Key Factors for Selecting an Agile Method: A Systematic Literature Review	Mashal Alqudah, Rozilawati Razali (2017)
	ACM Library	
32.	Towards a common Agile Software Development Model (ASDM)	André Janus (2012)
33.	A Systematic Literature Review of Improved Knowledge Management in Agile Software Development	Mochamad Umar Al Hafidz, Dana Indra Sensuse (2019)
34.	Measuring Productivity in Agile Software Development Process: A Scoping Study	Syed Muhammad Ali Shah Et al. (2014)
35.	Alignment of Stakeholder Expectations about User Involvement in Agile Software Development	Jim Buchan Et all (2017)
36.	On the Benefits/Limitations of Agile Software Development: An Interview Study with Brazilian Companies	Fernando Kamei Et all (2017)
37.	Studying Communication in Agile Software Development	Tuomas Niinimäki Et All (2009)
38.	Communication Factors for Speed and Reuse in Large- Scale Agile Software Development	Antonio Martini, Lars Pareto, Jan Bosch (2013)
39.	Scaling Agile Software Development to Large and Globally Distributed Large-scale Organizations	Abheeshta Putta (2011)

TABLE 33. *(Continued.)* **Titles of final selected papers.**

SIENCE DIRECT Library		
40.	Agile methods tailoring – A systematic literature review	Amadeu Silveira Campanelli, Fernando Silva Parreiras (2015)
41.	The relationship between organizational culture and the deployment of agile methods	Juhani Iivari , Netta Iivari (2010)
42.	Agile methods rapidly replacing traditional methods at Nokia: A survey of opinions on agile transformation	Maarit Laanti , Outi Salo , Pekka Abrahamsson (2010)
43.	Empirical studies of agile software development: A systematic review	Tore Dyba , Torgeir Dingsøyr (2008)
44.	Challenges and success factors for large-scale agile transformations: A systematic literature review	Kim Dikert , Maria Paasivaara , Casper Lassenius (2016)
45.	Interpretative case studies on agile team productivity and management	Claudia de O. Melo Et Al (2012)
46.	A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case	Kai Petersen , Claes Wohlin (2009)
47.	The impact of inadequate customer collaboration on self-organizing Agile teams	Rashina Hoda , James Noble, Stuart Marshall (2010)
48.	Quality requirements challenges in the context of large-scale distributed agile	Wasim Alsaqaf , Maya Daneva , Roel Wieringa (2019)
49.	Coordination artifacts in Agile Software Development	Anna Zaitsev, Uri Gal, Barney Tan (2020)
50.	Understanding agile software development practices using shared mental models theory	Xiaodan Yu , Stacie Petter (2014)
51.	Empirical studies of geographically distributed agile development communication challenges: A systematic review	Yehia Ibrahim Alzoubi , Asif Qumer Gill , Ahmed Al-Ani (2016)
52.	Conceptual model of working space for Agile (Scrum) project team	Pawel Rola , Dorota Kuchta , Dominika Kopczyk (2015)
53.	Group development and group maturity when building agile teams: A qualitative and quantitative investigation at eight large companies	Lucas Gren, Richard Torkar, Robert Feldt (2016)
54.	Teamwork quality and project success in software development: A survey of agile development teams	Yngve Lindsjørn Et Al (2016)
55.	Leveraging creativity in requirements elicitation within agile software development: A systematic literature review	Ainhoa Aldave, JuanM. Vara, David Granada, Esperanza Marcos (2019)
56.	Categorization of risk factors for distributed agile projects	Suprika V. Shrivastava , Urvashi Rathod (2014)
57.	Identifying some important success factors in adopting agile software development practices	Subhas Chandra Misra , Vinod Kumar , Uma Kumar (2009)
IEEE		
58.	Practical Implications from a Preliminary Theory of Simplicity in Agile Software Development Based on a Qualitative Study	Wylliams Barbosa Santos Et all (2017)
59.	Risks of Agile Software Development: Learning from Adopters	Amany Elbanna, Suprateek Sarker (2016)
60.	What Do We Know about Agile Software Development?	Tore Dybå and Torgeir Dingsøyr (2009)
61.	Agile Software Development: Impact on Productivity and Quality	A. Ahmed Et Al (2010)
62.	Significance of Agile Software Development and SQA Powered by Automation	Bilal Gonen, Dipali Sawant (2020)
63.	Productivity in Agile Software Development: A Systematic Mapping Study	Sandra L. Ramírez-Mora, Hanna Oktaba (2017)
64.	Challenges in Agile Software Development: A Systematic Literature Review	Widia Resti Fitriani Et Al (2016)
65.	Factors Influencing Productivity of Agile Software Development Teamwork: A Qualitative System Dynamics Approach	Israt Fatema , Kazi Sakib (2017)
66.	Factors that Impact Implementing an Agile Software Development Methodology	Jeffrey A. Livermore (2007)
67.	An Adaptive Model Of Health Diagnosis For Agile Software Development	JING-FAN TANG (2008)
68.	An Extended Adaptive Process Model for Agile Software Development Methodology	Sameera Sadaf Et Al (2017)
69.	A Bayesian Based Method for Agile Software Development Release Planning and Project Health Monitoring	Ahmed Nagy, Mercy Njima and Lusine Mkrtchyan (2010)
70.	A Mapping Study on Requirements Engineering in Agile Software Development	Ville T. Heikkil Et Al (2015)
71.	A Methodology for Assessing Agile Software Development Methods	Shvetha Soundararajan Et Al (2012)
72.	Experience Report of Teaching Agile Collaboration and Values, Agile Software Development in Large Student Teams	Martin Kropp et all (2016)
73.	Deepening Our Understanding of Communities of Practice in Large-Scale Agile Development	Maria Paasivaara and Casper Lassenius (2014)
74.	BuildBot: Robotic Monitoring of Agile Software Development Teams	Ruth Ablett Et Al (2007)

categories: i.e. process, team, organization management and technology. These Demotivators were practically validated through questionnaire survey using the responses of 54 respondents. In the conclusion the rakings of the study showed that Exhaustive Pair Programming is most critical Demotivator and Lack of team orientation is least critical factor adopting agile methods from perspective of software project manager. The rankings and weights of the Demotivators and their categories deliver a framework for project managers to evaluate their management tactics for successfully adopting agile methodologies in large scale development teams.

TABLE 34. Practices of de-motivators.

Sr.no	Practices / Solutions
1.	The team behavior and organizational culture should be aligned to agile values. Organizational culture should be flexible to adopt agile methods.
2.	Reduce the organizational complexity to adopt agile at a large scale. Adopting the Agile method is difficult due to resistance to change. Effective approaches should be used to make changes in the organization
3.	Support and involvement from management on regular basis have a vital impact on the adaptation of agile methods
4.	The organization should provide an environment that facilitates communication among customers and team members.
5.	These resources should be utilized in team training and team member memberships to prepare them to use ASDM.
6.	Top-level management's participation in Agile Coaching sessions was a successful strategy for increasing their knowledge and demonstrating their commitment to the Agile transformation
7.	Domain knowledge and technology aspects should be balanced by teams to avoid ambiguities and organizational processes, management practices requirements, and design confusion.
8.	Management support in Agile has a significant role in terms of facilitating knowledge sharing, building trust, and improving communication.
9.	Another study concluded that engineers were motivated by agile principles and project managers need motivation and training in agile methods
10.	Training and coaching sessions to education personal to make them agile experts
11.	Provide training on agile methods and coach teams as they learn by doing
12.	Training in the methodology can produce experts in the methodology
13.	Adaptation of agile methods in practice and experience in applying these methods will have a positive effect on agile adopter's resistance
14.	The change should be perceived as easy enough and understandable to the team with good reasons to adopt
15.	Management is unwilling to adopt change because of a lack of understanding and lack of involvement in agile methods.
16.	Customer involvement impacts development team feedback on regular bases.
17.	Frequent communication with the customer also leads to effective communication and good customer relationship
18.	Team members should develop a relationship with customers based on trust.
19.	Increase the ability to meet customer requirements and enhance customer relationships
20.	Iterative development and retroactive meetings with the client make the team respond to the customer's requirements.
21.	The developer can select the best method for the customer if they did not have a way of working requirements. A good experience of interaction with customers should be required for elicitation requirements.
22.	Improved communication between stakeholders and developers, instant releases, and enhanced flexibility can lead to enhance better understanding of requirements.
23.	One of the key aspects of ASD is the collocation of the development teams. To react quickly to frequently changing and ambiguous requirements.
24.	User stories are used to force customers to provide clear requirements with clear business drives
25.	The instant messaging tool was integrated with the backlog tool where users can reference requirements in the backlog via a direct link
26.	Close customer collaboration can lead to closely aligned end users so that they can freely provide valuable knowledge
27.	Knowledge can be transferred in a better way due to frequent feedback at each iteration and good communication
28.	Customer involvement on regular bases impacts feedback of the development team. Small teams enable effective communication and fast feedback
29.	The team was instructed to change their work practice so that it involves a short feedback cycle. Short iterations improve team feedback as well as customers feedback
30.	The demonstration is the best tool to get feedback
31.	The strict control of procedure leads to productivity. A motivating workplace provides the needed opportunity it also leads to productivity
32.	Developing software in short iterations can provide delivery of a small part of the project to the customer before the deadline.
33.	The management should act as a facilitator to the teams so that they will focus on the delivery of the product
34.	Overcoming communication gaps enhances the productivity
35.	Experienced and flexible peoples contribute more to productivity. New team members may bring new ideas, solutions, and energy to make improvements and establish agile practices. Professional training is also the most significant factor in productivity.
36.	Pair programming has a positive impact on productivity and best quality
37.	Developers keep code as straightforward, simple, and technically advanced as possible, and lessening documentation can also enhance productivity.
38.	Agile accepts changes at the implementation step and it can also increase productivity
39.	Automation can enhance productivity
40.	Productivity is related to effectiveness and efficiency. And also depends on agile teamwork.
41.	Close collaboration with the customer and customer representatives produces quality software within a period and enhances customer relationship
42.	Software developers value onsite customers and prefer to be available to onsite customers. The Regular collaboration of customers with the team allows for a shared goal between them
43.	The incorrect arrangement could provide miscommunication and conflict among stakeholders that can result in ineffective customer presence
44.	To uncover problems facing customer presence electronic collaboration is the best factor for large customers. Onsite customer is also agile best practice.
45.	Using pair programming there is no need for distributed team environment

TABLE 34. *(Continued.)* **Practices of de-motivators.**

46.	Pair programming facilitates learning if partners are exchanged regularly in this way to achieve optimal learning and increase collective code ownership.
47.	Pair programming in GDAD can be achieved using screen sharing and audio calls
48.	The arrangement of a Development team cell in which resources are seated in a circle facilitates that team members can use these resources mutually and can work together. Open workspace allows sharing of resource
49.	The agile team should be trained on formal specifications of components interfaces for the purpose to enhance the communication among the team members when they are separated from developers.
50.	To remove the stress from the team freedom should be given to the team and the organization in findings between autonomy and control.
51.	Training sessions should be arranged on the basics knowledge of Agile.
52.	The training of agile practice should give to development teams because of the implementation of agile practices at the stage of development teams.
53.	To organize overall strategy, the Agile team must be continuously guided by strategic inputs to take optimum decisions
54.	One person on the team who got training will train other team members.
55.	Conduct regular conferences for the integration of various parts of the project on specific milestones. Modifications to the code should be integrated into the code base regularly and examined for integration issues. Frequent check-ins are performed.
56.	Testing practices can be an automated test run,test-driven development, refactoring, and unit testing.
57.	Introducing the practice of early and continuous testing because it takes time and effort to introduce this properly
58.	By widening the knowledge of the other components to reduce team orientation. Team members should work in pairs. Pair programing facilitates learning.
59.	Inclusion or exclusion of new members to the team may cause the orientation of the team already been established.
60.	Delivering presentations team members share their experiences and expertise with can lead to removing team orientation problems. Mutual support of team members can overcome team orientation
61.	Team support each other when they communicate frequently and share knowledge
62.	Cross-training is an instructional strategy in which each team member is trained in the duties of his or her teammates
63.	Frequent meetings on daily biases enhance communication among team members and improve collective problem-solving. Who has the right information should be communicated.
64.	Agile collaboration tools enhance the exchange of information among team members and promote communication as well for the project as a sharing vision. For effective communication, appropriate communication tools are important for team members
65.	For better communication, the teams should be divided into sub-teams. Communication will be successful but consistency and redundancy of data may occur that there is another practice team members rotation
66.	Team members should be collocated to promote frequent interactions between team members for effective communication.
67.	To reduce challenges in larger-scale distributed agile software development various communication channels are available. I.e. telephones, video conferencing calls etc.
68.	Face to face communication is possible in collocated agile software development with team members to exchange knowledge.
69.	Improve large scale agile development team collaboration by maintaining valuable documentation. Documentation helps in communicating requirements changes to project team members or customer at a large scale at distributed locations. Teamwork enables using performance models enables coordination, collaboration, and communication among agile software development teams.
70.	The project manager should ensure that all obtained information were organized in effective way for team. Changes in requirements were always organized and regular feedback was provided on how the team is performing.
71.	Good relationships between team members can also reduce language barriers. Effective relationship between developer and customer can enhance communication Customer collaboration
72.	Informal communication helps to develop a strong team
73.	Project backlog is shared between distributed development teams to share project requirements and knowledge.
74.	Sharing common and open working space for all team members can make communication effortless and quick.
75.	Give training to team member to improve their communication skills. For language barrier set English as an official language and give trainings to team members to overcome language problems
76.	The information that should be communicated to the entire team should be presented in a continuous manner

Our future work is to categorize these practices using the Analytical Hierarchy process (AHP). These studies identified the practices which will be the input to the AHP process. AHP is the multi-criteria process in which research prioritizes the factors according to their best suitable needs.

APPENDIX A

See Table 33.

APPENDIX B

See Table 34.

REFERENCES

[1] M. Faisal Abrar, M. Sohail, S. Ali, M. F. Majeed, I. A. Shah, N. Rashid, and N. Ullah, "De-motivators for the adoption of agile methodologies for large-scale software development teams: An SLR from management perspective," *J. Softw., Evol. Process*, vol. 32, no. 12, p. e2268, Dec. 2020.

[2] M. I. Khan and S. U. Khan, "Critical barriers in project management faced by offshore software multi-sourcing vendors: A detailed study," *Organization*, vol. 7, p. 27, Jan. 2016.

[3] N. Rashid and S. U. Khan, "Green agility for global software development vendors: A systematic literature review protocol," *Proc. Pakistan Acad. Sci.*, vol. 52, no. 4, pp. 301–313, 2015.

[4] K. Schwaber and M. Beedle, *Agile Software Development With Scrum*. Upper Saddle River, NJ, USA: Prentice-Hall, 2002.

[5] J. A. Livermore, "Factors that significantly impact the implementation of an agile software development methodology," *J. Softw.*, vol. 3, no. 4, pp. 31–36, Apr. 2008.

[6] S. Keele, "Guidelines for performing systematic literature reviews in software engineering," 2007.

[7] M. Paasivaara, B. Behm, C. Lassenius, and M. Hallikainen, "Towards rapid releases in large-scale XaaS development at ericsson: A case study," in *Proc. IEEE 9th Int. Conf. Global Softw. Eng.*, Aug. 2014, pp. 16–25.

[8] B. Boehm and R. Turner, "Management challenges to implementing agile processes in traditional development organizations," *IEEE Softw.*, vol. 22, no. 5, pp. 30–39, Sep. 2005.

[9] T. Dyba and T. Dingsoyr, "What do we know about agile software development?" *IEEE Softw.*, vol. 26, no. 5, pp. 6–9, Sep. 2009.

[10] T. Dingsøyr and N. B. Moe, "Towards principles of large-scale agile development: A summary of the workshop at XP2014 and a revised research agenda," in *Agile Methods, Large-Scale Development, Refactoring, Testing, and Estimation*. Berlin, Germany: Springer, 2014, pp. 1–8.

[11] T. Dingsøyr, T. T. E. Fægri, and J. Itkonen, "What is large in large-scale? A taxonomy of scaling in agile software development," in *Proc. 15th Int. Conf. Product-Focused Softw. Process Improvement (PROFES)*, Helsinki, Finland, Dec. 2014, pp. 273–276.

[12] M. Paasivaara, C. Lassenius, V. T. Heikkilä, K. Dikert, and C. Engblom, "Integrating global sites into the lean and agile transformation at Ericsson," in *Proc. IEEE 8th Int. Conf. Global Softw. Eng.*, Aug. 2013, pp. 134–143.

[13] S. Nerur, R. Mahapatra, and G. Mangalaraj, "Challenges of migrating to agile methodologies," *Commun. ACM*, vol. 48, no. 5, pp. 72–78, May 2005.

[14] M. Paasivaara, S. Durasiewicz, and C. Lassenius, "Using scrum in a globally distributed project: A case study," *Softw. Process: Improvement Pract.*, vol. 13, no. 6, pp. 527–544, Nov. 2008.

[15] H. Berger and P. Beynon-Davies, "The utility of rapid application development in large-scale, complex projects," *Inf. Syst. J.*, vol. 19, no. 6, pp. 549–570, 2009.

[16] E. Bjarnason, K. Wnuk, and B. Regnell, "A case study on benefits and side-effects of agile practices in large-scale requirements engineering," in *Proc. 1st Workshop Agile Requirements Eng.*, Jul. 2011, p. 3.

[17] T. Dingsøyr, T. E. Fægri, and J. Itkonen, "What is large in large-scale? A taxonomy of scale for agile software development," in *Product-Focused Software Process Improvement*. Berlin, Germany: Springer, 2014, pp. 273–276.

[18] K. Petersen and C. Wohlin, "The effect of moving from a plan-driven to an incremental software development approach with agile practices," *Empirical Softw. Eng.*, vol. 15, no. 6, pp. 654–693, Dec. 2010.

[19] M. Fowler, "Put your process on a diet-as a reaction to cumbersome approaches to development, new methodologies have appeared. These methods attempt a compromise between no process and too much process," *Softw. Develop.*, vol. 8, no. 12, pp. 32–39, 2000.

[20] H. Koehnemann and M. Coats, "Experiences applying agile practices to large systems," in *Proc. Agile Conf.*, 2009, pp. 295–300.

[21] A. Elshamy and A. Elssamadisy, "Divide after you conquer: An agile software development practice for large projects," in *Extreme Programming and Agile Processes in Software Engineering*. Berlin, Germany: Springer, 2006, pp. 164–168.

[22] S. C. Misra, V. Kumar, and U. Kumar, "Identifying some critical changes required in adopting agile practices in traditional software development projects," *Int. J. Quality Rel. Manage.*, vol. 27, no. 4, pp. 451–474, Apr. 2010.

[23] M. Cohn and D. Ford, "Introducing an agile process to an organization [software development]," *Computer*, vol. 36, no. 6, pp. 74–78, Jun. 2003.

[24] B. Boehm, "Get ready for agile methods, with care," *Computer*, vol. 35, no. 1, pp. 64–69, 2002.

[25] C. Wohlin, E. Mendes, K. R. Felizardo, and M. Kalinowski, "Guidelines for the search strategy to update systematic literature reviews in software engineering," *Inf. Softw. Technol.*, vol. 127, Nov. 2020, Art. no. 106366.

[26] W. Afzal, R. Torkar, and R. Feldt, "A systematic review of search-based testing for non-functional system properties," *Inf. Softw. Technol.*, vol. 51, no. 6, pp. 957–976, Jun. 2009.

[27] R. A. Khan, M. F. Abrar, S. Baseer, M. F. Majeed, M. Usman, S. U. Rahman, and Y.-Z. Cho, "Practices of motivators in adopting agile software development at large scale development team from management perspective," *Electronics*, vol. 10, no. 19, p. 2341, Sep. 2021.

[28] M. K. Alqudah and R. Razali, "Key factors for selecting an agile method: A systematic literature review," *Int. J. Adv. Sci., Eng. Inf. Technol.*, vol. 7, no. 2, pp. 526–537, Apr. 2017.

[29] K. Dikert, M. Paasivaara, and C. Lassenius, "Challenges and success factors for large-scale agile transformations: A systematic literature review," *J. Syst. Softw.*, vol. 119, pp. 87–108, Sep. 2016.

[30] Y. I. Alzoubi, A. Q. Gill, and A. Al-Ani, "Empirical studies of geographically distributed agile development communication challenges: A systematic review," *Inf. Manag.*, vol. 53, no. 1, pp. 22–37, Jan. 2016.

[31] E. Al Kautsar, N. Salleh, R. Hoda, and A. L. Asnawi, "Challenges in adopting agile practices: Perceptions of software practitioners in Indonesia," in *Proc. 5th Int. Conf. Internet (ICONI)*, 2013, pp. 1–9.

[32] S. Dorairaj, J. Noble, and G. Allan, "Agile software development with distributed teams: Senior management support," in *Proc. IEEE 8th Int. Conf. Global Softw. Eng.*, Aug. 2013, pp. 197–205.

[33] S. C. Misra, V. Kumar, and U. Kumar, "Identifying some important success factors in adopting agile software development practices," *J. Syst. Softw.*, vol. 82, no. 11, pp. 1869–1890, Nov. 2009.

[34] J. Stapleton, *DSDM, Dynamic Systems Development Method: The Method in Practice*. Cambridge, U.K.: Cambridge Univ. Press. 1997.

[35] P. Xu, "Coordination in large agile projects," *Rev. Bus. Inf. Syst. (RBIS)*, vol. 13, no. 4, pp. 273–288, Oct. 2009.

[36] M. Laanti, O. Salo, and P. Abrahamsson, "Agile methods rapidly replacing traditional methods at nokia: A survey of opinions on agile transformation," *Inf. Softw. Technol.*, vol. 53, no. 3, pp. 276–290, Mar. 2011.

[37] B. Rumpe and A. Schröder, "Quantitative survey on extreme programming projects," in *Proc. Conf. Extreme Program. Flexible Processes Softw. Eng.*, Alghero, Italy, 2002, pp. 95–100.

[38] Graffin. (2001). *A Customer Experience: Implementing XP*. Accessed: Apr. 3, 2007. [Online]. Available: www.agilealliance.org/system/article/file/930/file.pdf

[39] Jensen. (2002). *Time Constrained Requirements Engineering—The Cooperative*. Accessed: Apr. 3, 2007. [Online]. Available: http://www.agilealliance.org/system/article/file/913/file.pdf

[40] W. B. Santos, J. A. O. G. Cunha, H. Moura, and T. Margaria, "Practical implications from a preliminary theory of simplicity in agile software development based on a qualitative study," in *Proc. Latin Amer. Comput. Conf. (CLEI)*, Sep. 2017, pp. 1–10.

[41] R. Green, T. Mazzuchi, and S. Sarkani, "Communication and quality in distributed agile development: An empirical case study. world academy of science," *Eng. Technol.*, vol. 61, pp. 322–328, 2010.

[42] A. Kumar and B. Goel, "Factors influencing agile practices: A survey," *Int. J. Eng. Res. Appl.*, vol. 2, no. 4, pp. 1347–1352, 2012.

[43] R. Hoda, J. Noble, and S. Marshall, "The impact of inadequate customer collaboration on self-organizing agile teams," *Inf. Softw. Technol.*, vol. 53, no. 5, pp. 521–534, May 2011.

[44] A. Zaitsev, U. Gal, and B. Tan, "Coordination artifacts in agile software development," *Inf. Org.*, vol. 30, no. 2, Jun. 2020, Art. no. 100288.

[45] M. Jovanović, A. Mas, A.-L. Mesquida, and B. Lalić, "Transition of organizational roles in agile transformation process: A grounded theory approach," *J. Syst. Softw.*, vol. 133, pp. 174–194, Nov. 2017.

[46] D. Russo, "The agile success model: A mixed methods study of a large-scale agile transformation," *ACM Trans. Softw. Eng. Methodol.*, vol. 30, no. 4, pp. 1–46, 2021.

[47] T. Dyba and T. Dingsøyr, "Empirical studies of agile software development: A systematic review," *Inf. Softw. Technol.*, vol. 50, pp. 833–859, Aug. 2008.

[48] C. Khalil and S. Khalil, "Exploring knowledge management in agile software development organizations," *Int. Entrepreneurship Manag. J.*, vol. 16, no. 2, pp. 555–569, Jun. 2020.

[49] C. de O. Melo, D. S. Cruzes, F. Kon, and R. Conradi, "Interpretative case studies on agile team productivity and management," *Inf. Softw. Technol.*, vol. 55, no. 2, pp. 412–427, Feb. 2013.

[50] N. Keshta and Y. Morgan, "Comparison between traditional plan-based and agile software processes according to team size & project domain (a systematic literature review)," in *Proc. 8th IEEE Annu. Inf. Technol., Electron. Mobile Commun. Conf. (IEMCON)*, Oct. 2017, pp. 567–575.

[51] S. L. Ramírez-Mora and H. Oktaba, "Productivity in agile software development: A systematic mapping study," in *Proc. 5th Int. Conf. Softw. Eng. Res. Innov. (CONISOFT)*, Oct. 2017, pp. 44–53.

[52] K. Petersen and C. Wohlin, "A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case," *J. Syst. Softw.*, vol. 82, no. 9, pp. 1479–1490, Sep. 2009.

[53] A. Putta, "Scaling agile software development to large and globally distributed large-scale organizations," in *Proc. IEEE/ACM 13th Int. Conf. Global Softw. Eng. (ICGSE)*, May 2018, pp. 136–139.
[54] A. Ahmed, S. Ahmad, N. Ehsan, E. Mirza, and S. Z. Sarwar, "Agile software development: Impact on productivity and quality," in *Proc. IEEE Int. Conf. Manag. Innov. Technol.*, Jun. 2010, pp. 287–291.
[55] I. Fatema and K. Sakib, "Factors influencing productivity of agile software development teamwork: A qualitative system dynamics approach," in *Proc. 24th Asia–Pacific Softw. Eng. Conf. (APSEC)*, Dec. 2017, pp. 737–742.
[56] N. Rashid and S. U. Khan, "Developing green and sustainable software using agile methods in global software development: Risk factors for vendors," in *Proc. 11th Int. Conf. Eval. Novel Softw. Approaches Softw. Eng.*, 2016, pp. 247–253.
[57] M. Korkala, P. Abrahamsson, and P. Kyllonen, "A case study on the impact of customer communication on defects in agile software Development," in *Proc. AGILE*, 2006, p. 11.
[58] X. Yu and S. Petter, "Understanding agile software development practices using shared mental models theory," *Inf. Softw. Technol.*, vol. 56, no. 8, pp. 911–921, Aug. 2014.
[59] J. Chong, R. Plummer, L. J. Leifer, S. R. Klemmer, O. Eris, and G. Toye, "Pair programming: When and why it works," in *Proc. PPIG*, 2005, p. 5.
[60] A. A. Khan, J. Keung, M. Niazi, S. Hussain, and M. Shameem, "GSEPIM: A roadmap for software process assessment and improvement in the domain of global software development," *J. Softw., Evol. Process*, vol. 31, no. 1, p. e1988, Jan. 2019.
[61] L. Gren, R. Torkar, and R. Feldt, "Group development and group maturity when building agile teams: A qualitative and quantitative investigation at eight large companies," *J. Syst. Softw.*, vol. 124, pp. 104–119, Feb. 2017.
[62] S. Dorairaj, J. Noble, and P. Malik, "Effective communication in distributed Agile software development teams," in *Proc. Int. Conf. Agile Softw. Develop.* Berlin, Germany: Springer, 2011, pp. 102–116.
[63] J. M. Bass, "How product owner teams scale agile methods to large distributed enterprises," *Empirical Softw. Eng.*, vol. 20, no. 6, pp. 1525–1557, Dec. 2015.
[64] M. Majchrzak, L. Stilger, and M. Matczak, "Working with agile in a distributed environment," *Softw. Eng. Res. Pract. Perspective*, vol. 7, no. 3, pp. 41–54, 2014.
[65] T. L. Dickinson and R. M. McIntyre, "A conceptual framework of teamwork measurement," in *Team Performance Assessment and Measurement: Theory, Methods, and Applications*, M. T. Brannick, E. Salas, C. Prince, Eds. Old Dominion Univ., 1997, pp. 19–43.
[66] V. G. Stray, N. B. Moe, and T. Dingsøyr, "Challenges to teamwork: A multiple case study of two agile teams," in *Proc. Int. Conf. Agile Softw. Develop.* Berlin, Germany: Springer, 2011, pp. 146–161.
[67] Y. Lindsjørn, D. I. K. Sjøberg, T. Dingsøyr, G. R. Bergersen, and T. Dyba, "Teamwork quality and project success in software development: A survey of agile development teams," *J. Syst. Softw.*, vol. 122, pp. 274–286, Dec. 2016.
[68] T. L. Saaty, *What is the Analytic Hierarchy Process? Mathematical Models for Decision Support*. Berlin, Germany: Springer, 1988, pp. 109–121.
[69] D. Russo, P. Ciancarini, T. Falasconi, and M. Tomasi, "A meta-model for information systems quality: A mixed study of the financial sector," *ACM Trans. Manag. Inf. Syst.*, vol. 9, no. 3, pp. 1–38, Sep. 2018.
[70] M. F. Abrar, M. S. Khan, I. Khan, G. Ali, and S. Shah, "Digital information credibility: Towards a set of guidelines for quality assessment of grey literature in multivocal literature review," *Appl. Sci.*, vol. 13, no. 7, p. 4483, Apr. 2023.
[71] M. F. Abrar, M. S. Khan, I. Khan, M. ElAffendi, and S. Ahmad, "Towards fake news detection: A multivocal literature review of credibility factors in online news stories and analysis using analytical hierarchical process," *Electronics*, vol. 12, no. 15, p. 3280, Jul. 2023.

FARMAN ALI was born in Swat, Pakistan, in 1993. He is currently pursuing the M.Sc. degree with the Department of Computer Software Engineering, University of Engineering and Technology, Peshawar, Peshawar, under the supervision of Assoc. Prof. M. U. Khan.

He is also teaching with the Elementary and Secondary Education Department, Khyber Pakhtunkhwa. His research interests include agile software development methodologies, systematic literature review, analytical hierarchy process, and machine learning.

MUHAMMAD USMAN (Senior Member, IEEE) received the B.Sc. degree in computer information systems engineering from the University of Engineering and Technology, (UET), Peshawar, Peshawar, Pakistan, in 2004, the M.Sc. degree in computer engineering from the Center of Advanced Studies in Engineering, Islamabad (CASE), Pakistan, in 2007, and the Ph.D. degree from the University of Ulsan, South Korea, in 2016. He is currently an Associate Professor with UET, Mardan, Mardan. His research interests include security and energy efficiency in wireless networks, next-generation net-works, and wireless sensor networks. In 2015, he received the Best SCI(E) Paper Award from the Korean Government through BK21+ Project. He is a Reviewer of various reputable international journals, such as IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, IEEE ACCESS, IEEE TRANSACTIONS ON COGNITIVE COMMUNICATION AND NETWORKING, IEEE SENSORS JOURNAL, IEEE SYSTEMS JOURNAL, *Applied Mathematics and Information Sciences*, and *The Computer Journal*.

MUHAMMAD FAISAL ABRAR previously held the position of a Lecturer with the Department of Computer Science, University of Swat, and the Department of Computer Science, University of Buner. He is currently a Lecturer with the Department of Computer Science, University of Engineering and Technology, Mardan, Mardan, Pakistan. His research interests include agile software development, software quality assurance, software outsourcing partnership, empirical software engineering, systematic literature review, big data, the Internet of Things, and information sciences. He has acted as a Reviewer for esteemed international journals, such as IEEE ACCESS, *Scientific Programming*, *Electronics*, *Mobile Information Systems*, and *Journal of Software: Evolution and Process*.

SHAMS UR RAHMAN, photograph and biography not available at the time of publication.

INAYAT KHAN received the Ph.D. degree in computer science from the Department of Computer Science, University of Peshawar, Pakistan. He is currently an Assistant Professor of computer science with the University of Engineering and Technology, Mardan, Mardan, Pakistan. His current research interests include the design and development of context-aware adaptive user interfaces for minimizing drivers' distractions, lifelogging, healthcare, deep learning, ubiquitous computing, accessibility, and mobile-based assistive systems for people with disabilities.

BADAM NIAZI received the M.S. degree from the Department of Computer Science, University of Peshawar. He is currently an Assistant Professor of computer science with the Faculty of Computing, Nangarhar University, Jalalabad, Afghanistan. He has published several research papers in well reputed international journals and conferences. His research interest includes mobile based assistive technologies for the special with special needs.