IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 8, 2019, accepted December 26, 2019, date of publication January 6, 2020, date of current version January 14, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2964432

Towards Taxonomical-Based Situational Model to Improve the Quality of Agile Distributed Teams

AMBER SARWAR[1], YASER HAFEEZ[1], SHARIQ HUSSAIN[2], AND SHUNKUN YANG[3]
[1]University Institute of Information Technology, Pir Mehr Ali Shah Arid Agriculture University, Rawalpindi 46000, Pakistan
[2]Department of Software Engineering, Foundation University Islamabad, Islamabad 44000, Pakistan
[3]School of Reliability and Systems Engineering, Beihang University, Beijing 100191, China

Corresponding author: Shunkun Yang (ysk@buaa.edu.cn)

This work was supported by the National Natural Science Foundation of China under Grant 61672080.

ABSTRACT At present, software organizations are developing software products that employ global software development (GSD) teams. Organizations tend to adopt new methodologies for global software development, among which is the use of agile in the GSD industry, which yields both benefits and challenges. However, software development teams do not consider situational needs that delay software delivery, resulting in the late discovery of incompatible assumptions and architecture level rework. In this study, we conduct a systematic literature review (SLR) to identify the situational factors that need to be considered by software development teams before developing a software product. We further present taxonomical classification and comprehensively map the situational factors that impact design development and advancement in the proposed Situational Agile Distributed Development (SADD) model. We propose 18 directed hypotheses against each situational factor that supports our SADD model. In order to evaluate our directed hypotheses, statistical analysis method is used, and the level of confidence of each directed hypothesis is validated. The result of our study confirms that global software development teams are highly reliant on the SADD Model. Our study will largely contribute by devising a multilevel taxonomy of situational factors that elevate the performance of global software development teams. This taxonomical classification will allow to better map the relationships between multiple situational factors and elevate the process of creating a holistic model to handle situational needs in the context of Agile Distributed Software Development (ADSD).

INDEX TERMS Agile distributed taxonomy, agile distributed teams, situational agile distributed development, situational model, taxonomical model.

I. INTRODUCTION

Over the years, global software development (GSD) has taken over the co-located software development model because of the former's increased benefits, such as increased quality and decreased cost [1]. In GSD projects, both the development teams and stakeholders may be located in different time zones [2]–[5]. However, because of the culture variance and communication barrier between the development teams, software development can become difficult in GSD [6]. In addition, with the passage of time, traditional approaches have become less effective in managing complex software development because of the many pitfalls and changes and the lack of proper management. Similarly, traditional approaches fail to generalize respective software development processes for multiple contexts, and the evidence confirms the lack of a unique approach to software development [1]. Agile methodology is one of the adaptive methods that can overcome the issues of GSD, such as culture variance and communication barrier. By contrast, emerging releases of agile makes it questionable to achieve a one-size-fits-all strategy [2]. Whereas agile techniques may seem to be the ''silver bullet'' for complex software development having tight constraints [7], the issue with this ''silver bullet'' is amplified when agile software development is used in a distributed environment [8]. The fundamental responsibility of Agile Distributed Software Development (ADSD) is that it should fit the needs of the project [9]. In addition, the needs of the project highly depend on the situational context, which may keep on changing. Situational context defines the context in which project is being developed. The context may change on the basis of project needs. In this research, situational context is ''Agile

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

Distributed Software Development''. Software development teams need to consider situational needs that highly rely on product quality [2]. Situational needs drive the context. i.e., situational context may change on basis of situational needs. In this research, situational needs are Team, Organization and Customer etc. Distributed software development teams must consider a wide range of situational factors that fulfill their situational needs ahead of deciding the most suitable process to adopt in situational context (ADSD). In this research, situational need is 'Team' and situational factors that drive the need are 'Team size, 'Technical Experience' etc. However, development teams may be unable to identify these related situational factors [2]. Another challenge lies with the view that is shared in agile distributed software on the basis of situational context, where the chosen situational approach should best fit the conditions, product, view, and goals of the markets and organization [10]. Similarly, whereas software architecture serves as the central concept to the whole software engineering, the evolution of software architecture with situational needs that comply with the ADSD environment is still lacking [11]. There is an obvious need for a software architectural design approach in the ADSD environment [6]. As of this review, no well-established software design methodology has been proposed in the literature. The previously stated issues are the major problems of software architecture in ADSD [12]. Therefore proper classification of situational factors is mandatory for selecting, analyzing, and evolving software architecture work products [13].

Previous studies somehow support the fact that architectural changes are directly proportional to frequent situational variations [10]. The determinant of situational needs on agile software architecture is an aggregation of development team performance and decline in software quality. Despite its significance, there is a lack of techniques, approaches, and experimental studies concerning the impact of situational needs on software architecture [7]. Whereas there exist many studies in the literature that examine the evolution of software development in terms of situational needs, the evolution of agile software architecture with situational needs is still under progress [6]. Furthermore, as advancement continues, tailoring and evolving software architecture for a given set of situational factors is becoming a complex problem [7]. Whereas some progress has recently been noted in regard to the evolution of software architecture with a given set of situational factors, the association of situational factors to software development teams is still an issue [2], [8]. In order to unravel this problem, a set of situational factors affecting software development teams were identified and taxonomically classified on the multilevel. Each situational factor represents a unique situation and is intelligently harmonized with software architectures that have exploited both the situational factors and situational awareness [9]. Although there exists a continuous stream of reported research in literature regarding software architectures for ADSD, systematically analyzing and taxonomically classifying the collective impact of situational factors that exist in literature on architectural solutions for distributed development teams in Situational Agile Distributed Development (SADD) require a timely effort.

In this study, we first conducted a systematic literature review (SLR) [10] for the period of 2013 to 2019 to analyze the current state-of-the-art approaches regarding situational factors and needs of ADSD teams. The objective of this SLR was to systematically identify the situational factors that need to be considered by software development teams before developing the software product. Further, this will help to analyze and present a taxonomical classification and comprehensively map the situational factors that impact design, development, and advancement in the proposed SADD. This research work is initiated and motivated by a number of research questions, and their findings are anticipated to distribute systematized knowledge among ADSD teams and researchers interested in SADD. The conducted research and proposed SADD model will further be beneficial for development teams and researchers in SADD. A systematic organization of research transfers a body of knowledge to develop and propose new theories and solutions and evolve future dimensions. In addition, it will be helpful for practitioners aiming to further extend work in SADD and propose new models and techniques.

SLR is conducted to identify the situational factors required in SADD, therefore providing quality and maturity to development teams for identifying innovative trends, research areas, and future magnitude of SADD. The main reason for applying SLR is to identify, classify, and synthesize current state-of-the-art research and transfer knowledge in the research community [10]. The following are the key contributions of this study.

Systematic identification and analysis of the aggregate findings on situational needs and factors existing in literature for SADD teams for characterizing (i) prevailing situational factors, (ii) multilevel taxonomical classification of those factors, (iii) SADD model proposed on the basis of taxonomical classification, (iv) and statistical evaluation, and empirical and practical implications of the SADD Model. We intend to achieve our common objective by answering the following questions by conducting a systematic literature review of the current literature:

RQ1: What are the current challenges and practical barriers of SADD?

RQ2: How can the identified challenges be classified taxonomically for attaining harmonization of software architecture in SADD?

RQ3: What are the practical implications of the proposed SADD model?

Our search string affirms to research questions, as RQ1 will uncover the challenges and situational factors which cause situational variations in Agile Software Development. The search string incorporates the key words related to our research questions. The remaining paper is structured as follows: Section II presents and explains our research methodology in detail, along with the findings of SLR.

Section III discusses the taxonomic classification of identified situational factors from the former section. Section IV describes our proposed SADD Model. Section V presents the results and discussions. Section VI concludes the paper and outlines plans for future work.

II. SYSTEMATIC LITERATURE REVIEW

This investigation proposes SLR for recognizable proof of situational variables required by different ADSD groups. SLR revealed situational factors in the literature and afterward fundamentally broke down the recognized information. SLR additionally raised the conduct of research in an increasingly precise manner and decreases the bias of methodological advances. It was a methodical and particular strategy for distinguishing and investigating distributed research in a similar field to accomplish a general goal and answer an explicit research question. SLR was an assorted writing audit because it was a relatively arranged and orderly survey when contrasted with standard writing survey. SLR additionally shaped a reason for taxonomical grouping that was exceedingly legitimate and helpful and that probably will not be conceivable in different techniques. The connected research system had been presented in Fig. 1. SLR depended on remarkable, checked, and approved audit conventions for the extraction, investigation, and documentation of results. This SLR utilized the rules proposed by Kitchenham *et al.* [14] with three-advance procedure that incorporated: making arrangements for the ideal SLR, directing the arranged procedure, and recording the discoveries. The result of each progression of the audit had been approved remotely to make the survey progressively solid. On the premise of SLR discoveries, taxonomical grouping has been proposed. Each period of the proposed procedure is outlined in the following subsections.

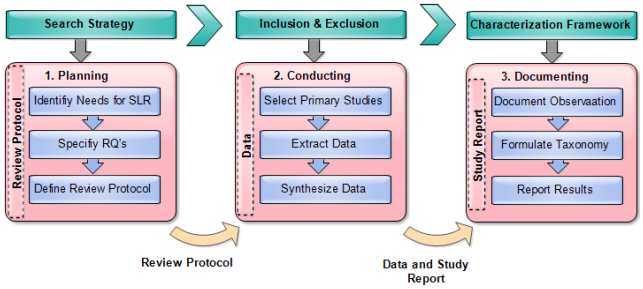

FIGURE 1. Overview of research methodology.

A. PLANNING THE REVIEW

Arranging began with recognizable proof of the requirements for a deliberate audit and produced a survey convention:

1) IDENTIFY THE REQUIREMENTS FOR SLR

The need was distinguished in phase 1. We likewise detailed the general objective and extent of the investigation through Population, Intervention, Comparison, Outcome and Context (PICOC) criteria [15], along with inclusion and exclusion criteria as shown in Table 1.

TABLE 1. Inclusion and exclusion criteria.

	Criteria	Rationale
Inclusion Criteria	Scientifically peer reviewed papers	A scientific paper guarantees a certain level of quality through peer review because it possesses a certain amount of content.
	Studies which discuss situational factors / environmental factors / contextual factors in ADSD	Specific solutions, metrics, and analysis of situational factors form the basis for the proposed solution.
	Studies that cover agile or distributed software development	Our context is agile distributed software development.
Exclusion Criteria	Non-peer-reviewed studies, white papers, and non-English publications	We tend to exclude these papers because they do not cover our objectives.
	Paper(s) will be excluded if it explains success or critical factors only.	
	Paper(s) will be excluded if it does not discuss agile architecture or software architecture.	
	Paper(s) will be excluded if it does not discuss software development.	

2) SPECIFY RESEARCH QUESTIONS

Three research questions were produced in order to investigate the answers with respect to situational needs in ADSD. These examination questions altogether infer the quest procedure for writing mining.

3) DEFINE REVIEW PROTOCOL

In view of the targets, we stipulate the exploration questions and the survey extension to plan quest strings for writing extraction. The inquiry string was created based on characterized research questions. Furthermore, the search string involved watchwords, topics, and ideas. Afterwards, databases were selected according to expert opinions from specialists. The method of creating a search string was shown in Fig. 2. According to Fig. 2 first step was generating search string i.e., identification of main concepts, which includes finding keywords and synonyms. Second step was identification of journals and databases that includes library catalogue. Third step was searching of resources that includes boolean operations like, 'AND' and 'OR'. Final step was to present results and refine results on review basis. The produced inquiry string was presented in Fig. 3 formulated on basis of Fig. 2 alongside databases. First level was string composition based on the steps shown in Fig. 2. Then came phase 1 of string execution on multiple databases. Phase 2 shows the extracted studies based on inclusion and exclusion studies. Phase 3 showed the screened studies. We additionally settle a convention for a methodical audit by following [16] and

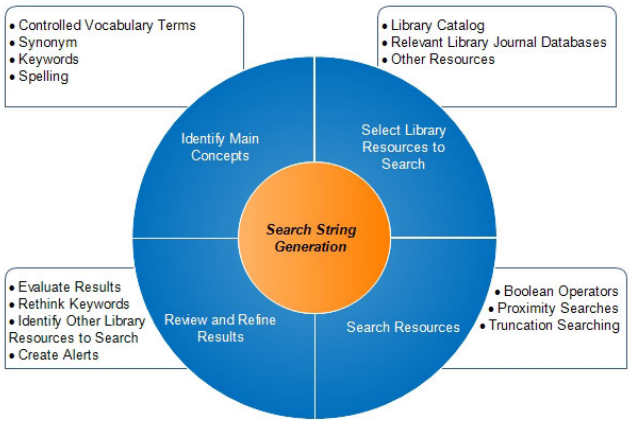

FIGURE 2. Search string generation.

our involvement with SLR [15], [17], [18]. Based on the proposal by Brereton *et al.* [16], we remotely assess the convention before its usage. We approach an external expert for input, who had involvement in directing SLR in a region with underlying ADSD situational settings. We additionally play out a starter investigation of the deliberate audit with five (around 25 percent) of the included examinations. The goal for leading a fundamental report is to initially decrease the predisposition between the scientists and continually improve the portrayal plot for information accumulation. We augment the survey scope, improve hunt techniques, and refined the incorporation/avoidance criteria during the pilot thinks about.

B. CONDUCTING THE REVIEW

Following are the subsequent stages, beginning with study choice and produces filtered and integrated data:

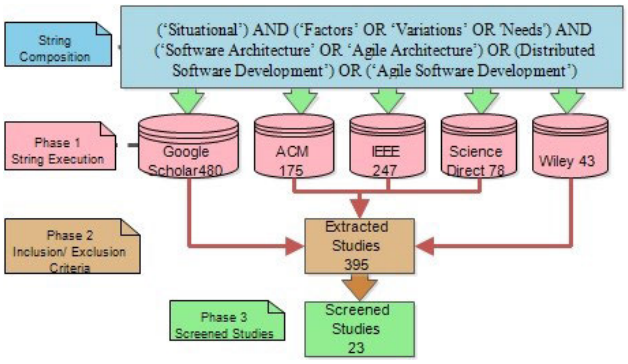

FIGURE 3. Search string, database result, and primary studies.

1) SELECT PRIMARY STUDIES

The pursuit terms utilized are based on the work proposed in [16] and are guided by research questions. Fig. 3 demonstrates the outcomes against an inquiry string on five unique databases. However, the search string as mentioned in Fig. 3 has been generated on the basis of Fig. 2. The generated search string identified main concepts relevant to our study, sources and refinement of results. We mined 1,023 literature resources from 2013 to 2019. The year 2013 is picked because a starter search found no outcomes prior to that year that were connected to the exploration questions. Because we utilize our prime inquiry criteria on title and conceptual aspect, this method brings about a high number of disconnected investigations, which are additionally refined with an auxiliary hunt.

2) EXTRACT DATA

The choice stage involves three phases: introductory hunt in databases, incorporation/avoidance, and last choice dependent on quality appraisal as shown in Fig. 4. This procedure incorporates screening of titles, edits compositions of potential essential, and is performed by the researchers against the inclusion/exclusion criteria in Table 1.

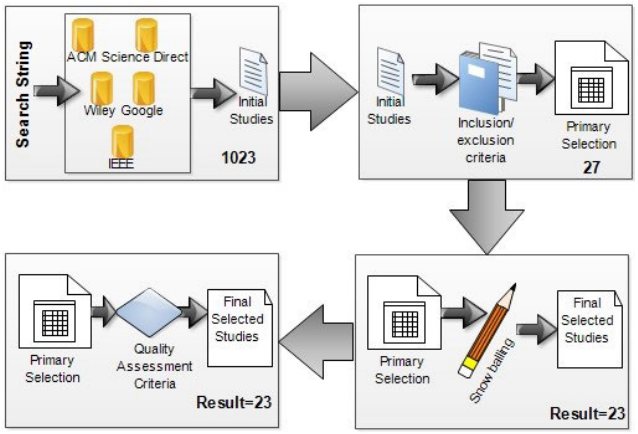

FIGURE 4. Search string, database result, and primary studies.

For nearly 20 percent of the studies, no choice could be made. In such cases, both rejection and continuation to a conclusive determination included looking at the full content. Lastly, determination depends on an approval sweep of the investigations, methods for movement, and apparatus backing and subtleties of the assessment approach. Subsequent to playing out this progression, 27 studies are chosen. During the optional inquiry procedure, references for the 27 studies are explored. Rules [14] suggest that snowballing from the reference arrangements of the recognized articles ought to be utilized, despite looking in databases to distinguish extra significant articles through the reference arrangements of the articles discovered by utilizing search strings. This method prompted the exclusion of four progressively pertinent examinations. Thus, 23 studies are incorporated for subjective evaluation. Subjective evaluation of the included investigations is shown in Fig. 4. Fig. 4 explains the detailed process of final selected studies. It also shows the quality assessment of primary studies and selection of final studies. Quality assessment was done by giving a value to each paper on basis of their work. Paper with same domain was given high value.

TABLE 2. Situational factors.

#	Situational Categories	Situational Factors	Value	Paper ID
1	Business Context and Requirement	Standards Requirement Stability	None, low, medium and high	[11]
2	Organization	Size	Small, medium, large	[13, 19]
		Maturity	Low, medium, high	
		Management Commitment	Low, medium, high	
		Structure	Functional, dysfunctional, matrix	
3	Team	Size	Small, medium, large	[20-22]
		Geographic Distributions	Co-located, distributed	
		Domain Experience	None, low, medium, high	
		Technical Experience	None, low, medium, high	
4	Customer	Availability	Single located, frequent, rare, planned	[20]
		Domain Experience Resistance	None, low, medium, high	
5	System	Size	Small, medium, large	[20]
		Complexity	Low, medium, high	
		Re-usability	None, component, application, domain specific, product line	
		Technology Maturity	None, existent, emergent	
		Existing Situational Approaches	Yes, no	
		Existing back-end services	No, yes	
		Innovation Degree	Low, medium, high	

3) SYNTHESIZE DATA

For the 23 included examinations, we essentially focus on the specialized thoroughness of the substance exhibited.

C. DOCUMENTING SLR FINDINGS

The review findings are efficiently organized, which recognizes gaps in literature with respect to situational approach, and are presented in Table 2. Table 2 answers our first research question as well. In Table 2 we have categorized situational factors on the basis of their concepts and themes along with their 'Value' choices for defined cases. Paper ID defines their references. Additionally, some essential and optional investigations that are identified through SLR, and which drive our motivation towards the target goal, are examined.

Organizations have been utilizing conventional programming advancement approaches as a way to execute their business techniques, in order to improve the achievement of their activities, and require the following professions: analysts and specialists [2]. Within the scope of programming advancement situations, numerous investigations have been led to distinguish elements that add to the customary activity disappointments. One of the serious issues with conventional programming improvement is the difference in venture advancement innovations and business settings [6]. New strategies and methodologies for programming advancement ventures have been proposed. Among these is an advancement procedure that spins around different improvements, along with emphasis on advancement cycle that can encourage nonstop associations with clients and address vulnerabilities. This iterative procedure is ordinarily one of the light-footed strategies that have been progressively actualized by numerous organizations to supplant customary programming advancement [12]. According to [15], [23], a couple of programming organizations all in all have received a particularly deft procedure. In this manner, a chronicle of light-footed strategy pieces, which arranges the obvious information as indicated by their destinations and necessities, is presented. The depiction of spry strategies in different venture circumstances is accomplished through a deliberate writing survey, which is dependent on observational investigations. Likewise, the paper recommends a learning technique that will facilitate the adaptation of new methods by the practitioners of ADSD. Furthermore, their methodology is limited to assembled associations. Yagüe *et al.* [6] gave guidance with respect to exercises gained from fitting spry techniques for huge-scale disseminated advancement. The exercises depended on one of the biggest improvement programs in Norway, where 12 scrum groups consolidated coordinated practices with customary undertaking by the executives. The Perform program conveyed 12 discharges over a four-year term, completing on a spending plan and on schedule. The creators abridged 10 key exercises on five pivotal themes that were important to other enormous improvement ventures looking to consolidate scrum with customary undertaking by the board. In addition, their exercises demonstrated that product improvement does not involve a fixed course; rather, it needs to fulfill changing needs and circumstances. As of late, two of the creators have delivered and distributed an underlying reference system that will more likely guide process planners to tailor venture explicit procedures just as much as to comprehend the hidden

choices behind programming procedure fitting. The reference structure involves 44 variables characterized into 8 primary classifications and further explained into 170 sub-factors [24], [25]. The reference system of O'Conner *et al.* [24] creators is expanded, and the way that there is certifiably not a solitary procedure splendidly fitting in all specific circumstances is settled upon; therefore, the product procedure ought to suit the prerequisites of its situational setting. In their past work, they combined an underlying reference system of the situational elements influencing the product improvement process. They controlled a contextual analysis and connected the recognized situational factors for steady programming advancement and conveyance. Their outcomes demonstrated that setting is an unpredictable and key witness for programming process choices. Their results inferred that situational antiquities have a stern connection with item design and procedure depiction [25], [26].

Similarly, Razavian *et al.* [27] conducted empirical research in software architecture decision-making and classified software architecture decision-making that involves humans, their behavioral issues, and practical situations. However, interactions of these situations in agile software architecture decision-making is still under progress [28], [29]. Overcoming the aforementioned challenges of coordination between situational factors and software architecture, Giray and Tekinerdogan [30] proposed a situational method engineering (SME) approach, which can be reprocessed to develop customized methods. The authors agreed upon the fact that there is not a single methodology that encourages a ''one size fits all'' approach. The constructed list of situational factors identified the method part, which expressed the architecture and meta-model. Whereas their work supported the Internet of Things domain, there is still no exact approach that handles the same issue in the domain of SADD. The stability of a system is highly dependent on its software architecture [28], [31]. However, in distributed systems, there are situations when the architecture is unavailable; thus, there should be a way to define the view of the system. Many different methods and tools exist to provide such a view. Whereas there have been taxonomies of different recovery methods and survey results of those methods analyzing how those results conform to the expert opinions on the systems, there has not been a survey that goes beyond a simple automatic comparison. Instead, this paper seeks to answer questions about the viability of individual methods in given situations [29], the quality of their results, and whether these results can be used to indicate and measure the quality and quantity of architectural situational changes. The authors evaluated their solution through case studies of Android, Apache Hadoop, and Apache Chukwa, obtained by running PKG, ACDC, and ARC. Their approach was unable to solve the architectural situational changes in distributed environment. Software architecture and early design decisions can take various forms, and their optima demand that these must be synchronized with the needs of the given situational context. Although earlier theoretical proposition is practically

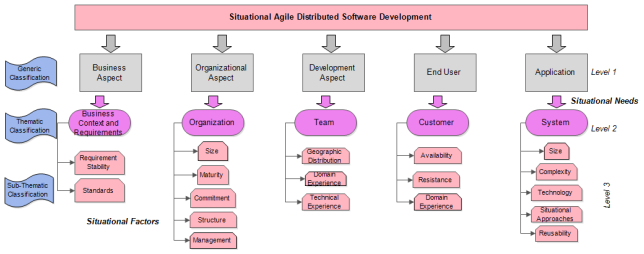

FIGURE 5. Taxonomic classification.

clear, the granularity of the communication between the software architecture and the factors of the situational context is less apparent. In view of the previously discussed reasoning, situational factors with less change (traditional) and with more change (agile) have high impact. Thus, during the construction of software architecture, different situational factors need to be considered [23], [30], [32].

III. TAXONOMIC CLASSIFICATION AND MAPPING OF THE RESEARCH

In this section we answer our second research question. Based on the SLR findings, the topics and their sub-arrangement are accomplished. To break down the situational requirements for ADSD groups, first we recognize the transcendent research subjects by applying topical investigation [33]. Staggered taxonomical arrangement is performed by following the modified rules proposed by Usman *et al.* [34]. According to [34], [35], ideas that are of the extent of scientific categorization advancement are excluded. This decision enables us to prohibit strategies and methodologies that are not upheld by the consequences of our SLR and fuse new discoveries to respond to the examination questions. This method helps in the investigation of the subjects and ideas to distinguish new angles that could be additionally improved for ADSD advancement groups. For example, the arrangement of the situational approach and mindfulness and whether exercises could be converged to encourage the utilization of the situational model are determined. The scientific classification in Fig. 5 arranges the recognized topics as a diagram for the current research and aides the result exchange for an efficient writing survey. The scientific categorization in Fig. 5 gives a precise distinguishing proof, with naming and arrangement of different research subjects dependent on the similitude or refinements of their relative commitments for an efficient writing audit in form of a taxonomy. In order to present in form of taxonomy, level based approach has been proposed i.e., first level describes its general classifications, second level defines the thematic classification based on related concepts and third level defines sub thematic classification. Through the breaking down of some pertinent examinations alongside the following of a portion of the rules from the ACM Computing Classification System and Computing Research Repository, and through the following of the rules for taxonomical arrangement [36], the taxonomical classification, as shown in the following figure, is determined.

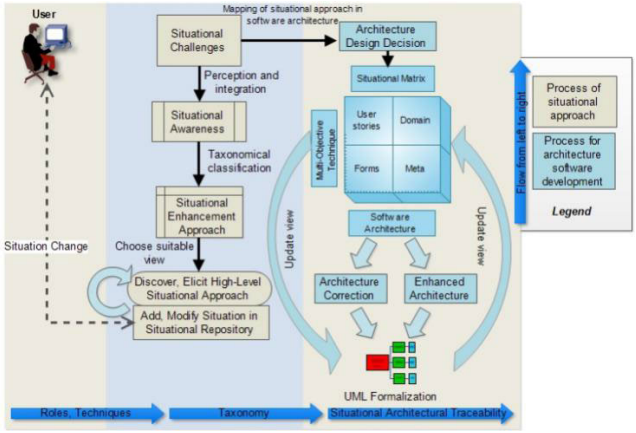

FIGURE 6. Proposed model.

IV. PROPOSED SADD MODEL

Based on the taxonomical arrangement, we propose a SADD model, which is shown in Fig. 6. In our proposed approach all situational challenges are handled by applying situational awareness and situational enhancement approach. Situational awareness is done on basis of perception and integration. Afterwards taxonomical classification situational enhancement approach should be applied. Situational challenges should also be applied in architecture design decision. Architecture correction and architecture enhancement, results in meta model that incorporates situational factors applied to each software process. The recognized situational factors in the proposed scientific categorization, will not just assist one's individual basic leadership, but will also additionally make venture creation stable through considering both situational and natural variables. The proposed SADD model enables the client to deal with situational needs, e.g., venture size and group size. Based on situational discernment and combination, situational mindfulness is mapped on the situational challenges. Situational mindfulness incorporates the ID of situational needs and related variables identified with ADSD [37], [38]. It will likewise upgrade the exhibition and nature of ADSD groups because it will raise the ADSD procedure. Later, when taxonomical grouping circumstance is considered, upgrade is done through applying multilevel taxonomical mapping. Through the use of taxonomical order, situational requirements are distinguished, and related situational components are mapped. In the event that a change happens, at that point the client, without much additional effort, can include or adjust the circumstance in the situational vault, thus utilizing situational components extricated from proposed scientific classification. On the premise of situational challenges, an engineering plan choice is done through making a situational network and choosing a reasonable building view.

The situational framework will not just deal with situational changes, but in addition, will accomplish harmonization of programming engineering advancement in light-footed appropriated programming improvement. Nonexclusive classification features numerous parts of situational technique, building to help with the product engineering tasks, advancement (post-sending), and improvement (pre-arrangement) periods of ADSD. Conventional characterization is utilized to compose the outcomes into three particular stages. Our proposed model has been validated according to validation process proposed by [38]. The model life-cycle should have development, implementation and operation along with modelling elements. Modelling elements comprise of input, process and output. Roles and responsibilities have also been defined in our proposed model. All of the above have been shown in Figure 6.

V. RESULTS AND DISCUSSIONS

In this section, the findings of this study are discussed. In order to obtain validated results, both case study and questionnaires approaches were used. Afterward, statistical techniques for analysis and synthesis of results were applied.

A. CASE 1: SITUATIONAL ADSD FOR SMALL-SCALE FARM MANAGEMENT SYSTEM

In order to conduct a case study, we used a published case study of a small-scale farm management system [23], [39] and mapped an improved version of the meta model used in [40] for representing the situational needs. The improved representation of the meta-model conforms to the ISO/IEC 24744 standard. With the guidelines proposed by Farwick et al. being followed [40], partial representation of a conceptual model is formed as shown in Fig. 7(a). The process is formed using our proposed SADD model. A small-scale farm needs an information system to support its agricultural decision-making process. A small team with a high domain knowledge in precision farming will develop the system. The farmers are the foremost clients, and two of them are picked as full-time client delegates to be engaged with this venture. The level of development is medium, henceforth some exploratory methodology is expected to moderate venture dangers. Existing methodologies and backend administrations will be utilized in the venture. When the procedure represented in Fig. 7(a) are followed and the distinguished arrangement of situational factors (recorded in Table 2) are utilized, situational needs and related work unit in regards to situational variables were recognized for the pertinent ADSD groups and are shown in Fig. 7(b). Because of contextual analysis mapping, Table 3 and Table 4 communicate with those distinguished situational factors from the contextual analysis following the total choice procedure, as appears in Fig. 8, and show the essential portrayals of situational needs and factors. These subtleties give direction to ADSD groups in recognizing new circumstances and utilizing the current situational needs with their methodology. ADSD groups ought to have the option to look at situational factors, which should be utilized in making new situational approaches, in order to approach the up-and-coming entrepreneur. The related work units and work items to every circumstance were made on the premise of existing rules [40]. Thematic classification expands the conventional order by

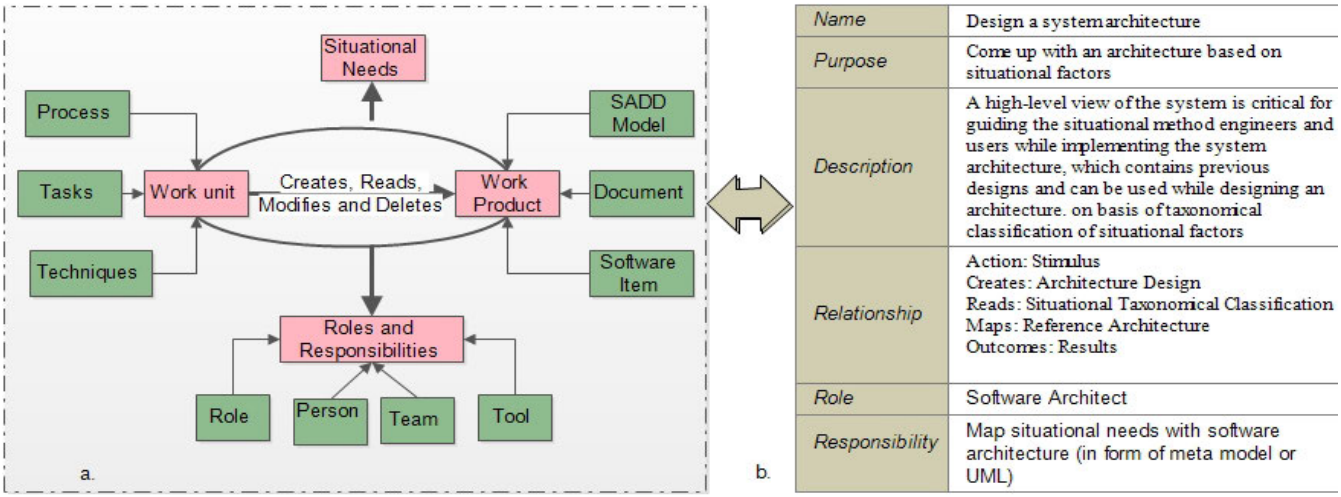

FIGURE 7. (a) A partial conceptual model of ISO/IEC 24744 standard, and (b) An example of work unit.

TABLE 3. A set of example situational needs and factors defined for the published case.

#	Situational Factor	Value
SN1	[Team].[Size]	Large
SN2	[Team].[Domain experience]	High
SN3	[Customer].[Availability]	Distributed
SN4	[System].[Degree of innovation]	High
SN5	[System]. [Size]	Large
SN6	[System]. [Existing Situational Approaches]	Yes

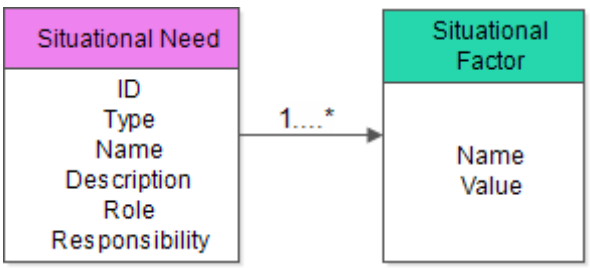

FIGURE 8. Situational need descriptor.

including subtleties based on the essential focal point of the examination in an accumulation of related investigations to distinguish and speak to the repetitive research subjects utilizing topical examination. We recognized five dominating topics.

For constructing work units and work products mentioned in Table 3 and Table 4, a descriptor is defined for each situational need in order to make it reusable. The situational need descriptor is illustrated in Fig. 8. A situational need is identified by assigning a unique name, a type (work unit, work product, producer) as defined in Fig. 7(a), description regarding situational need, and associated role and responsibility. The description of the situational need defines the situational factor associated, as shown in Fig. 7(b). The descriptor defines the level of granularity for each situational factor.

When the situational need descriptor in Fig. 8 is utilized, the search inquiry test to be executed on a situational store is according to the following: SELECT situational need WHERE Type = work unit OR work item OR maker AND Description = ADSD group AND FOR EACH Situational Factor = [Team][Size]. It is expected to choose all the situational elements related with situational need 'Group.' In the wake of choosing these reusable situational needs, an ADSD group has many applicant situational components to be utilized during SADD. Following the total determination process, as appears in Fig. 9, circumstance needs could be effectively mapped in programming engineering and raise the presentation of ADSD teams. Sub-topical classification gives a fine-grained refinement of the previously mentioned three subjects with seventeen particular sub-subjects.

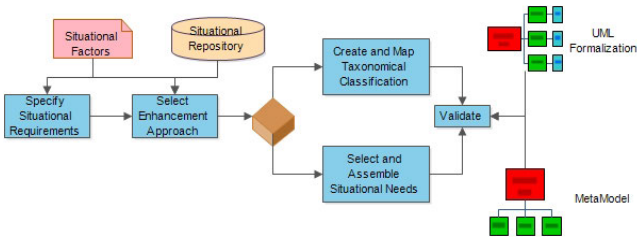

FIGURE 9. Constructing and mapping situational needs for software architecture.

Fig. 9 represents the complete process of selection using the proposed SADD model. According to the process, situational factors are defined after specifying associated situational needs. Afterward, situational enhancement approach is applied if required; otherwise, existing approach is reused. Once situational needs are properly identified, taxonomical

TABLE 4. Identified situational need descriptor details.

ID	Name	Type	Description	Roles and Responsibility
SD01	Produce an SADD Team	Work unit	Business context and requirements	Especially recommended when [Team].[Domain experience]=high; [Business].[Standards]= high
SD02	ADSD Situational Repository	Work product	Business context and requirements	Especially recommended when [Team].[Domain experience]=high; [Business].[Requirement stability]= None, Low
SD03	SADD Process Spec.	Work product	Business context and requirements	Especially recommended when [Team].[Domain experience] = none, low
SD04	SADD Glossary	Work product	Business context and requirements	Especially recommended when [Team].[Domain experience] = none, low
SD05	Specify SADD Requirements	Work unit	Requirements	Especially recommended when [Team].[Domain Experience] = high; [Customer].[Availability] = as per plan,
SD06	User Story for SADD	Work product	Requirements	Recommended when [Team].[Size] = small; [Customer].[Availability] = on-site
SD07	System UML	Work product	Architecture	Recommended when [Team].[Size] = small; [Customer].[Availability] = on-site
SD08	System Meta-model	Work product	Architecture	Especially recommended when [Team].[Size] = small; [Customer].[Availability] = as per plan, rare
SD09	Design a System Architecture	Work unit	Analysis and design	Especially recommended when [Team].[Size]=small; [Customer].[Availability] = as per plan, rare

classification is mapped against each situational factor, along with the assembling of situational needs. Afterward, the situational factors are validated and mapped on software architecture in the form of meta-model or UML formalization.

After the complete selection process, as shown in Fig. 9, is followed, a complete set of situational needs and associated situational factors are identified along with their value, as shown in Table 3. The identified situational needs for the mapped published case are used with their associated value.

Through the use of the conceptual model, as shown in Fig. 7(a), and the situational need descriptor, as shown in Fig. 8, Table 4 is generated. Table 4 explains the fields of each situational need descriptor in detail. Each descriptor is assigned a unique ID (e.g. SD01) that defines the situational descriptor for the first situational need as stored in the situational repository. The descriptor detail defines the roles and responsibilities for the mapped case, along with the description and type of situational need.

A comparison of our results with the published case study results indicate that our results are more refined and efficient. The comparative results for both approaches used for the published case are shown in Fig. 10. As shown in the figure, both approaches share the same situational needs but used different methodologies. It is evident from the figure that the published case is only applicable to small teams, whereas the SADD model is applicable for large teams. Similarly, the domain experience remained high for both approaches. Further, the published case is limited to on-site situations, whereas SADD model is for distributed teams. The published case did not use any situational approach, whereas the

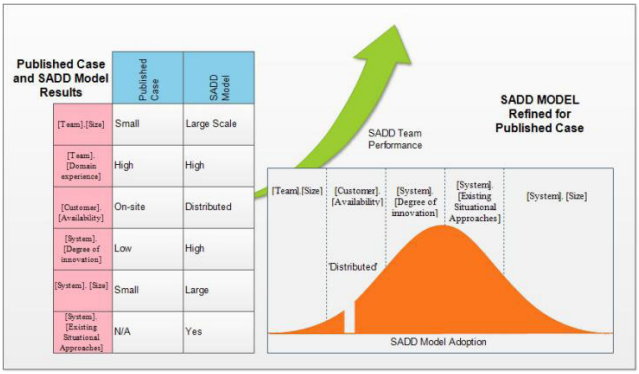

FIGURE 10. Result comparison for published case and SADD model.

SADD model identified a complete set of situational needs and proposed situational factors and descriptors, as shown in Figs. 7, 8, and 9 respectively. As a result, using the SADD model enhanced team performance and the overall quality of the SADD team.

Similarly if we compare our SADD model with existing situational Agile Distributed Software Development model (SADSD) as presented in [2]. It could be seen that (SADSD) only used scrum and XP practices to overcome the issues of agile architecture, however, we have used software architecture as agile architecture practice to overcome the harmonization of situational variations with software architecture. Also in SADD model situational perception and situational integration have also been linked in order to apply situational enhancement approach. Similarly, our SADD model

also stores user stories in form of matrix and further apply architecture correction in order to synchronize the situational variation in software architecture. This outputs a meta-model also shown in our mapped case study.

B. TAXONOMIC CLASSIFICATION EVALUATION

In order to evaluate the taxonomic classification, interviews from agile distributed software development practitioners were conducted. A set of directed hypotheses and specific questions was provided to the practitioners. The following are the directed hypotheses generated against identified situational factors.

H_o^A :- Requirements stability results in the upgrade of business context => Business aspect of architecture.

H_o^B :- Standards contribute to the successful business value of software architecture.

H_o^C :- Size of organization tends to contribute towards the complexity of software architecture.

H_o^D :- Commitment to the standards result in successful architecture deployment.

H_o^E :- Lack of maturity of organization leads to the missing of situational factors in agile distributed environment.

H_o^F :- Inconsistent structure of organization may lead to agile development in distributed environment.

H_o^G :- There should be a certain level of management of software architecture to fulfill organization standards.

H_o^H :- Geographic distribution does result in situational variations.

H_o^I :- Lack of technical dependencies within team leads to inconsistent architecture modules.

H_o^J :- Domain experience affects the overall performance of the team.

H_o^K :- Lack of availability of customer makes communication a barrier in agile distributed environment.

H_o^L :- The higher the resistance of customers, the higher the level of inconsistent modules of architecture.

H_o^M :- Lack of domain experience of customer makes understandability difficult.

H_o^N :- The more complex the size of the system, the higher the performance of application.

H_o^O :- The higher the complexity of system, the higher the efficiency of application.

H_o^P :- Lack of situational approaches affect the deployment of system in agile distributed context.

H_o^Q :- Incorporation of technology elevates the mapping and traceability of system.

H_o^R :- Reusability of software system components makes application development easy in agile distributed context.

In order to evaluate the directed hypotheses, the following list of questions was derived. The questions followed guidelines as identified by Krosnick [41]. The list of directed hypotheses and questions were circulated worldwide through online survey. Experts from different countries participated in the survey. The experts were contacted through LinkedIn and snowballing technique [42]. The distribution of participating respondents is shown in Fig. 11.

FIGURE 11. Respondents.

A. To what degree does requirements stability affect the business aspect of architecture?
B. Rate how much standards do contribute to the successful business value of software architecture.
C. How much does the size of organization contribute towards the complexity of software architecture?
D. To what rate does commitment to the standards affect successful architecture deployment?
E. To what rate does lack of maturity of organization lead to the missing of situational factors in agile distributed environment?
F. How likely is it that inconsistent structure of organization leads to agile development in distributed environment?
G. Rate how necessary a certain level of management of software architecture is to fulfilling organization standards.
H. How much is geographic distribution likely to result in situational variations?
I. Rate whether lack of technical dependencies within team leads to inconsistent architecture modules.
J. Rate how much domain experience does affect the overall performance of the team.
K. How much does lack of availability of customer make communication a barrier in agile distributed environment?
L. How much does higher resistance of customers lead to higher level of inconsistent modules of architecture?
M. How likely is that lack of domain experience of customer makes understandability difficult?
N. Rate how much the complexity of system size affects the performance of application.
O. How much does higher complexity of system make the application more efficient?
P. To what level does lack of situational approaches affect the deployment of system in agile distributed context?
Q. Rate how much the incorporation of technology elevates the mapping and traceability of system.
R. How much does the reusability of software system components make application development easy in agile distributed context?

Total 18 respondents both from agile distributed software development and situational agile distributed development,

TABLE 5. Statistical analysis of given responses.

Pr.	*T*	Mean	Min/ Max	Med	M.rank	S.D	*U*	*p*	*Z*	Level of confidence for directed hypothesis
H_o^A : Requirements Stability	AD	2.6	1/4	3	2.30	1.10	8.5	0.50	0.88	80.96%
	SAD	3.2	2/4	3	3.10	0.74				
H_o^B : Contribution of Standards	AD	2.2	1/3	2	2.04	0.75	9.5	0.63	0.66	74.45%
	SAD	2.6	1/4	3	2.30	1.01				
H_o^C : Size of organization	AD	2.6	2/3	3	2.50	0.48	7.0	0.29	1.25	89.39%
	SAD	3.2	2/4	3	3.10	0.74				
H_o^D : Commitment to standards	AD	1.8	1/3	2	1.64	0.74	10.5	0.75	0.45	67.49%
	SAD	2.0	1/3	2	1.80	0.63				
H_o^E : Organization Maturity	AD	1.8	1/3	2	1.64	0.75	5.0	0.14	1.64	94.98%
	SAD	2.8	2/4	3	2.70	0.75				
H_o^F : Structure	AD	2.4	2/3	2	2.30	0.48	12.0	1.00	0.10	54.36%
	SAD	2.4	1/4	3	2.10	1.20				
H_o^G: Geographic distribution	AD	3.4	3/4	3	3.30	0.48	10.0	0.67	0.60	72.57%
	SAD	3.6	3/4	4	3.50	0.49				
H_o^H : Level of Management	AD	1.8	1/2	2	1.70	0.40	8.5	0.46	0.95	82.86%
	SAD	2.2	1/3	2	2.04	0.74				
H_o^I : Technical Dependencies	AD	2.2	1/3	2	2.04	0.75	5.0	0.14	1.64	94.98%
	SAD	3.2	2/4	3	3.10	0.75				
H_o^J : Domain Experience	AD	3.6	3/4	4	3.50	0.49	9.0	0.52	0.81	79.05%
	SAD	3.0	2/4	3	2.80	0.89				
H_o^K : Customer Availability	AD	3.0	2/4	3	2.80	0.63	12.0	1.00	0.11	54.36%
	SAD	2.8	1/4	3	2.40	1.16				

TABLE 5. *(Continued.)* **Statistical analysis of given responses.**

H_o^L : Customer Resistance	AD	2.4	2/3	2	2.30	0.48	7.0	0.29	1.25	89.39%
	SAD	1.8	1/3	2	1.60	0.74				
H_o^M : Customer Experience	AD	2.2	1/3	2	2.10	0.75	5.0	0.14	1.64	94.98%
	SAD	3.2	2/4	3	3.10	0.74				
H_o^N : System Size	AD	3.2	2/4	3	3.10	0.75	12.5	0.92	0.10	50.00%
	SAD	3.2	2/4	3	3.10	0.75				
H_o^O : System Complexity	AD	2.2	1/3	3	1.90	0.97	6.0	0.21	1.46	92.77%
	SAD	3.2	2/4	3	3.10	0.74				
H_o^P : Situational Approaches	AD	1.6	1/2	2	1.50	0.48	13.5	0.12	2.69	99.65%
	SAD	3.6	3/4	4	3.50	0.75				
H_o^Q : Technology Incorporation	AD	2.0	1/3	2	1.80	0.63	7.5	0.34	1.09	86.17%
	SAD	2.8	1/4	3	2.40	1.16				
H_o^R : Component Reusability	AD	2.6	2/3	3	2.50	0.48	7.0	0.29	1.25	89.39%
	SAD	3.2	2/4	3	3.10	0.74				

were collected. In order to normalize data, responses were recorded on a rating scale of 1–5. Another reason for normalizing the responses was to make the data suitable for statistical analysis. Statistical analysis is explained later in this section. The most common method used for statistical analysis in software development across multiple tests of the directed hypotheses is Fisher's combined probability test. On the other hand, an alternative method called the weighted Z-test has more power and more precision than does Fisher's test. Further, in contrast to some statements in the literature, the weighted Z-test is superior to the un-weighted Z-transform approach. The results show that, when P-values from multiple tests of the directed hypotheses are combined, the weighted Z-method should be preferred [43].

Normalized responses extracted from the questionnaires were statistically analyzed in order to prove the proposed directed hypothesis. Statistical analysis was performed online; Z-test, U-test, and probability of each variable were also calculated in order to confirm the level of confidence for each directed hypothesis, as shown in Table 5. In Table 5, AD represents agile development, whereas SAD represents situational agile development. A total of 26 practitioners took part in interviews and gave their responses on the defined rating scale. The practitioners belonged under both agile distributed software development and situational agile software development. To conduct statistical analysis, mean, min/max, median, and mean rank were calculated. Further, standard deviation was calculated, and U-test was performed. In the end, the probability was checked on the basis of significance value. A significance value of 0.05 was set for comparing the probability. Cultural variation and instant work were not accepted because their probability was less than the significance value; therefore, an alternate hypothesis was selected for both variables. Z-test was also calculated in order to check the level of confidence of the directed hypothesis [43], [44]. The reason for generating directed hypothesis was to show the relation between dependent and independent variables. With respect to our approach the 18 identified situational factors were independent variables, however, the software architecture and agile distributed software development are the dependent variable. After all levels of confidence for the directed hypothesis were calculated, the results are presented graphically, as shown in Fig. 12.

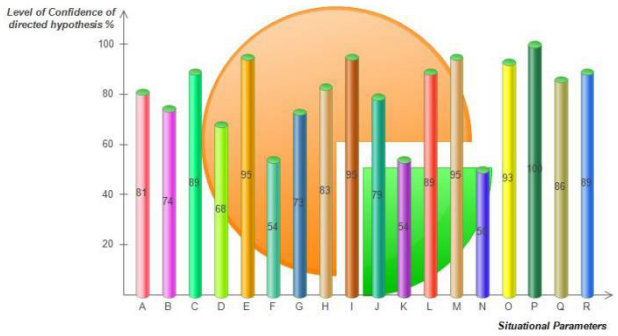

FIGURE 12. Graph against level of confidence for direct hypothesis.

The level of confidence for each directed hypothesis is shown in Fig. 12. Each bar in the figure represents a situational parameter. The highest level of confidence is for situational approaches. This means that when situational variation in agile distributed development is not considered, the level of confidence would be less satisfactory. Our contribution emphasized the identification of situational factors highlighted in literature. Similarly, the effect of situational factors on SADD architecture was also handled. Protocol review was evaluated from practitioners, and questionnaires were validated. The questionnaire was designed by the authors and validated by the experts. Validation process involved the understanding the problem and generating questions relevant to the problem. Further taxonomy was generated on the basis of identified situational factors.

C. IMPLICATION OF THE STUDY

In this section, we have answered our third research questions. The study provides a state-of-the-art overview of situational requirements as barrier in an agile distributed software development environment. This study provides a taxonomical classification of situational factors. It also proposes a SADD model of situational barriers, which presents the key categories of the situational barriers that can serve as knowledge for academics, researchers, and practitioners working on situational method engineering and software architecture in the agile distributed software development. This model will assist distributed software development firms into paying more attention on the situational barriers with respect to software architecture. Moreover, this research work provides a deep understanding of the situational and contextual barriers in relation to software architecture and on how to achieve harmonization between both sides. The reported situational barriers can assist the practitioners into considering the most relevant situational barriers with respect to their context. In summary, this study provides a detailed overview of a literature survey of the available situational factors in the context of an agile distributed software development that harmonizes between situational factors and software architecture, which has not been conducted before. Finally, this study contributes to the development of software architecture in agile development, which assists distributed software development organizations in assessing and improving their software development programs effectively.

D. THREATS AND VALIDITY

We mapped the following threats as internal, external, and reliability threats.

1) RELIABILITY

In this study, we used literature review approach in order to investigate the situational barriers in agile distributed software development. This may be a threat towards the validity of the investigated situational barriers. This threat has been addressed by using the systematic literature review approach.

In addition, among responses gathered through online survey, only 18 were complete. This may be considered as a small sample size. However, through the referencing of existing empirical studies, the sample of the present study is enough to justify the results of empirical study. Similarly, an informal method was adopted to categorize the investigated situational barriers into five situational aspects. This may be a threat towards the validity of the situational-barrier categorization process. However, most of the researchers of other domains of agile distributed software development also adopted the same process in order to classify the identified situational factors/parameters into different categories [26], [29].

2) INTERNAL VALIDITY

Internal validity is related to the cause and effect relationship. An internal threat is present on the basis of the results generated by using published case study. We spent a substantial amount of time in identifying situational needs and factors associated with the published case study. The questionnaire results were bound to have biases in opinions, current knowledge, and attitudes of the respondents, and generate a very narrow viewpoint as a whole. Multiple backgrounds, domain knowledge, and levels of expertise of the respondents with regard to ADSD are also likely to affect their adoption and understanding of SADD.

3) EXTERNAL VALIDITY

Another threat related to our study is the external threat. This threat is amplified when inaccurate conclusions are drawn from the literature and applied to published cases with different contexts and settings. The published case data obtained might have lost valuable information and might not be sufficient for making any assumption or recommendation. However, we feel the SADD model can be implemented with a good understanding of situational needs.

VI. CONCLUSION

The increasing trend of agile distributed software development paradigm motivated us to investigate the situational factors associated with software architecture. The software architecture is not an initial phase of the software development life cycle; however, it requires more intellection for

producing the quality software. By using systematic literature review approach and snowballing technique, a total of 18 situational factors were identified. Moreover, to validate the key findings of systematic literature survey, taxonomical classification was done and an empirical study (statistical analysis) was conducted. We generated directed hypothesis against each situational factor. In order to validate the hypothesis, questions were generated and circulated worldwide through LinkedIn and online survey. We received a total of 18 complete survey responses. The responses proved that situational approach has the greatest impact on the agile distributed software development. The responses showed comparison between traditional and ADSD domain considering the same situations. Moreover, the situational hypothesis "F," i.e., structure of the organization, has the minimum importance. Furthermore, we generated five aspects of situational factors on the basis of taxonomical classification. First, we identified the situational barriers and proposed a solution in the form of taxonomical classification. Afterward, we proposed a SADD model that incorporated situational enhancement approach and harmonization with software architecture in ADSD domain. Further, we proposed five aspects of situational factors on the basis of SADD model and taxonomical classification. To support our findings, statistical analysis was performed. Z-test and probability were also calculated. We believe that the findings of the current study are helpful for addressing the challenges faced by ADSD organizations in the software architecture development process. It was also evident from the case study that software architecture is especially recommended when working in different situations. Similarly HoC resulted in 89.39% of Level of contribution, thus it was clear that situational factor 'size of organization' contributes 89.39% to software to architecture.

FUTURE WORK

The basic motive of this study is to develop a situational taxonomic model in the context of agile distributed software development. The proposed SADD model is based on existing literature studies. The situational awareness of the SADD model is based on perception and integration, which may lead to biases if not handled properly and deeply. The present study contributed in the area of situational factors and harmonization of those situational factors in software architecture. However, in the future, we plan to conduct a systematic mapping review and an empirical study in order to investigate the additional situational factors, which affect the complete development of software, in terms of design and implementation in the ADSD domain. In addition, we also plan to conduct a systematic mapping study in order to identify the success factors and best practices of software architecture in the ADSD domain, which may be useful in addressing the software components for situational method engineering. We believe that the proposed SADD model is useful for assessing and managing the situational factors in the ADSD environment.

REFERENCES

[1] 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, Jan. 2019, Art. no. e1988.

[2] A. S. Hashmi, Y. Hafeez, M. Jamal, S. Ali, and N. Iqbal, "Role of situational agile distributed model to support modern software development teams," *Mehran Univ. Res. J. Eng. Technol.*, vol. 38, no. 3, pp. 655–666, 2019.

[3] M. Shameem, C. Kumar, B. Chandra, and A. A. Khan, "Systematic review of success factors for scaling agile methods in global software development environment: A client-vendor perspective," in *Proc. 24th Asia–Pacific Softw. Eng. Conf. Workshops (APSECW)*, Dec. 2017, pp. 17–24.

[4] M. Shameem, C. Kumar, and B. Chandra, "Challenges of management in the operation of virtual software development teams: A systematic literature review," in *Proc. 4th Int. Conf. Adv. Comput. Commun. Syst. (ICACCS)*, Jan. 2017, pp. 1–8.

[5] A. Khan, C. Kumar, M. Shameem, and B. Chandra, "Impact of requirements volatility and flexible management on GSD project success: A study based on the dimensions of requirements volatility," *Int. J. Agile Syst. Manage.*, vol. 12, no. 4, pp. 199–227, 2019.

[6] A. Yagüe, J. Garbajosa, J. Díz, and E. González, "An exploratory study in communication in agile global software development," *Comput. Standards Inter.*, vol. 48, pp. 184–197, Nov. 2016.

[7] B. Murphy, C. Bird, T. Zimmermann, L. Williams, N. Nagappan, and A. Begel, "Have agile techniques been the silver bullet for software development at microsoft?" in *Proc. ACM / IEEE Int. Symp. Empirical Softw. Eng. Meas.*, Oct. 2013, pp. 75–84.

[8] S. Fraser and D. Manci, "No silver bullet reloaded: Report on XP 2017 panel session," *SIGSOFT Softw. Eng. Notes*, vol. 43, no. 4, p. 53, Jan. 2019.

[9] K. Suryaatmaja, D. Wibisono, and A. Ghazali, "The missing framework for adaptation of agile software development projects," in *Eurasian Business Perspectives*. Cham, Switzerland: Springer, 2019, pp. 113–127.

[10] O. Sievi-Korte, S. Beecham, and I. Richardson, "Challenges and recommended practices for software architecting in global software development," *Inf. Softw. Technol.*, vol. 106, pp. 234–253, Feb. 2019.

[11] A. A. Alsanad, A. Chikh, and A. Mirza, "A domain ontology for software requirements change management in global software development environment," *IEEE Access*, vol. 7, pp. 49352–49361, 2019.

[12] G. Rong, B. Boehm, M. Kuhrmann, E. Tian, S. Lian, and I. Richardson, "Towards context-specific software process selection, tailoring, and composition," in *Proc. Int. Conf. Softw. System Process (ICSSP)*, 2014, pp. 183–184.

[13] L. Przybilla, M. Wiesche, and H. Krcmar, "The influence of agile practices on performance in software engineering teams: A subgroup perspective," in *Proc. ACM SIGMIS Conf. Comput. People Res. (SIGMIS-CPR)*, 2018, pp. 33–40.

[14] B. Kitchenham, R. Pretorius, D. Budgen, O. P. Brereton, M. Turner, and M. Niazi, "Systematic literature reviews in software engineering—A tertiary study," *Inf. Softw. Technol.*, vol. 52, pp. 792–805, Aug. 2010.

[15] S. D. Vishnubhotla, E. Mendes, and L. Lundberg, "An insight into the capabilities of professionals and teams in agile software development: A systematic literature review," in *Proc. 7th Int. Conf. Softw. Comput. Appl. (ICSCA)*, 2018, pp. 10–19.

[16] P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, and M. Khalil, "Lessons from applying the systematic literature review process within the software engineering domain," *J. Syst. Softw.*, vol. 80, no. 4, pp. 571–583, Apr. 2007.

[17] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," in *Ease*, vol. 8, pp. 68–77, Jun. 2008.

[18] R. V. O'connor, P. Elger, and P. M. Clarke, "Continuous software engineering—A microservices architecture perspective," *J. Softw. Evol. Proc*, vol. 29, no. 11, Nov. 2017, Art. no. e1866.

[19] Y. I. Alzoubi, A. Q. Gill, and B. Moulton, "A measurement model to analyze the effect of agile enterprise architecture on geographically distributed agile development," *J. Softw. Eng. Res. Develop.*, vol. 6, p. 4, Mar. 2018.

[20] R. Hoda, N. Salleh, and J. Grundy, "The rise and evolution of agile software development," *IEEE Softw.*, vol. 35, no. 5, pp. 58–63, Sep. 2018.

[21] J. Tripp, J. Saltz, and D. Turk, "Thoughts on current and future research on agile and lean: Ensuring relevance and rigor," in *Proc. 51st Hawaii Int. Conf. Syst. Sci.*, 2018.

[22] E. H. Trainer and D. F. Redmiles, "Bridging the gap between awareness and trust in globally distributed software teams," *J. Syst. Softw.*, vol. 144, pp. 328–341, Oct. 2018.

[23] G. Giray, M. Yilmaz, R. V. O'Connor, and P. M. Clarke, "The impact of situational context on software process: A case study of a very small-sized company in the online advertising domain," in *Systems, Software and Services Process Improvement* (Communications in Computer and Information Science). Cham, Switzerland: Springer, 2018, pp. 28–39.
[24] R. V. O'connor, P. Elger, and P. M. Clarke, "Exploring the impact of situational context: A case study of a software development process for a microservices architecture," in *Proc. Int. Workshop Softw. Syst. Process. (ICSSP)*, 2016, pp. 6–10.
[25] P. Clarke and R. V. O'Connor, "Changing situational contexts present a constant challenge to software developers," in *Systems, Software and Services Process Improvement* (Communications in Computer and Information Science). Cham, Switzerland: Springer, 2015, pp. 100–111.
[26] P. Clarke and R. V. O'Connor, "The situational factors that affect the software development process: Towards a comprehensive reference framework," *Inf. Softw. Technol.*, vol. 54, no. 5, pp. 433–447, May 2012.
[27] M. Razavian, B. Paech, and A. Tang, "Empirical research for software architecture decision making: An analysis," *J. Syst. Softw.*, vol. 149, pp. 360–381, Mar. 2019.
[28] G. Borrego, A. L. Morán, R. R. Palacio, A. Vizcaíno, and F. O. García, "Towards a reduction in architectural knowledge vaporization during agile global software development," *Inf. Softw. Technol.*, vol. 112, pp. 68–82, Aug. 2019.
[29] S. Dhir, D. Kumar, and V. Singh, "Success and failure factors that impact on project implementation using agile software development methodology," in *Software Engineering*. Singapore: Springer, 2019, pp. 647–654.
[30] G. Giray and B. Tekinerdogan, "Situational method engineering for constructing Internet of Things development methods," in *Proc. Int. Symp. Bus. Model. Softw. Design*, 2018, pp. 221–239.
[31] J. Jia, H. Mo, L. F. Capretz, and Z. Chen, "Grouping environmental factors influencing individual decision-making behavior in software projects: A cluster analysis," *J. Softw., Evol. Process*, vol. 30, Jan. 2018, Art. no. e1913.
[32] C. Coulin, D. Zowghi, and A.-E.-K. Sahraoui, "A situational method engineering approach to requirements elicitation workshops in the software development process," *Softw. Process, Improve. Pract.*, vol. 11, no. 5, pp. 451–464, Sep. 2006.
[33] H. Munir, K. Wnuk, and P. Runeson, "Open innovation in software engineering: A systematic mapping study," *Empir Softw. Eng*, vol. 21, no. 2, pp. 684–723, Apr. 2016.
[34] M. Usman, R. Britto, J. Börstler, and E. Mendes, "Taxonomies in software engineering: A systematic mapping study and a revised taxonomy development method," *Inf. Softw. Technol.*, vol. 85, pp. 43–59, May 2017.
[35] A. A. Zafar, S. Saif, M. Khan, J. Iqbal, A. Akhunzada, and A. Wadood, "Taxonomy of factors causing integration failure during global software development," *IEEE Access*, vol. 6, pp. 22228–22239, 2018.
[36] M.-L. Ryan, "Introduction: On the why, what and how of generic taxonomy," *Poetics*, vol. 10, pp. 109–126, 1981.
[37] A. Aharoni and I. Reinhartz-Berger, "A domain engineering approach for situational method engineering," in *Proc. Int. Conf. Conceptual Modeling*, 2008, pp. 455–468.
[38] P. J. R. De Jongh, J. Larney, E. Mare, G. W. Van Vuuren, and T. Verster, "A proposed best practice model validation framework for banks," *South Afr. J. Econ. Manage. Sci.*, vol. 20, pp. 1–15, Jun. 2017.
[39] G. Marks, R. V. O'Connor, and P. M. Clarke, "The impact of situational context on the software development process-a case study of a highly innovative start-up organization," in *Proc. Int. Conf. Softw. Process Improvement Capability Determination*, 2017, pp. 455–466.
[40] M. Farwick, C. M. Schweda, R. Breu, and I. Hanschke, "A situational method for semi-automated enterprise architecture documentation," *Softw. Syst. Model.*, vol. 15, no. 2, pp. 397–426, May 2016.
[41] J. A. Krosnick, "Questionnaire design," in *Institute of Mathematical Statistics*. Cham, Switzerland: Palgrave Macmillan, 2018, pp. 439–455.
[42] W. I. King, "The annals of mathematical statistics," *Ann. Math. Statist.*, vol. 1, no. 1, pp. 1–2, Feb. 1930.
[43] M. C. Whitlock, "Combining probability from independent tests: The weighted Z-method is superior to Fisher's approach," *J. Evol. Biol.*, vol. 18, no. 5, pp. 1368–1373, Aug. 2005.
[44] G. Sun, H. Zhang, J. Fang, G. Li, and Q. Li, "A new multi-objective discrete robust optimization algorithm for engineering design," *Appl. Math. Model.*, vol. 53, pp. 602–621, Jan. 2018.

AMBER SARWAR is currently pursuing the Ph.D. degree with the University Institute of Information Technology, PMAS-Arid Agriculture University Rawalpindi, Pakistan.

YASER HAFEEZ received the Ph.D. degree from International Islamic University Islamabad, Pakistan. He is currently working as an Associate Professor with the University Institute of Information Technology, PMAS-Arid Agriculture University Rawalpindi, Pakistan. His research interests include software engineering and knowledge management.

SHARIQ HUSSAIN received the master's degree in computer science from PMAS Arid Agriculture University, Rawalpindi, Pakistan, in 2007, and the Ph.D. degree in applied computer technology from the University of Science and Technology Beijing, Beijing, China, in 2014. Since 2014, he has been with the Department of Software Engineering, Foundation University Islamabad, Rawalpindi Campus, where he is currently an Assistant Professor. His main research interests include web services, QoS in web services, web service testing, the IoT, context awareness, and e-learning.

SHUNKUN YANG received the B.S., M.S., and Ph.D. degrees from the School of Reliability and Systems Engineering, Beihang University, in 2000, 2003, and 2011, respectively. He was an Associate Research Scientist with Columbia University, from September 2014 to September 2015. He has been an Associate Research Professor with Beihang University, since 2016. His main research interests include reliability, testing and diagnosis for embedded software, CPS, the IoT, and intelligent manufacturing.