Hindawi
Scientific Programming
Volume 2022, Article ID 8312253, 19 pages
https://doi.org/10.1155/2022/8312253

Research Article

A Critique on Task Allocation Processes in Distributed Agile Software Development

Farah Ijaz,[1] Waqar Aslam,[1] Abeer Abdulaziz AlSanad,[2] Zahid Aslam,[1] Insaf Ullah,[3] and Fazl ullah[4]

[1]*Department of Computer Science and Information Technology, The Islamia University of Bahawalpur, Bahawalpur, Pakistan*
[2]*Information Systems Department, College of Computer and Information Sciences, Imam Mohammad Ibn Saud Islamic University, Riyadh 11432, Saudi Arabia*
[3]*Hamdard Institute of Engineering and Technology, Hamdard University, Islamabad 44000, Pakistan*
[4]*Department of Information Technology, Khana-e-Noor University, Pol-e-Mahmood Khan, Shashdarak, Kabul, Afghanistan*

Correspondence should be addressed to Waqar Aslam; waqar.aslam@iub.edu.pk and Fazl ullah; fazlullahumar@gmail.com

Received 9 April 2022; Accepted 11 May 2022; Published 28 May 2022

Academic Editor: Sikandar Ali

Distributed agile software development is implemented to tackle the increasing competition and changing requirements of the software industry. It inspires the development of emergent software applications and supports user satisfaction with fast delivery to market at low cost. Distributed software development presents a complex distributed team structure of the software industry having various workplaces. Distributed agile software development requires effective and well-organized decisions during task allocation. Task allocation is a challenging activity across the spectrum of committed software developers in a distributed environment. Effective task allocation under the pressure of achieving project objectives is a complex management problem in distributed agile software development. Task allocation process is proposed to support effective task allocation in distributed agile software development. Firstly, we identify different aspects of the task allocation context to explore the basic requirements of decision-making in DASD. Based on the information related to the identified aspects, different types of process activities are defined; those should be fulfilled to support the task allocation decisions. Task allocation process during software project planning requires comprehensive information about various factors that influence decision-making. The proposed task allocation process is evaluated based on the process activities. None of the current state-of-the-art research works focus on an approach for task allocation that completely fulfills all activities related to high-level processes or low-level processes. We fill this gap by identifying, classifying, sequencing, and proposing these activities that can be used to realize a methodical approach for a generic task allocation process in a distributed agile software development setup.

1. Introduction

Distributed software development (DSD) offers different ways to reach a larger pool of experts, optimize costs, and decrease time to delivery. Challenges encountered in DSD and the distance between employees explicitly disturb communications, coordination, and control. Agile software development (Scrum) has been used since the early 1990s for the development of complex software products and has also been adopted to support the management of distributed projects in distributed software [1].

Agile software development (ASD) offers a positive response to expected or unexpected changes, is flexible and rapid to adapt to changes at any time, is lean in terms of minimizing cost without affecting the quality, and concentrates on improvement and innovation [2]. According to evidence, ASD also offers improved results compared to traditional methods in DSD, which presents distributed agile software development (DASD). However, when agile is followed in DSD, a main challenge is to decide how to grip the higher complexity of distribution and required smart evaluation of organizational factors and agile principles [3].

Software project is a people-intensive activity, and its related resources are mainly human resources. Different project tasks require different skills with better level of proficiency of development team members. Best fitted task to member allocation is a very challenging responsibility for a software project manager [4]. Quality, productivity, and motivation are negatively affected by the low transparent and unjustified mode of task allocation [5, 6]. More than 40% of projects failed in China due to ineffective arrangements related to human resources and tasks of software projects [4]. Task to member allocation decision can control cost, time, and quality concerns of software projects [6]. Task allocation is performed based on the ability and availability; that decision is taken by the project manager or Scrum master [7]. Task allocation becomes a more critical activity due to limited understanding of the factors that affect the consequences of task allocation. Decision about the distribution of tasks to remote teams is a main challenging action for the successful development in DSD [8]. Normally, DSD teams have coordination issues due to which they may take 2.5 times longer to complete tasks than collocated teams. Coordination requires managing all dependencies for successful project development [9].

Sometimes, different types of allocation strategies and objectives conflict with each other when there is a lack of knowledge about the activities required to complete an effective task allocation process in DASD. According to the complexity of DASD, a complete process must be properly defined with supporting activities to handle the complexity of DASD.

Here, we remark on considering extra functional properties in software projects through composability. Composability for extra functional properties may be classified in terms of the composition types such as directly composable properties, architecture related properties, derived properties, usage dependent properties, and system environment context properties [10]. Software architecture should be described with better usability of architectural language [11]. Technical architect can support architectural design decisions in DASD [12].

We proposed a task allocation process to support DASD, based on the identified information related to the aspects of task allocation context in DASD. This work is based on selecting publications in which models, methods, approaches, frameworks, and algorithms for task allocation in different software development setups (DSD, ASD, and DASD) were presented. These publications present factors, dependencies, roles, capabilities, and challenges/risks that influence the task allocation in DASD.

The research question addressed in this paper is as follows:

(Q) Which activities should be fulfilled for a complete and effective task allocation process in DASD?

The main objectives of this work are as follows:

(i) To evaluate the existing research works

(ii) To identify the existing methods, processes, models, approaches, and frameworks that support task allocation in DASD

(iii) To define a complete task allocation process in the context of DASD

2. Software Development Process and Contemporary Approaches

2.1. Distributed Software Development

2.1.1. Model. Based on the extension of Bokhari's algorithm using Bayesian networks, a decision support model is presented. This model supports multiple objectives and characteristics of DSD, and a number of scenarios are simulated with random costs [13].

A decision support model using a genetic algorithm for task allocation is presented. Task, resources, objectives, and different parameters are used as main entities. Based on a genetic algorithm, a chromosome is formed for dual fragments, initially the information about the allocation of resources to separate tasks and further the information about the allocation of timelines to individual task resource combination [14].

The virtual team effectiveness model (adapted from Cohen's self-managing work team model) is presented for distributed teams (virtual teams). A case study of six distributed teams is conducted, and the impact of task design, team characteristics, organizational context, and team process on the team performance is described [15].

Multicriteria model for task allocation and MCDA (multicriteria decision analysis)-based decision support system are proposed to support distributed team of the global software industry. Implementation of the model improved the efficiency of decision-making, and this model is acknowledged by the company as part of the standard project initiation procedure [16].

The global teaming model is proposed to support global software development. The mapping of uncertainty management approach is presented. Different types of recommendations are presented to resolve ambiguity in GSD [17].

2.1.2. Method. The application of the verbal decision analysis method ZAPROS III-i is presented to support the sorting of factors related to DSD. This work supported task allocation and improved the performance of the tasks in a fast as well as practical way. ARANAÚ tool (for graphical support, used to offer the graph presenting the dominance relations between the alternatives and used to support the ZAPROS III-i methodology) is used [18].

A hybrid methodology based on methods of verbal decision analysis (VDA) is presented to support task allocation activity in DSD. ORCLASSWEB tool (to automate the comparison process of alternatives and provide the result for a problem to support the decision-making) and ARANAÚ tool are used [19].

2.1.3. Framework. Framework for the allocation of work packages between team members is presented to support GSD. Qualitative study through Delphi study is done for the identification of a single metric for each factor. According to the results, after implementation of the proposed framework, higher software quality is achieved [20].

Framework is presented for task allocation in global software development to help decision-makers control delay and reallocation. Framework is validated with different members from around the world [21].

2.1.4. Algorithm. Task allocation is presented for software crowdsourcing a collaborative development with the solution for a multitask to multiworker allocation. A series of algorithms are proposed, and experimental results prove the effectiveness of the offered solution [22].

A dynamic utility task allocation algorithm is proposed for software crowdsourcing development. Kuhn–Munkres algorithm with a weighted bipartite graph is used for optimal matching of tasks to members. According to the results, better performance in total allocation utility with an improved level of success is achieved [23].

To solve heterogeneous task allocation issues in the crowd intelligence software development system, three algorithms are proposed: GANs (generative adversarial networks) for text generation to generate decision data and amplify data features, the Baum–Welch algorithm for obtaining model parameters, and the Viterbi algorithm for obtaining optimal task allocation strategy. Based on the Agile Manager game platform (simulated platform), this approach is evaluated experimentally [24].

2.1.5. Qualitative Research. Qualitative study is presented to identify the different criteria that are applied to task allocation in DSD. Applied criteria are identified through the interviews of project managers from different software development companies. QSR NVivo (qualitative data analysis tool) is used for data analysis [25].

2.1.6. Systematic Literature Review. Systematic literature review (SLR) is presented to identify multicriteria models, decision-making models, and approaches for task allocation in DSD. Different types of used tools are described, and future work is suggested based on the results of SLR [26].

Systematic mapping is presented to discover challenges along with possible solutions related to GSD. The StArt tool (to conduct a staggering selection and filtering process of the publications) is used. Research work is classified according to the COBIT (control objectives for information and related technologies) framework. Mapping by COBIT governance objectives and mapping by COBIT governance components are done [27].

Coordination requirements of teams in DSD are investigated, concentrating on the role of team knowledge in the coordination. According to the findings, shared knowledge of the team and shared task knowledge are very important to resolve coordination issues [28].

A systematic literature review and survey questionnaire are performed for the identification of different factors of DSD. A conceptual cost estimation model is also presented based on these factors [29].

2.1.7. Survey. Web-based survey is conducted with 54 participants from global software development. Different factors are identified, and their relative importance is described in the task allocation context. This study also explores the value of situation-based decision-making throughout task allocation [30].

Thorough survey about the global virtual team performance is presented. According to the survey results, team trust and team cohesion have a relationship with a reciprocal effect on each other. AMOS 19.0 (statistical software for analysis of a moment structure, used for structural equation modeling) is used [31].

2.1.8. Grounded Theory. Grounded theory is presented to explore the impact of software tools on the performance of a distributed team. Fifteen distributed teams are observed and interviewed over two years. Different software tools are recommended to solve issues related to the performance of virtual project teams and enhance the performance of virtual teams in a distributed environment [32].

2.1.9. Task Allocation Strategy. The task allocation strategy for DSD is presented based on the information about system architecture and external factors with a traceability perspective. According to this study, a strategy for task allocation should consider important factors to reduce communication and coordination overhead and time delay [33].

2.1.10. Ontology. Ontology is proposed to represent concepts about task allocation in DSD and provides an understanding of different factors. Three perspectives according to an organizational structure, namely, product perspective, functional perspective, and project perspective, are presented, and relationships between these perspectives are also described. StarUML tool (to design the graphical model that illustrates the proposed ontology) is used [34].

2.1.11. Taxonomy. Taxonomy is presented to explore the integration failure factors which cause the project failure in DSD. Nonoptimal task allocation for distributed teams, inadequate task dependencies related to systemic properties and ineffective modularization of work, and many other factors that can cause project failure are described [35].

2.1.12. Task Coordination Portfolios. Optimal IT-mediated task coordination portfolios are proposed to support the various levels of task dependencies, temporal dispersion, and perceived time constraints of DSD. They are tested through a survey and analysis of 95 globally distributed teams towards their performance enhancement. This research contributes to both theory and practice [9].

2.1.13. Software Tool. TAMRI (tool) is presented for task allocation, which is based on distributed systems method and Bayesian networks. It supports multiple criteria to

improve weighted objectives. Summary of task distribution approaches, implementation of the tool, and three application scenarios are also presented in this paper [36].

2.2. Agile Software Development

2.2.1. Model. Mathematical optimization model for ASD is presented, and algorithms using Bayesian networks are proposed to support agile iteration scheduling. The quality of planning in ASD is improved by considering planning factors such as dependencies and capacities of resources [37].

A prediction model for project performance is proposed, reflecting the impact of leadership and organization factors on the agility and flexibility of the organization. A framework is proposed to obtain the information (about leadership factors, agility factors, flexibility factors, organizational factors, and project performance) that is the basis for the proposed model. Artificial intelligence techniques (radial basis function method and Bayesian networks) and statistical tools are used. Proper understanding is offered as to how to achieve the highest project performance through the model application [38].

2.2.2. Method. Method is presented for the individual capability calculation of every team member. 18 parameters are presented with criteria, values, and weight factors for weighing individual capability. The method is validated in a small ASD company [39].

Roles of the software development team are defined based on the personality types. Myers–Briggs Type Indicator (MBTI) is used to measure the personality types of software development team participants. Rules for the project manager and programmer are defined for team composition. Results are validated with different classification techniques and offer better accuracy [40].

2.2.3. Framework. Framework is presented for capability-centric web tool; this tool helps keep track of capabilities for team composition and task allocation schedules [41].

2.2.4. Algorithm. Algorithmic method which considers different factors is presented for the estimation of cost, time, and effort. Three cases in which levels of factors are different are presented to show the effectiveness of the presented method. According to these three cases, the estimated values are changed due to changes in levels of factors [42].

2.2.5. Qualitative Research. Individual and team capability measures are identified through qualitative research and used to form productive teams. Interviews were carried out with 14 practitioners in a telecom company; interviewees were employed in a mobile money transfer system developed in Sweden and India [43].

2.2.6. Empirical Research. Exploratory empirical research about requirement engineering in ASD is presented and concentrated on requirements dependencies. A hybrid approach (with agile and plan driven methods) to the architecture is used to support scalability and management of dependencies [44].

2.2.7. Systematic Literature Review. Based on a systematic review of empirical studies, five factors that strongly affect the performance of the software development team are identified. Furthermore, comparing these factors with the Agile Manifesto of ASD team, performance can be improved by focusing on these five factors, namely, team coordination, goal orientation, team cohesion, shared mental models, and team learning [45].

A review of different task allocation approaches adopted in ASD quantitatively is presented. Different types of applied techniques for task allocation along with their strengths and weaknesses are described. Most techniques followed quantitative approaches, but qualitative aspects are not addressed vastly [46].

A systematic mapping study is presented, and factors that affect the productivity in ASD are identified. Mostly identified factors depend on agile teams and affect productivity positively [47].

2.2.8. Survey. An interview survey from various participants of eight organizations is done to determine the onboarding activities of ASD teams. Different types of onboarding objectives are identified in relation to team norms, culture context, job responsibility, understanding of responsibilities of other team members, standard of work, agile techniques, and project domain knowledge. Mapping onboarding techniques with high or very high contributions to support different onboarding objectives are presented [48].

Semi-structured interviews with team members of different agile software development teams are conducted. Four categories of iteration objectives are described and contributed to the software development and project management literature [49].

A survey is conducted on various software companies and concluded that using ASD (Scrum) has a positive impact on software project management [50].

2.2.9. Grounded Theory. Theory including 58 agile experts belonging to 23 software companies in New Zealand and India is presented. Informal, implicit, transient, and spontaneous roles that are used for composing self-organizing ASD teams are identified. These roles are mentor, coordinator, translator, champion, promoter, and terminator. Understanding these roles helps teams and their managers to perform their responsibilities as self-organizing teams [51].

Grounded theory is applied, and 23 practitioners of ASD from 19 teams across thirteen countries are interviewed. Sixteen factors related to the five categories and having an impact on the selection of agile methods are considered.

Various recommendations according to the situations are presented to support the selection of the agile method [52].

2.2.10. Pilot Study. To understand the motivating self-assignment factors in ASD, interviews are conducted with 12 participants of ASD. Three groups of factors, namely, task-based, developer-based, and opinion-based, are identified; according to the analysis, task-based and developer-based factors are more important than opinion-based [53].

2.2.11. Taxonomy. Taxonomy is proposed, and dependency analysis is presented. Taxonomy is used to evaluate which agile practices address dependencies in three projects of ASD [54].

2.2.12. Decision Support System (DSS). Sprint planning decision support system (SPESS tool) for the improvement of sprint planning is presented. SPESS is based on the planning poker and Hungarian algorithm, along with factors such as team member capability level and task dependencies [55].

2.2.13. Controlled Experiment. According to the results of the controlled experiment, software quality improved significantly when participants applied more granular task descriptions in test-driven development [56].

2.2.14. Software Tool. Human-Centered Agile Software Engineering (HASE) online tool is proposed for agile project management. HASE tool is used to undertake data analysis approach for task allocation. The dynamic approach reflects the impact of factors such as competence, equality, workload variation, completion time, and confidence of team members [57].

The software tool BAIS (Bayesian Agile Iteration Scheduling) using Java programming language is developed, used to control schedules, and supports a set of strategies for an arrangement of tasks in ASD [37].

A multi-agent simulation is presented for p value (possible highest value) based on task allocation for the Scrum team in ASD [58].

2.3. Distributed Agile Software Development

2.3.1. Model. The multicriteria decision model is presented based on multicriteria decision analysis (MCDA) for project planning in distributed Scrum software development. Cognitive mapping and MACBETH (measuring attractiveness by a categorical based evaluation technique) are used. Experiments with the model are described and evaluated three plans, and general criteria, global software development criteria, and Scrum criteria are also presented for planning [1].

Architecting in GSD, a conceptual model, is proposed for architectural design. Check lists for practitioners are also presented in table format with a question, rationale, and example answer (considering problems for GSD using Scrum). Findings are related to the importance of architecture design practices and validated through interviews with architects from seven different companies following GSD [59].

Effort estimation model is presented for DASD, based on the different aspects of user story such as size, complexity, quality, novelty, type, and risks. Proposed model is verified through creating different project scenarios used to estimate effort with or without risks [60].

Conceptual model is presented to control scope creep and evaluate the impact of change on the cost, time, quality, stakeholder involvement, and design reworking. Effectiveness of the proposed model is validated through expert judgment and case study [61].

A task allocation model based on supervised machine learning is proposed and implemented for DASD. According to the findings, it supports decision-making and is supportive in terms of task allocation [62].

2.3.2. Method/Approach. Hybrid methodology using the verbal decision analysis methods is described for the classification and ranking of factors to support the DSD. The verbal decision analysis methods support the decision-making process through multicriteria qualitative analysis. Criteria and their values are identified through interviews with experts, and then each factor is characterized with the help of a questionnaire. After this, the ORdinal CLASSification (ORCLASS) method is tracked for the division of factors to be inserted in the preference groups. Finally, the ZAPROS III-i is applied to order the important factors. ORCLASSWEB tool and ARANAÚ tool are used [8].

Situational DASD approach is proposed for the development of a situation-based software architecture framework. Survey and interviews are arranged to understand the effect of situational variations within the DASD context. Experimental results are obtained through practical and statistical analysis [63].

Role-based task allocation is proposed for prediction related to the allocation of an incoming task to a suitable role. The performance of the proposed task allocator is compared with contemporary NN (neural network) and ML (machine learning) models, and it is better than the compared models with an overall accuracy of 69.3% [64].

2.3.3. Framework. Quantitative framework is presented to support task allocation in DASD. Different types of factors, dependencies, and technical and nontechnical capabilities are identified to support DASD. The role assignment model is also proposed [6].

2.3.4. Case Study. Multi-case study is conducted with nine teams on two projects belonging to two companies, company (A) worked in agile software development and Company (B) worked in V-model development methodology. According to the results, satisfaction of requirements related to expertise coordination and work coordination

influences the team performance. Accordingly, the satisfaction of the team's coordination requirements is essential to achieve desired performance [65].

Case study is conducted using agile and Scrum in a distributed environment. Different types of challenges and recommendations to grip these challenges are presented to support DASD. Jira (software project management web-based tool) and the GoDaddy web hosting are used [66].

A case study is presented in large-scale agile projects, and twenty coordination mechanisms related to synchronization activities and synchronization artifacts are defined. QSR NVivo (qualitative data analysis tool) and tools for communications, such as Slack and Skype, are used [67].

Coordination challenges are identified in DASD within the context of installation and customization of commercial off-the-shelf (COTS) software system. Coordination mechanisms and tools are described to address different types of dependencies with the main coordination challenges [68].

2.3.5. Qualitative Research. The impact of agile practices in low geographical distribution in Austria and Germany is investigated through interviews with experts. The inductive category method is applied to draw a conclusion from collected data. Challenges and recommendations are presented [69].

Empirical research: collaborative tools used by software companies for collaboration are presented in DASD. Collaborative tools are categorized and suggested for coping with three dimensions, namely, communication, workspace, and life cycle, in DASD [70].

2.3.6. Systematic Literature Review. Systematic literature review is done for the identification of factors, and then SWOT (strengths, weaknesses, opportunities, and threats) framework is proposed to support global agile software development. According to the proposed framework, strengths and weaknesses are the internal aspects of the organization, while opportunities and threats are the external aspects of the organization supporting global agile software development [71].

A literature review is conducted to identify practices, issues, and success factors when scaling agile in large organizations. Literature review is used as input to conduct the action research study in a software company that tracked the process of scaling ASD processes [72].

2.3.7. Survey. The relationships between agile enterprise architecture, active communication, and performance are identified. The relation between the active communication dimensions and their effect on the global distributed agile development is described through the theoretical and empirical aspects. The quantitative data analysis approach PLS (partial least squares) is used. According to the results, communication efficiency supports positively the completion of a project within time and budget, and communication effectiveness supports positively functionality and quality [2].

Empirical research is presented, and a survey is done for the validation of relevant hypotheses to address issues of GSD. Software process abilities such as rigor, standardization, agility, and customizability are identified. These abilities cope with the challenge of increasing task environment complexity [73].

Important success factors are identified through a systematic mapping study and validated by a questionnaire survey from experts. Main findings of the study provide information related to the high priority success factors to support agile requirement change management in DASD [74].

2.3.8. Grounded Theory. Grounded theory is conducted, and participants from 38 software companies were interviewed. According to the study, trust between team members in DASD is essential for coping with challenges. Different types of techniques for improving trust between team members are presented to support DASD [75].

2.3.9. Taxonomy. The taxonomy of dependencies is proposed for the DASD environment, and many types of dependencies are identified throughout the software development process. How the task allocation according to the objectives can be changed and how the impact of dependencies is changed due to objective-based task allocation are also described with examples [76].

2.3.10. Decision Support System (DSS). Decision support system for risk management in DSD is proposed, and guidelines can be retrieved through the proposed tool. Impact of different factors on decision-making and risk management strategies to handle identified risk factors are presented. Furthermore, the system can also be extended by adding the feature of task allocation across sites in DSD according to the results of risk assessment [5].

3. Proposed Task Allocation Process for DASD

Critique is conducted to observe the different aspects of task allocation in DASD. Based on the current information related to the task allocation context, various aspects related to task allocation are identified. Figure 1 presents these identified aspects of task allocation context in DASD. These aspects are objectives of software development project, tasks related to the software project development, roles of members in a team, capabilities of members in a team, factors related to the different objects of software development environment, dependencies among the different objects of software development environment, and risks/challenges of software project development.

These aspects explore the crucial information that provides a better understanding related to the requirements of effective decision-making. They provide the foundation concept of our proposed task allocation process. Based on

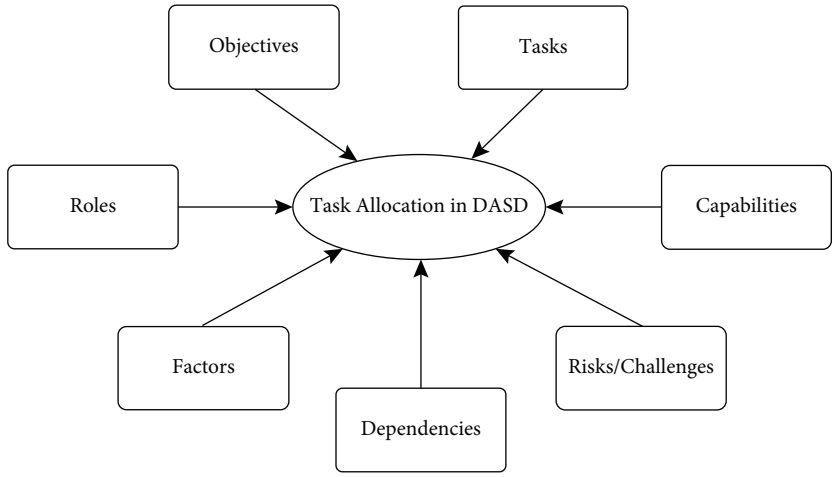

Figure 1: Basic aspects of task allocation context in DASD.

these aspects related to the task allocation context, we identify requirements of different types of activities that must be satisfied for successful task allocation decisions.

Based on these basic aspects, we define multiple process activities to support decision-making throughout task allocation in DASD. We categorized these process activities into different levels: high-level process activities and low-level process activities. The proposed task allocation process is defined in three phases: high-level process activities are considered in phase 1, low-level process activities are considered in phase 2, and task to member allocation is done in phase 3.

Basic aspects of task allocation context in DASD provide input for high-level process activities. They highlight the main activities of concern without details. They provide a very basic and general explanation or presentation of the activities.

High-level process activities provide input for low-level process activities. Low-level process activities provide a detailed description of each activity.

For example, factors included in high-level process activities are identified with a basic concept rather than a detailed explanation, and low-level process activities provide the detailed description of factors with their categories.

(1) High-Level Process Activities (Phase 1). Defining objectives, task identification, capability identification, defining roles, factor identification, dependency identification, and challenge/risk identification are high-level process activities.

(2) Low-Level Process Activities (Phase 2). Task requirements identification, categories of capabilities, categories of factors, categories of dependencies, identifying dependencies between tasks and resolving dependency problems, and identifying challenges/risks of different categories and recommendations are low-level process activities.

(3) Task to Member Allocation (Phase 3). Task to member allocation is the final phase of the proposed process. After completing the high-level process activities and low-level process activities, tasks are allocated to team members.

Figure 2 presents the sequence of the complete task allocation process in the three phases. Low-level process activities depend on the completion of high-level process activities, and task to member allocation depends on the completion of low-level process activities. At the starting phase of the process, high-level process activities are completed; in the second phase, low-level process activities are completed; and then in the final phase, task to member allocation is done.

3.1. High-Level Process Activities (Phase 1). During phase 1 of task allocation process, high-level process activities are completed. High-level activities include defining objectives, task identification, capability identification, defining roles, factor identification, and dependency identification.

3.1.1. Defining Objectives. Objectives throughout the start-up life cycle of software engineering are presented, for example, forming an effective team, validating ideas, balancing customer value with time to market, supporting business needs, external (customer) perceived value with underlining the functionality, internal market potential value, financial value (revenue), good quality, mitigating technology risk, and measuring product performance in the market [77]. Software project managers have to define multiple objectives throughout the software project development plan. These software project objectives are minimizing cost, minimizing defects to support higher-quality products, minimizing the time for completion, maximizing the team member's utilization, and maximizing customer satisfaction [78].

Many objectives are defined in proposed researches, for example, minimizing the development effort [79], minimizing defects [78, 80], minimizing cost [1, 4, 8, 13, 14, 16, 18, 19, 25, 33, 36, 76, 81–85], minimizing time [8, 13, 14, 18, 19, 33, 36, 37, 55, 78, 81, 83, 85], increasing quality [8, 13, 14, 18, 19, 24, 33, 36, 55, 56, 76], timelines of completion of tasks [57], increasing team members' productivity [42, 43], improving work relationship and maximizing project's requirements [83], on-time completion, on-

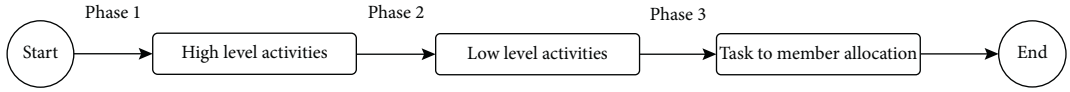

Figure 2: Task allocation process.

budget completion, functionality completion, and satisfying required quality [2].

3.1.2. Task Identification. The software project is comprised of multiple tasks. A task can be defined as the smallest piece of work with clear functionality, such as writing a document, developing a component, and testing the code, or can be any type of required effort during software development [86]. Task allocation process basically relies on task identification. In ASD, user stories are identified, and each user story is decomposed into different tasks [6].

Development tasks are identified in proposed research such as [4, 13, 19, 20, 23, 25, 36, 55, 57, 58, 65, 78, 79, 81, 84, 85].

3.1.3. Defining Roles of Software Development Team. Different types of roles are assigned to people by the project manager or project leader to form a software project development team. According to the project manager, that activity is based on the experience related to people, availability, and required skills [87]. Correctly assigning roles to members of a development team is essential for forming a productive team [88]. Based on the DASD (agile with Scrum method), different types of roles are identified such as Scrum master, product owner, quality manager, designer, programmer, maintenance specialist, tester, and configuration manager [6].

Roles are defined with proper criteria in previously proposed researches such as [6, 29, 39, 40, 51, 55, 88–91] and [87].

3.1.4. Factor Identification. "Factors are characteristics or features related to anything; these are used to understand the nature of the thing (object, entity, or context). For a better understanding of task allocation in DASD, different factors should be identified.

For a better understanding of task allocation, factors are identified in proposed researches [1, 2, 4, 6, 8, 9, 13, 15, 18–25, 28, 29, 32–34, 38, 40, 42–46, 51–55, 65–69, 71–74, 79, 80] and [82, 83, 85, 87, 90–95].

3.1.5. Dependency Identification. Task dependencies declare the situation while the success of specific task is enforced to start the following task or the success of specific action depends on the presence of a specific object (object can be an artifact, a person, or a piece of information). Coordination satisfies the management of dependencies among activities. Distribution of teams internationally is the root of extensive challenges for coordination among dependent tasks. Coordination has a noticeable impact on task performance when tasks are extremely interdependent. Unsatisfactory coordination between distributed teams is the source of cost and time overruns as well as reduced quality.

For a better understanding of task allocation, dependencies are identified in proposed researches [6, 9, 13, 33, 37, 44, 54, 55, 65, 67, 68, 73, 76, 80, 81, 85, 95].

3.1.6. Challenge/Risk Identification. Merging ASD with DSD causes the severity of risk existence because these two methods for project development are in contrast with each other. Therefore, using ASD with DSD makes the development process riskier and harder to control. DASD has various risks related to software development life cycle and project management [96]. Risks related to sprint, team management, team interaction, communication, location, geographical distribution, and task in DASD are identified [60].

3.2. Low-Level Process Activities (Phase 2). During phase 2 of task allocation process, low-level process activities are completed. Low-level process activities include task requirement identification, defining categories of factors, defining categories of dependencies, identifying dependencies between tasks and resolving dependency problems, and identifying challenges/risks and recommendations.

3.2.1. Task Requirements' Identification. Software project scheduling has to deal with the fact of assigning limited resources (humans, time, technology, and money) to tasks over time. This can be considered as an optimization problem in which the resource allocation consists in assigning time intervals to the execution of the activities (realization tasks) while taking into consideration both temporal (precedence between tasks) and resource constraints (resource availability) and the minimum execution time objective [37].

Development task can have different types of requirements according to the nature of task. These task requirements must be identified timely to overcome later issues of development. Some task requirements are identified, for example, task complexity [6, 14–16, 20–25, 55, 58, 65, 79], task type [6, 13, 15, 19, 22–25, 30, 36, 41, 55, 65, 67, 78, 84, 95], task with the strict timeline [30], required budget [14], required skills [4, 6, 13–15, 23, 24, 24, 30, 34, 35, 44, 65, 81, 84], coupling between tasks [4, 6, 9, 13, 14, 28, 34, 35, 37, 44, 54, 55, 65, 68, 73, 80, 81, 85, 89, 95], deadline [34], required resources [34], required effort [14, 20, 24, 24, 34], task size [6, 28, 30, 68], time required for task completion [35, 37, 55, 58, 81, 95], and task similarity [35, 85, 95].

3.2.2. Categories of Capabilities. Capabilities can be categorized into two domains: technical capabilities and nontechnical capabilities.

(1) Technical Capabilities. In DASD context, technical capabilities are identified in different areas:

(1) Technical knowledge: a team member must have technical knowledge according to his role in a specific field such as project supervision, requirement engineering, designing, programming, testing, quality configuration management, and maintenance.

(2) Tools knowledge: a team member must be capable of practice and selection of tools to support influenced areas.

(3) Agile knowledge: a team member working in the DASD setup must have knowledge and understanding of the values of ASD such as maintaining product backlog and sprint backlog, arranging daily meetings, arranging sprint planning meetings, arranging sprint review meetings, organizing sprint burndown charts, and contribution during planning poker.

(4) DSD knowledge: a team member working in the DASD setup must have knowledge and understanding of the values of DSD such as distributed communication, distributed cooperation, raising and keeping trust, practice of tools designed for distributed teamwork, and intercultural cooperation.

(5) Methodological knowledge: methodological knowledge is the abilities of team members such as learning and working strategies, reading techniques and information processing, testing strategies, presentation and reporting, and researching.

(2) Nontechnical Capabilities. In DASD context, nontechnical capabilities are identified in different areas:

(1) Interpersonal capabilities: interpersonal capabilities include communication skills, sociability, customer service, and concentration.

(2) Team cooperativeness: team cooperativeness capabilities include self-responsibility, understanding, offering ideas and attending to the concepts of others, curiosity, and self-motivation.

(3) Solving conflicts: capabilities related to solving conflicts include the ability to listen to others, common sense, managing emotions, resolution of conflicts, and initiation.

(4) Professional development: capabilities related to professional development include the ability to learn alone, take risks, balance the required assets to fulfill numerous intentions, resist stress, supervise the progress, and create improvements throughout the development [6].

Capabilities are defined with proper criteria in previously proposed researches such as [6, 23, 39, 43, 51, 57, 78, 87, 88, 90, 91, 97, 98].

3.2.3. Categories of Factors. There are different categories of factors that are related to task allocation in DASD, such as project related factors, individual related factors, site related factors, team related factors, task related factors, agile related factors, environment related factors, and technology related factors. All of these types of factors must be identified to satisfy the context of task allocation in DASD.

(1) Project Related Factors. Type of project [6, 42, 52], scope [50, 52], quality requirements [6, 42], hardware/software requirements [6, 42], complexity [6, 42], product architecture [6, 25], project cost [6, 50, 52], cost of change [52], and required time and effort for project completion [6] are project related factors.

(2) Individual Related Factors. Communication skills [6, 42, 52], awareness in team [6, 42], management expertise [6, 42], security [6, 42], working time [6, 42], past experience [6, 42], technical skills [6, 42, 52], proficiency [4, 6, 30], personality type [40], seniority level [55], competency level [55], job involvement [92], commitment with organization [92], status (status of person in team such as permanent staff member or temporary staff member) [4], and salary (salary paid according to the status of team member) [4] are people related factors.

(3) Site Related Factors. Knowledge and skills of team members [6, 34], availability of team members [6, 30, 34], task site specificity [6, 30], contractor reliability [6, 25], labor cost [25] and [6, 30], workload at site [29, 30], working time [6, 34], time and cultural difference [6, 25, 29, 30, 34], language difference [29], and proximity to client and market [25] are site related factors.

(4) Task Related Factors. Task size [6, 30], task type [6, 19, 22–25, 65, 84, 92], task complexity [6, 14, 20, 22–25, 55, 58, 65, 79], required budget [14], task environment complexity [73], proximity to customer requirement [6, 34], required skills [4, 6, 13, 14, 23, 24, 24, 30, 34, 34, 35, 44, 65, 81, 84], required resources [6, 34], task deadline [6, 34], task priority [6], task novelty [9], task familiarity [65], task analyzability [9], and task variability [9] are task related factors.

(5) Agile Related Factors. Transparency and openness [6, 99], prioritized delivery [6, 100], enough documentation [6, 100], strong communication [6, 74, 100], coordination [74], customer involvement [6, 52], flexibility to support changes [6], better risk analysis [71], and self-organizing team [71] are agile related factors.

(6) Team Related Factors. Team size [2, 9, 15, 16, 20, 28, 29, 32, 34, 44, 46, 51, 54, 66, 67, 72, 73, 79, 83, 93], team concurrency [79], team intensity [79], team fragmentation [79], team efficiency [9], team effectiveness [9], team velocity [43], competence diversity [43, 92], team formation age [1], team skills fit [1], team trust [31, 32], team potency [15], team spirit [15], team cohesion [31, 32, 45], team performance, team flexibility [92], and maturity of the team [19] are team related factors.

(7) Environment Related Factors. Personal causes [6, 25], political purposes [6, 25], communication and coordination overhead, nature of DSD [30], organizational purposes [6], organizational culture [52], number of distributed sites [5], geographical distance between distributed sites [5], number of projects in progress [6], number of teams, and type of development methodology [6] are environment related factors.

(8) Technology Related Factors. Interaction medium [5], collaboration modes [5], Internet medium [5], and tools [5] are technology related factors.

3.2.4. Categories of Dependencies. There are different categories of dependencies, and these dependencies should be identified to improve proper coordination during task allocation.

(1) Basic Dependencies. Flow, fit, and sharing are basic dependencies.

(1) Flow dependency: it is when an activity forms something that is used by further activity; for example, software designer makes the software design that is further used by the developer.

(2) Fit dependency: many activities offer outputs that have to be joined together, such as the integration level where individual components have to be joined together.

(3) Sharing dependency: many activities are required to use any sort of means, such as the time of technical expert [6, 54, 76, 80].

(2) Component Dependency. Relation of dependency defines the path between two components, and the layered structure of a system can be defined based on dependencies among components [101]. External dependencies of components as fan-in and fan-out are used as metrics of software component reusability [102]. Individual components are generally developed at distributed sites independently, and their integration needs to be synchronized to complete system requirements and features [103]. Software components might have intense level of dependencies. Managing these dependencies is very important because these have strong impact on other components in the software architecture. Dependencies among the software components influence the working environment of the development team [76].

(3) Software Dependencies

(1) Syntactic dependency: it defines relations between source code files, and its types are data and functional syntactic dependencies. It can be denoted as D_{ab}, which specifies the number of data, function, and method references that a program file makes to b program file.

(2) Logical dependency: it defines semantic or indirect relations between program files, as well as approximately clear relations. This makes a relation between program files that are reformed together due to modification constraints by single or many software project team participants [6, 76, 80, 104].

(4) Work Dependencies (WD). The software quality can be affected by the work dependencies.

(1) Workflow dependency: it defines definite relations between software project team participants influenced by workflows as well as processes. Workflow dependencies relate to the development team members throughout the modification request. Due to workflow dependencies, modifications made by one team member can affect others' work [6, 80].

(2) Coordination requirements' dependency: it represents the degree to which one team member requires coordination with another team member throughout the given assignments of development tasks [80].

(5) Inter-task Dependencies. Pooled, sequential, reciprocal, and team dependence tasks are inter-task dependencies.

(1) Pooled dependence task: independently, team member finishes his work before work is aggregated. For example, independent team members complete limited coding of modules earlier to be integrated with other modules.

(2) Sequential dependence task: independently, team member has to complete his task before offering the work to the next member of the team. For example, test cases are designed during the test plan by team members; these test cases are used as input during the testing phase and used by the next team member.

(3) Reciprocal dependence task: the work passes back and forth between team members. For example, all through the debugging practice, work flows side to side between participants of the team; these participants are programmers who are answerable for testing.

(4) Team dependence task: team members work simultaneously to detect issues and form solutions. For example, all through the requirement analysis, team members have to identify the requirements of the customer [6, 9, 54, 76].

(6) Agile Dependencies. Information dependency, process dependency, and resource dependency are agile dependencies.

(1) Information dependency: information is a core requirement during software project development; it declares different dependencies such as requirements, expertise, historical, and task-member assignment.

(2) Process dependency: it is when a task is required to be completed before another task proceeds. It takes account of activity and business process dependencies.

(3) Resource dependency: when an object is required for project success, this includes entity and technical dependencies [6, 54, 76].

(7) Distributed Software Development Dependencies. Geographical, temporal, cultural, technical, knowledge, organizational, people, resource, site, communication, and coordination dependencies are DSD dependencies [6, 30, 73].

3.2.5. Identification of Dependencies between Tasks and Resolving Dependency Problem. Dependencies between tasks are identified in previously proposed research studies [4, 6, 9, 13, 28, 30, 36, 37, 44, 55, 65, 73, 76, 80, 81, 85, 89, 95]. Dependency problems are resolved in previously proposed research studies [4, 9, 13, 36, 37, 44, 55, 65, 73, 76, 80, 81, 85, 89, 95].

3.2.6. Challenges/Risks in Various Classes of Software Development Setups. There are several challenges/risks identified and categorized into DSD, ASD, and DASD. Recommendations related to the challenges of each category are also described.

(1) Challenges/Risks in Distributed Software Development. These include priority conflicts, overload problem, getting time and attention of people on time, low opportunities of informal communication, low richness of communication medium, lack of previous knowledge [28], communicating with distributed team members, time zones and working hours, cultural differences, language differences, tools, software engineering practices, team dynamics, IT infrastructure for collaboration, different knowledge levels, asymmetry in processes policies and standards, tracking and control, intellectual property issues, creating team spirit [1], low quality decision-making, issue for plan recurrent decisions, absence of information sharing, insecurity related to performances [16], lack of transparency related to task management, low information sharing, not enough visibility of virtual members of team inside/outside the organization, lack of transparency about the project progress, low efficiency of communication [32], lack of coordination, low trust, poor cohesion [31], absence of strong component or project ownership, component dependencies, poor process to cope with requirements changes, organizational cultural differences, absence of effective task allocation, handling bugs and faults, complex licenses issues, inefficient cost and effort estimation, conflict management, poor team interaction opportunities [103], corporate environment, ownership, relationship management, project management and planning, scope, requirements, funding, scheduling, development processes, staffing, technology, external dependencies [105], ensuring governance framework setting, maintenance, profits delivery, risk optimization, resource optimization, stakeholder engagement [27], communication, coordination, control [106], task distribution, knowledge management, geographical distribution, collaboration structure, cultural distribution, stakeholder relations, communication infrastructure, technology [107], and temporal, geographical, and sociocultural related issues [108].

(1) Recommendations

Different types of propositions are presented to handle challenges of coordination, shared knowledge of task, shared knowledge of team, task awareness, and presence awareness in DSD [28].

Different types of software tools are recommended to solve these issues related to the performance of virtual project teams. Proper selection and use of software tools can support communication, collaboration, and project management activities (task management such as task planning, task assignment, task transparency, and task tracking) [32].

Different types of recommendations are described according to different practices such as global task management, knowledge and skills management, global project management, management between locations, and collaboration between locations. These recommendations are presented to resolve ambiguity in GSD [17].

Possible solutions for challenges related to these objectives (ensuring governance framework setting, maintenance, profits delivery, risk optimization, resource optimization, and stakeholder engagement) are identified to support GSD [27].

Effective coordination in teams with higher trust and cohesion supports positive feedback and improves the project performance positively [31]. Different strategies are presented to handle risks related to communication, coordination, and control in GSD [106].

Different types of risk resolution techniques are presented to cover planning, control, and social integration areas [107]. Policy recommendations are presented to mitigate different types of challenges related to different aspects of DSD [109].

(2) Challenges/Risks in Agile Software Development. Challenges during agile adoption include changes on regular basis from client, lack of requirements, daily Scrum meetings, difficulty in tailoring the agile process after adoption, selection of agile to integrate different types of projects to satisfy end users' requirements, time efficiency compromises, lack of team cooperation, lack of management support, poor architectural planning, creeping requirements, insufficient training [110], scope creep, overly optimistic schedules, requirements or developer gold-plating, short-changed quality, inadequate design, friction between developers and customers, budget risk, cancelation cost, requirement error, technology risk, security risk, and contractor failure [111].

Different development and deployment risks (technical debt, separation of development and IT operations, and increased defect in new ASD teams) and project management risks (unstandardized project management tools, lack of knowledge retention) are identified [112].

According to the empirical research, dependencies must be considered as a part of risk management because they create issues (prioritization changes, negative effect on project planning, and late delivery) in case they are found in later development stage [44].

(1) Recommendations: risk mitigation strategies reported by respondents are related to schedule risks, budget/financial risks, technology risks, architectural

risks, security risks, deployment risks, people risks, requirement change risks, performance risks, and programmatic risks [113].

Organizations experienced risks related to project development and management, and multiple ways are described as to how the organizations dealt with them [112].

Continuous communication and collaboration are recommended to mitigate the risks due to dependencies, and a hybrid approach to architecture also helps to manage dependencies [44].

Different agile methods are described such as Scrum, Extreme Programming (XP), and Dynamic Systems Development Method (DSDM) for risk management. DSDM and Scrum offer better risk management practices than XP [111].

(3) Challenges/Risks in Distributed Agile Software Development. These include communication and coordination issues due to coordinating multiple agile teams on the same project and synchronizing sprints in a large-scale agile development program, complex software architecture issues due to integration concerns, dependencies with other subsystems and teams, managing and integrating heterogeneous subsystems of different teams, geographical issues related to distance between agile teams and lack of team cohesion at different locations, different cultures and mindsets creating issues related to doubts in people about changes, encouraging teams to talk about tasks and impediments, dealing with higher-level management interferences, lacking sense of ownership responsibilities and loss of management, methodology challenges related to incorrect practices of agile development and establishing a common understanding of agile thinking and practices, project management challenges related to creating a teamwork centric rewarding model, defining clear roles and responsibilities, considering required competencies when assigning teams to tasks, dealing with increasing workload of key stakeholders, fixed price contracts in ASD, unplanned requirements/risks and decreased predictability [114], organizing Scrum teams, creating and prioritizing the backlog, estimating the stories as a team, creating the release plan, sprint length, managing dependencies, product owner effectiveness, continuous integration, keeping the pace [1], failure to match architectural design to organization structure and practices and allocate work consequently, problems in clear architectural decisions, poor awareness and communication between distributed teams and unsatisfactory knowledge management, unsatisfactory quality assurance, failure to have a stable architecture, incomplete or incorrect practices applied for design and development, problems in finding architectural dependencies as well as decoupling components to separate tasks, problems with tasks spanning across multiple sites [59], lack of face to face communication, improper task allocation, absence of agile coaching, time and linguistic differences, insufficient knowledge sharing between team members, costly implementation, skepticism toward agile development, inappropriate management commitments, lack of agile process evaluation mechanism [71], resistance to change, distributed locations, quality assurance, integration with non-agile parts of organization, unsatisfactory commitment as well as teamwork, excessive pressure and workload, absence of required knowledge, coaching and training, requirements management hierarchy, measuring progress [72], team members at times not identifying responsibilities of others, progress being blocked or postponed when waiting for resources and information, frustrating condition when one has done his work (deliverables) and has to wait for activities (external part) from other members to be completed and new resources without required historical information [67], competing priorities, language and cultural differences, process misalignment, unclear required information, lack of availability and responsiveness caused by time zone differences, indefinite progress, testing becoming a bottleneck, firewall and security barrier and rework due to more risk of misunderstanding [68], task environment complexity [73], different cultures having dissimilar methods to solve the same problems, lack of face to face communication, late instead of initial top-level design and incomplete high-level requirements becoming cause of waste of resources and time, and poor tools and a poor environment leading to a poor project [66].

(1) Recommendations: types of recommendations are presented in four dimensions, namely, task, structure, actors, and technology; decision support system for risk management is also proposed [5].

Different types of strategies are presented to handle risks (uncoordinated activities, lack of group awareness, low quality of communication bandwidth, insufficient tool support, large number of people in a team, lack of collaborative environment, and increased number of sites) in DASD [115].

Challenges related to spatial, time-based, and sociocultural differences in DASD can be handled by improving the trust level in order to work as one team. Different types of techniques for improving trust between team members are presented [75].

Coordination mechanisms and tools are described to address different types of dependencies with the main coordination challenges [67, 68].

Software process abilities such as rigor, standardization, agility, and customizability are identified. These abilities lessen the negative impact of team distribution and dynamic nature of requirements and cope with the challenge of increasing task environment complexity [73].

Communication challenges can be resolved with agile enterprise architecture in GSD. According to the empirical evidence, agile enterprise architecture supports the efficiency and effectiveness of communication and increases the performance of global distributed agile development [2].

Seven recommendations related to challenges of communication, language, awareness, technology, and connectivity are described. Using practices and tools to radiate information, colocating the team regularly, providing channels and encouraging communication, planning for communication, preparing alternative communication strategies, adapting the process model, and sticking to the defined processes and practices are recommended to support DASD [69].

Different types of challenges such as different cultures having dissimilar methods to solve the same problems, lack of face to face communication, late instead of initial top-level design and incomplete high-level requirements becoming a cause of waste of resources and time, and poor tools and a poor environment leading to a poor project are presented. Recommendations to tackle different challenges are presented to support DASD [66].

Recommendations for the challenges related to different concerns such as organization, ways of working, architectural knowledge management, quality management, change management, design practice, modularity, and task allocation are described in GSD using Scrum [59].

(4) Challenges/Risks Which Have Direct Impact on Task Allocation Process. Challenges/risks are prioritized from the identified challenges/risks in DSD, ASD, and DASD, which have direct impact on task allocation in DASD. These challenges/risks which require attention from the management are given below.

Challenges/risks that have direct impact on task allocation in DASD are prioritized from the identified challenges/risks in DSD, ASD, and DASD. They require attention from the management. They include lack of transparency related to task management, absence of effective task allocation and distribution, insufficient shared knowledge of tasks, and lack of task planning and tracking. Risks due to global task management, unresolved task dependencies and inability to find architectural dependencies may also lead to bad decisions.

3.3. Task to Member Allocation (Phase 3). During phase 3 of task allocation process, task to member allocation is performed. Task to member allocation is presented in [1, 4, 8, 13, 14, 16, 18–20, 22–24, 24, 36, 37, 55, 57, 58, 65, 78, 79, 81, 84, 85, 95, 116].

4. Evaluation and Discussion Based on the Proposed Task Allocation Process for DASD

In Table 1, we include works that presented task to member allocation. In this table, we investigate the presented task to member allocation to find out how many high-level process activities and low-level process activities are considered before the task to member allocation. In all works presented in Table 1 except for [6], task to member allocation is done; we include this work because it completed many other process activities of task allocation process to support task allocation in DASD.

We evaluate all task to member mapping based on our proposed task allocation process for DASD. None of the presented research works in Table 1 completely considered all high-level process activities and low-level process activities before the task to member allocation in DASD. All of them considered very few activities before allocation. In the presented works in Table 1, some low-level process activities such as defining objectives, task identification, and identifying task requirements are considered except in [6]. Capabilities are defined with criteria in [6, 23, 57, 78], roles are defined with criteria in [6, 55], and other works just identified capabilities and roles. The works did not offer explanation for criteria or even mention the name of any criterion during the identification of capabilities and roles.

Very few factors and dependencies (dependencies between tasks) are presented. Many factors are identified in these research works [1, 8, 13, 18–20, 36], but categories of these factors are not defined. Categories are defined just in [6], but team factors and technology factors are not included in this work. Task dependency problem is resolved in some presented works, but categories of dependencies are identified just in [6, 76]. Identified categories of factors and categories of dependencies are not properly considered before task allocation in any presented work. Challenges/risks are identified in [24, 81, 95] and [8, 13, 16, 18–20, 55, 76].

Following are suggestions for future work:

(i) There is a need to propose methods, models, and approaches to support high-level process activities and/or low-level process activities of task allocation process in DASD

(ii) These types of methods, models, and approaches which consider all categories of factors related to DASD before task to member allocation should be proposed

(iii) These types of methods, models, and approaches which consider all categories of dependencies related to DASD before task to member allocation should be proposed

(iv) Challenges/risks-based task allocation approaches are required for DASD

Software development (SD), distributed software development (DSD), agile software development (ASD), and distributed agile software development (DASD) are development setups. Task identification (TI), defining objectives (DO), defining capabilities (DC), capability identification (CI), defining roles (DR), role identification (RI), factor identification (FI), and dependency identification (DI) are related to high-level activities. Task requirements' identification (TR), categories of factors, categories of dependencies, resolve problem of dependencies (RD), and challenges/risk identification (C/R) are low-level activities. Project-related factors (PF), individual-related factors (ID), site-

Table 1: Evaluation based on proposed task allocation process.

Development setup	Proposed methodology	High-level process activities									Low-level process activities															
											CF											CD				
		DO	TI	DC	CI	DR	RI	FI	DI	TR	PF	IF	SF	TF	AF	EF	TmF	TecF	BD	SD	WD	AD	TD	DSD-D	RD	C/R
SD	Estimation model [79]	✓	✓	✗	✓	✗	✓	✓	✗	✓	—	—	✗	—	✗	✗	—	✗	✗	✗	✗	✗	✗	✗	✗	✗
SD	Mathematical model for scheduling [85]	✓	✓	✗	✓	✗	✓	✓	✓	✓	—	—	✗	—	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗
SD	Scheduling approach [4]	✓	✓	✗	✓	✗	✓	✓	✓	✓	—	—	✗	—	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗
SD	Systematic literature review [81]	✓	✓	✗	✓	✗	✓	✓	✓	✓	—	—	✗	—	✗	✗	—	✗	✗	✗	✗	✗	✗	✗	✓	✓
SD	Framework [82]	✓	✓	✗	✓	✗	✓	✓	✗	✓	—	—	✗	—	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
SD	AITA with negotiation tree [116]	✓	✓	✗	✓	✗	✓	✓	✗	✓	—	—	✗	—	✗	✗	✗	—	✗	✗	✗	✗	✗	✗	✗	✗
SD	Multi-objective optimization approach [78]	✓	✓	✓	✗	✗	✓	✓	✗	✓	—	—	✗	—	✗	✗	—	✗	✗	✗	✗	✗	✗	✗	✗	✗
SD	Approach for mining task dependencies [95]	✓	✓	✗	✓	✗	✓	✓	✓	✓	—	—	✗	—	✗	✗	✗	—	✗	✗	✗	✗	✗	✗	✓	✓
SD	Model for job design [84]	✓	✓	✗	✓	✗	✓	✓	✗	✓	—	—	✗	—	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
DSD	Models and algorithms [24]	✓	✓	✗	✓	✗	✓	✓	✗	✓	—	—	✗	—	✗	—	✗	—	✗	✗	✗	✗	✗	✗	✗	✓
DSD	Verbal decision analysis method [18]	✓	✓	✗	✓	✗	✓	✓	✗	✓	—	—	—	—	✗	—	—	—	✗	✗	✗	✗	✗	✗	✗	✓
DSD	Planning tool [36]	✓	✓	✗	✓	✗	✓	✓	✓	✓	—	—	—	—	✗	—	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗
DSD	Decision support model [13]	✓	✓	✗	✓	✗	✓	✓	✓	✓	—	—	—	—	✗	—	✗	—	✗	✗	✗	✗	✗	✗	✓	✓
DSD	Task assignment method [22]	✓	✓	✗	✓	✗	✓	✓	✓	✓	—	—	✗	—	✗	—	✗	—	✗	✗	✗	✗	✗	✗	✓	✗
DSD	Task allocation algorithm [23]	✓	✓	✓	✗	✗	✓	✓	✗	✓	—	—	✗	—	✗	—	✗	—	✗	✗	✗	✗	✗	✗	✗	✗
DSD	Framework for allocation [20]	✓	✓	✗	✓	✗	✓	✓	✗	✓	—	—	—	—	✗	—	✗	—	✗	✗	✗	✗	✗	✗	✗	✓
DSD	Hybrid methodology [19]	✓	✓	✗	✓	✗	✓	✓	✗	✓	—	—	—	—	—	—	—	—	✗	✗	✗	✗	✗	✗	✗	✓
DSD	Multicriteria model [16]	✓	✓	✗	✓	✗	✓	✓	✗	✗	—	—	—	—	✗	—	—	—	✗	✗	✗	✗	✗	✗	✗	✓
ASD	Decision support system [55]	✓	✓	✗	✓	✓	✗	✓	✓	✓	—	—	✗	—	✓	—	✗	—	✗	✗	✗	✗	✗	✗	✓	✓
ASD	Multi-agent simulation[58]	✓	✓	✗	✓	✗	✓	✓	✗	✓	—	—	✗	—	✓	—	✗	—	✗	✗	✗	✗	✗	✗	✗	✗
ASD	Algorithms [37]	✓	✓	✗	✓	✗	✓	✓	✓	✓	—	—	✗	—	✓	—	✗	—	✗	✗	✗	✗	✗	✗	✓	✗
ASD	Software tool for agile management [57]	✓	✓	✓	✗	✗	✓	✓	✗	✓	—	—	✗	—	✓	—	—	—	✗	✗		✗	✗	✗	✗	✗
DASD	Case study [65]	✓	✓	✗	✓	✗	✓	✓	✓	✓	—	—	—	—	✓	—	—	—	✗	✗	✓	—	—	—	✓	✓
DASD	Taxonomy [76]	✓	✓	✗	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓	✓	✗	✓	✓	✓	✓	✓
DASD	Decision model [1]	✓	✓	✗	✓	✗	✓	✓	✓	✓	—	—	—	—	✓	—	—	—	✗	✗	✗	✗	✗	—	✓	✓
DASD	Hybrid methodology [8]	✓	✓	✗	✓	✗	✓	✓	✗	✓	—	—	—	—	—	—	—	—	✗	✗		✗	✗	✗	✗	✓
DASD	Framework for task allocation [6]	✓	✗	✓	✗	✓	✗	✓	✓	✗	✓	✓	✓	✓	✓	✓	✗	✗	✓	✓		✓	✓	✓	✗	✗

related factors (SF), task-related factors (TF), agile-related factors (AF), environment-related factors (EF), team-related factors (TmF), and technology related factors (TecF) are categories of factors. Basic dependencies (BD), software dependencies (SD), workflow dependencies (WD), agile dependencies (AD), task dependencies (TD), and DSD dependencies (DSD-D) are categories of dependencies. ✓: completed; —: partially completed; ✗: not completed.

5. Conclusion

Distributed agile software development presents many benefits for software development, but it is also a source of very complicated setup for decision-making due to a distributed environment. Ineffective planning of task allocation through an imperfect task allocation process creates many problems for successful project development. Regarding the research question, task allocation process is presented to support decision-making in DASD. Based on the different aspects of the task allocation context, different types of activities are defined and categorized into high-level process activities and low-level process activities to support task allocation in DASD.

According to the objectives of this research, a complete task allocation process is proposed and evaluated against existing relevant works. Our evaluation considers detailed activities and is hence comprehensive enough for a variety of approaches and methods. To the best of our knowledge, none of the previous works support completeness in the sense of our pointed out high-level activities or low-level process activities that enable better matching of actual tasks to the capabilities of members. Through such detailed work, a stronger understanding related to all activities of task allocation process is developed. Such an understanding is mandatory for the correct decision-making of a project manager or a Scrum master, who steers projects towards successful completion. It also methodically guides keeping detailed history of project decisions for strength and weakness analysis of teams at various sites. At a higher view, failures of future projects can be addressed, and optimality of resources can be achieved objectively.

Task allocation agnostic with respect to relevant factors, dependencies, and risks and capabilities of team members per task requirements gives unrealistic and suboptimal solution that is highly likely to miss out the target objectives of a project. Detailed information about the context eases decision-making and provides an understanding of expected outcomes. Our proposed process will help the software industry to improve its processes by considering all activities before the task to member allocation. It will support the final decision-making with complete information on the task allocation context in DASD. Even task prioritization is possible to reflect high-level preferences of the management.

Data Availability

All the data used to support the findings of the study are imcluded within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] L. H. Almeida, A. B. Albuquerque, and P. c. R. Pinheiro, "A multi-criteria model for planning and fine-tuning distributed Scrum projects," in *Proceedings of the 2011 IEEE Sixth International Conference on Global Software Engineering*, pp. 75–83, 2011.

[2] Y. I. Alzoubi and A. Q. Gill, "An empirical investigation of geographically distributed agile development: the agile Enterprise architecture is a communication enabler," *IEEE Access*, vol. 8, pp. 80269–80289, 2020.

[3] G. Papadopoulos, "Moving from traditional to agile software development methodologies also on large, distributed projects," *Procedia - Social and Behavioral Sciences*, vol. 175, pp. 455–463, 2015.

[4] W. N. Chen, J. Zhang, and S. Member, "Ant colony optimization for software project scheduling and staffing with an event-based scheduler," *IEEE Transactions on Software Engineering*, vol. 39, no. 1, pp. 1–17, 2013.

[5] A. Aslam, N. Ahmad, T. Saba et al., "Decision support system for risk assessment and management strategies in distributed software development," *IEEE Access*, vol. 5, pp. 20349–20373, 2017.

[6] W. Aslam and F. Ijaz, "A quantitative framework for task allocation in distributed agile software development," *IEEE Access*, vol. 6, pp. 15380–15390, 2018.

[7] R. Hoda and L. K. Murugesan, "Multi-level agile project management challenges: a self-organizing team perspective," *Journal of Systems and Software*, vol. 117, pp. 245–257, 2016.

[8] M. Simão Filho, P. R. Pinheiro, and A. B. Albuquerque, "Analysis of task allocation in distributed software development through a hybrid methodology of verbal decision analysis," *Journal of Software: Evolution and Process*, vol. 29, no. 7, pp. e1867–18, 2017.

[9] J. Sutanto, A. Kankanhalli, and B. C. Yian Tan, "Investigating task coordination in globally dispersed teams," *ACM Transactions on Management Information Systems*, vol. 6, no. 2, pp. 1–31, 2015.

[10] P. Banerjee, A. Sarkar, and N. C. Debnath, "Generalized extra-functional properties composition for component based system," in *Proceedings of the - 2017 International Conference on Recent Advances in Signal Processing*, vol. 2017, pp. 71–76, Telecommunications and Computing, 2016.

[11] I. Malavolta, P. Lago, H. Muccini, P. Pelliccione, and A. Tang, "What industry needs from architectural languages: a survey," *IEEE Transactions on Software Engineering*, vol. 39, no. 6, pp. 869–891, 2013.

[12] J. M. Bass, "Agile method tailoring in distributed enterprises: product owner teams," in *Proceedings of the - IEEE 8th International Conference on Global Software Engineering*, pp. 154–163, 2013.

[13] A. Lamersdorf, J. Münch, and D. Rombach, "A decision model for supporting task allocation processes in global software development," in *Proceedings of the International Conference on Product-Focused Software Process Improvement*, Springer, 2009.

[14] J. Fernandez and M. Basavaraju, "Task allocation model in globally distributed software projects using genetic algorithms," in *Proceedings of the 2012 IEEE Seventh*

International Conference on Global Software Engineering, p. 181, 2012.

[15] D. S. Staples and A. F. Cameron, "The effect of task design, team characteristics, organizational context and team processes on the performance and attitudes of virtual team members," in *Proceedings of the 38th Hawaii International Conference on System Sciences*, pp. 1–10, 2005.

[16] A. Barcus and G. Montibeller, "Supporting the allocation of software development work in distributed teams with multi-criteria decision analysis," *Omega*, vol. 36, no. 3, pp. 464–475, 2008.

[17] M. Marinho, J. Noll, and S. Beecham, "Uncertainty management for global software development teams," in *Proceedings of the 11th International Conference on the Quality of Information and Communications Technology*, pp. 238–246, QUATIC), 2018.

[18] M. S. Filho, P. R. Pinheiro, and A. Bessa Albuquerque, "Applying verbal decision analysis in distributed software development: rank ordering the influencing factors in task allocation," in *Proceedings of the 11th Iberian Conference on Information Systems and Technologies*, vol. 2016, CISTI, 2016.

[19] M. Simão Filho, P. R. Pinheiro, and A. B. Albuquerque, "Task assignment to distributed teams aided by a hybrid methodology of verbal decision analysis," *IET Software*, vol. 11, no. 5, pp. 245–255, 2017.

[20] M. Ruano-Mayoral, C. Casado-Lumbreras, H. Garbarino-Alberti, and S. Misra, "Methodological framework for the allocation of work packages in global software development," *Journal of Software: Evolution and Process*, vol. 26, no. 5, pp. 476–487, 2014.

[21] S. Imtiaz and N. Ikram, "Framework for task allocation in global software development," *IEEE Access*, vol. 8, pp. 206235–206247, 2020.

[22] D. Yu, Z. Zhou, and Y. Wang, "Crowdsourcing software task assignment method for collaborative development," *IEEE Access*, vol. 7, pp. 35743–35754, 2019.

[23] D. Yu, Y. Wang, and Z. Zhou, "Software crowdsourcing task allocation algorithm based on dynamic utility," *IEEE Access*, vol. 7, pp. 33094–33106, 2019.

[24] X. Yin, J. Huang, W. He, W. Guo, H. Yu, and L. Cui, "Group task allocation approach for heterogeneous software crowdsourcing tasks," *Peer-to-Peer Networking and Applications*, vol. 14, no. 3, pp. 1736–1747, 2020.

[25] A. Lamersdorf, J. Münch, and D. Rombach, "A survey on the state of the practice in distributed software development: criteria for task allocation," in *Proceedings of the - 2009 4th IEEE International Conference on Global Software Engineering*, pp. 41–50, 2009.

[26] M. Simão Filho, P. R. Pinheiro, A. B. Albuquerque, and J. J. P. C. Rodrigues, "Task allocation in distributed software development: a systematic literature review," *Complexity*, vol. 2018, pp. 1–13, 2018.

[27] A. Manjavacas, A. Vizcaíno, F. Ruiz, and M. Piattini, "Global software development governance: challenges and solutions," *Journal of Software: Evolution and Process*, vol. 32, no. 10, pp. 1–26, 2020.

[28] J. A. Espinosa, S. A. Slaughter, R. E. Kraut, and J. D. Herbsleb, "Team knowledge and coordination in geographically distributed software development," *Journal of Management Information Systems*, vol. 24, no. 1, pp. 135–169, 2014.

[29] J. A. Khan, S. U. R. Khan, J. Iqbal, I. U. Rehman, J. Iqbal, and I. U. R. Rehman, "Empirical investigation about the factors affecting the cost estimation in global software development context," *IEEE Access*, vol. 9, pp. 22274–22294, 2021.

[30] S. Imtiaz and N. Ikram, "Dynamics of task allocation in global software development," *Journal of Software: Evolution and Process*, vol. 29, no. 1, pp. e1832–17, 2017.

[31] I. Ullah, N. U. Amin, A. Almogren, M. A. Khan, M. I. Uddin, and Q. Hua, "A lightweight and secured certificate-based proxy signcryption (CB-PS) scheme for E-prescription systems," *IEEE Access*, vol. 8, pp. 199197–199212, 2020.

[32] I. Ullah, S. Zeadally, N. U. Amin, M. Asghar Khan, and H. Khattak, "Lightweight and provable secure cross-domain access control scheme for internet of things (IoT) based wireless body area networks (WBAN)," *Microprocessors and Microsystems*, vol. 81, p. 103477, 2021.

[33] J. Żywiołek, J. Rosak-Szyrocka, M. A. Khan, and A. Sharif, "Trust in renewable energy as part of energy-saving knowledge," *Energies*, vol. 15, no. 4, p. 1566, Feb. 2022.

[34] A. B. Marques, J. R. Carvalho, R. Prikladnicki, and S. Marczak, "An ontology for task allocation to teams in distributed software development," in *Proceedings of the 8th International Conference on Global Software Engineering*, pp. 21–30, IEEE, Italy, 2013.

[35] A. A. Zafar, S. Saif, M. Khan et al., "Taxonomy of factors causing integration failure during global software development," *IEEE Access*, vol. 6, pp. 22228–22239, 2018.

[36] A. Lamersdorf, "Tamri : a tool for supporting task distribution in global software development projects jürgen münch," in *Proceedings of the Fourth IEEE International Conference on Global Software Engineering*, pp. 322–327, IEEE, 2009.

[37] N. N. Tuan and H. Q. Thang, *Iteration Scheduling Using Bayesian Networks in Agile Software Development*, pp. 300–308, 2019.

[38] M. A. Oliveira, L. V. O. Dalla Valentina, A. H. Futami, O. Possamai, and C. A. Flesch, "Project performance prediction model linking agility and flexibility demands to project type," *Expert Systems*, vol. 38, no. 4, pp. 1–27, 2021.

[39] M. Turi and L. Vickovi in *Proceedings of the 22nd Telecommun. forum TELFOR 2014*, pp. 1134–1137, IEEE, 2014.

[40] A. R. Gilal, J. Jaafar, S. Basri, M. Omar, and M. Z. Tunio, "Making programmer suitable for team-leader : software team composition based on personality types," in *Proceedings of the International Symposium on Mathematical Sciences and Computing Research (iSMSC)*, pp. 78–82, IEEE, Malaysia, 2015.

[41] S. D. Vishnubhotla, E. Mendes, and L. Lundberg, "Designing a capability-centric web tool to support agile team composition and task allocation: a work in progress," in *Proceedings of the ACM/IEEE 11th International Workshop on Cooperative and Human Aspects of Software Engineering Designing*, pp. 41–44, 2018.

[42] R. Popli and N. Chauhan, "Agile estimation using people and project related factors," in *Proceedings of the 2014 International Conference on Computing for Sustainable Global Development*, pp. 564–569, New Delhi, India, 2014.

[43] E. Mendes and L. Lundberg, "Realising individual and team capability in agile software development : a qualitative investigation," in *Proceedings of the 2018 44th Euromicro Conference on Software Engineering and Advanced Applications*, pp. 183–190, SEAA), Prague, 2018.

[44] A. Martakis and M. Daneva, "Handling requirements dependencies in agile projects: a focus group with agile software development practitioners," in *Proceedings of the 7th International Conference on Research Challenges in Information Science*, pp. 1–11, IEEE, France, 2013.

[45] T. Dingsoyr, T. E. Faegri, T. Dyba, B. Haugset, and Y. Lindsjorn, "Team performance in software development:

research results versus agile principles," *IEEE Software*, vol. 33, no. 4, pp. 106–110, 2016.

[46] M. Singh, N. Chauhan, and R. Popli, *A Review on Quantitative Task Allocation in Agile Software Development*, pp. 268–273, SSRN Electronic Journal, 2019.

[47] S. L. Ramirez-Mora and H. Oktaba, "Productivity in agile software development: a systematic mapping study," in *Proceedings of the 5th International Conference in Software Engineering Research and Innovation*, pp. 44–53, CONISOFT, 2018.

[48] J. Buchan, S. G. MacDonell, and J. Yang, "Effective team onboarding in Agile software development: techniques and goals," in *Proceedings of the 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, p. 1, Brazil, 2019.

[49] M. L. Drury-Grogan, "Performance on agile teams: relating iteration objectives and critical decisions to project management success factors," *Information and Software Technology*, vol. 56, no. 5, pp. 506–515, 2014.

[50] F. Hayat, A. Ur Rehman, K. S. Arif, K. Wahab, and M. Abbas, "The, nfluence of agile methodology (scrum) on software project management," in *Proceedings of the 20th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, pp. 145–149, SNPD), Japan, 2019.

[51] R. Hoda, J. Noble, and S. Marshall, "Self-organizing roles on agile software development teams," *IEEE Transactions on Software Engineering*, vol. 39, no. 3, pp. 422–444, 2013.

[52] M. Alqudah, R. Razali, and M. Kasim, "Agile methods selection model: a grounded theory study," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 7, pp. 357–366, 2019.

[53] Z. Masood, R. Hoda, and K. Blincoe, "Motivation for self-assignment : factors agile software developers consider," in *Proceedings of the IEEE/ACM 10th International Workshop on Cooperative and Human Aspects of Software Engineering*, pp. 11-12, CHASE) Motivation, Argentina, 2017.

[54] D. E. Strode, "A dependency taxonomy for agile software development projects," *Information Systems Frontiers*, vol. 18, no. 1, pp. 23–46, 2016.

[55] A. Alhazmi and S. Huang, "A decision support system for sprint planning in scrum practice," in *Proceedings of the SoutheastCon 2018*, pp. 1–9, 2018.

[56] I. Karac, B. Turhan, and N. Juristo, "A controlled experiment with novice developers on the impact of task description granularity on software quality in test-driven development," *IEEE Transactions on Software Engineering*, vol. 47, no. 7, pp. 1315–1330, 2021.

[57] J. Lin, H. Yu, Z. Shen, and C. Miao, "Studying task allocation decisions of novice agile teams with data from agile project management tools," in *Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering*, pp. 689–694, Sweden, 2014.

[58] Z. Wang, "P-value based task allocation in a scrum team: a multi-agent simulation," in *Proceedings of the IEEE 10th International Conference on Software Engineering and Service Science*, pp. 1–4, ICSESS), China, 2019.

[59] O. Sievi-Korte, S. Beecham, and I. Richardson, "Challenges and recommended practices for software architecting in global software development," *Information and Software Technology*, vol. 106, pp. 234–253, 2019.

[60] I. Technology, "Risk aware and quality enriched effort estimation for mobile applications in distributed agile software development," *Journal of Information Science and Engineering*, vol. 1500, pp. 1481–1500, 2017.

[61] F. Aizaz, S. U. R. Khan, J. A. Khan, J. A. L. I. Inayat-Ur-Rehman, and A. Akhunzada, "An empirical investigation of factors causing scope creep in agile global software development context: a conceptual model for project managers," *IEEE Access*, vol. 9, pp. 109166–109195, 2021.

[62] P. William, P. Kumar, G. S. Chhabra, and K. Vengatesan, "Task allocation in distributed agile software development using machine learning approach," in *Proceedings of the 2021 International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications (CENTCON)*, vol. 1, pp. 168–172, Bengaluru India, 2021.

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

[64] S. Shafiq, A. Mashkoor, C. Mayr-Dorn, and A. Egyed, "TaskAllocator: a recommendation approach for role-based tasks allocation in agile software development," *ICSSP/ICGSE* in *Proceedings of the - 2021 IEEE/ACM Joint 15th International Conference on Software and System*, vol. 2021, pp. 39–49, Spain, 2021.

[65] A. Sablis, D. Smite, and N. Moe, "Team-external coordination in large-scale software development projects," *Journal of Software: Evolution and Process*, vol. 33, no. 3, pp. 1–26, 2021.

[66] Y. Khmelevsky, X. Li, and S. Madnick, "Software development using agile and scrum in distributed teams," in *Proceedings of the 11th Annual IEEE International Systems Conference*, pp. 1–4, Canada, 2017.

[67] V. Stray, N. B. Moe, and A. Aasheim, "Dependency management in large-scale Agile: a case study of DevOps teams," in *Proceedings of the 52nd Hawaii International Conference on System Sciences*, pp. 7007–7016, Hawaii, HI, USA, 2019.

[68] J. Buchan, A. B. M. N. A. Talukder, and M. Senapathi, "Coordination in distributed agile software development: insights from a COTS-based case study," in *Proceedings of the Australasian Conference on Information Systems*, pp. 942–952, Fremantle, 2019.

[69] M. Stadler, R. Vallon, M. Pazderka, and T. Grechenig, "Agile distributed software development in nine central European teams:challenges, benefits and recommendations," *International Journal of Computer Science and Information Technology*, vol. 11, no. 01, pp. 01–18, 2019.

[70] F. Calefato and C. Ebert, "Agile collaboration for distributed teams [software technology]," *IEEE Software*, vol. 36, no. 1, pp. 72–78, 2019.

[71] R. Sinha, M. Shameem, and C. Kumar, "SWOT: strength,weaknesses, opportunities, and threats for scaling agile methods in global software development," in *Proceedings of the 13th Innovations in Software Engineering Conference (Formerly Known as India Software Engineering Conference) (ISEC 2020)*, pp. 1–10, ACM, India, 2020.

[72] M. Kalenda, P. Hyna, and B. Rossi, "Scaling agile in large organizations: practices, challenges, and success factors," *Journal of Software: Evolution and Process*, vol. 30, no. 10, pp. e1954–24, 2018.

[73] G. Gwanhoo Lee, J. A. Espinosa, and W. H. Delone, "Task environment complexity, global team dispersion, process capabilities, and coordination in software development," *IEEE Transactions on Software Engineering*, vol. 39, no. 12, pp. 1753–1771, 2013.

[74] T. Kamal, Q. Zhang, M. A. Akbar, M. Shafiq, A. Gumaei, and A. Alsanad, "Identification and prioritization of agile requirements change management success factors in the domain of global software development," *IEEE Access*, vol. 8, pp. 44714–44726, 2020.

[75] S. Dorairaj and J. Noble, "Agile software development with distributed teams: agility, distribution and trust," in *Proceedings of the 2013 Agile Conference*, pp. 1–10, 2013.

[76] F. Ijaz and W. Aslam, "Identification of dependencies in task allocation during distributed agile software development," *Sindh University Research Journal -Science Series*, vol. 51, no. 01, pp. 31–36, 2019.

[77] E. Klotins, M. Unterkalmsteiner, P. Chatzipetrou et al., "A progression model of software engineering goals, challenges, and practices in start-ups," *IEEE Transactions on Software Engineering*, vol. 47, no. 3, pp. 498–521, 2021.

[78] J. Duggan, J. Byrne, and G. J. Lyons, "A task allocation optimizer for software construction," *IEEE Software*, vol. 21, no. 3, pp. 76–82, 2004.

[79] R. K. Smith, J. E. Hale, A. S. Parrish, and A. S. Parrish, "An empirical study using task assignment patterns to improve the accuracy of software effort estimation," *IEEE Transactions on Software Engineering*, vol. 27, no. 3, pp. 264–271, 2001.

[80] M. Cataldo, A. Mockus, J. A. Roberts, and J. D. Herbsleb, "Software dependencies, work dependencies, and their impact on failures," *IEEE Transactions on Software Engineering*, vol. 35, no. 6, pp. 864–878, 2009.

[81] A. V. Rezende, L. Silva, A. Britto, and R. Amaral, "Software project scheduling problem in the context of search-based software engineering: a systematic review," *Journal of Systems and Software*, vol. 155, pp. 43–56, 2019.

[82] H. Y. Chiang and B. M. T. Lin, "A decision model for human resource allocation in project management of software development," *IEEE Access*, vol. 8, pp. 38073–38081, 2020.

[83] A. Costa, F. Ramos, M. Perkusich et al., "team formation in software engineering: a systematic mapping study," *IEEE Access*, vol. 8, pp. 145687–145712, 2020.

[84] S. Moon, "Skill development, bargaining power, and a theory of job design," *Journal of Economics and Management Strategy*, vol. 27, no. 2, pp. 270–296, 2018.

[85] O. A. Arık, "Project scheduling and staff allocation problem with time-dependent learning effect: a mixed integer nonlinear programming approach," *A - Applied Sciences and Engineering*, pp. 204–215, 2019.

[86] J. Kroll, S. Friboim, and H. Hemmati, "An empirical study of search-based task scheduling in global software development," in *Proceedings of the - 2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track*, pp. 183–192, Argentina, 2017.

[87] M. André, M. G. Baldoquín, and S. T. Acuña, "Formal model for assigning human resources to teams in software projects," *Information and Software Technology*, vol. 53, no. 3, pp. 259–275, 2011.

[88] S. T. Acuna, N. Juristo, A. M. Moreno, A. M. Moreno, and U. P. De Madrid, "Emphasizing human capabilities in software development," *IEEE Software*, vol. 23, no. 2, pp. 94–101, 2006.

[89] H. Zhu and S. Member, "Group multi-role assignment with coupled roles," in *Proceedings of the 2019 IEEE 16th International Conference on Networking, Sensing and Control*, pp. 281–286, ICNSC), Canada, 2019.

[90] B. Sergiy and V. Anastasiia, "Determination of competences that take affect the formation of creative capabilities of team of managers," in *Proceedings of the 2019 IEEE 14th International Conference on Computer Sciences and Information Technologies (CSIT)*, vol. 3, pp. 122–125, Ukraine, 2019.

[91] Y. Sunaga, H. Washizaki, K. Kakehi, Y. Fukazawa, S. Yamato, and M. Okubo, "Relation between combinations of personal characteristic types and educational effectiveness for a controlled project-based learning course," *IEEE Transactions on Emerging Topics in Computing*, vol. 5, no. 1, pp. 69–76, 2017.

[92] P.-C. Chen, C.-C. Chern, and C.-Y. Chen, "Software project team characteristics and team performance: team motivation as a moderator," *2012 19th Asia-Pacific Software Engineering Conference*, vol. 1, pp. 565–570, 2012.

[93] L. R. Vijayasarathy and C. W. Butler, "Choice of software development methodologies: do organizational, project, and team characteristics matter?" *IEEE Software*, vol. 33, no. 5, pp. 86–94, 2016.

[94] M. Shameem, C. Kumar, and B. Chandra, "A proposed framework for effective software team performance: a mapping study between the team members' personality and team climate," in *Proceeding of the - IEEE International Conference on Computing, Communication and Automation, ICCCA 2017*, pp. 912–917, India, 2017.

[95] U. Ashraf, C. Mayr-Dorn, and A. Egyed, "Mining cross-task artifact dependencies from developer interactions," in *Proceedings of the 2019 IEEE 26th International Conference on Software Analysis*, pp. 186–196, Evolution, and Reengineering, China, 2019.

[96] S. V. Shrivastava and U. Rathod, "Risks in distributed agile development: a review," *Procedia - Social and Behavioral Sciences*, vol. 133, pp. 417–424, 2014.

[97] M. Irfan, M. Hassan, and N. Hassan, "The effect of project management capabilities on project success in Pakistan: an empirical investigation," *IEEE Access*, vol. 7, pp. 39417–39431, 2019.

[98] J. G. Rivera-Ibarra, J. Rodríguez-Jacobo, J. A. Fernández-Zepeda, and M. A. Serrano-Vargas, "Competency framework for software engineers," in *Proceedings of the 23rd IEEE Conference on Software Engineering Education and Training*, pp. 33–40, Pennsylvania, PA, USA, 2010.

[99] M. Kropp and A. Meier, "Collaboration and human factors in software development: teaching agile methodologies based on industrial insight," *IEEE Global Engineering Education Conference EDUCON*, vol. 10-13, pp. 1003–1011, 2016.

[100] K. M. B. Da Silva and S. C. Dos Santos, "Critical factors in agile software projects according to people, process and technology perspective," in *Proceedings of the - 6th Brazilian Workshop on Agile Methods, WBMA 2015*, pp. 48–54, Brazil, 2015.

[101] X. Gai, C. Shen, and Y. Liu, "Trust estimation method via dependencies between components," in *Proceedings of the 2012 International Conference on Industrial Control and Electronics Engineering*, pp. 44–47, China, 2012.

[102] F. Mohd, S. Ismail, M. M. A. Jalil, F. I. N. M. Zulkarnain, and N. Zamin, *A Guidelines for Controlled Experimental Design to Evaluate the Metrics of Software Component Reusability* in *Proceedings of the - CAMP 2021: 2021 5th International Conference on Information Retrieval and Knowledge Management: Digital Technology for IR 4*, no. 59504, pp. 85–90, 2021.

[103] S. Mahmood, M. Niazi, and A. Hussain, "Identifying the challenges for managing component- based development in global software development : preliminary results," in

Proceedings of the Science and Information Conference, pp. 933–938, IEEE, 2015.

[104] G. A. Oliva and M. A. Gerosa, "On the interplay between structural and logical dependencies in open-source software," in *Proceedings of the - 25th Brazilian Symposium on Software Engineering*, pp. 144–153, Brazil, 2011.

[105] J. M. Erickson and R. Evaristo, "Risk factors in distributed projects," in *Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06)*, pp. 1–10, Washington, DC, USA, 2006.

[106] D. Damian, "Risk identification and risk mitigation instruments for global software development : systematic review and survey results," in *Proceedings of the Sixth IEEE International Conference on Global Software Engineering Workshops*, pp. 36–41, IEEE, Finland, 2011.

[107] J. S. Persson, L. Mathiassen, J. Boeg, T. S. Madsen, and F. Steinson, "Managing risks in distributed software projects: an integrative framework," *IEEE Transactions on Engineering Management*, vol. 56, no. 3, pp. 508–532, 2009.

[108] S. Y. Chadli, A. Idri, and L. Fern, "Frameworks for risk management in GSD projects : a survey," in *Proceedings of the 10th International Conference on Intelligent Systems: Theories and Applications (SITA)*, pp. 1–6, IEEE, Morocco, 2015.

[109] M. Hassan, "A policy recommendations framework to resolve global software development issues," in *Proceedings of the International Conference on Innovative Computing (ICIC)*, pp. 1–10, IEEE, India, 2019.

[110] S. A. Ruk, M. F. Khan, S. G. Khan, and S. M. Zia, "A survey on adopting agile software development: issues & its impact on software quality," in *Proceedings of the 6th IEEE International Conference on Engineering Technologies and Applied Sciences*, pp. 1–5, ICETAS), Malaysia, 2019.

[111] A. Albadarneh, I. Albadarneh, and A. Qusef, "Risk management in agile software development : a comparative study," in *Proceedings of the 2015 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies*, pp. 1–6, AEECT), 2015.

[112] A. Elbanna and S. Sarker, "The risks of agile software development: learning from adopters," *IEEE Software*, vol. 33, no. 5, pp. 72–79, 2016.

[113] M. Hammad, "Risk management in agile software development : a survey," in *Proceedings of the International Conference on Frontiers of Information Technology (FIT) Risk*, pp. 162–166, Pakistan, 2019.

[114] M. Kleehaus, C. Caprano, and F. Matthes, "Identifying and structuring challenges in large-scale Agile development based on a structured literature review," in *Proceedings of the 2018 IEEE 22nd International Enterprise Distributed Object Computing Conference*, pp. 191–197, EDOC), Sweden, 2018.

[115] E. Hossain, M. A. Babar, H. Paik, and J. Verner, "Risk identification and mitigation processes for using scrum in global software development : a conceptual framework," in *Proceedings of the 16th Asia-Pacific Software Engineering Conference*, pp. 457–464, IEEE, Washington, DC, USA, 2009.

[116] Z. Zahedi, S. Sengupta, and S. Kambhampati, "'Why not give this work to them?' explaining ai-moderated task-allocation outcomes using negotiation trees," pp. 1–8, 2020.