Received November 30, 2017, accepted January 26, 2018, date of publication February 16, 2018, date of current version April 4, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2803685

A Quantitative Framework for Task Allocation in Distributed Agile Software Development

WAQAR ASLAM AND FARAH IJAZ
Department of Computer Science and Information Technology, The Islamia University of Bahawalpur Pakistan, Bahawalpur 63100, Pakistan
Corresponding author: Waqar Aslam (waqar.aslam@iub.edu.pk)

ABSTRACT Distributed agile software development is a promising paradigm, addressing the necessities of emergent software application markets that are described by huge user base and small time to market characteristics. A key decision involved during the development process, is task allocation to team members. An appropriate task-member assignment facilitates project management, lessens the complexities and influences chances of project success. Task allocation becomes a more challenging activity in a distributed agile software development due to insufficient understanding of different factors and dependencies involved. We propose a task allocation framework comprising of two phases: one, identifying factors and dependencies that strongly influence the task allocation decision; two, proposing a quantitative method that allocates tasks to team members who best match the task requirements. Task requirements are expressed as capabilities, catering for different aspects, such as technical, personal, and environment. Our method stays transparent to the targeted objectives, in this case the best match. Other objectives, such as quality and cost may be introduced conveniently; even multiple objectives can be addressed. Such a method also allows quality evaluation of task-member assignment during and after the project completion, toward minimizing related risks.

INDEX TERMS Software engineering, Agile software development, distributed software development, task allocation method, task allocation framework.

I. INTRODUCTION

Software development process is organized to deliver software products faster and economical with enhanced quality. Agile methods, focusing on customer satisfaction, leverage achieving these objectives efficiently. They emphasize cooperation of individuals, sustain quick and cheap changeability and concentrate on producing functional software rather than strictly following recommended guidelines [1]. Distributed Software Development (DSD) involves teams working together to achieve project targets. In DSD, development sites can be physically distributed within a country or around the world involving multiple countries [2]. In DSD projects are developed out of various parts, where each part may be developed at a separate site. DSD offers global opportunities to further optimize Software Company objectives; for instance reduced time, less budget and enhanced quality, by relieving the confinement due to a single team.

Executing Agile manifesto (Agile Software Development is termed as ASD) in DSD gives rise to Distributed Agile Software Development (DASD), which offers excellent paybacks; for instance low development budget, possibility to engage increasing number of developers with proven better capabilities and experiences gained from around the world and adopting best practices. We exclusively abbreviate Accruing benefits from DASD is not straight forward, as it also gives rise to challenges that span on both paradigms: Agility and Distributivity [2]. These challenges include those due to multiplicity of teams, such as their interdependencies, poor communication and coordination skills, mistrust among teams, poor association among various sites due to spatial and temporal distance and differences in development cultures and languages [3]. In such a setup, task allocation to team members has a vital role that controls software cost and time concerns, which remain underpinned as top objectives of Software Companies.

Task allocation in non-transparent and unjustified manners can be a quality issue and causes decrease in productivity/motivation [4]. Whereas proper project planning is required for successful project development, in reality more than 40% of projects failed in China due to ineffectual arrangement about development tasks of software project and human assets [5].

Generally, the process of task allocation primarily relies on tasks identification. In Agile setups, a set of user stories are identified, with each user story decomposed into set of tasks that ought to deliver new functionalities. Iteration plans involve the details of responsible parties and corresponding task allocation [6]. In each iteration, the development team members pull out tasks by themselves and make self-assignments to their wish for completion. Self-inspiration in the development team members is exposed through this self-pull method. Adopting Agile practices efficiently by teams distributed over large geographic locations is difficult as timely decisions about task allocation are hampered due to absence of direct eye to eye discussions. Coordination required amongst team members to agree on task assignments, is weakened [7].

DSD being relatively an emerging paradigm, associated risks are non-trivial and even new risks emerge [8]. These risks can easily lead to sub optimal tasks allocation to team members. Task requirements call for capabilities that are unevaluated. Mainly the increased complexity of task allocation in DSD is caused by insufficient awareness about scattered locations, diverse time regions and possibly conflicting cultures: all this in contrast to collocated teams where skills of team member are well known, space is shared, as also culture and time zone [9]. An effective task allocation has an increased significance for optimal decisions [9] that influence benefits and minimize risks of DSD [10]. Generally task allocation is performed by the support of Scrum Master (Scrum software development) or Project Manager, whereas it is based on ability and availability. In such cases, it is commonly driven by past experiences of team members.

Another task related challenge is different types of dependencies. In Agile setups, there are task-task dependencies within same Sprints or different Sprints [11]. Task coordination among team members is required to manage these dependencies [12], [13]. In DSD, distributed sites have important dependencies on each other. These dependencies have strong communication as well as coordination requirements. Insufficient interaction among teams causes lack of team spirit and create rework problem. Other than these, many factors affect task allocation in DSD and strongly affect the project success [9]. A survey done on Project Managers highlights ad-hoc approaches used. Mostly the focus is on cost cutting that leads to project failures [10].

A task allocation plan should consider characteristics and relationships between distributed teams. These characteristics are strongly coupled and have strong impact on project duration and product quality [14]. One key characteristic is the expertise levels of team members. There is a need to assign tasks to the best available experts. Thus, effectively experts assume best possible roles [9], [15]. The expertise levels should be calculated deterministically considering capabilities of team members per roles.

Task allocation can be done as one-to-one, mapping to each team member a task or it can be done as many-to-one, mapping to each team member multiple tasks. An important objective of task allocation is to lessen the effort of task completion [16]. Similarly other objectives can be defined to cope up needs.

This leads us to define our research questions addressed in this paper:

Q1. Which factors and dependencies influence task allocation in DASD?

Q2. How to allocate tasks to team members quantitatively and without ambiguity?

To tackle Q1, we identify all factors impacting the development process in DASD. These factors have interdependencies, which must be considered during task allocation. Thus dependencies are also identified, so is their influence on software development process. To tackle Q2, we develop a mathematical method that allows task allocation per their requirements to best available experts per their capabilities to handle those tasks. The proposed method takes into account important determinants of tasks allocation optimality: experience length and past appraisal to perform specific tasks and relative importance of task requirements.

The structure of the paper is as follows. Section II offers related literature and highlights how our work departs from it. Section III and Section IV identify factors and dependencies in context of DASD. Our proposed method of task allocation in DASD is explained in Section V, while Section VI presents discussions about our framework and limitations of our method. Finally conclusions are drawn in Section VII.

II. RELATED WORK

Some observations in task allocation experiences are highlighted. Three organizations following DASD, have identified major communication challenges in their task related activities such as task allocation structure, coordination, management and levying the delivery responsibility of completed tasks [2]. Similarly time is wasted during re-orientation of tasks. Switching between tasks distracts team members. For instance, sometimes switching between four or five small tasks require more extra time for re-orientation than to work on these tasks [17]. Globally dispersed teams generally require about 2.5 times longer than collocated teams to complete their work due to lack of task coordination [13]. Self-organizing Agile team have reported that task level challenges of project management are lack of acceptance criteria and task dependencies. Lack of acceptance criteria means incomplete understanding of requirements that create task level problems. Task dependencies on other teams need toleration to interruptions. Work cancellation issue causes errors in effort estimation, triggering re-sizing and re-work [11].

Next, we report on the emergence of task allocation approaches. We identify different proposed methodologies and approaches taken in context of task allocation. For brevity, it suffices to list them in Table 1 along with brief descriptions.

TABLE 1. Proposed methodologies in the context of task allocation.

Year	Development Environment	Proposed Methodology	Description
2001 [16]	Software Development	Model	Predictability of a cost/effort estimation is improved (over COCOMO model) by considering various factors related to task assignment (development team, concurrency, intensity, fragmentation, etc.).
2002 [18]	Workflow Management System	Model	Evaluation of multi-criteria (ability, workload, tasks similarity, relationships among team members, etc.) based suitability of team members to support role based task allocation.
2004 [19]	Software Development	Genetic algorithm based optimization	Multi-objective optimization using genetic/evolutionary algorithms for task allocation. Allows objectives trade-offs to Project Managers and generates improved work schedules for team members.
2006 [20]	DASD	Method	Task allocation and coordination for day to day decisions using current practices of Software Project Management; for instance, object oriented process modeling and critical path analysis. A Java/Eclipse based distributed tool, NextMove is built.
2009 [21]	DSD	Bayesian networks based decision support model	Task allocation uses Bayesian networks and probability distributions (an extension of Bokhari's Algorithm). Model is validated using simulations of scenarios with random costs.
2009 [22]	Complex Software Systems	Resource negotiation based model	Physical and social contexts are considered for task allocation. Also features load balancing and communication costs reduction between allocated agents.
2009 [23]	DSD	Bayesian networks based method.	Task allocation optimizes weighted project objectives. A tool, TAMRI is built.
2013 [5]	Software Development	Event based scheduler	Plans of employees and task scheduling are based on ant colony optimization algorithm.
2013 [14]	DSD	Ontology of task allocation	The presented ontology validates the relationships in real distributed projects.
2014 [24]	ASD	Agile Project Management	Task allocation is based on exploratory data analysis that considers factors such as competence, equality, workload variation, time of completion and confidence of team members. A tool, HASE (Human-Centered Agile Software Engineering) is built.
2014 [25]	Software Development	Genetic algorithm based model	Collaborative software development using genetic algorithm, forms pairs of tasks towards optimal task-member assignments.
2015 [26]	DSD	Systematic literature review	Identifies influential task allocation factors (technical capabilities of experts, variance of time regions, vendor credibility, task volume, task dependency, vendor maturity level, etc.) and ignored task allocation factors (resident government rules, requirements solidity, product architecture, intellectual property rights, etc.).
2015 [13]	DSD	Task coordination portfolios	IT-mediated optimal task synchronization portfolios are based on circumstantial factors (different levels of task dependencies, perceived time constraints, temporal dispersion of development teams, etc.). 95 DSD teams are analyzed towards their performance enhancement.
2016 [27]	Distributed Systems	Survey study	A summary of task allocation approaches, highlights distributed systems, load balancing, control models, optimal resource utilization, dependability methods, models for network structures and coordination between heterogeneous nodes.
2016 [9]	DSD	Survey study	A summary of task allocation approaches highlights factors influencing related decisions and task importance ranking.
2017 [15]	DSD	Survey study	62 practitioners are surveyed about their task allocation experiences. Results highlight important factors such as location of experts, time dissimilarity between teams, task dependency/size, and vendor dependability.

III. TYPES OF FACTORS

Different factors are identified that influence task allocation in DASD (see Table 2). Understanding the nature of project and people has a foundational role, so some basic project and people related factors should be recognized [28]. A general analysis about task allocation in DSD environment also highlights the importance of these factors [9], [13]–[15], [26].

Expertise and knowledge are other key factors that mostly play a vital role in development activities. At times, specific expertise may be needed, so are technical skills and domain expertise, which have a strong effect on site selection decision. Site characteristics, such as analyst intelligence and adaptability, programmer approach to problem solutions based on available software/hardware tools and client proximity remain important during task allocation.

Task characteristics, for instance, application experience and platform experience should also be considered. Labor cost, considered as key aspect and meant for initiating DSD, has a significant impact on task allocation decisions, especially those related to distribution of tasks to various sites or teams. Practitioners define that labor cost is not the deciding factor; as communication and coordination cost

TABLE 2. Task allocation related factors (DSD and Agile related are shown in bold).

Project	People	Site	Task	Agile	Environment
Type, Quality, Hardware Software, Ease of operation, Complexity, Data transactions, Multiple site [28], **Product architecture** [10], Required budget, effort & deadline	Communication skills, Familiarity in team, Managerial skill, Security, Working Time, Experience of previous projects, **Technical ability** [28], **Expertise** [9]	**Team members knowledge and skills** [14], **Personnel availability** [9], **Task site specificity** [9], **Vendor reliability**, **Labor cost** [9], [10], **Workload at site[9]**, **Working time** [14]**, Time and cultural difference** [9], [10], [14], **Proximity to client/market** [10], **Number of sites** [4]	**Task size** [9], **Proximity to customer requirement** [14], **Required skills** [14], **Required resources** [14], **Task deadline** [14], Task priority [29]	**Transparency [30], Prioritized delivery, Enough documentation, Strong communication [31], Customer collaboration, Changeability**	**Personal reasons** [10], **Political reasons** [10], **Communication & coordination overhead** [9], **Type of DSD** [9], Organizational objective, Number of sites, Projects in progress, Specific development methodology

should also be considered, so that deciding factor should be development cost rather than only the labor cost. A strong coordination requirement among distributed teams is a critical factor for task allocation. Availability of team members should be assessed realistically for their consideration of engagement in other projects and related commitments. Thus workload factor should be managed to prevent overloading and choking the deliverables deadlines. Location, cultural and time difference create site dependencies, are also deciding factors during work assignment. In DSD, time difference can be defined as work day differences, start and end of the workday, holidays and cultural differences along with language differences. Political reasons among teams in DSD can also affect communication and coordination negatively.

Task related factors have a direct impact on allocation of resources to be consumed during development. An important aspect is prioritization of tasks, which can reflect business outcome value of the task. Sometimes customer demand is also addressed by priority [29].

Previous works differentiate between types of DSD environments: offshoring (same company but having offices in distributed locations), offshoring outsourcing (different companies and different countries) and companies supporting both types (offshoring and offshore outsourcing). These types of DSD have impact on development in terms of communication protocol they use. Offshoring companies having prior working relationships, most probably follow the same communication protocol for better coordination. The situation is different when outsourcing is involved [9], [10], [15]. It is observed that DSD control structures are either centralized or distributed. In centralized structure, team members report directly to the project manager, who performs coordination as well as control tasks using collaborative tools. However in distributed structure, team members interact directly with local coordinators, who report to the project manager regularly [15]. These two structure types leverage development of different cultures in companies.

The described factors are success factors. Their impact on project success may be ranked as critical, normal or low. Such ranks may be used to assign weights to path in Path Analysis or coefficients in Regression Analysis. Another form of factors, failure factors may be defined to build company history on full/partial case studies of projects that need to be avoided. Study of failure factors is significant especially when contradicting success factors emerge, hence tradeoff points with certain failure chances may be useful.

IV. TYPES OF DEPENDENCIES

Task dependencies refer to the condition when one task needs to be completed in order to initiate the subsequent task [11], or the progress of an action relies on the presence of a thing, where thing can be an artifact or a person or a piece of information [12]. Different categories of dependencies are described in Table 3; these should be recognized to achieve proper coordination during task allocation.

TABLE 3. Task allocation dependencies (DSD and Agile related are shown in bold).

Dependency	Category
Flow, Fit and Sharing	Basic [12], [33]
Workflow and Coordination	Work/Social [34]
Knowledge (Requirements, Expertise, Historical and Task-member assignment)	**ASD** [12]
Process (Activity and Business process)	
Resource (Entity and Technical)	
Tasks (Pooled, Sequential, Reciprocal and Team)	Inter-task [12], [13]
Distributed Environment (Geographical, Social/Cultural, Technical, Temporal, Informational, Communication and Organizational)	**DSD** [9], [35]

Coordination among activities is a key issue in software development, because coordination allow management of dependencies among activities [12]. Dispersion of teams globally presents extensive challenges to coordinating interdependent tasks. Coordination has strong impact on task performance when tasks have highly dependent activities. Insufficient coordination among development team is the major source of budget overruns and delays in addition to compromised quality of software product. Process rigor, process standardization and agility supports to solve issues due to distributed teams and requirements dynamism [35]. There are different categories of coordination issues in DSD, and these issues depend on the nature of concerned dependencies: for instance, technical, temporal, and process related coordination issues. Technical coordination problems arise when software inter-component dependencies are not managed effectively: for example, due to redundant code and incompatible interfaces, integration problems arise.

Temporal coordination problems occur when time dependencies are not efficiently managed: for example, finish-to-start type of dependencies. When software activities or software parts are not completed according to project schedule, other works are affected, such as testing phase cannot start prior to completion of coding. Occurrence and identification of dependencies during software development is normal. When these dependencies are not managed timely and systematically, process coordination problems arise. For example, problems arise due to lack of adhering to the established software processes and non-resolvability of priority conflicts [36]. Three basic types of dependencies are defined: flow, fit and sharing [12], [33]:

A. FLOW DEPENDENC

A situation in which an output of an activity is used by other activity, e.g., designer creates the design specification that is then used by developer.

B. FIT DEPENDENCY

A situation when multiple activities create outputs that have to fit together, e.g., the integration phase where all individual components have to fit together.

C. SHARING DEPENDENCY

A situation when multiple activities need to use some resource, usually limited, such as the time of an expert technical architect.

After defining the three basic dependencies, now more specialized dependencies are defined. Workflow dependencies relate developers throughout the evolution of modification request (MR). Work related dependencies emerge as work is done in different parts of a system. For instance, two developers might work on two different MRs involving files that are syntactically or logically interdependent. In that case, modifications made by each developer may affect others work. These types of work related dependencies are significantly complex in nature and involve more effort to identify and supervise. Results show that logical and work dependencies are more important, impacting the likelihood of source code files to exhibit field defects.

D. WORKFLOW DEPENDENCY

It represents definite relationships between project members based on workflows and/or processes.

E. COORDINATION DEPENDENCY

It corresponds to less explicit relationships between project members based on their past contributions to the development effort. Also technical dependencies of the system under development are considered [33].

From here on, we identify ASD based dependencies, which can support significant coordination practices. Knowledge, process and resource are three dependencies that have impact on ASD, though first one has a strong impact.

F. KNOWLEDGE DEPENDENCY

When information is required for project progress, it comes with dependencies such as requirements, expertise, historical and task-member assignment.

1) REQUIREMENT DEPENDENCY

Knowing requirements and in correct sense have a key role, absence of which affects the project progress.

2) EXPERTISE DEPENDENCY

It occurs whenever task related technical proficiency is limited to some individual or group. Three categories of experts are recognized in the field of software development: technical, design and domain. These types of knowledge are recommended to be shared in an ASD environment to lessen dependence on one person.

3) HISTORICAL DEPENDENCY

A situation in which knowledge about past decisions is required. There must be somebody who acts as a log of past events and related decisions.

4) TASK-MEMBER ASSIGNMENT

It reflects assigning task(s) to members, due to which project progress takes certain course.

G. PROCESS DEPENDENCY

A situation when a task is necessarily finalized prior to another task can proceed. It includes activity and business process dependencies.

1) ACTIVITY DEPENDENCY

A situation where in an activity cannot continue till another specific activity is finished, which has an impact on project development.

2) BUSINESS PROCESS DEPENDENCY

A situation in which a prevailing business practice mandates tasks to be carried out in a certain order.

H. COORDINATION DEPENDENCY

It corresponds to less explicit relationships between project members based on their past contributions to the development effort. Also technical dependencies of the system under development are considered [33].

From here on, we identify ASD based dependencies, which can support significant coordination practices. Knowledge, process and resource are three dependencies that have impact on ASD, though first one has a strong impact.

I. KNOWLEDGE DEPENDENCY

When information is required for project progress, it comes with dependencies such as requirements, expertise, historical and task-member assignment.

1) REQUIREMENT DEPENDENCY

Knowing requirements and in correct sense have a key role, absence of which affects the project progress.

2) EXPERTISE DEPENDENCY

It occurs whenever task related technical proficiency is limited to some individual or group. Three categories of experts are recognized in the field of software development: technical, design and domain. These types of knowledge are recommended to be shared in an ASD environment to lessen dependence on one person.

3) HISTORICAL DEPENDENCY

A situation in which knowledge about past decisions is required. There must be somebody who acts as a log of past events and related decisions.

4) TASK-MEMBER ASSIGNMENT

It reflects assigning task(s) to members, due to which project progress takes certain course.

J. PROCESS DEPENDENCY

A situation when a task is necessarily finalized prior to another task can proceed. It includes activity and business process dependencies.

1) ACTIVITY DEPENDENCY

A situation where in an activity cannot continue till another specific activity is finished, which has an impact on project development.

2) BUSINESS PROCESS DEPENDENCY

A situation in which a prevailing business practice mandates tasks to be carried out in a certain order.

K. RESOURCE DEPENDENCY

When an object is required for project success: it includes entity and technical dependencies.

1) ENTITY DEPENDENCY

A situation in which a resource (person, place or thing) waits on the availability of another resource. It can cause a busy block situation, which must be identified resolved.

2) TECHNICAL DEPENDENCY

A situation in which project progress is influenced by technical characteristics of development, such as due to interaction of one software component with another [12].

From here on, we identify four types of task interdependencies: pooled, sequential, reciprocal and team.

L. POOLED DEPENDENCE TASK

Individual team members finalize their work independently, which are then aggregated. For instance, each team member writes few modules of code independently before they are integrated.

M. SEQUENTIAL DEPENDENCE TASK

Individual team members finalize their work and deliver it to other team members. For instance, team members might design test cases relevant to his specialization in the overall test plan. After passing the test, the code and the test cases used, are input to the next team member.

N. RECIPROCAL DEPENDENCE TASK

The work alternates between team members. For instance, work alternates between programmers and testers throughout the modification process.

O. TEAM DEPENDENCE TASK

All the team work simultaneously for problem identification and proposing solution. For example, during the software requirement phase, all the team is engaged together in understanding the requirements and consensus building [12], [13], [37].

P. DISTRIBUTED ENVIRONMENT DEPENDENCY

Distributed sites have disparity issues of geographical, social/cultural, technical, temporal, informational, communication and organizational nature. Together, they are termed as DED. Due to these issues dependencies emerge, which when considered with other relevant factors, complexity increases during task allocation [9], [10].

Depending on the Software Development objective, a given task allocation strategy may produce conflicting or sub-optimal plan. An efficient Project Management in GSD may have underlying tradeoffs, which must be brought into limelight. For instance, characteristics of the sites and their relationships have significance during task allocation in DSD [21]. According to results of a systematic literature review [26], it is required to develop task distribution techniques and standards for Global Software Development to achieve potential benefits of development distribution such as lower costs and high quality.

It is important and pertinent to identify the stated dependencies during software development process. Most of these dependencies are illustrated in Fig. 1, which also shows how the relationships of team members are affected by these dependencies. It shows dependencies such as sequential dependency (SD), activity dependency (AD), fit dependency (FT), entity dependency (ED), technical dependency (TD), knowledge dependency (KD), team dependence task (TDT) and reciprocal dependence task (RDT). Product owner is the key stakeholder. According to Agile manifesto, software development process has Product Backlog containing user stories (USs) and Sprint Backlog containing

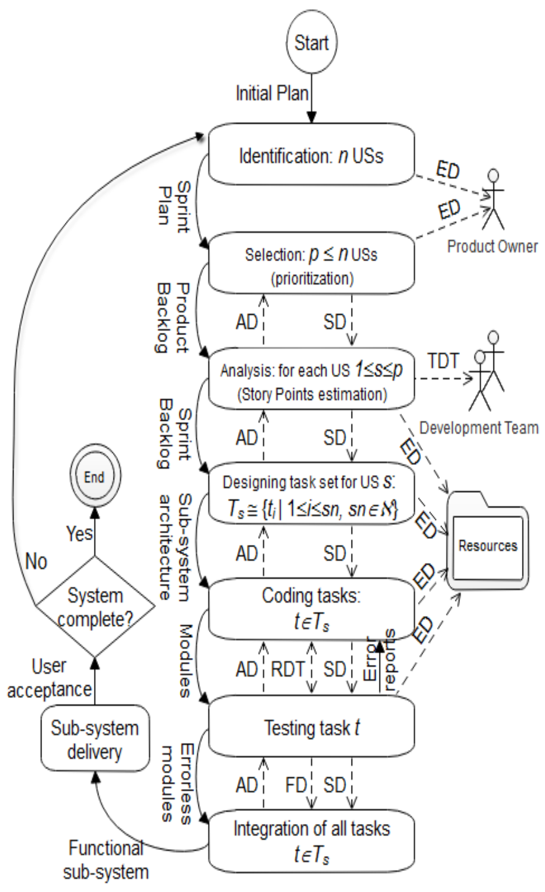

FIGURE 1. Software development process showing different types of dependencies. Given a project, n User Stories are identified, out of which p User Stories are selected on priority for the next Sprint s for which the task set is denoted by T_s. ℵ denotes the set of natural numbers. Other abbreviations are explained in text. Dependencies and flows are shown by dotted lines and solid lines respectively.

tasks list. As the project starts, an initial plan is chalked out, which after identification of USs, develops into a Sprint plan. After selection of user stories for Sprint, software development phases such as analysis, designing, coding, testing and integration having sequential and activity dependencies. Analysis phase also have team dependence tasks that depend on DT for maintaining Sprint Backlog. Integration phase includes SD and AD, also having FD. RDT affect coding and testing phases with two way forward and backward dependency. All the tasks for completion also depend on ED and KD. During Task allocation, these dependencies should be considered to decrease inconsistencies that may arise due to neglecting effect of these dependencies in DASD.

Dependencies between different sites in DSD are shown in Fig. 2, wherein distributed environment dependencies (DED) are shown. DED stays a strong candidate for consideration in a distributed setup, leveraging timely addressing the arising issues.

V. TYPES OF CAPABILITIES

Whatever method is used to allocate tasks to team members, ultimately factors and dependencies are translated into

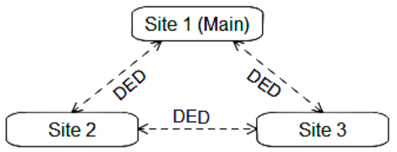

FIGURE 2. Distributed environment dependencies.

requirements, which are handled by appropriate expertise, hence its role has prime importance [9], [15]. This section focuses on this area and determines the required capabilities for various software roles.

Capabilities refer to the set of skills, awareness, attitudes, abilities, and manners that are used to assess performance and proficiency in a specific profession. They allow to compare suitability of team members to fulfill certain roles, even quality criteria demand per task can be imposed [38], [39]. Other than identifying suitability, we also achieve comparability using a quantitative method as developed in the next section. Different types of required capabilities per software roles are listed in Table 4. Mainly we are interested in listing required capabilities per software role, including those due to DASD environment.

VI. PROPOSED MODEL

This model assigns roles to team members based on their specific capabilities and experiences. A given team member may vary on these two aspects from other team members, irrespective of deployment sites. Independent of team member capabilities, the roles pose specific requirements. Based on their capabilities and experiences, these requirements are met by the assigned team members, fully or partially. We aim to evaluate the strength of assignment method quantitatively. Such quantification also enables comparability with other assignment methods.

In this paper, we use $\boldsymbol{M}$ notation to represent matrices. Let there be m roles needed to develop the given project with a possibility of each role having a varying number of requirements. We collect unique requirements of all roles. Let there be n requirements in this universal collection. A Boolean matrix representing requirements of all roles is created: $\boldsymbol{R} = \left[r_{i,j}\right] \in \{0, 1\}^{m \times n}$. $r_{i,j} = 0$ implies absence of requirement j in role i. $r_{i,j} = 1$ implies presence of requirement j in role i. For clarity $\boldsymbol{R} = \begin{bmatrix} r_{1,1} & \cdots & r_{1,n} \\ \vdots & \ddots & \vdots \\ r_{m,1} & \cdots & r_{m,n} \end{bmatrix}$.

Next we tag an arbitrary role and execute our method, which quantitatively evaluate the suitability of all team members to fulfill that role. Most suitable team member, one with the highest score, is selected to fulfill that role. In case of a tie between multiple team members, one with the lowest index value of some order is selected. The tagged role once assigned, is subtracted from the next evaluation. The process is repeated till all roles are assigned.

Let there be t team members ready to take m roles. In case $t \neq m$, either there are too few team members

TABLE 4. Types of required capabilities for software roles (DSD and Agile related are shown in bold).

Required Capabilities			Software Roles							
			Scrum Master	Product Owner	Quality Manager	Designer	Programmer	Maintenance Specialist	Tester	Configuration Manager
Technical	Technical Knowledge [39], [40]	Project supervision	✓							
		Requirement engineering		✓						
		Designing				✓				
		Programming					✓			
		Testing							✓	
		Quality			✓					
		Configuration management								✓
		Maintenance						✓		
	Tools [39]	Software tools competency	✓	✓	✓	✓	✓	✓	✓	✓
	Agile (Scrum)	**Maintain Product/Sprint Backlog**		✓						
		Organize Sprint Burn Down Chart	✓							
		Contributing during Planning Poker	✓	✓	✓	✓	✓	✓	✓	✓
	Methodological [41]	Making learning/working strategies	✓	✓	✓	✓	✓	✓	✓	✓
		Making testing strategies							✓	
		Presenting and reporting	✓	✓	✓	✓	✓	✓	✓	✓
		Researching	✓	✓	✓	✓	✓	✓	✓	✓
Non-Technical	Distributed [42]	**Distributed communication**	✓	✓	✓	✓	✓	✓	✓	✓
		Distributed cooperation	✓	✓	✓	✓	✓	✓	✓	✓
		Trust building	✓	✓	✓	✓	✓	✓	✓	✓
		Distributed tools competency	✓	✓	✓	✓	✓	✓	✓	✓
		Acculturation	✓	✓	✓	✓	✓	✓	✓	✓
	Interpersonal [39], [43]	Communication skills	✓	✓	✓	✓	✓	✓	✓	✓
		Concentration	✓	✓	✓	✓	✓	✓	✓	✓
		Customer service	✓	✓						
		Arrange Sprint meetings (daily, planning, review and retrospective)	✓	✓	✓	✓	✓	✓	✓	✓
	Team Cooperativeness [39], [41]	Self-responsibility	✓	✓	✓	✓	✓	✓	✓	✓
		Understandability	✓	✓	✓	✓	✓	✓	✓	✓
		Ideas share-ability and understandability	✓	✓	✓	✓	✓	✓	✓	✓
		Curiosity and self-motivation	✓	✓	✓	✓	✓	✓	✓	✓
	Solving conflicts [39] , [40]	Listen-ability	✓	✓	✓	✓	✓	✓	✓	✓
		Emotions control-ability	✓	✓	✓	✓	✓	✓	✓	✓
		Conflicts resolve-ability	✓	✓	✓	✓	✓	✓	✓	✓
	Professional development [39], [43]	Learnability	✓	✓	✓	✓	✓	✓	✓	✓
		Endeavour risks	✓	✓	✓	✓	✓	✓	✓	✓
		Resources management objectively	✓	✓	✓	✓	✓	✓	✓	✓
		Stress resistance-ability	✓	✓	✓	✓	✓	✓	✓	✓
		Progress monitoring	✓	✓						
		Adjustments for improvements	✓	✓	✓			✓		✓

or too many team members as compared to roles. This situation is termed as 'unBalanced Load' (uBL). If $t = m$, the situation is termed as 'Balanced Load' (BL). Ideally t task sets need to be generated so that team members can assume one role each. Multiple roles of same nature, for instance Programmer, can result, but unique identification of all roles, resolves the issue. Based on decision preferences such as those due to time constraints on project completion duration, more team members may be hired on temporary basis.

A simple suitability criterion for role assignment is given in [40]. We extend it by considering two determinants:

a. Past experience of team members (measured in same unit such as month) to address specific requirements of the tagged role i. Experience lengths are specified conveniently by a matrix. Let $\mathbf{E}^{(i)} = \left[e_{j,k}^{(i)}\right] \in \mathbb{R}^{t\times n}$ be the matrix of experiences of all team members corresponding to the universal collection of roles requirements. Here $\mathbb{R}$ is the set of Real numbers. Thus $e_{j,k}^{(i)}$ is the experience length of team member j to address the role requirement k. From this general representation, experience lengths extra to role requirements can be filtered out by using the Boolean matrix $\boldsymbol{R}$.
b. Past performance appraisal points corresponding to addressability of specific requirements of roles. These points are increasingly important, whereby management driven evaluation leverages objective development of employees. Let $\mathbf{P}^{(i)} = \left[p_{j,k}^{(i)}\right] \in \mathbb{R}^{t\times n}$ be the matrix of performance appraisal points of all team members corresponding to the universal collection of roles requirements. Thus $p_{j,k}^{(i)}$ is the performance appraisal points of team member j to address the role requirement k. Similar to the case of experience lengths, extra performance appraisal points can be filtered out by using the Boolean matrix $\mathbf{R}$.

The weighted sum of these determinants is termed as score, while it seeks an appropriate balance between the influences of these two determinants. The balance reflects on the past technical history of the Software Development Company in context of projects already completed. Thus score $=$ (experiencelength) $\times\, \alpha +$ (appraisalpoints) $\times\, (1-\alpha)$, where $0 \le \alpha \le 1$ is the weight used to balance between these two determinants. Let $s_{j,k}^{(i)}$ denote the score achieved by the team member j for addressing the role i requirement j, then

$$s_{j,k}^{(i)} = e_{j,k}^{(i)} \times r_{i,j} \times \alpha + p_{j,k}^{(i)} \times r_{i,j} \times (1-\alpha)\,. \qquad (1)$$

In (1), multiplication with $r_{i,j}$ filters out requirements irrelevant to a particular role. Scores of team members can vary drastically, so we consider its relative sense. It is determined relative to maximum score of all team members for a given requirement. Relative score is given by $rs_{j,k}^{(i)} = \frac{s_{j,k}^{(i)}}{\max_{1\le k\le t} s_{j,k}^{(i)}}$. As pointed out in [44], requirements don't have equal impact, so they may be weighted accordingly: $rs_{j,k}^{(i)} \times w_{j,k}^{(i)}$, where $w_{j,k}^{(i)}$ is the relative weight of role i requirement k for team member j. Given a role and a team member, it is the ratio between the relative score and sum of all relative scores per role requirements: $w_{j,k}^{(i)} = \frac{rs_{j,k}^{(i)}}{\sum_{k=1}^{n} 0 rs_{j,k}^{(i)}}$.

Our method, mapping a given role to a team member, is computed as

$$\mathrm{f} : \mathbb{N} \to \mathbb{N},$$
$$\mathrm{f}\,(i) = \arg\left(\max_{1\le j\le t}\left(\sum\nolimits_{k=1}^{n} rs_{j,k}^{(i)}\right)\right). \qquad (2)$$

$\arg(\cdot)$ maps role i to the index of team member with maximum cumulative relative score. $\mathbb{N}$ is the set of Natural numbers. Best members for all roles are given by the mapping $\mathrm{f}\,(i)\,, 1 \le i \le m$.

VII. DISCUSSION AND LIMITATIONS

We are working on a task allocation framework in DASD that can consider influential factors and dependencies. To this end, we identified them with a focus on DASD setup, which tends to introduce complexities for decision making. One important decision is allocation of tasks to team members. This decision is mostly taken qualitatively and lacks in a deterministic way, hence involves risks that stay for future/subsequent projects. The gap is covered by aiming for a quantitative method that we introduced. This method considers the capability requirements per role needed to execute the project. For each role, there is a selection of the best team member based on the past experience length, past appraisal for undertaking similar tasks and relative importance of capability requirements.

Our framework requires a pool of tasks and team members per site. Goodness of our method depends on appropriate pooling. The solution space is restricted by the boundaries of the defined pool. A key point to note is that our task-member mapping is individual. It does not consider site collective views. These views can be formed, for instance, to cater for cost and quality concerns at the site levels. Another point to note is that our method stays valid when required project effort needs to be divided among various sites. Local decisions can be left to the site managers, who can map tasks to members.

VIII. CONCLUSIONS

There is an increasing trend of software development in distributed Agile setups, especially in the emerging software application market with immense user bases. In contrast to two decades back and earlier, when software usage patterns would take years to emerge and settle, nowadays patterns are sharp with exponential times. Such software usage market has a direct impact on software development processes. Of all, those based on two paradigms, Agility and distributiveness, have competitive development advantage, as markets require. We support this notion by forming our first research question (Q1.), which is addressed by identifying factors and dependencies in DASD setups. To lessen the time to market for emerging software products, we deem it important to quantify the process of task allocation objectively. This aspect we pose as second research question (Q2.), which is addressed by defining an objective of task allocation to the best member from the available pool.

Currently we search in an n dimension solution space. In future we plan to extend this framework by forming collective site views to define our site-global objectives. Such task allocation objectives will extend our solution space to wider areas. Also it would be interesting to compare results of the proposed method with other searching algorithms such as those based on heuristics and genetics.

REFERENCES

[1] K. N. Rao, G. K. Naidu, and P. Chakka, "A study of the agile software development methods, applicability and implications in industry," *Int. J. Softw. Eng. Appl.*, vol. 5, no. 2, pp. 35–45, 2011.

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

[3] S. V. Shrivastava and U. Rathod, "Categorization of risk factors for distributed agile projects," *Inf. Softw. Technol.*, vol. 58, pp. 373–387, Feb. 2015.

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

[5] W.-N. Chen and J. Zhang, "Ant colony optimization for software project scheduling and staffing with an event-based scheduler," *IEEE Trans. Softw. Eng.*, vol. 39, no. 1, pp. 1–17, Jan. 2013.

[6] S. Amjad *et al.*, "Calculating completeness of agile scope in scaled agile development," *IEEE Access*, to be published.

[7] J. Lin, "Context-aware task allocation for distributed agile team," in *Proc. 28th IEEE/ACM Int. Conf. Autom. Softw. Eng. (ASE)*, Nov. 2013, pp. 758–761.

[8] W. Aslam, F. Ijaz, M. I. Lali, and W. Mehmood, "Risk aware and quality enriched effort estimation for mobile applications in distributed agile software development," *J. Inf. Sci. Eng.*, vol. 33, no. 6, pp. 1481–1500, 2017.

[9] S. Imtiaz and N. Ikram, "Dynamics of task allocation in global software development," *J. Softw., Evol. Process*, vol. 29, no. 1, Jan. 2016.

[10] A. Lamersdorf, J. Munch, and D. Rombach, "A survey on the state of the practice in distributed software development: Criteria for task allocation," in *Proc. 14th IEEE Int. Conf. Global Softw. Eng.*, Jul. 2009, pp. 41–50.

[11] R. Hoda and L. K. Murugesan, "Multi-level agile project management challenges: A self-organizing team perspective," *J. Syst. Softw.*, vol. 117, pp. 245–257, Jul. 2016.

[12] D. E. Strode, "A dependency taxonomy for agile software development projects," *Inf. Syst. Frontiers*, vol. 18, no. 1, pp. 23–46, Feb. 2016.

[13] J. Sutanto, A. Kankanhalli, and B. C. Y. Tan, "Investigating task coordination in globally dispersed teams: A structural contingency perspective," *ACM Trans. Manage. Inf. Syst.*, vol. 6, no. 2, pp. 1–31, Jul. 2015.

[14] A. B. Marques, J. R. Carvalho, R. Rodrigues, T. Conte, R. Prikladnicki, and S. Marczak, "An ontology for task allocation to teams in distributed software development," in *Proc. IEEE 8th Int. Conf. Global Softw. Eng.*, Aug. 2013, pp. 21–30.

[15] S. Mahmood, S. Anwer, M. Niazi, M. Alshayeb, and I. Richardson, "Key factors that influence task allocation in global software development," *Inf. Softw. Technol.*, vol. 91, pp. 102–122, Nov. 2017.

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

[17] M. Korkala and F. Maurer, "Waste identification as the means for improving communication in globally distributed agile software development," *J. Syst. Softw.*, vol. 95, pp. 122–140, Sep. 2014.

[18] M. Shen, G.-H. Tzeng, and D.-R. Liu, "Multi-criteria task assignment in workflow management systems," in *Proc. 36th Annu. Hawaii Int. Conf. Syst. Sci.*, Jan. 2003, pp. 1–9.

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

[20] D. K. M. Mak and P. B. Kruchten, "Task coordination in an agile distributed software development environment," in *Proc. Can. Conf. Elect. Comput. Eng.*, May 2006, pp. 606–611.

[21] A. Lamersdorf, J. Münch, and D. Rombach, "A decision model for supporting task allocation processes in global software development," in *Proc. 10th Int. Conf. Product-Focused Softw. Process Improvement (PROFES)*, Oulu, Finland, Jun. 2009, pp. 332–346.

[22] Y. Jiang and J. Jiang, "Contextual resource negotiation-based task allocation and load balancing in complex software systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 20, no. 5, pp. 641–653, May 2009.

[23] A. Lamersdorf and J. Munch, "TAMRI: A tool for supporting task distribution in global software development projects," in *Proc. 4th IEEE Int. Conf. Global Softw. Eng.*, Jul. 2009, pp. 322–327.

[24] 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 *Proc. 29th ACM/IEEE Int. Conf. Autom. Softw. Eng.*, Vasteras, Sweden, 2014, pp. 689–694.

[25] S. Chakraverty, A. Sachdeva, and A. Singh, "A genetic algorithm for task allocation in collaborative software developmentusing formal concept analysis," in *Proc. Int. Conf. Recent Adv. Innov. Eng. (ICRAIE)*, May 2014, pp. 1–6.

[26] S. Mahmood, S. Anwer, M. Niazi, M. Alshayeb, and I. Richardson, "Identifying the factors that influence task allocation in global software development: Preliminary results," in *Proc. 19th Int. Conf. Eval. Assessment Softw. Eng.*, Nanjing, China, Apr. 2015, pp. 1–6.

[27] Y. Jiang, "A survey of task allocation and load balancing in distributed systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 2, pp. 585–599, Feb. 2016.

[28] R. Popli and N. Chauhan, "Agile estimation using people and project related factors," in *Proc. Int. Conf. Comput. Sustain. Global Develop. (INDIACom)*, Mar. 2014, pp. 564–569.

[29] Z. Masood, R. Hoda, and K. Blincoe, "Motivation for self-assignment: Factors agile software developers consider," in *Proc. 10th Int. Workshop Cooperat. Human Aspects Softw. Eng.*, Buenos Aires, Argentina, May 2017, pp. 92–93.

[30] M. Kropp and A. Meier, "Collaboration and human factors in software development: Teaching agile methodologies based on industrial insight," in *Proc. IEEE Global Eng. Educ. Conf. (EDUCON)*, Apr. 2016, pp. 1003–1011.

[31] K. M. B. da Silva and S. C. dos Santos, "Critical factors in agile software projects according to people, process and technology perspective," in *Proc. 6th Brazilian Workshop Agile Methods (WBMA)*, Oct. 2015, pp. 48–54.

[32] T. W. Malone *et al.*, "Tools for Inventing Organizations: Toward a Handbook of Organizational Processes," *Manage. Sci.*, vol. 45, no. 3, pp. 425–443, 1999.

[33] M. Cataldo, A. Mockus, J. A. Roberts, and J. D. Herbsleb, "Software dependencies, work dependencies, and their impact on failures," *IEEE Trans. Softw. Eng.*, vol. 35, no. 6, pp. 864–878, Nov./Dec. 2009.

[34] S. Imtiaz, "Architectural task allocation in distributed environment: A traceability perspective," in *Proc. 34th Int. Conf. Softw. Eng. (ICSE)*, Jun. 2012, pp. 1515–1518.

[35] G. Lee, J. A. Espinosa, and W. H. DeLone, "Task environment complexity, global team dispersion, process capabilities, and coordination in software development," *IEEE Trans. Softw. Eng.*, vol. 39, no. 12, pp. 1753–1771, Dec. 2013.

[36] J. A. Espinosa, S. A. Slaughter, R. E. Kraut, and J. D. Herbsleb, "Team knowledge and coordination in geographically distributed software development," *J. Manage. Inf. Syst.*, vol. 24, no. 1, pp. 135–169, Jul. 2007.

[37] N. A. Staudenmayer, "Interdependency: Conceptual, empirical, & practical issues," Tech. Rep., 1997.

[38] T. Ley and D. Albert, "Identifying employee competencies in dynamic work domains: Methodological considerations and a case study," *J. Univ. Comput. Sci.*, vol. 9, no. 12, pp. 1500–1518, 2003.

[39] J. G. Rivera-Ibarra, J. Rodríguez-Jacobo, and M. A. Serrano-Vargas, "Competency framework for software engineers," in *Proc. 23rd IEEE Conf. Softw. Eng. Edu. Training*, Mar. 2010, pp. 33–40.

[40] S. T. Acuna, N. Juristo, and A. M. Moreno, "Emphasizing human capabilities in software development," *IEEE Softw.*, vol. 23, no. 2, pp. 94–101, Mar. 2006.

[41] V. Thurner, A. Böttcher, and A. Kämper, "Identifying base competencies as prerequisites for software engineering education," in *Proc. IEEE Global Eng. Edu. Conf. (EDUCON)*, Apr. 2014, pp. 1069–1076.

[42] C. Gold, J. Abke, and Y. Sedelmaier, "A retrospective course survey of graduates to analyse competencies in software engineering," in *Proc. IEEE Global Eng. Edu. Conf. (EDUCON)*, Apr. 2014, pp. 100–106.

[43] M. Paasivaara, C. Lassenius, D. Damian, P. Räty, and A. Schröter, "Teaching students global software engineering skills using distributed Scrum," in *Proc. 35th Int. Conf. Softw. Eng. (ICSE)*, May 2013, pp. 1128–1137.

[44] S. Čelar, M. Turić, and L. Vicković, "Method for personal capability assessment in agile teams using personal points," in *Proc. 22nd Telecommun. Forum Telfor (TELFOR)*, Nov. 2014, pp. 1134–1137.

WAQAR ASLAM received the Ph.D. degree in computer science from The Eindhoven University of Technology, The Netherlands. He is currently an Assistant Professor with the Department of Computer Science and Information Technology, The Islamia University of Bahawalpur Pakistan. His research interests include performance modeling of (distributed) software architectures, effort/time/cost estimation of software development in (distributed) agile setups, performance modeling and QoS of wireless/computer networks, and social network data analysis. He received HEC Oversees Scholarship for the Ph.D. degree.

FARAH IJAZ received the M.S. degree in computer science from the Department of Computer Science and Information Technology, The Islamia University of Bahawalpur Pakistan. She is currently a Visiting Lecturer with The Islamia University of Bahawalpur Pakistan, where she is also involved in research. Her research interests include software requirement management in distributed environment using agile methods with focus on effort/time/cost estimation.