Information Technology and Management (2022) 23:167–192
https://doi.org/10.1007/s10799-022-00365-9

Task allocation and coordination process in distributed agile software development: an ontology based approach

Chitra Nundlall[1] · **Soulakshmee D. Nagowah**[1]

Accepted: 9 April 2022 / Published online: 10 May 2022

Abstract
Distributed agile software development (DASD) has gained much popularity over the past years. It relates to Agile Software Development (ASD) being executed in a distributed environment due to factors such as low development budget, emerging software application markets and the need for more expertise. DASD faces a number of challenges with respect to coordination and communication issues. Task allocation in such an environment thus becomes a challenging task. Adopting proper task allocation strategy is crucial to overcome challenges and issues in DASD. Various studies highlight the challenges being faced by DASD and have proposed solutions in the form of framework or models. Knowledge models in the form of ontologies can help to solve certain issues and challenges by providing a proper representation of data that is shareable among distributed teams. Several ontologies with respect to task allocation exist. However, ontologies incorporating factors and dependencies influencing task allocation process in DASD are limited. An ontology representing the knowledge related to task allocation and coordination is important for proper decision making in organizations. Based on an in-depth literature review and a survey conducted among professionals in industry, this paper proposes an ontology, *OntoDASD*, that incorporates relevant factors and dependencies to be considered in task allocation and coordination process in DASD environment. The ontology facilitates team coordination through effective communication and task allocation by defining the concepts to share knowledge and information in an appropriate way. *OntoDASD* has been properly evaluated and validated by professionals in the field.

Keywords Distributed agile software development · Task allocation · Coordination · Ontology

1 Introduction

Distributed software development (DSD) has gained popularity from the last decade due to the high competition in software market. It aims at serving customers around the world and exploiting the advantages of low-cost workforce [1]. DSD allows work to be done simultaneously by various teams spread across the world [2]. ASD is another paradigm that became popular in the past decade due to its benefits such as customer satisfaction and cooperation of individuals to achieve objectives efficiently [3]. DSD companies, which have expanded their business globally, are following the same philosophy of ASD such as self-assigning tasks, open floor discussions and coordination [4]. This agile principle is best suited for collocated teams as it promotes effective coordination amongst various teams working at the same geographical location [5]. However, when applied in distributed environments, intra and inter team coordination challenges crop up, leading to delays in communication [2, 6]. Layman et al. [7] reported that lack of informal communication, in turn, leads to low levels of trust and awareness of work and progress at remote sites.

Task allocation in a distributed setting is another critical activity to be tackled. Task allocation in DASD is shared within an empowered team and is not the sole responsibility of the project manager [8]. Frustration among team members, lack of motivation, low quality and inaccurate estimates are the issues that result from task allocation performed by solely the project manager [8]. Proper task allocation attempts to reduce the time spent for completing projects [53]. In DASD, a number of factors exist that impact task allocation and coordination process [9]. Studying

✉ Soulakshmee D. Nagowah
s.ghurbhurrun@uom.ac.mu

Chitra Nundlall
shwetanundlall@gmail.com

[1] Department of Software and Information Systems, Faculty of Information, Communication and Digital Technologies, University of Mauritius, Moka, Mauritius

these factors is crucial so that the project success is not hindered [9]. Literature highlights the need to consider dependencies as well during decision-making [10–12]. Insufficient attention to dependencies can contribute to unsatisfactory progress as they may constrain work progress [10]. Researchers are still undergoing several studies in the area and are still looking for effective solutions to solve the task coordination challenges. As mentioned by Rocha et al. [13], lack of communication, management and cultural differences are the obstacles for efficient distribution of team to carry out a certain activity. Some of the challenges in task assignment are documentation, pair programming, working hours of different sites and cultural differences [14]. Lack of documentation in distributed setting can cause rework and waste of time [14]. Additionally, the team tends to miss out important information [14]. Different working hours lead to poor coordination between teams. Planning challenges also impact task allocation process [11]. Lack of planning or requirement gathering gives rise to changes in current plan, increasing backlog, which has to be reprioritized.

In a distributed environment consisting of different teams and different sites, it is important to share high-level knowledge regarding projects information and staff details among others so that there is a homogeneous comprehension of the project's information as well as staff involved (their expertise and availability). There is a need for a common understanding of the terms and concepts related to the task allocation and coordination process in DASD projects. Research has shown that the integration of software engineering and semantic web technologies has led to the sharing and reuse of knowledge especially when teams are located in different geographical areas [15, 16]. A knowledge-based model such as an ontology can help in knowledge sharing and knowledge transfer among these distributed teams in DASD and thus resolve communication and coordination issues [17, 18]. Rocha et al. [18] add that such an approach enhances task allocation and coordination process by preventing task misinterpretation. Ontologies have been used in different phases of the software development process as classified by Vizcaíno et al. [16] and Martínez-García et al. [19]. Gruber [20] defines an ontology as an "explicit specification of a conceptualization". An ontology represents "the effort to formulate an exhaustive and rigorous conceptual schema within a given domain, typically a hierarchical data structure containing all the relevant elements and their relationships and rules (regulations) within the domain." [21]. It is a knowledge representation model that defines concepts and properties in the domain [22].

The general question used for this research was: *Is it possible to create an ontology to resolve coordination and task allocation challenges in DASD?* In an attempt to address this research question, an ontology, entitled *OntoDASD* is proposed, developed and evaluated. The ontology aims to represent relevant factors and dependencies to enhance the task allocation and coordination process across teams in a distributed environment. In order to develop the knowledge–based model, the following objectives were defined:

- Review existing ontologies from miscellaneous papers in the field and perform an in-depth analysis
- Describe the methodology for ontology development
- Define a list of competency questions that the ontology should answer
- Provide a common vocabulary for team members and project managers to assist them in the decision-making by defining concepts and properties
- Develop the ontology in OWL
- Demonstrate the use of the ontology to answer above competency questions by presenting SPARQL query results
- Evaluate the ontology

The remainder of the paper is organized as follows: Sect. 2 provides a summary of the related works. Section 3 describes the methodology adopted to develop the ontology. Section 4 describes the evaluation process where the ontology has been assessed. Section 5 concludes on the work done and the results.

2 Related work

In this section, ontologies and taxonomy in the field of task allocation and coordination are described. The main goal of studying the related works was to investigate whether the existing ontologies satisfy the requirements to meet our objectives defined in Sect. 1. The ontologies are further compared in Table 1.

Rajpathak et al. [23] proposed a generic task ontology for scheduling problems. Though the ontology is developed for scheduling application, the key concepts adopted are constraints, cost, activity, task and resource, which are the factors necessary for task allocation and coordination. The relationships between the various entities demonstrate the conceptual model for task scheduling.

Almeida et al. [24] proposed concept mapping consisting of three different concept hierarchies namely general, Global Software Development (GSD) and scrum criteria. The authors used the term GSD to represent DSD. The model was based on the input and judgement of project managers and applied multi-criteria approach focusing on cost reduction and task allocation [24]. The study targeted DASD using scrum methodology. It describes general and DSD related factors along with factors related to the agile methodology adopted in scrum.

Table 1 Analysis of related works

Papers	Factors	Dependencies	Task Allocation process	DSD	Agile software development	Taxonomy	Ontology
[25]	✓		✓	✓			✓
[24]	✓		✓	✓	✓		
[10]		✓	✓	✓	✓	✓	
[12]		✓	✓		✓	✓	
DKDOnto [13]	✓			✓			✓
[23]	✓		✓				✓
[19]	✓				✓	✓	✓
[55]	✓						✓

Strode and Huff [12] have investigated on dependencies in agile software development. The paper proposed a taxonomy consisting of task, resource and knowledge dependencies. Requirement, expertise, task allocation and historical dependencies were grouped into knowledge dependency. Activity and business process were grouped into task dependency. Entity and technical dependencies were grouped into resource dependency. The taxonomy was proposed to project stakeholders as a tool for decision making to prevent them from facing issues in their projects.

Marques et al. [25] proposed an ontology for task allocation to teams in distributed software development based on the concepts proposed by Mak and Kruchten [26]. The ontology is composed of 6 fragments: *Artifacts*, *Activities*, *Competences*, *Teams*, *Organization* and *Project team*. *Artifacts*, *Activities* and *Teams* are the concepts proposed by Mak and Kruchten [26] for task coordination in an agile distributed software development. The work by Marques et al. [25] has been considered most relevant to this research as it focused on DSD and had similar objectives. However, the authors have taken into consideration some but not all factors that affect task allocation and coordination in their model. The authors presented their ontology in fragments with SPO label for *Software Process Ontology* [27] and SEO label *for Software Enterprise Ontology* [28]. *Artifact* englobes the resulting products in the project. *Activity* factor was taken into consideration in the ontology. Other factors like competence, time, cost and activity constraint were also considered. *Competences* include factors like knowledge, skills, experience, fluency level, language, and person and performance history. *Team* incorporates factors like cost, historical work, team work period, cultural alignment for an effective coordination in DSD. Factors like time zone, site locations, and organization role are linked to *Organization*. *Project team* relates goal, human resource, team and project showing that different projects have different priorities. The realization of goal is done by activity.

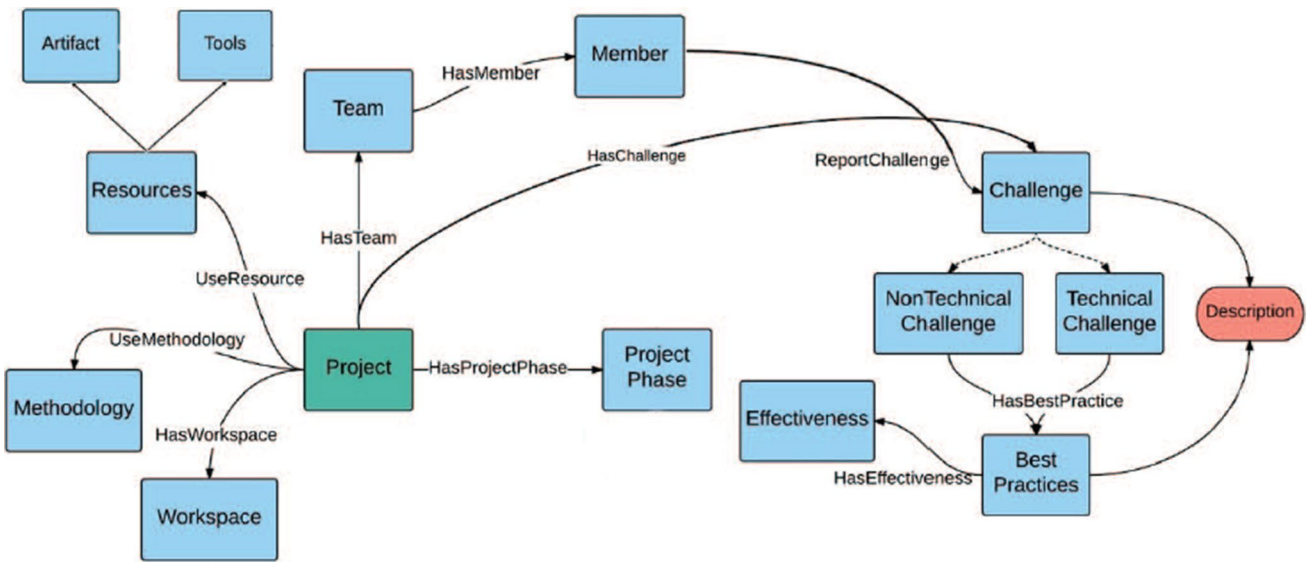

Fig. 1 DKDOnto ontology [13]

Rocha et al. [13] proposed an ontology named *DKDOnto* to support software development with distributed teams. The main elements of the ontology as shown in Fig. 1 are project, member, best practices, challenges, skills, place, artifact, tools and methodology. *Project* stores all information related to each phase in the project. *Member* includes team member and any other resource who form part of the project. *BestPractices* are the measures to be adopted in order to overcome the challenges in the project. A tool based on *DKDOnto* has been implemented to support the software development process with distributed teams [54].

Ijaz and Aslam [10] studied the dependencies in task allocation during DASD with the aim to recognize different types of issues on time before they influence the software product achievement. In DASD, insufficient attention to these dependencies at both task level and project level can lead to agile sprint cancellation or delay and this has a direct impact on customer satisfaction [10]. The authors proposed a taxonomy on different types of dependencies such as basic ones (flow, fit, sharing and component), software and task related, agile process and distributed environment related. The authors conducted an in-depth study on the dependencies in DASD.

As minimal documentation is prioritized in agile software development, knowledge tends to vaporize leading to issues like poor understanding of requirements and technical solutions, delay in software development projects [19]. It is also difficult to search for artefacts and experts. Martinez-Garcia et al. [19] aimed at condensing knowledge by using an ontology to provide an information structure that can perform automated reasoning about knowledge. The authors adopted Methontology approach to come up with the ontology. Factors like profile of the expert, artifacts, programming knowledge and projects are captured by the ontology.

Anzures-Garcia et al. [55] proposed a workflow ontology for a group's organizational structure. The aim was to help computer supported cooperative work (CSCW) teams to overcome the problems of communication, collaboration and coordination. This work was found related as it aims at achieving some of our objectives. The ontology is composed of an appropriate base of knowledge to represent CSCW systems development. The workflow ontology demonstrates the steps to develop these kinds of systems and allows adaptations. The factors considered in the research work are tasks and stage of group's organizational structure, status, policy, right, activity and resource, priority of each stage and roles at each stage.

Table 1 presents the comparison between existing ontologies based on the most relevant criteria for this research work:

- *Factors*: Are all the main factors found in the literature combined in an ontology?
- *Dependencies*: Are all the dependencies incorporated in an ontology?
- *Task Allocation*: Is the main focus of the ontology to solve task allocation problem?
- *DSD:* Did the study extend their approach to distributed teams?
- *Agile software development*: Did the study cater for agile software development?
- *Taxonomy:* Did the study propose a taxonomy?
- *Ontology*: Was an ontology developed?

Despite the existence of ontologies and taxonomies in the domain, none have really proposed a task allocation and coordination ontology that achieves all the objectives of this study. It can be observed from Table 1 that while some research works made use of factors namely [13, 23–25], others have incorporated dependencies in the task allocation namely [10, 12]. Both factors and dependencies influence the task allocation process [9]. There is thus the need for a knowledge model that incorporates both factors and dependencies to ensure a proper task allocation and coordination. There exist a few studies in this field namely [4, 10, 11, 29–31] in the literature about factors, challenges and methods for task allocation and coordination in agile distributed software environment. However, these studies did not propose an ontology in that area.

Marques et al. [25] have proposed an ontology for DSD but did not incorporate agile software development process. The authors only presented the concepts and their relationships and sought expert opinions to validate the ontology. They did not present the implementation in detail. Almeida et al. [24] presented important factors that should be considered in agile methodology but was limited to concept hierarchy mapping. Ijaz and Aslam [10] have proposed a taxonomy on DASD and Strode and Huff [12] on ASD focusing only on dependencies. Rocha et al. [13] proposed *DKDOnto*, open for any ontology but the authors did not consider dependencies in their ontology. In the ontology evaluation, the authors used reasoners and other tools for verification. The research work by Rajpathak et al. [23] was considered least relevant as it was developed for scheduling applications. Anzures-Garcia et al. [55] do not consider overcoming task allocation issues and the work proposed does not cater for distributed agile software development. The proposed workflow ontology only shows the steps for the development of a CSCW system. The factors considered are limited for CSCW systems. The ontology by Martinez-Garcia et al. [19] is limited to coding phase in the software development process. A limited number of factors was considered for the ontology development. The authors focused more on illustrating the methodology adopted to come up with the ontology than the criteria taken into

consideration. The ontologies [19] and [55] have been used for informational purpose and not transactional.

The aim of this research work is therefore to address the shortcomings of existing works and to propose an ontology that incorporates all relevant factors and dependencies affecting task allocation and coordination in DASD. Variations in the concept mapping presented in the existing studies lead to challenges for common and shareable representation of factors. The objectives are to minimize variations in the concept mapping by reusing terms and vocabularies already in the literature. The ontology also aims to illustrate how the factors and dependencies help in assisting task allocation and coordination.

3 Methodology

This section describes the methodology adopted to develop *OntoDASD* ontology for DASD. NeOn methodology by Suarez-Figueroa et al. [32] has been chosen due to its advantages of reusing existing ontological resources and re-engineering concepts to transform non ontological resources components into ontology representation style [33]. The overview of the development approach is shown in Fig. 2. There are seven stages namely initiation, reuse, reengineering, merging, modelling, implementation and maintenance phase. The detailed description of each phase is explained in the following sections.

3.1 Initiation phase

This phase describes the knowledge acquisition process for the ontology domain though a literature review as well as gathering information from experts in the domain. As initial procedures, a Systematic Literature Review (SLR) was conducted to find out the factors and dependencies affecting task allocation and coordination in DASD [35]. A survey was then designed to get experts' opinions to validate these factors and dependencies [36]. The output of this phase is a motivation scenario and an Ontology Requirements Specification Document (ORSD). Competency questions present in the ORSD, are later used for evaluation purposes.

3.1.1 Literature review

After analyzing the related works, an SLR was necessary to identify the factors influencing task allocation and coordination in DASD. People characteristics, site characteristics, environment, task characteristics, project and agile factors were identified in the SLR [35]. Table 2 depicts the list of shortlisted factors, which were found in different publications. The shortlisting was based on the number of occurrences in previous work. The factors in Table 2 are the most frequently mentioned factors in literature.

While analyzing the related works in the previous section, it was found that only two studies by Ijaz and Aslam [10] and Strode and Huff [12] researched on dependencies in DASD. A dependency is referred to a process or task that relies on an action in a project to happen in order to progress [9]. Dependencies between tasks, people, resources, requirements, expertise and others are required for an effective communication and coordination strategy for a good flow of information. This motivated us to research on the different dependencies that exist within a project and within a team. Dependencies related to knowledge, process and resources in DASD were retrieved from literature [9]. Table 3 depicts the list of shortlisted dependencies, which were found in previous articles. The shortlisting was based on the number of occurrences in previous work. The dependencies in Table 3 are the most frequently mentioned dependencies in literature.

3.1.2 Survey

To validate the results obtained from literature review, a survey was conducted with 25 agile participants globally

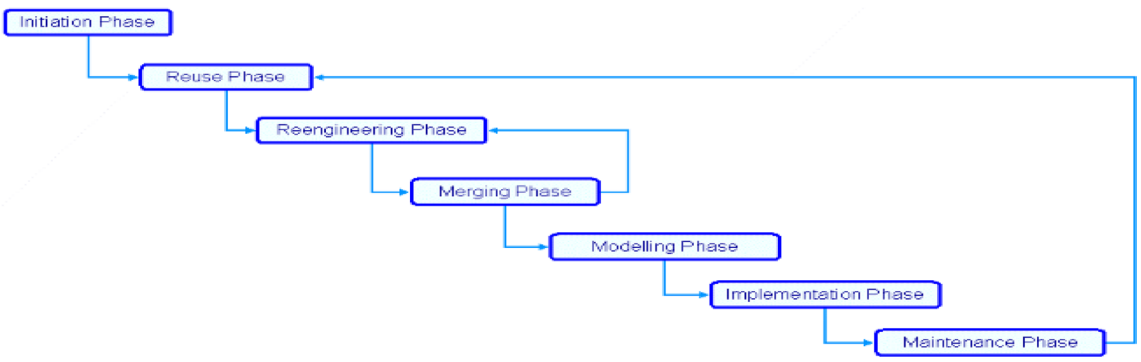

Fig. 2 NeOn waterfall ontology network life cycle model [34]

Table 2 Factors [35]

Factors	Source
Expertise	[4, 9, 19, 24, 37–41]
Technical Ability	[4, 9, 19, 40, 42–44]
Team members knowledge and skills	[4, 9, 29, 40, 43–45]
Personnel availability	[9, 19, 38, 41]
Project Manager Maturity	[44, 45]
Team Maturity	[29, 40]
Task Site Specificity	[9, 38, 40, 46]
Labour cost	[9, 38, 39, 41, 44, 46]
Workload at site	[4, 9, 38, 41, 46]
Working time	[9, 29, 39, 42]
Cultural differences	[9, 29, 38, 40, 42, 44]
Site locations	[9, 47]
Team willingness	[4, 39, 44, 45, 48]
Communication	[9, 24, 29, 30, 38, 42, 47, 49]
Coordination	[9, 24, 29, 30, 42, 45]
Task Size	[4, 9, 29, 38, 40, 42]
Proximity to customer requirement	[9, 41]
Required resources	[4, 9, 41, 48]
Task deadline	[4, 9, 46, 48]
Effort	[4, 39, 41, 48]
Product Architecture	[9, 19, 40]
Product	[9, 40]
Transparency	[9, 42]
Prioritized delivery	[4, 9, 29, 40, 55]
Enough Documentation	[9, 19]
Customer collaboration	[9, 39]
Language Fluency	[29]

Table 3 Dependencies [36]

Dependencies	Source
People	[10, 11, 30, 38]
Requirements	[9, 10, 40]
Activity	[10, 11, 49, 55]
Task	[9, 10, 30, 38, 40, 55]
Technical aspects	[10, 11]
Expertise	[9, 10, 40]
Resources	[10, 30, 38, 55]
Sites	[10, 37, 38]

[36]. The practitioners were asked to rank the importance of the factors and dependencies based on their importance in task allocation and coordination in agile distributed software development. It was found that though literature highlights a number of factors influencing task allocation process in distributed agile environment, organizations considered only a few. This resulted in project failures, communication and coordination issues and project delays. Figure 3 depicts the relative scores for the factors in descending order. Weights were assigned that is 6 being most important factor and 1 being least important. The value scores were calculated by summing the weights given by each practitioner for each factor.

As for dependencies, the practitioners were given a list obtained from literature and were asked to choose those that they usually consider while assigning tasks. Figure 4 depicts the number of responses obtained in the survey. The count represents the number of practitioners who have chosen the dependencies.

The gap between the literature and the software industry practice shows that DASD software industry is not paying enough attention to all the identified factors from literature. The results of the survey have provided useful feedback, which helped in the creation of *OntoDASD* ontology.

3.1.3 Motivation scenario

This section presents a motivation scenario to provide a clear picture of the scope of the ontology and to help gather concepts, terms and relationships for the conceptual model.

Vinci Ltd. is a global agile project located in France, England, Germany, Russia and India. Vinci Ltd. is composed of Jerome (a product owner) and Philip (a scrum master and quality manager) who performs project supervision from distance. There are 3 programmers namely Jean, Michelle and Stephanie from Russia. Michelle and Stephanie are in Germany. There are 3 testers namely Francois, Marco and Meidie. Jerome and Philip are in France close to the customer, performing requirement engineering together. Jean is a designer who works in England. There is a team specialized in configuration management and maintenance in India. The team is composed of Ijaz, a configuration manager and Salim, a maintenance specialist. Philip performs project supervision. In typical agile environments, when each day comes, team members pull the next highest priority task from the product backlog and stick their name to the task on the board. They just say 'yes' they will do it, and no one argues. Being in a team of 3 to 5, people do not necessitate to be strict. They are free to choose the tasks they are more comfortable with. The team knows when, who is doing what, and who is best in tackling which tasks. In DSD, such as in Vinci Ltd, for an effective coordination, good communication is a must. Distance affects communication leading to poor association among various sites and in turn, to improper task assignment. It is difficult for Philip to figure out when and who is doing what in Vinci Ltd. It becomes difficult to adopt these agile practices to share knowledge between remote team members. In a project, individuals might want to know which task depends on the completion of another as well as the relationships between tasks. Face to face discussions, whereby team members self-assign tasks, are not possible

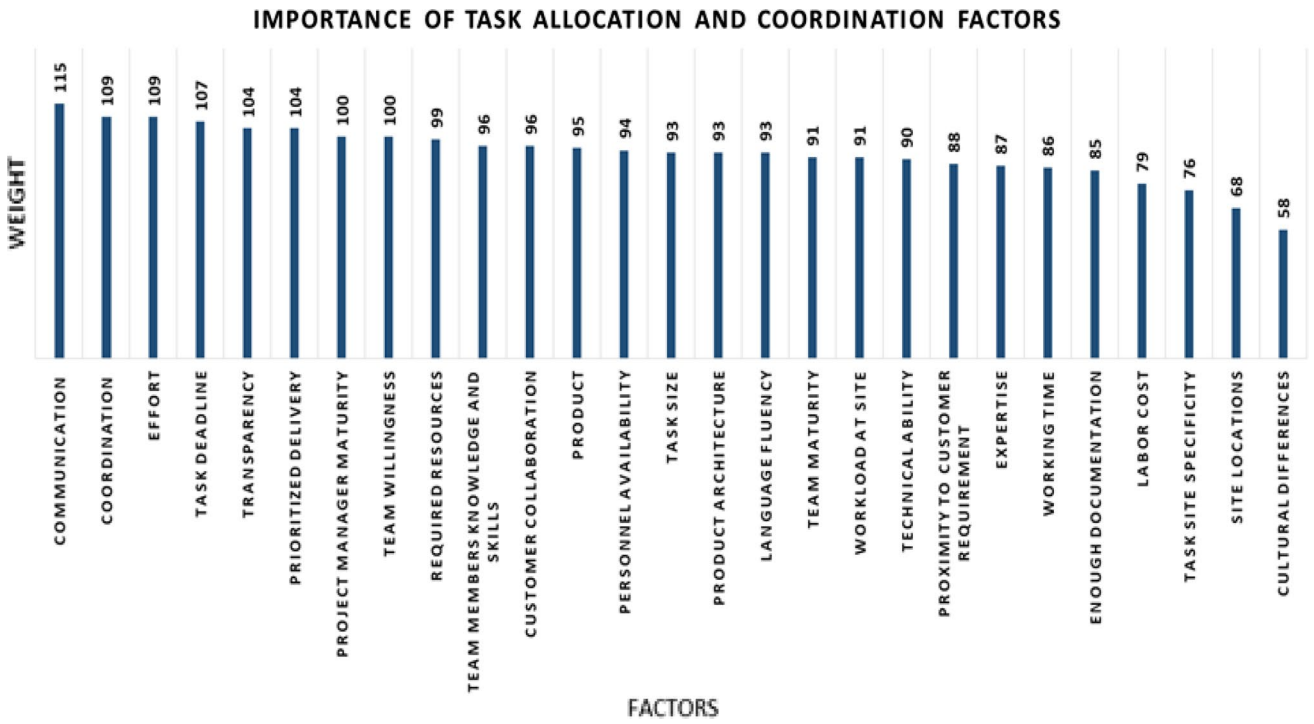

Fig. 3 Survey results on factors [36]

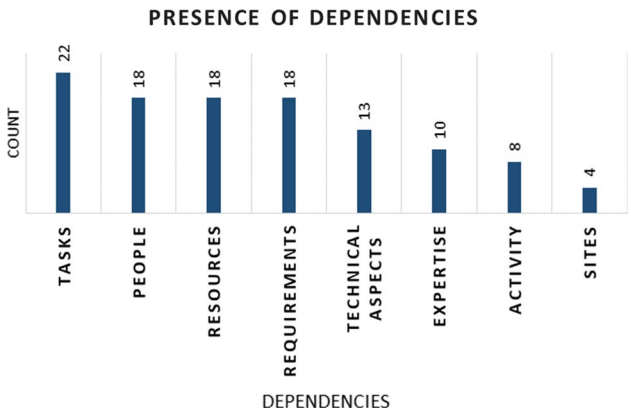

Fig. 4 Survey results on dependencies [36]

in Vinci Ltd. This gives rise to multiple dependencies that cannot be ignored.

3.1.4 Ontology requirements specification document (ORSD)

Based on the motivation scenario, the ORSD was defined in Table 4. It contains the knowledge necessary to capture ontology requirements. Non-functional requirements as well as competency questions are defined in the ORSD.

3.2 Reuse phase

This phase deals with reusing one or multiple ontological resources for the ontology to be developed. The output of this phase is a data dictionary (as shown in Table 5) showing concepts of *OntoDASD* ontology based on motivation scenario and ORSD from initiation phase. Source reference shows the concepts that have been reused from existing ontologies.

3.3 Reengineering phase and merging phase

In this phase, the ontological resources (concepts from existing ontologies) and non-ontological resources (concepts not used in ontology previously) are transformed into a formal model. Figure 5 depicts the output of this phase. The part reengineered concerns factors and dependencies from the survey.

3.4 Ontology description and modelling phase

This phase deals with describing and designing of the ontology based on the requirements. A taxonomy has been developed based on the results of systematic literature and survey. It has been used as a starting point for building domain concepts for our ontology named *OntoDASD*. To ease the understanding of the taxonomy, it has been presented in fragments composing of the main concepts.

Table 4 Ontology Requirement Specification Document

Ontology Requirements Specification Document		
1	*Purpose*	
	To represent task allocation and coordination knowledge	
2	*Scope*	
	It should incorporate the factors and dependencies affecting task allocation and coordination identified in the literature. It should have the attributes required to be used in an algorithm for allocating task	
3	*Implementation Language*	
	The ontology should be implemented in an ontology language OWL using Protégé tool	
4	*Intended End-Users*	
	The end users will be agile professionals such as project managers, team leaders and team members in software industry	
5	*Intended Uses*	
	The ontology will be used in a local java application to allocate task to end users	
6	*Ontology Requirements*	
	(a) Non-Functional Requirements	
	Code	Description
	Accuracy	
	NFR1	The concepts should be represented in a logical form that comply with the expertise of the users
	NFR2	The ontology should capture and correctly represent aspects of the real world
	Clarity	
	NFR3	The ontology should effectively communicate the intended meaning of the defined terms
	NFR4	The definitions should be documented
	NFR5	The ontology should be understandable
	Completeness	
	NFR6	The domain of interest should be appropriately covered
	NFR7	Competency questions should be defined and the ontology should answer them
	NFR8	The ontology should include all relevant concepts
	Conciseness	
	NFR9	The ontology should not contain redundant and useless definitions
	NFR10	The ontology should define only essential terms
	(b) Functional Requirements: Groups of Competency Questions	
	ICQ-1	What is the teamwork reputation of employees and their period?
	ICQ-2	Which employees has competence, x?
	ICQ-3	What employees have competence, x and skills, y and what is the period of time?
	ICQ-4	What is the communication level, coordination level, teamwork reputation and cultural alignment of employees?
	ICQ-5	How much effort is allocated to a task?

Table 5 Data Dictionary for *OntoDASD* Ontology

Term	Description	Source reference	Prospective entity
People Characteristics	Refers to the characteristics for team personnel	SEO_Human Resource [25]	Class
Site Characteristics	Refers to the characteristics for different office locations		Class
Task Characteristics	Refers to the characteristics of activities involved in a project		Class
Project Characteristics	Refers to a process in which a product is developed	SEO_Project [25]	Class
Agile Characteristics	Refers to a project management process for software development		Class
Expertise	Refers to the competency of team members in a particular field	SEO_Competence [25]	Class
Technical ability	Refers to skills such as knowledge and ability on methods, programming languages, tools etc.	DASD_Technology skills [21]	Class
Team members knowledge and skills	Refers to knowledge and skills team members require to coordinate within a team and remain aligned	SEO_Knowledge Domain [25]	Class
Personnel availability	Refers to the availability of the team members during the decision making process as well as free to perform tasks		Class
Project Manager Maturity	Refers to project manager experience in the field and maturity in his profession	DASD_PM Experience [21]	Class
Team Maturity	Refers to a certain age experience in the domain	DASD_Team experience [21]	Class
Task Site Specificity	Refers to application and platform experience possessed by the team members		Class
Labor cost	Refers to cost of professionals and other resources	DASD_Cost [21]	Class
Workload at site	Refers to backlog of personnel and commitments	DASD_Backlog Strategy [21]	Class
Working time	Refers to time zone when the various teams are executing their task	TADSD_Timezone [25]	Class
Cultural differences	Refers to cultures and different working habits of personnel	TADSD_Cultural Alignment level [25]	Class
Site locations	Refers to geographical location of office	TADSD_Country [25]	Class
Team willingness	Refers to the motivation and interest of the team to complete a task		Class
Communication	Refers to the means to convey an information in a project environment	DASD_DSD Communication [21]	Class
Coordination	Refers to a comprehensive understanding of the project and what's going on and when, what other team members are doing and what they should be doing for their work to fit in with other team members work	DASD_Control and Coordination level Required [21]	Class
Task Size	Refers to the size of the task and helps to determine the amount of time a resource will complete the task		Class
Proximity to customer requirement	Refers to how close the task is to the requirement		Class
Required resources	Refers to hardware, software and human requirements	DASD_IT Infrastructure for Collaboration [21]	Class
Task deadline	Refers to time allocated to complete a task	DASD_Delivery Estimate [21]	Class

Table 5 (continued)

Term	Description	Source reference	Prospective entity
Effort	Refers to effort level that is require to perform a task is important while assigning task		Class
Product Architecture	Refers to modules view, components and connectors		Class
Product	Refers to the nature of the product	SPO_Artifact [25]	Class
Transparency	Refers to visiblity of actions taken in the project		Class
Prioritized delivery	Refers to priority provided to a particular task		Class
Enough Documentation	Refers to artifacts, product deliverables and manuals to convey missing and unclear information		Class
Customer collaboration	Refers to customer knowledge and involvement		Class
Language Fluency	Refers to the proper use of foreign languages accents	TADSD_Language [25]	Class
Factors	Refers to the factors affecting task allocation and coordination in DASD		Class
Dependencies	Refers to the process where the progress of one action relies upon the timely output of another action		Class
Knowledge	Refers to the information required for the advancement of a project		Class
Process	Refers to the process when a task needs to be finalized to allow another task to start or proceed		Class
Resource	Refers to what is needed for tasks to be carried out such as personnel and tools	DKD_Resource [13]	Class
Requirements	Refers to customer needs with respect to the project		Class
Activity	Refers to a series of works to complete a task	SPO_Activity [25]	Class
Task	Refers to a single output from work breakdown structure in a project		Class
Technical aspects	Refers to the technical specifications needed to complete a task	DKD_Technical challenge [13]	Class
Agile Software Process	Refers to the software methodology used to implement a project		Class
User Stories	Refers to a very high-level definition of a requirement and contains just enough information so that the developers can produce a reasonable estimate of the effort to implement it	DASD_User Stories [21]	Class
Person	A project stakeholder who has a role in the project	[25]	Class
Scrum Master	Refers to a person in an agile development who manages the team and processes	[9]	Class
Product Owner	Refers to a person in an agile development who maximize value of the products	[9]	Class
Quality Manager	Refers to a person in an agile development who works towards enhancing quality in the organization	[9]	Class
Designer	Refers to a person in an agile development who designs the products in an understandable manner for the programmer	[9]	Class

Table 5 (continued)

Term	Description	Source reference	Prospective entity
Programmer	Refers to a person in an agile development who implements the product	[9]	Class
Maintenance Specialist	Refers to a person in an agile development who is responsible for maintenance	[9]	Class
Tester	Refers to a person in an agile development who performs testing	[9]	Class
Configuration Manager	Refers to a person in an agile development who do configuration	[9]	Class
Software tools competency	Refers to competence in terms of capability to use a tool	[9]	Class
Task Allocation and Coordination	Refers to the process of assigning tasks to team members and communicating smoothly within the team		Class
Stage	Refers to agile stage	[50]	Class
Initiate Stage	Refers to agile initiate stage	OntoAgile_InitiateStage [50]	Class
Sprint Stage	Refers to agile sprint stage	OntoAgile_SprintStage [50]	Class
Release Stage	Refers to agile release stage	OntoAgile_ReleaseStage [50]	Class
Implement	Refers to agile implement phase	OntoAgile_Implement [50]	Class
Review Retrospect	Refers to agile review retrospect phase	OntoAgile_ReviewRetrospect [50]	Class
Plan	Refers to agile plan phase	OntoAgile_PlanEstimate [50]	Class

The ontology is being used for both transactional purpose, task allocation and informational purposes, providing coordination mechanisms. Rules are implemented to show task allocation. The factors and dependencies that are needed to effectively coordinate within team and between teams, have been considered in the ontology.

Agile software process, dependencies, and factors describe the domain of the ontology as demonstrated in Fig. 6. The relationships between the subclasses describe the task allocation and coordination process.

(A) Factors

This section describes the factors incorporated in the ontology.

Agile related factors These factors are needed for promoting flow of information between and within team from different locations as illustrated in Fig. 7.

Task related factors Task-related factors as shown in Fig. 8 such as *Task Size* and *effort* are required for performing the task. Other factors like *delivery day estimate* and whether the task requires proximity to customer, are crucial for task assignment process. Knowing the maximum effort a team member can apply and the high-quality output probability are important. Before allocating a task to someone, it is necessary to ensure that the person has the required infrastructure to perform the task.

Project related factors Knowing how many years of experience the project manager, the team or team member have before starting a project is important. Whether the team or team member is willing to undertake a particular project helps to determine whether the person is committed to perform the task. Project related factors are shown in Fig. 9.

Human resource related factors These factors as shown in Fig. 10 will help to determine the capabilities of the person with respect to the task requirements. Knowing who is available to perform the task and who has limited availability, are necessary for task allocation process.

Site Related Factors In DASD, knowing your reporting line is important to ensure that proper reporting. Sprint planning meetings should be performed via videoconference to gather project status from team members. Meetings help to raise alerts on time to avoid delay. The classes are composed of different mechanisms for inter-team coordination, which will be provided as information to the target audience. Examples of instances include vertical communication, horizontal communication, meetings, personal feedback and group mode feedback. Knowing the cost of resources from different countries is important before choosing resource. Language, country and time zone are factors essential for distributed settings. When working hours are not aligned, teams may lack coordination. Figure 11 shows the factors pertaining to the work site.

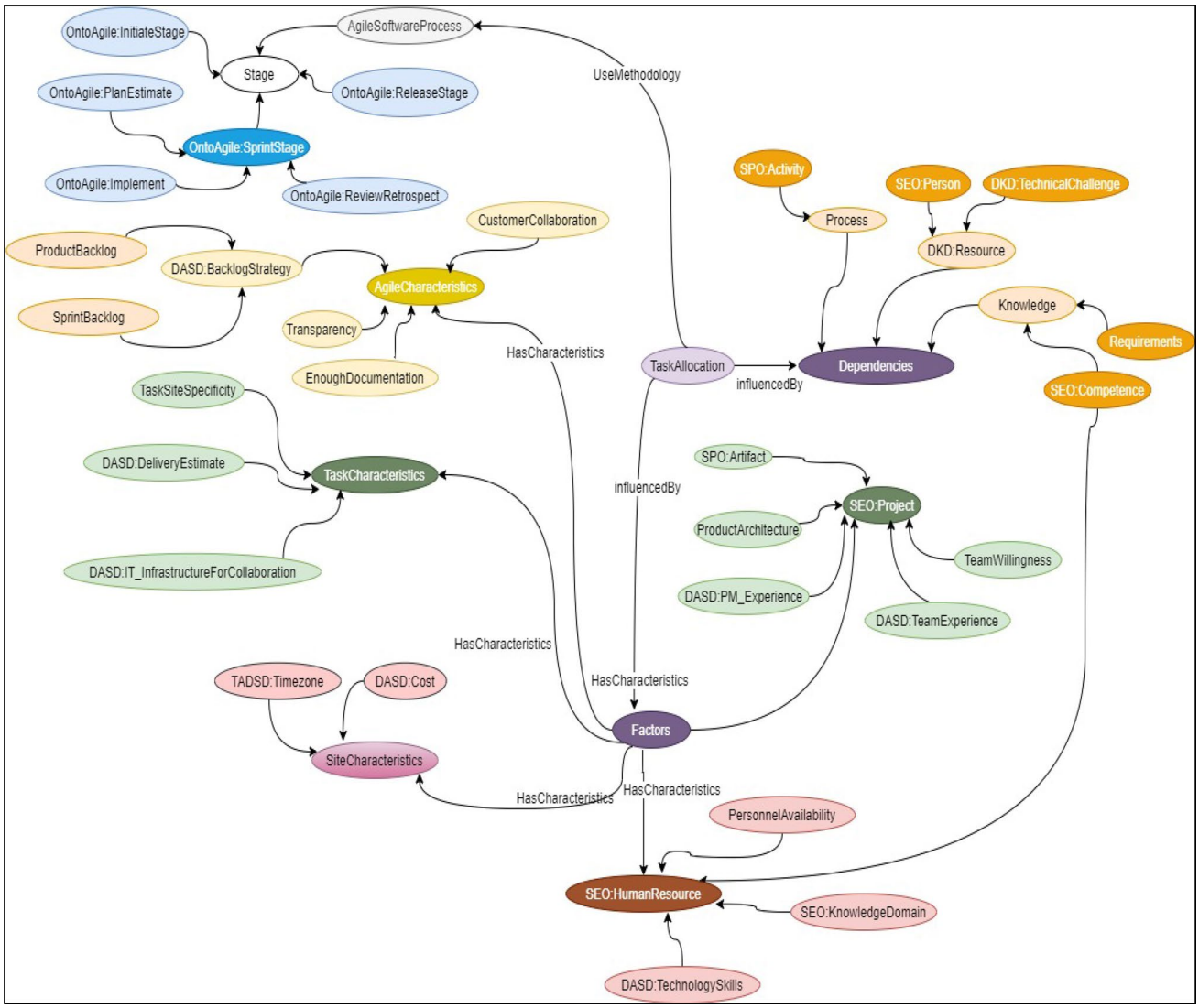

Fig. 5 *OntoDASD* concept mapping

(B) Dependencies

As we deal with coordination in distributed environment, managing dependencies between teams in distributed environment is critical. The dependencies as shown in Fig. 12, are described as follows:

Knowledge Knowledge dependency is the information required to perform task allocation. It is then broken down into two main knowledge required: competence and requirements dependency. Knowing who is good in performing which task and who is stuck with a particular task are essential for the smooth running of an organization. When insufficient information have been gathered on a requirement, this causes a delay in the process as details need to be identified. The concepts will help decision makers to consider the dependencies by prioritizing requirements and gathering competence information properly of employees.

Process Process dependency is when a process needs to be completed before another process to start. It is therefore divided into two main dependencies applicable for agile software development: user stories and activities. When an activity depends on the completion of another activity and the latter is delayed, this affect project progress. An example is the start of testing depends on the completion of programming. When user stories need to be implemented on a specific order then any problem may delay the whole process.

DKD_Resource Resource dependency is what is needed for a task to be carried out. Resource dependency is divided into personnel and technical challenge. Having to wait for a person or unavailability of personnel delay the project

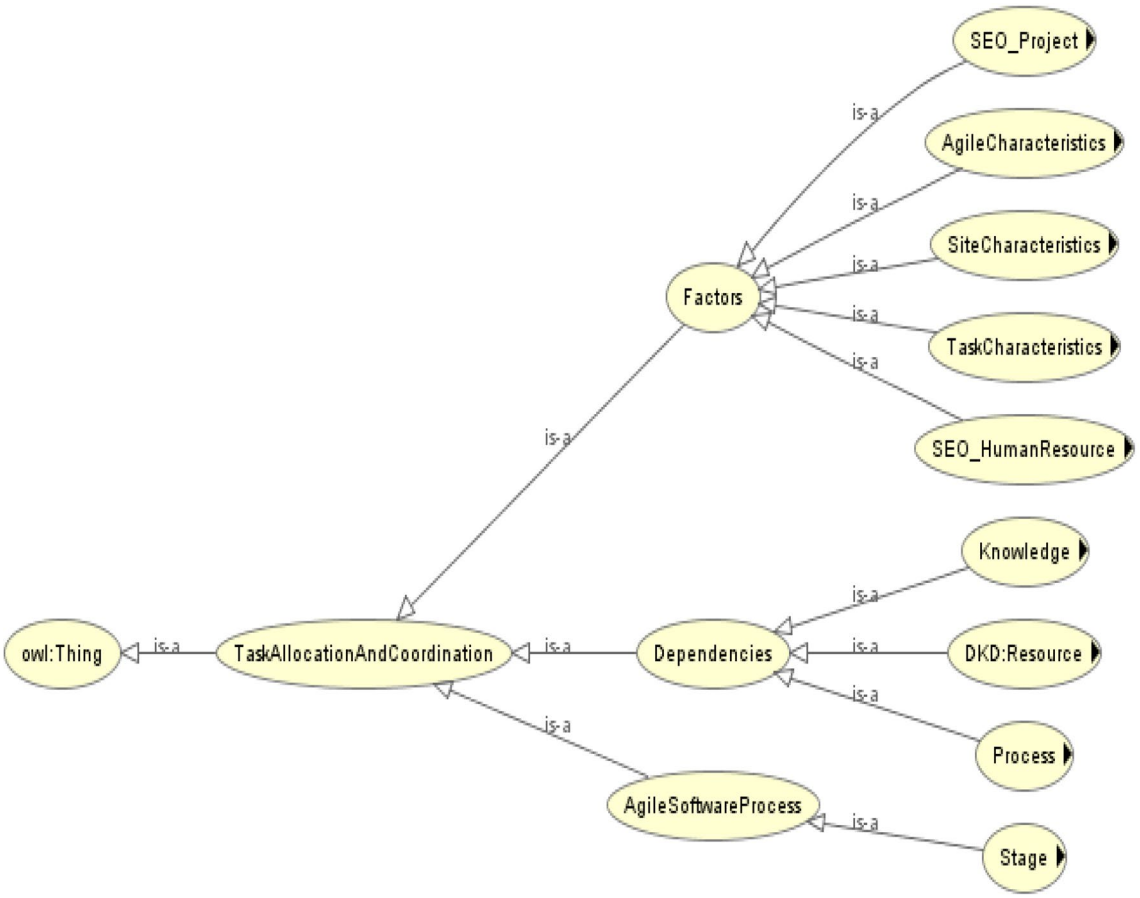

Fig. 6 OntoDASD-task allocation and coordination

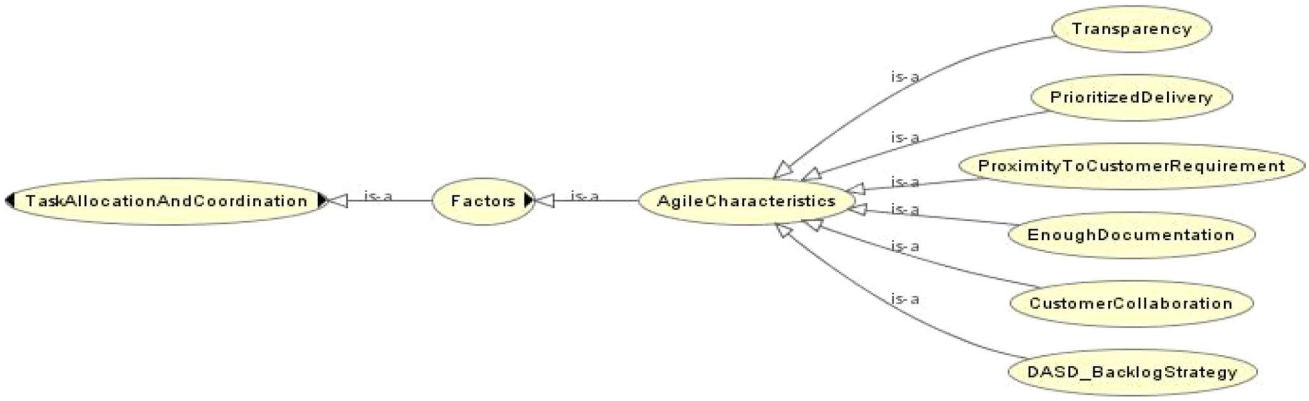

Fig. 7 Agile related factors

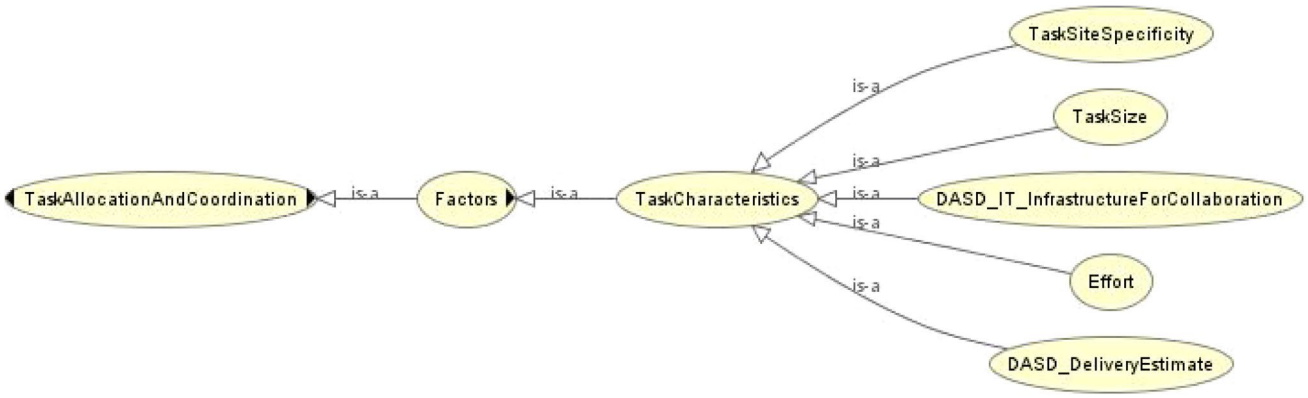

Fig. 8 Task related factors

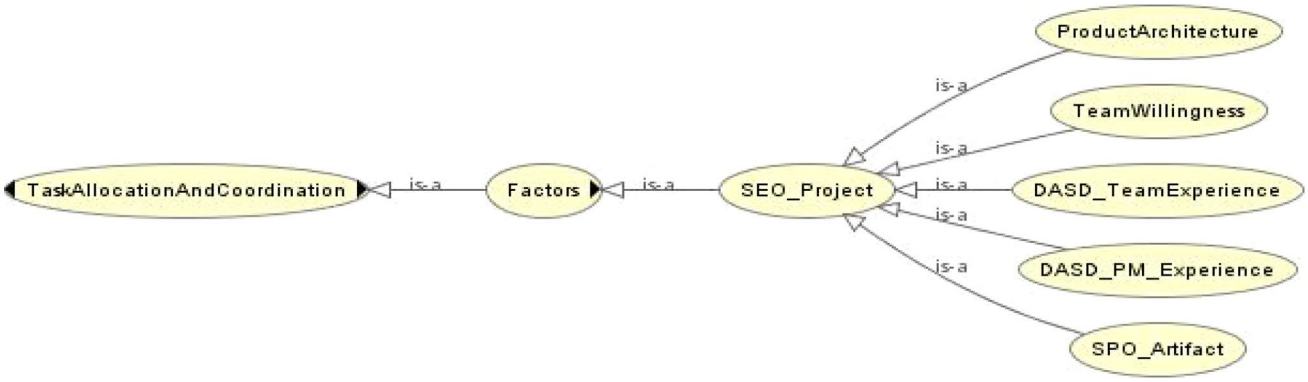

Fig. 9 Project related factors

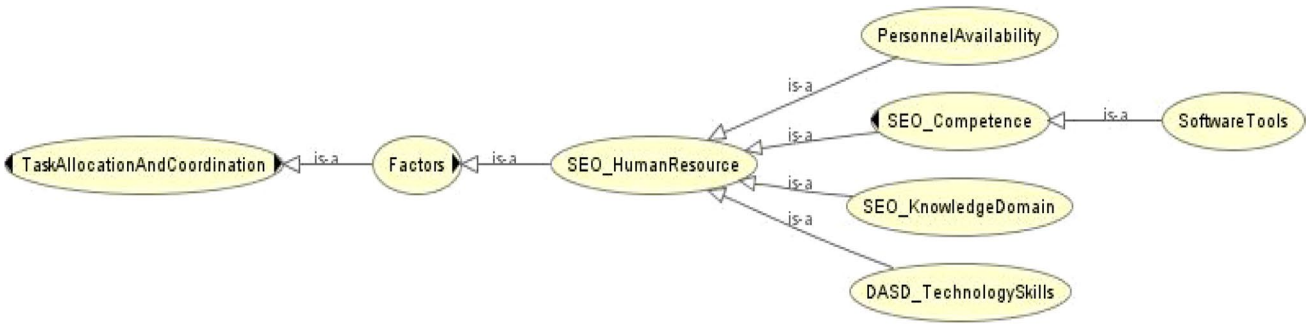

Fig. 10 Human resource related factors

progress. Lack of technical tools to perform task, prevents a task from being done.

(C) Agile Software Development Process

Since the *OntoDASD* is designed for agile practitioners, it is necessary to include concepts and relationships related to agile software development process. The concepts and relationships were reused from *OntoAgile* [50] as shown in Fig. 13.

3.5 Implementation phase

In this phase, the formal model from the previous phase is implemented in an ontology language OWL using Protégé tool. OWL was developed to model business domain and

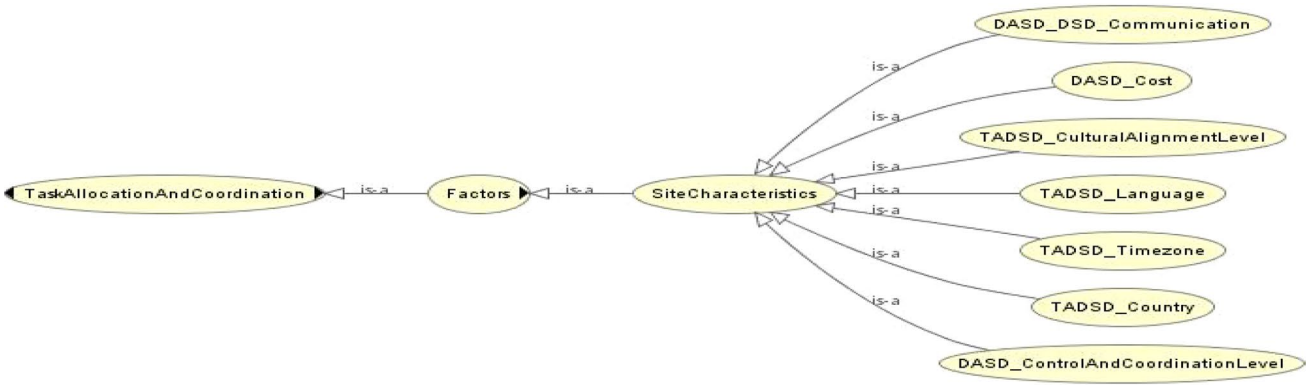

Fig. 11 Site related factors

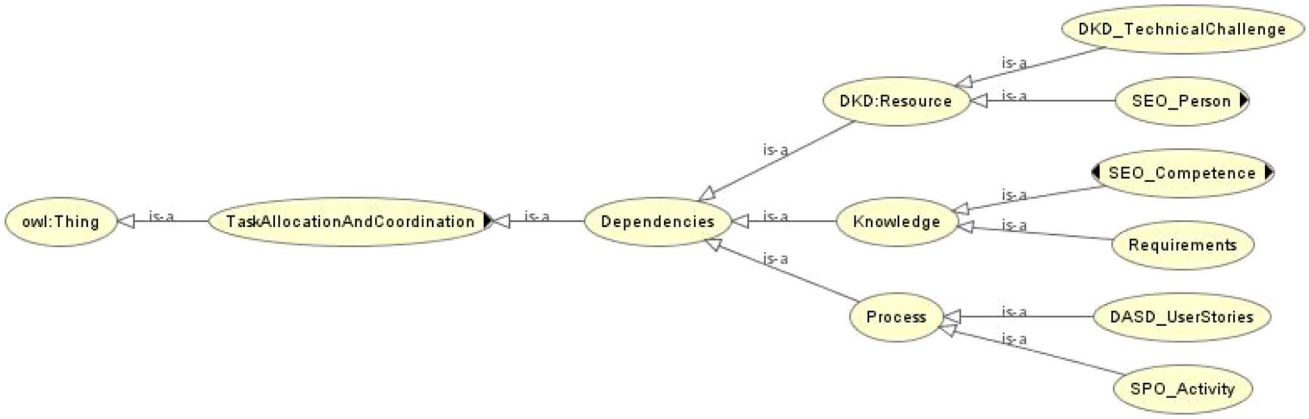

Fig. 12 Dependencies

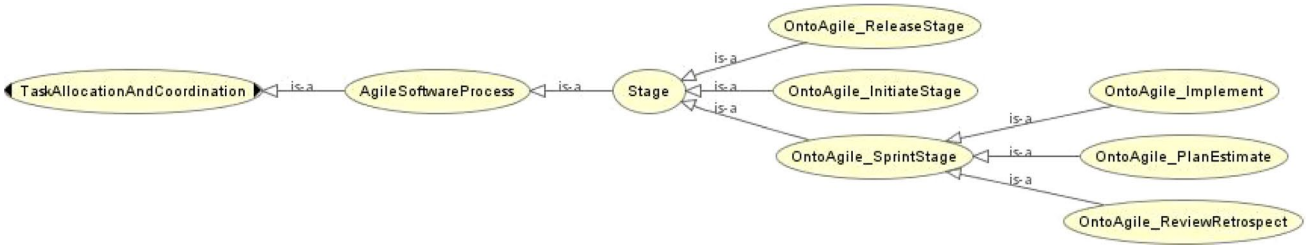

Fig. 13 Agile software development process

bridge business-IT gap by providing vocabulary to describe business knowledge using formal semantics, understandable by both business and computer programs [51].

The taxonomy in Sect. 3.4 is formalized, whereby the hierarchy of concepts are translated into parent–child relationships of classes. Classes represent concepts in the domain. Features and attributes of the concepts are represented as properties. The ontology constituted a knowledge base with a set of individual instances of classes. A top-down development approach was adopted where the most general concepts such as *AgileSoftwareProcess*, *Dependencies*, *Factors* and *Tasks* were defined followed by creating the subclasses. Figure 14 shows the breakdown among the different levels of generality.

Due to space constraints only selected sub classes are depicted. The class *AgileSoftwareProcess* and subclass *SiteCharacteristics* show that our focus is on DASD. By incorporating knowledge of relationships among agile software process stages, dependencies, factors and tasks, the ontology allows reasoning about task allocation in DASD.

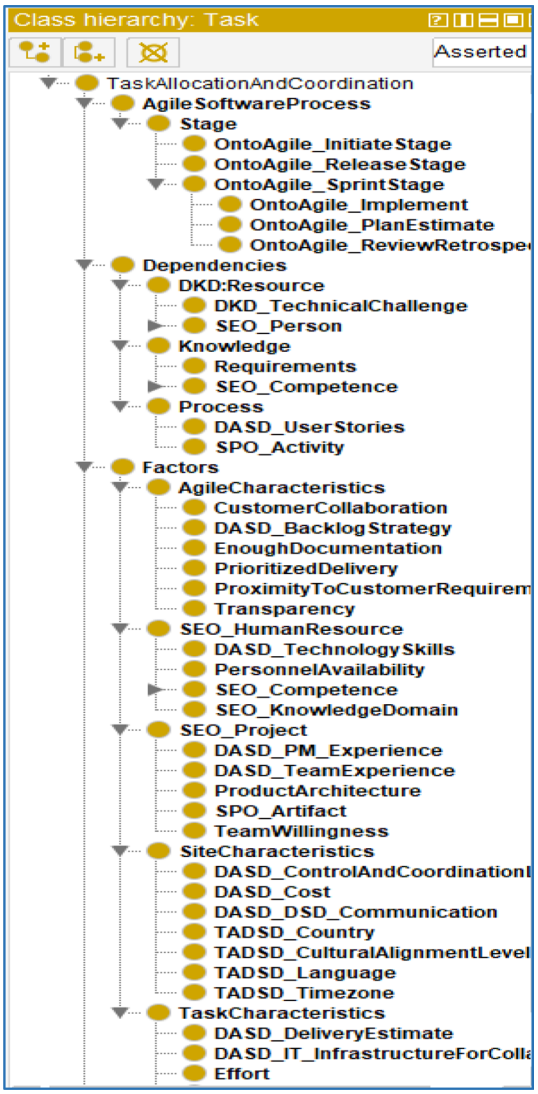

Fig. 14 Classes levels of *OntoDASD*

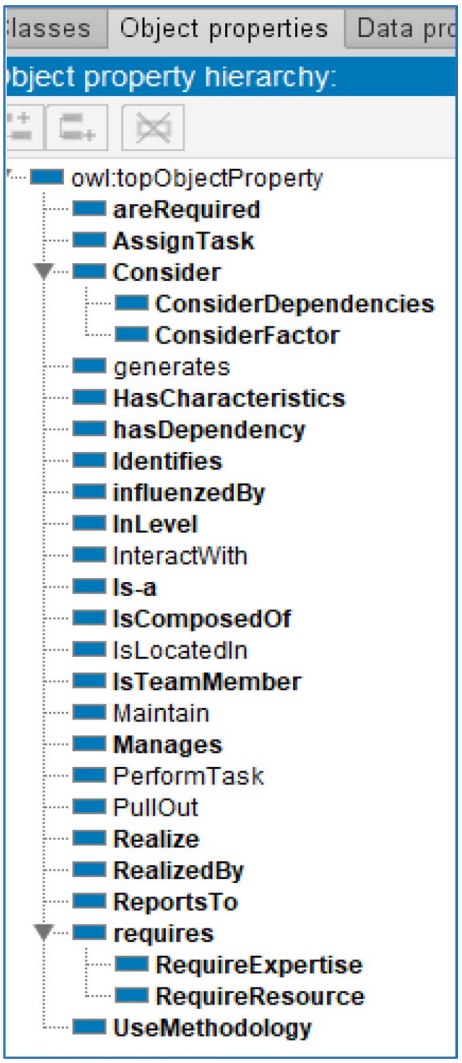

Fig. 15 Object properties

3.6 Rules and relationships between classes

An ontology with only classes is useless without relations between them. Figures 15 and 16 show object properties and datatype properties, which are used to create relationships between instances of the class created. In addition to axioms, SWRL (Semantic Web Rule Language) are rules expressed in terms of OWL concepts (classes, properties, and individuals). SWRL has been used to model parts of *OntoDASD* that was not possible with OWL. SWRL rules can also be used for validation by computing outputs. With the help of SWRL built-ins, a pool of domain-specific rules can be used and modelled [51].

Rule 1 If a person p is to be assigned a task t, check if the backlog effort, e together with the effort required, r for the task do not exceed the maximum capacity that is the productivity, m of the person.

Listing 1 SWRL Rule for assigning task taking into consideration backlog properties

```
swrlb:add(?eff, ?r, ?e) ^ OntoDASD:Task(?t) ^ OntoDASD:EffortRequired(?t, ?r) ^ OntoDASD:MaxProductivity(?p, ?m)
^ OntoDASD:BacklogEffort(?p, ?e) ^ OntoDASD:SEO_Person(?p) ^ swrlb:lessThan(?eff, ?m) -> OntoDASD:AssignTask(?p, ?t)
```

Rule 2 If a person *x* is a team member of project *z* and a project manager *y* manages project *z* this implies that person *x* reports to project manager *y*.

Listing 2 SWRL Rule for reporting line within a team

```
OntoDASD:SEO_Person(?x) ^ OntoDASD:IsTeamMember(?x, ?z) ^ OntoDASD:Manages(?y, ?z) ^ OntoDASD:SEO_Project(?z) ^ OntoDASD:ProjectManager(?y)
-> OntoDASD:ReportsTo(?x, ?y)
```

Rule 3 If a team member has applied efforts less than that assigned to a task and has good reputation and has delivered before deadline, this implies team member willingness.

Listing 3 SWRL Rule for determining team member willingness before allocating task

```
OntoDASD:SEO_Person(?p) ^ OntoDASD:AppliedEffort(?p, ?e) ^ OntoDASD:Task(?t) ^ OntoDASD:PerformTask(?p, ?t) ^
OntoDASD:EffortRequired(?t, ?r) ^ swrlb:lessThan(?e, ?r) ^ OntoDASD:PersonReputation(?p, ?g) ^ swrlb:greaterThanOrEqual(?g, 10) ^
OntoDASD:Deadline(?t, ?d) ^ OntoDASD:Completion(?t, ?c) ^ temporal:before(?c, ?d) -> OntoDASD:IsWilling(?p, ?t)
```

Rule 4 If a task *t* requires a competence *c* and technology skills *s* and a person has experience is greater than 1 year and has performed task in the past, this implies that team member is capable to perform task.

Listing 4 SWRL Rule for determining capability of a team member to perform the task

```
OntoDASD:Task(?t) ^ OntoDASD:SEO_Competence(?c) ^ RequireExpertise(?t,?c) ^
OntoDASD:DASD_TechnologySkills(?s) ^ OntoDASD:RequireSkills(?t,?s) ^ OntoDASD:SEO_Person(?p)
^ PerformTask(?p,?t) ^ OntoDASD:Years(?p,?y) ^ swrlb:greaterThanOrEqual(?y, 1)
->OntoDASD:Capable(?p,?t)
```

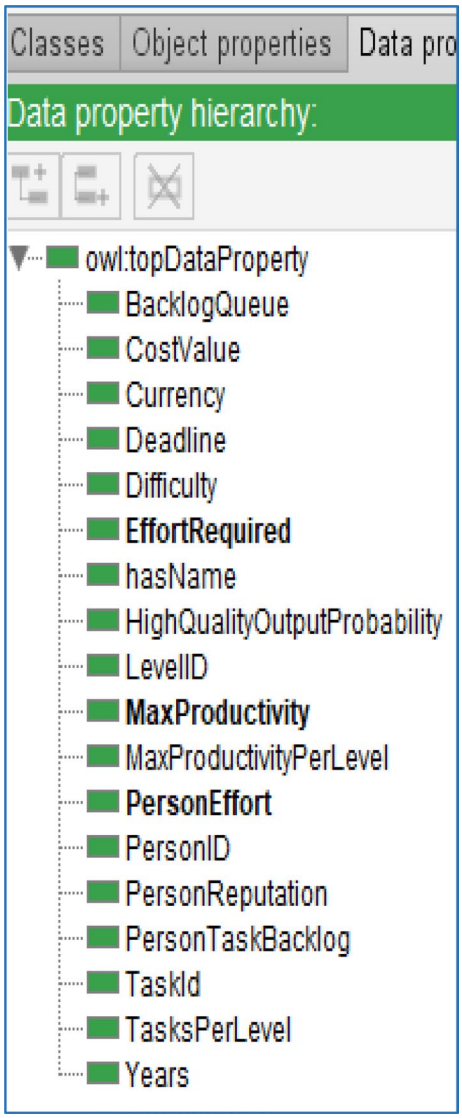

Fig. 16 Datatype properties

4 Ontology evaluation

In this phase, the ontology is evaluated based on a set of metrics and with respect to competency questions defined in Sect. 3.1.4.

4.1 Metrics and formal validation

According to [50], the evaluation process consists of three main elements related to performance and correct definition: consistency, completeness and conciseness. With the ontology implemented in Protégé in OWL Language, the Pellet reasoner was used to evaluate both the computational efficiency and consistency. The ontology is processed in 463 ms by Pellet. This does not only show how fast the standard reasoning processes can be applied to the ontology but also the lack of inconsistencies in the implementation. The ontology was adapted to answer almost all the questions in Table 6.

Table 6 Evaluation criteria

Criteria	Questions	Yes
Accuracy	Have the concepts been represented in a logical form that comply to the expertise to the users?	✓
	Does the ontology capture and correctly represent aspects of the real world?	✓
Clarity	Does the ontology effectively communicate the intended meaning of the defined terms?	
	Are the definitions documented?	✓
	Is the ontology understandable?	✓
Completeness	Is the domain of interest appropriately covered?	✓
	Are competency questions defined and can the ontology answer them?	✓
	Does the ontology include all relevant concepts?	✓
Conciseness	Does the ontology not contain redundant and useless definitions?	✓
	Does the ontology define only essential terms?	

Table 7 shows an evaluation of the *OntoDASD* Ontology based on "*Knowledge coverage and popularity measures*" proposed by [52]. In other words, it shows the number of distinct classes, object properties, data properties and individuals present in the ontology.

4.2 Evaluation of requirements and answer to competency questions

The RDF Query Language (SPARQL) was also used to evaluate the effectiveness of the ontology. Protégé in built SPARQL tool was used to execute queries to the ontology. The ability of the ontology to answer competency questions in Sect. 3.1 was evaluated. The competency questions (*ICQ1-ICQ5*) were translated from natural language to SPARQL queries. The data obtained as a result of execution validates the purpose fulfillment of the ontology.

The following was used as PREFIX when running the SPARQL Queries:

OntoDASD: http://www.semanticweb.org/shweta/ontologies/2020/2/TaskAllocationAndCoordinationOntology#

Figures 17, 18, 19, 20 and 21 show the results of the competency questions defined in ORSD defined in Sect. 3.1.4.

(A) ICQ-1

Table 7 Entity counts in OntoDASD

Metric	Value
Number of classes	62
Number of properties	41
Datatype properties	23
Object properties	18
Number of individuals	62

Figure 17 lists the employee, his teamwork reputation and years of experience.

(B) ICQ-2

Figure 18 shows all employees with competency in Java Programming.

SPARQL:

```
PREFIX : <http://www.semanticweb.org/shweta/ontologies/2020/2/TaskAllocationAndCoordinationOntology#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX vocab: <http://localhost:2020/resource/vocab/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX map: <C:/Users/chitra.nundlall/Documents/Chitra/d2rq-0.8.1/d2rq-0.8.1/outfile.ttl#>
PREFIX db: <http://localhost:2020/resource/>
SELECT ?employee ?teamworkreputation ?Years WHERE{
?t a :TeamWillingness .
?t :TeamworkReputation ?teamworkreputation .
?t :YearStart ?CYS .
?t :YearEnd ?CYE .
bind( year(?CYE)-year(?CYS) as ?Years )
?t :EmployeeId ?employee .
?e a :SEO_Person .
?e :EmployeeId ?emp .
```

Results: Browse Go! Reset

SPARQL results:

employee	teamworkreputation	Years
db:3	3	22
db:7	3	17
db:10	8	13
db:12	7	12
db:9	5	11
db:2	7	6
db:6	8	6
db:13	8	3
db:14	7	3
db:4	7	3
db:8	5	2

Fig. 17 Results of ICQ-1

Fig. 18 Results of ICQ-2

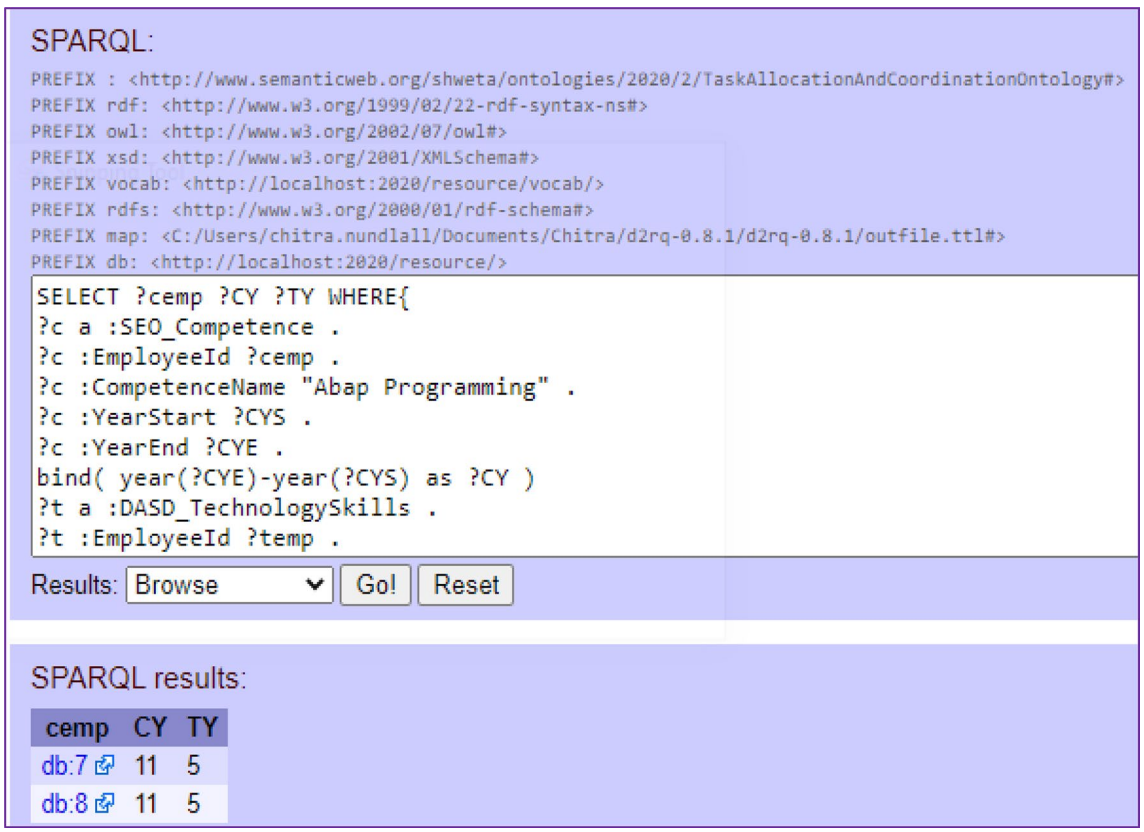

Fig. 19 Results of ICQ-3

SPARQL:

```
PREFIX : <http://www.semanticweb.org/shweta/ontologies/2020/2/TaskAllocationAndCoordinationOntology#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX vocab: <http://localhost:2020/resource/vocab/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX map: <C:/Users/chitra.nundlall/Documents/Chitra/d2rq-0.8.1/d2rq-0.8.1/outfile.ttl#>
PREFIX db: <http://localhost:2020/resource/>
SELECT ?temp ?trep ?tcom ?tcoor ?tcul ?TY WHERE{
?t a :TeamWillingness .
?t :TeamworkReputation ?trep .
?t :DASD_DSD_Communication ?tcom .
?t :DASD_ControlAndCoordinationLevel ?tcoor .
?t :TADSD_CulturalAlignmentLevel ?tcul .
?t :YearStart ?CYS .
?t :YearEnd ?CYE .
bind((year(?CYE) - year(?CYS)) as ?TY )
```

Results: Browse Go! Reset

SPARQL results:

temp	trep	tcom	tcoor	tcul	TY
db:3	3	5.0E0	7.0E0	9.0E0	22
db:7	3	2.0E0	4.0E0	6.0E0	17
db:10	8	9.0E0	6.0E0	7.0E0	13
db:12	7	9.0E0	2.0E0	6.0E0	12
db:9	5	7.0E0	6.0E0	8.0E0	11
db:6	8	8.0E0	7.0E0	7.0E0	6
db:2	7	9.0E0	8.0E0	7.0E0	6
db:14	7	5.0E0	3.0E0	4.0E0	3
db:13	8	7.0E0	8.0E0	5.0E0	3
db:4	7	9.0E0	5.0E0	8.0E0	3
db:8	5	6.0E0	4.0E0	3.0E0	2

Fig. 20 Results of ICQ-4

(C) ICQ-3

Figure 19 lists all employees having experience in Abap Programming and the relevant years of experience.

(D) ICQ-4

Figure 20 shows the communication level, coordination level, teamwork reputation and cultural alignment of all employees.

(E) ICQ-5

Figure 21 shows the effort that has been allocated to each task.

4.3 Domain-expert evaluation

OntoDASD has been assessed by agile professionals via a survey questionnaire. The structure of the survey has been designed in such a way that the ontology content and relationship have been presented to the participants. The ontology is evaluated based on criteria such as consistency, completeness and accuracy. It was difficult to collect feedback from domain experts due to COVID-19 pandemic where most experts were working from home and were not easily reachable. Only ten participants consisting of team members, lead software engineer, development team, product owner, scrum master, junior software engineer, senior software developer and senior application developer provided their feedback on *OntoDASD*. As shown in Table 8, the majority had more than 1 year experience in DASD. This gave us confidence that they were best suited to evaluate *OntoDASD*.

(A) Completeness

All the participants agreed that *OntoDASD* describes all concepts related to Task Allocation and Coordination in DASD as shown in Fig. 22.

(B) Consistency

Fig. 21 Results of ICQ-5

Table 8 Survey participants

Participant	Years of experience in DASD
Product owner	5
Scrum master	5
Team member	1
Lead software engineer	2
Development team member	3
Junior software engineer	1
Engineer	1
Senior software developer	4
Senior application developer	4.5
Team member	1

The majority of the participants agreed that all relevant concepts related to the DASD domain have been represented in the ontology as shown in Fig. 23.

(C) Accuracy

Figure 24 reports on the ontology accuracy. A rating of 1–5 was defined (1—*Inaccurate*, 2—*Moderately Inaccurate*, 3—*Moderately Accurate*, 4—*Accurate*, 5—*Very Accurate*. The majority of the participants agreed that *OntoDASD* captures and correctly represents aspects of the real world.

Most of the professionals reported that *OntoDASD* satisfies criteria of completeness, consistency and accuracy based on the graphs illustrated in Figs. 22, 23 and 24. One or two participants disagreed and argued that other factors such as

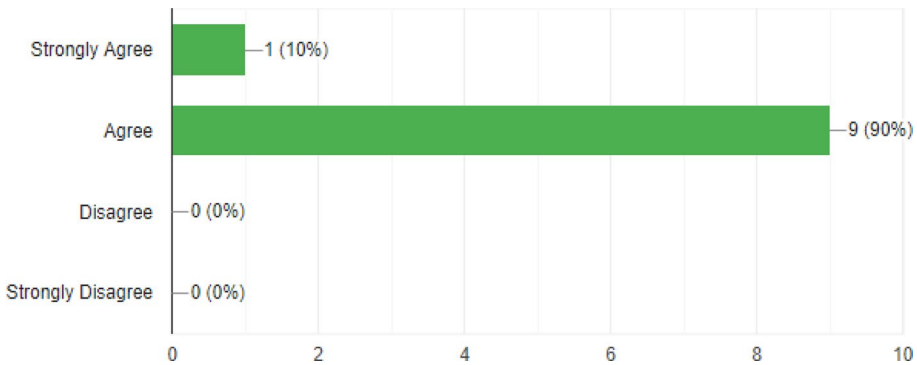

Fig. 22 Ontology completeness

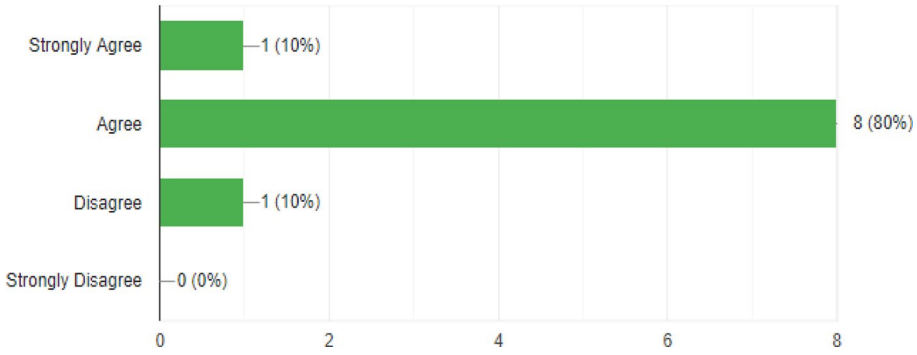

Fig. 23 Ontology consistency

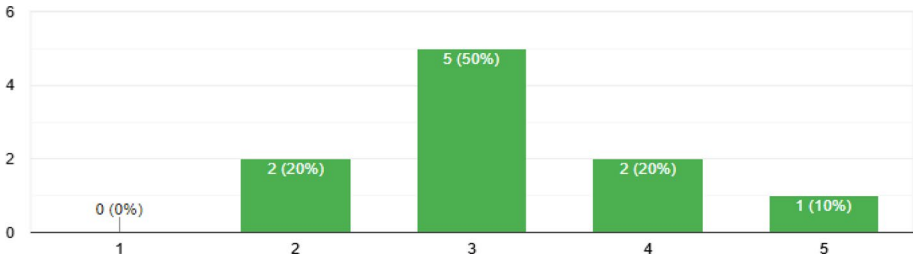

Fig. 24 Ontology accuracy

Daily Feedback and dependencies such as *Business Process Dependency* could have been considered in the process of task allocation and coordination in DASD. As described in Sect. 3.4, the most influential factors and dependencies affecting task allocation and coordination were incorporated in the ontology, based on a survey carried out in industry by Nundlall and Nagowah [36]. In future, the ontology can be improved by incorporating new factors and dependencies that evolve.

4.4 Discussion

OntoDASD meets all the criteria defined in Table 1 namely *Factors*, *Dependencies*, *Task Allocation process*, *DSD*, *Agile Software Development*, *Taxonomy* and *Ontology*. None of the existing studies found met all the criteria. This study complements the existing studies by further incorporating factors and dependencies related to task allocation and coordination in DASD. *OntoDASD* reused the findings of Rajpathak et al. [23] and Almeida et al. [24] in its ontology. *OntoDASD* incorporated some of the concepts from Ijaz and Aslam [10] and Strode and Huff [12] that are mostly used by organizations in the ontology. *OntoDASD* used both metrics and tools for evaluation purposes. *OntoDASD* also used SWRL rules and SPARQL queries as mentioned in Sects. 3.5 and 4.2 respectively.

4.4.1 Implications for practitioners

OntoDASD represents a knowledge model where there is a common understanding of the terms and concepts related to the task allocation and coordination process in DASD projects. High-level knowledge regarding project information, agile software process information and staff details among others are represented in the ontology so that there is a homogeneous comprehension of the project and process information as well as staff involved (their expertise and availability). Practitioners can thus use the ontology to perform task allocation based on task and employee profile. A project manager can use the ontology to search for most suitable employee for a particular task based on knowledge about agile software process stages, dependencies, factors and tasks. Several queries described in Sect. 4.2 can be performed using the ontology. Furthermore, the ontology makes use of SWRL rules for proper reasoning about task allocation in DASD as described in Sect. 3.5.

5 Conclusion

The main contribution of this research work is an ontology, *OntoDASD,* which provides a taxonomy of factors and dependencies influencing task allocation and coordination process in the area of distributed agile software development. The conceptualization of *OntoDASD* was based on an in-depth analysis of relevant literature and practitioners'

feedback via a survey. As a result, the knowledge model, *OntoDASD*.

1. provides agile team members and project managers located in distributed sites with a common understanding of the criteria and aspects necessary for effective task allocation and coordination process in DASD
2. adopts most relevant factors and dependencies based on experts in the field situated around the world
3. reuses vocabulary from existing ontologies in the field in an attempt to harmonize the terms.

The ontology was formally validated using metrics. Additionally, the competency questions were answered using SPARQL query language showing that the ontology met the requirements defined initially in ORSD. It was further evaluated by conducting a survey among experts in industry. Most of the professionals reported that *OntoDASD* satisfies criteria of completeness, consistency and accuracy. The ontology could be used as a reference point for academia and industry to pursue further research in that field.

5.1 Limitations

Despite the numerous advantages of *OntoDASD* described in previous sections, it cannot be claimed that *OntoDASD* is a complete ontology. Based on the survey carried out (as described in Sect. 3.1.2), *OntoDASD* considers the most relevant and influential factors and dependencies affecting task allocation and coordination in DASD. It does not cater for all existing factors and dependencies affecting task allocation and coordination. *OntoDASD* can further be extended in future by combining various other ontologies within the DASD domain and by incorporating new factors and dependencies.

5.2 Future works

OntoDASD has modelled the key concepts and knowledge of DASD domain. The vocabulary provided in the ontology illustrates the knowledge related to distributed agile software development in a generic manner. As future works, an ontology-based task allocation and coordination tool that utilizes *OntoDASD* will be implemented to support task allocation and coordination process among team members in DASD.

The tool will consist of a number of features namely *Task Recommendation*, *Task Self-Assignment*, *Viewing of project backlog and team member backlog at a site*, *Checking availability of team member of any site*, *Comparing site workforce cost*, *Determining most suitable team member for a task*, *Task completion notification*, *Viewing/editing of team activities by Project manager/team leader*, *Viewing of task dependencies, Assisting in manual task allocation by displaying comments*, *Prioritizing tasks based on factors*, *Productivity insights of team members* and *Task Tracking* amongst others.

References

1. Kamaruddin N, Arshad N, Mohamed A (2012) Comparison of drivers between global software development and agile global software development: a SURVEY. Computer and Mathematical Sciences Graduates National Colloqium 2013, At Faculty of Computer and Mathematical Sciences, UiTM Shah Ala
2. Stray V, Moe NB (2020) Understanding coordination in global software engineering: a mixed-methods study on the use of meetings and Slack. J Syst Softw 170:110717. https://doi.org/10.1016/j.jss.2020.110717
3. Zaitsev A, Gal U, Tan B (2020) Coordination artifacts in agile software development. Inf Organ 30(2):100288. https://doi.org/10.1016/j.infoandorg.2020.100288
4. Lin J (2013) Context-aware task allocation for distributed agile team. In: 2013 28th IEEE/ACM international conference on automated software engineering (ASE). IEEE, pp 758–761. https://doi.org/10.1109/ASE.2013.6693151
5. Sistla K, Sherry A (2016) A comparative study of collocated and distributed agile software development. Int J Adv Res 4:2320–5407. https://doi.org/10.21474/IJAR01/1904
6. Kudaravalli S, Faraj S, Johnson SL (2017) A configural approach to coordinating expertise in software development teams. MIS Q 41(1):43–64. https://doi.org/10.25300/MISQ/2017/41.1.03
7. Layman L, Williams L, Damian D, Bures H (2006) Essential communication practices for Extreme Programming in a global software development team. Inf Softw Technol 48(9):781–794. https://doi.org/10.1016/j.infsof.2006.01.004
8. Masood Z, Hoda R, Blincoe K (2020) How agile teams make self-assignment work: a grounded theory study. Empir Softw Eng 25(6):4962–5005. https://doi.org/10.1007/s10664-020-09876-x
9. Aslam W, Ijaz F (2018) A quantitative framework for task allocation in distributed agile software development. IEEE Access 6:15380–15390. https://doi.org/10.1109/ACCESS.2018.2803685
10. Ijaz F, Aslam W (2019) Identification of dependencies in task allocation during distributed agile software development. Sindh Univ Res J 51(01):31–36
11. Sekitoleko N, Evbota F, Knauss E, Sandberg A, Chaudron M, Olsson HH (2014) Technical dependency challenges in large-scale agile software development. In: International conference on agile software development. Springer, Cham, pp 46–61. https://doi.org/10.1007/978-3-319-06862-6_4
12. Strode DE, Huff SL (2012) A taxonomy of dependencies in agile software development. In: 23rd Australasian conference on information systems
13. Rocha R, Araujo A, Cordeiro D, Ximenes A, Teixeira J, Silva G, Duarte M (2018) DKDOnto: an ontology to support software development with distributed teams. Procedia Comput Sci 126:373–382. https://doi.org/10.1016/j.procs.2018.07.271
14. Shrivastava SV (2010) Distributed agile software development: a review. arXiv preprint arXiv:1006.1955
15. Bhatia MPS, Kumar A, Beniwal R (2015) Ontology based framework for automatic software's documentation. In: 2015 2nd international conference on computing for sustainable global development (INDIACom). IEEE, pp 421–424
16. Vizcaíno A, García F, Piattini M, Beecham S (2016) A validated ontology for global software development. Comput Stand Interfaces 46:66–78. https://doi.org/10.1016/j.csi.2016.02.004

17. Kumar SA, Kumar TA (2011) Study the impact of requirements management characteristics in global software development projects: an ontology based approach. Int J Softw Eng Appl 2(4):107. https://doi.org/10.5121/ijsea.2011.2410
18. Rocha R, Bion D, Azevedo R, Gomes A, Cordeiro D, Leandro R, Silva I, Freitas F (2020) A syntactic and semantic assessment of a global software engineering domain ontology. In: Proceedings of the 22nd international conference on information integration and web-based applications & services, pp 253–262. https://doi.org/10.1145/3428757.3429143
19. Martínez-García JR, Castillo-Barrera FE, Palacio RR, Borrego G, Cuevas-Tello JC (2020) Ontology for knowledge condensation to support expertise location in the code phase during software development process. IET Softw 14(3):234–241. https://doi.org/10.1049/iet-sen.2019.0272
20. Gruber TR (1993) A translation approach to portable ontology specifications. Knowl Acquis 5(2):199–220
21. Wongthongtham P, Chang E, Dillon T, Sommerville I (2008) Development of a software engineering ontology for multisite software development. IEEE Trans Knowl Data Eng 21(8):1205–1217. https://doi.org/10.1109/TKDE.2008.209
22. Pahl C (2007) An ontology for software component matching. Int J Softw Tools Technol Transf 9(2):169–178. https://doi.org/10.1007/s10009-006-0015-9
23. Rajpathak D, Motta E, Roy R (2001) A generic task ontology for scheduling applications. In: International conference on artificial intelligence (ICAI'2001)
24. Almeida LH, Pinheiro PR, Albuquerque AB (2011) Applying multi-criteria decision analysis to global software development with scrum project planning. In: International conference on rough sets and knowledge technology. Springer, Berlin, Heidelberg, pp 311–320. https://doi.org/10.1007/978-3-642-24425-4_41
25. Marques AB, Carvalho J R, Rodrigues R, Conte T, Prikladnicki R, Marczak S (2013) An ontology for task allocation to teams in distributed software development. In: 2013 IEEE 8th international conference on global software engineering. IEEE, pp 21–30. https://doi.org/10.1109/ICGSE.2013.12
26. Mak DK, Kruchten PB (2006) Task coordination in an agile distributed software development environment. In: 2006 Canadian conference on electrical and computer engineering. IEEE, pp 606–611. https://doi.org/10.1109/CCECE.2006.277524
27. Falbo RDA, Bertollo G (2009) A software process ontology as a common vocabulary about software processes. Int J Bus Process Integr Manag 4(4):239–250. https://doi.org/10.1504/IJBPIM.2009.032281
28. Santos G, Villela K, Montoni M, Rocha AR, Travassos GH, Figueiredo S, Amaral M (2005) Knowledge management in a software development environment to support software processes deployment. In: Biennial conference on professional knowledge management/Wissensmanagement. Springer, Berlin, Heidelberg, pp 111–120. https://doi.org/10.1007/11590019_14
29. Almeida LH, Albuquerque AB (2011) A multi-criteria model for planning and fine-tuning distributed scrum projects. In: 2011 IEEE sixth international conference on global software engineering. IEEE, pp 75–83. https://doi.org/10.1109/ICGSE.2011.36
30. Paasivaara M, Blincoe K, Lassenius C, Damian D, Sheoran J, Harrison F, Isotalo V (2015) Learning global agile software engineering using same-site and cross-site teams. In: 2015 IEEE/ACM 37th IEEE international conference on software engineering. IEEE, vol 2, pp 285–294. https://doi.org/10.1109/ICSE.2015.157
31. Sauer J (2010) Architecture-centric development in globally distributed projects. In: Agility across time and space. Springer, Berlin, Heidelberg, pp 321–329. https://doi.org/10.1007/978-3-642-12442-6_22
32. Suarez-Figueroa MC, Gomez-Perez A, Motta E, Gangemi A (2012) Introduction: ontology engineering in a networked world. In Ontology engineering in a networked world. Springer, Berlin, Heidelberg, pp 1–6. ISBN 978-3-642-24794-1
33. Sheth A (ed) (2012) Semantic-enabled advancements on the web: applications across industries: applications across industries. IGI Global. ISBN: 9781466601857
34. Suárez-Figueroa MC (2009) D5.3.2 Revision and extension of the NeOn development process and ontology life cycle. ISBN 3-540-44268-5. 28
35. Nundlall C, Nagowah SD (2021) Task allocation and coordination in distributed agile software development: a systematic review. Int J Inf Tecnol 13:321–330. https://doi.org/10.1007/s41870-020-00543-4
36. Nundlall C, Nagowah SD (2021) Factors affecting task allocation and coordination in distributed agile software development. In: Panigrahi CR, Pati B, Pattanayak BK, Amic S, Li K-C (eds) Progress in advanced computing and intelligent engineering. Advances in intelligent systems and computing, vol 1299. Springer, Singapore. https://doi.org/10.1007/978-981-33-4299-6_66
37. Bick S, Spohrer K, Hoda R, Scheerer A, Heinzl A (2017) Coordination challenges in large-scale software development: a case study of planning misalignment in hybrid settings. IEEE Trans Softw Eng 44(10):932–950. https://doi.org/10.1109/TSE.2017.2730870
38. Hashmi AS, Hafeez Y, Jamal M, Ali S, Iqbal N (2019) Role of situational agile distributed model to support modern software development teams. Mehran Univ Res J Eng Technol 38(3):655–666
39. Moe NB, Cruzes D, Dybå T, Mikkelsen E (2015) Continuous software testing in a globally distributed project. In: 2015 IEEE 10th international conference on global software engineering. IEEE, pp 130–134. https://doi.org/10.1109/ICGSE.2015.24
40. Moe NB, Šmite D, Šāblis A, Börjesson AL, Andréasson P (2014) Networking in a large-scale distributed agile project. In: Proceedings of the 8th ACM/IEEE international symposium on empirical software engineering and measurement, pp 1–8. https://doi.org/10.1145/2652524.2652584
41. Szőke Á (2010) Optimized feature distribution in distributed agile environments. In: International conference on product focused software process improvement. Springer, Berlin, Heidelberg, pp 62–76. https://doi.org/10.1007/978-3-642-13792-1_7
42. Collins E, Macedo G, Maia N, Dias-Neto A (2012) An industrial experience on the application of distributed testing in an agile software development environment. In: 2012 IEEE seventh international conference on global software engineering. IEEE, pp 190–194. https://doi.org/10.1109/ICGSE.2012.40
43. Papadopoulos G (2015) Moving from traditional to agile software development methodologies also on large, distributed projects. Procedia Soc Behav Sci 175(2):455–463. https://doi.org/10.1016/j.sbspro.2015.01.1223
44. Simão Filho M, Pinheiro PR, Albuquerque AB (2015) Task allocation approaches in distributed agile software development: a quasi-systematic review. In: Software engineering in intelligent systems. Springer, Cham, pp 243–252. https://doi.org/10.1007/978-3-319-18473-9_24
45. Nordio M, Estler HC, Meyer B, Aguirre N, Prikladnicki R, Di Nitto E, Savidis A (2014) An experiment on teaching coordination in a globally distributed software engineering class. In: 2014 IEEE 27th conference on software engineering education and training (CSEE&T). IEEE, pp 109–118. https://doi.org/10.1109/CSEET.2014.6816788

46. Banijamali A, Dawadi R, Ahmad MO, Similä J, Oivo M, Liukkunen K (2016) An empirical study on the impact of Scrumban on geographically distributed software development. In: 2016 4th international conference on model-driven engineering and software development (MODELSWARD). IEEE, pp 567–577. ISBN:978-1-5090-5898-3
47. Alzoubi YI, Gill AQ, Al-Ani A (2015) Distributed agile development communication: an agile architecture driven framework. JSW 10(6):681–694. https://doi.org/10.17706/jsw.10.6.681-694
48. McCarthy S, O'Raghallaigh P, Fitzgerald C, Adam F (2019) Towards a framework for shared understanding and shared commitment in agile distributed ISD project teams. In: ECIS 2019, proceedings of the 27th European conference on information systems. AIS Electronic Library (AISeL), pp 1–15. ISBN 978-1-7336325-0-8
49. Nyrud H, Stray V (2017) Inter-team coordination mechanisms in large-scale agile. In: Proceedings of the XP2017 scientific workshops, pp 1–6. https://doi.org/10.1145/3120459.3120476
50. Ortega-Ordoñez WA, Pardo-Calvache CJ, Pino-Correa FJ (2019) OntoAgile: an ontology for agile software development processes. DYNA 86(209):79–90. https://doi.org/10.15446/dyna.v86n209.76670
51. MacLarty I, Langevine L, Bossche MV, Ross P (2009) Using SWRL for rule-driven applications
52. Fernández M, Overbeeke C, Sabou M, Motta E (2009) What makes a good ontology? A case-study in fine-grained knowledge reuse. In: Asian semantic web conference. Springer, Berlin, Heidelberg, pp 61–75. https://doi.org/10.1007/978-3-642-10871-6_5
53. Smith RK, Hale JE, Parrish AS (2001) An empirical study using task assignment patterns to improve the accuracy of software effort estimation. IEEE Trans Softw Eng 27(3):264–271
54. Rocha R, Leandro R, Silva I, Araujo J, Bion D, Freitas F, Cordeiro D, Gomes A, Azevedo R (2021) DKD-S: an ontology-based tool for global software development. In: 2021 16th Iberian conference on information systems and technologies (CISTI). IEEE, pp 1–6. https://doi.org/10.23919/CISTI52073.2021.9476386
55. Anzures-García M, Sánchez-Gálvez LA, Hornos MJ, Paderewski-Rodríguez P (2018) A workflow ontology to support knowledge management in a group's organizational structure. Comput Sist 22(1):163–178. https://doi.org/10.13053/cys-22-1-2781