Received September 15, 2017, accepted October 18, 2017, date of publication November 8, 2017, date of current version December 5, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2771257

Ontology Based Multiagent Effort Estimation System for Scrum Agile Method

MUHAMMAD ADNAN AND MUHAMMAD AFZAL
Department of Computer Science and Engineering, University of Engineering and Technology, Lahore 54890, Pakistan
Corresponding author: Muhammad Adnan (adnan4any@hotmail.com)

ABSTRACT This paper emphasizes on software effort estimation and knowledge management of practicing Scrum methodology that are challenging tasks in agile context. Proposed approach improves software effort estimation and knowledge management of software projects by focusing on Scrum process and practices using ontology model in a multiagent estimation system. It also motivates project key stakeholders to regularly save significant tacit knowledge of unique situations in the form of lessons learnt during the project development. Various agents of the estimation system access the existing knowledge base and autonomously perform their inferencing activities using description logic as per requirements specified by the scrum master and respond with suitable estimate to him/her in the form of time, resources, and lessons learnt for the success of future projects. To validate our approach, an experiment, based on twelve web projects, was conducted using proposed approach, delphi and planning poker estimation methods. The obtained results by applying MMRE, PRED(x) evaluation measures reveals that proposed approach delivers more accurate estimates as compared with delphi and planning poker methods.

INDEX TERMS Knowledge management, multiagent effort estimation system, ontology, Scrum.

I. INTRODUCTION

Nowadays, most of the software companies are developing their small to medium size softwares using Scrum process and practice. They invest a lot of money on process improvement and resource management for upcoming projects that are necessary to be developed within time and cost. An experienced project leader is a major resource for the success of these projects. Other difficult to achieve essential ingredients are the accurate estimate of software size, team size, software effort and completion time [1]. Currently, in the software industry inaccurate estimation of effort and time are severe problems that impacts software projects [2]. The estimate accuracy depends upon good analysis of the software development context, experience of the project leader and previous successful projects of a software company. Anyhow, according to the IT leadership survey conducted by BluePrint Software Systems, more than fifty-nine per cent projects are typically delivered late and over budget, while only eleven percent of IT projects are completed within the original budget [3].

In our view, these problems in the software development result from two shortcomings in the Scrum software practice. Firstly, the software companies are rapidly developing their projects and products without saving complete information or insights acquired in a structured semantic format which can support easy comprehension of project's nature. Secondly, lessons learnt by scrum masters and other project stakeholders during the development of the software projects are not being saved which can really help the software development companies to resolve the complex situations of their future sprints and projects. The scrum masters or project stakeholders do gain experience and tacit knowledge through learnt lessons to be capable of making more reliable decisions. The software companies lose all that tacit knowledge body when their scrum masters or project stakeholders leave. To deal with these problems, we propose knowledge management components to comfortably capture and organize the project knowledge based on Scrum regularly in semantic and well-structured manners. This paper presents a multiagent estimation system which reasons through description logic and Scrum ontology model to help scrum masters or software companies in producing reliable estimates. The detailed proposed approach is explained in section-III. Furthermore, in the section-IV, we have explained a case study of shopping cart project estimation in the form of ontology graphs and describe how relevant multiple agents work altogether and produce an estimate after analyzing the knowledgebase. Conclusively, in order to validate our approach, an experiment and its obtained results are also presented in section-V.

II. BACKGROUND AND RELATED WORK

Multiagent systems (MAS) are evolving with time by integrating many disciplines to develop robust and intelligent softwares. Many MAS features are now being used in the development of agent-oriented softwares, ontology based softwares, large-scale industrial and business systems which involve hundreds of agents [4]. These are software entities which really help to monitor the events for the users and also assist them on how to perform their tasks effectively and efficiently. Key attributes that belong to MAS are: autonomy, reactivity, social-ability, communication, inferencing, coordination, collaboration, learning and goal orientation [5]. Moreover, agent-based systems have brought new insights and ways to look at modern softwares. Many agent based approaches are being developed to improve the process and work of various systems. In this paper, we are focusing on two aspects that are Scrum methodology and software effort estimation. In fact, Scrum is one of the dominated agile methods for software development that emphasizes on simplicity, flexibility, team coordination, customer involvement and certain productivity. It focuses on small team size that prioritizes the backlogs to be considered for developments in the form of reasonable sprints until the product is deployed with continuous customer improvements and feedbacks [6], [7]. Our approach uses ontology to provide structure and semantics to all core elements of Scrum process and practices. The purpose is to facilitate key stakeholders of the project to save their lessons learnt during the development of their projects and products into a semantically-aware structured repository. Secondly, multiagent effort estimation system is proposed in which description logic of each agent is defined and implemented to conduct the reliable effort estimation according to the scrum master's specifications. All agents in the system analyze and extract the relevant knowledge as per the specifications from the semantic repository and produce a suitable estimate for the upcoming Scrum project.

Many research studies show that various estimation models, tools and techniques are being used in the software industry as a solution to deal with the software effort estimation challenge including COCOMO, COCOMO-II , Cosmic function point analysis, Putnam, Delphi technique, analogy based estimation and expert judgment [8], [9]. Mahnic and Hovelja highlight a consensus–based effort estimation technique that leads to more accurate estimates called planning poker which is widely used in agile software development like Scrum and Extreme programming [10]. Janeth *et al.* focus on planning poker technique for estimating the size of user stories in the context of Scrum and propose a new model to estimate the complexity, importance of user stories and correlate Bayesian network to get an accurate estimation [11]. A study of Gupta and Kumar concentrates on the Delphi technique which is a looping process to collect the effort estimates from the judgments of experts having professional experience of relevant application areas. They introduced an improved Delphi technique to estimate the software development efforts more accurately [12]. A comparative study, conducted by Usman and Britto, indicates that agile practitioners, both in co-located and distributed agile software development contexts apply experts' subjective assessment to estimate the effort and uses story points for size metrics frequently. Likewise, effort is estimated mainly at iteration and release planning levels but underestimation is dominant for both the agile contexts [13]. Jorgensen suggested that software development effort estimates are usually expert judgment-based, effort estimation in work-hours unit leads to low estimates than in work-days unit and underestimation can be reduced by opting higher granularity effort units [14]. Binish, Liliana and Martin reveal that effort estimation is more challenging task in the agile context and important factors that impact the estimation accuracy are developer knowledge, experience, complexity and impacts of changes on the under-developed system [15]. Chetan and Asheesh have reported that approximately thirty percent software projects become successful and there are many reasons for software failure but effort estimation is one of them. They have proposed three point techniques with analogy-based estimation that allow making accurate estimates on the average with three different values [16]. Srdjana *et al.* propose a Bayesian network model for effort estimation to be suitable for any agile method and be useful at early project planning stage. Their technique apply statistical methods like MMRE, PRED(m). Their results show very good prediction accuracy [17]. Due to the increasing complexity of software engineering projects, many Agent Oriented Software Engineering (AOSE) methods have been proposed for estimation and development purposes in the last few years [18]. Ontology and agent-oriented technology are emerging research trends in open source, commercial and global software developments, as well as architectural design, networking, games and education. Jasper *et al.* studied the usage of ontology in the context of software development for the sake of communication, interoperability, specification, reusability and maintenance [19], [20]. Freitas and Vieira designed ontology to represent core knowledge of software performance testing, especially, to plan, guide and manage testing of softwares [21]. Rocha et al. identified a number of new challenges in global software development and proposed an ontology-based system which transforms software development process with distributed teams to be the best practice [22]. Feldmann *et al.* proposed an ontology-based approach that integrates the software requirements and test cases at early stage to ensure quality and correctness to complex functionalities of the system [23]. Quanwang *et al.* introduced the quality of service (QoS)-aware multigranularity ontology based service composition approach to meet the users' global QoS constraints and to maximize QoS value [24]. Simmons and Dillon proposed ontology to effectively manage semantically aware open source repositories for open source software development [25].

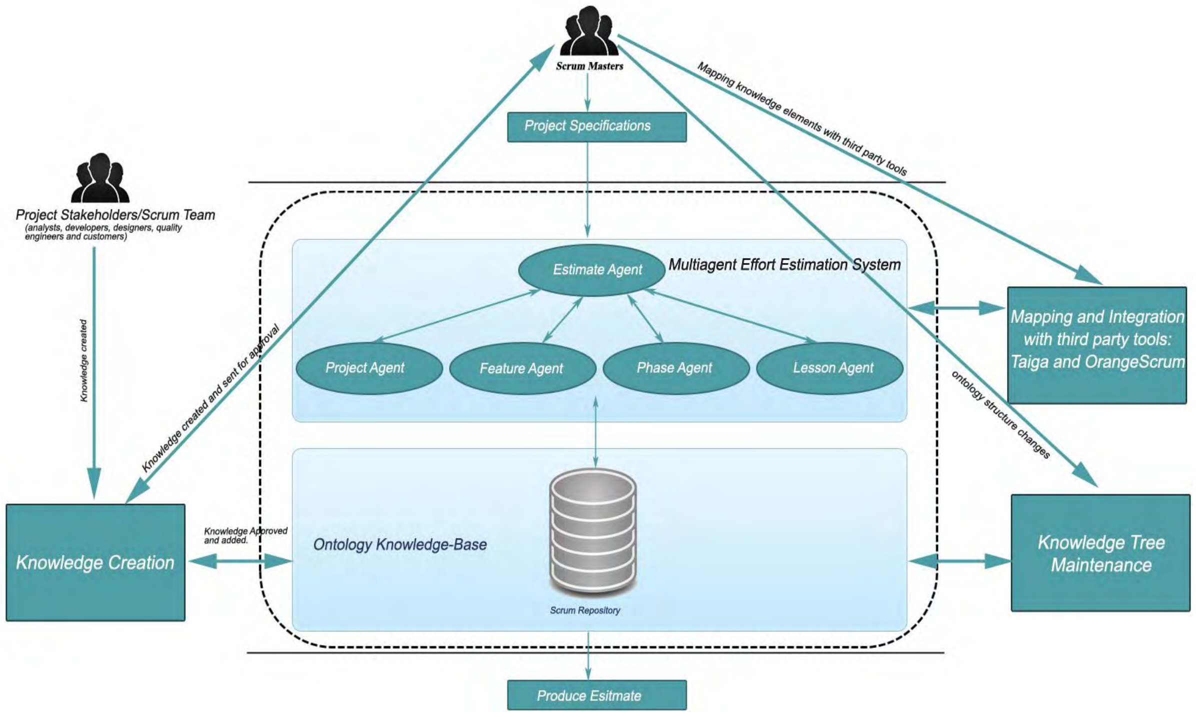

Scrum Masters
Project Specifications
Mapping knowledge elements with third party tools
Project Stakeholders/Scrum Team
(analysts, developers, designers, quality engineers and customers)
Knowledge created
Knowledge created and sent for approval
Estimate Agent
Multiagent Effort Estimation System
Project Agent
Feature Agent
Phase Agent
Lesson Agent
Mapping and Integration with third party tools: Taiga and OrangeScrum
ontology structure changes
Knowledge Creation
Knowledge Approved and added.
Ontology Knowledge-Base
Scrum Repository
Knowledge Tree Maintenance
Produce Esitmate

FIGURE 1. Detailed architecture of proposed approach.

Brain and Cesar suggested ontology for software development methodologies which include both product and process aspects [26]. Shilpa and Maya describe an ontology-oriented reliability (OnO-Reliability) development which deals with reliability issues from very start of software development process to enhance reliability as per products and process attributes [27]. Not many ontology languages have been developed so far. Some well-known languages are recommended by W3C for Semantic Web. These yet evolving languages are referred as ontology markup languages or web-based ontology languages. Most of these languages are based on Extensible Markup Language (XML) syntax [28]. Some examples of such ontology markup languages are XML, RDF, RDFS, OWL, and SPARQL. However, we could not find any ontology or agent based approach, in the literature, to facilitate the Scrum process management in semantic and structured manner for the software development and better estimation accuracy.

III. PROPOSED APPROACH

Our proposed approach comprises of five core components, namely, multiagent effort estimation system, ontology knowledge-base, knowledge creation, knowledge tree maintenance and facility to software companies to map and integrate project data with our system to make quality estimates. Our effort estimation system consists of multiple agents that inference from given project specifications and interact with ontology knowledge-base to produce a quality estimate. The Scrum masters specify software domain, a set of relevant projects already completed, top-k learnt lessons and their priority type to initiate estimation process. The multiagent estimation system then activates and assigns the task to the relevant agents. Multiple agents analyze the current knowledge-base by using description logic and OWL reasoner as per the scrum master requirements. Once all the agents achieve their assigned goals, their result is compiled into an estimate report to the Scrum masters as per request. Secondly, ontology knowledge-base consists of Scrum ontology model used to capture and organize the project knowledge into scrum repository regularly in well-structured and semantic manner. Thirdly, project stakeholders or Scrum team can create new knowledge using knowledge creation component. Fourthly, structural changes of Scrum ontology are carried out by knowledge tree maintenance component. Lastly, our approach also provide the facility to the software development companies to map and integrate with the third party tools to import the projects data based on scrum practices into our system that enables them to use knowledge management components and multiagent estimation system to produce the quality estimates for their upcoming projects. These components are explained as follows while detailed architecture of proposed approach is depicted in Fig. 1.

A. MULTIAGENT EFFORT ESTIMATION SYSTEM

The first core component is multiagent effort estimation system that initiates the estimation mechanism for the Scrum master in order to get the quality estimate for the upcoming project by receiving his or her given specifications which includes: software domain, a set of relevant projects already completed, top-k learnt lessons and their priority type.

After receiving project specifications from the Scrum master, the multiagent starts extracting and analyzing the already completed relevant projects from the ontology knowledge-base according to the given specifications by applying their defined description logic using HermiT OWL reasoner. Once all the agents accomplish assigned goals, their results are compiled into an estimate report and presented to the Scrum master. The main purpose of suggested underlying ontology model is to support the proposed estimation mechanism by providing a better structure and semantics to the knowledge of projects that are based on Scrum methodology. The underlying ontology facilitates application of defined description logic of the multiagent to extract information required to produce the quality estimates in minimum time and effort.

The multiagent effort estimation system consists of five agents named as project agent, phase agent, lesson agent, feature agent and estimate agent, respectively. The project agent analyzes the ontology knowledge-base to calculate the average number of sprints required for the project and number of days to complete each sprint from the relevant projects extracted from the ontology. The phase agent suggests the average team size and average length of each phase of Scrum process, named as, analysis, design, coding and testing. The lesson agent suggests the most relevant priority based lessons that closely relate to the phases of the project. The feature agent analyzes the knowledge-base to suggest the number of features that should be implemented, on the average, against each sprint of the project. The estimate agent forwards requirements specified by scrum master to other agents. The agents work in parallel and submit their inferred quality estimates to the estimate agent that calculates the resources, effort required in person-days and presents the final estimate to the scrum master. Fig. 2 represents the detailed description logic used by each agent to produce an estimate.

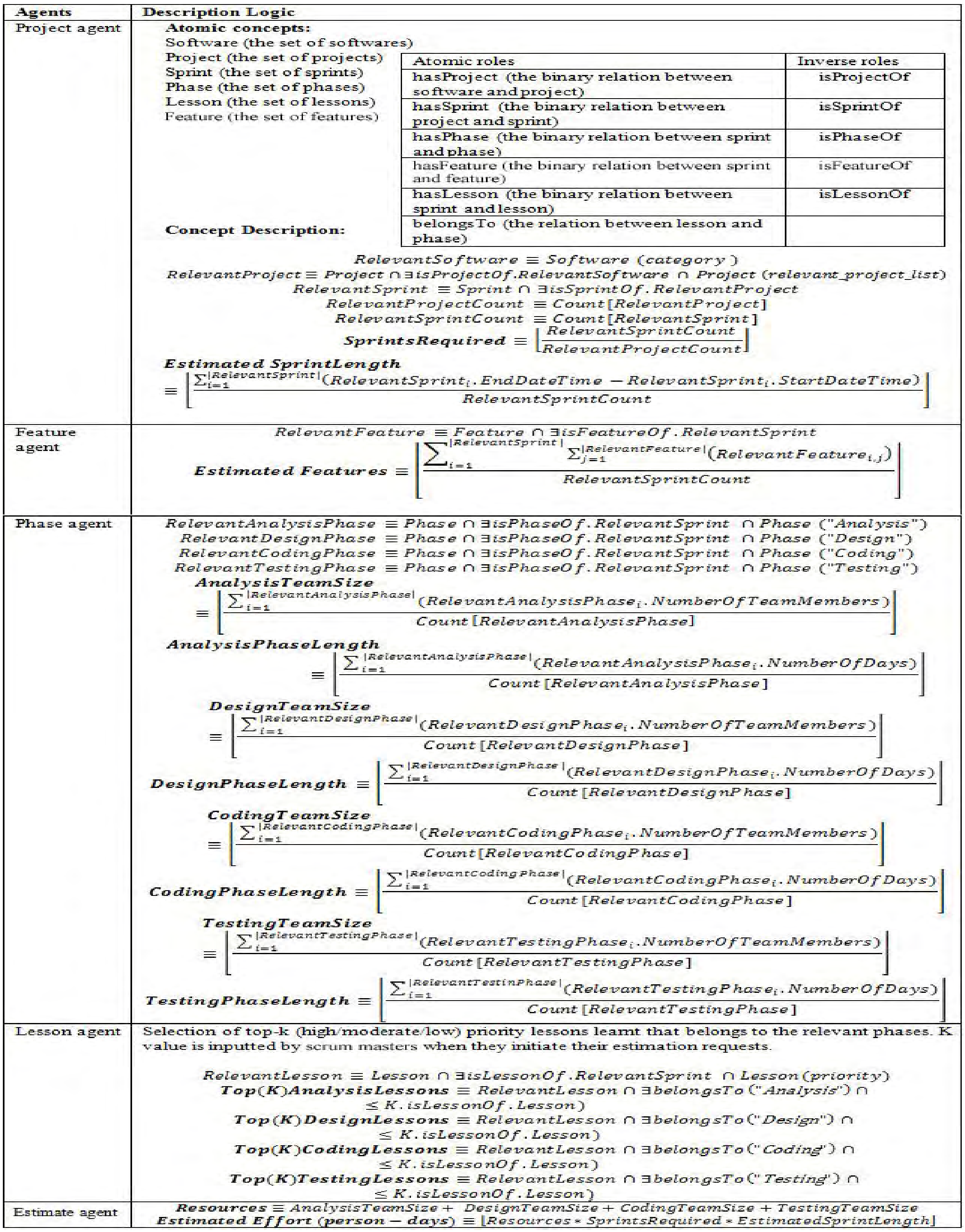
The description logic and the concepts used by the multiagent, as depicted in Fig. 2, are concrete formulas for the given estimation mechanism because the concepts used in the description logic conform to the knowledge classes and relevant terminologies of the underlying ontology model. However, it may evolve gradually depending upon the structural changes in the ontology model with the passage of time due to dynamic needs of software development companies practicing Scrum methodology.

According to Fig. 2, description logic used by multiagent consists of six atomic concepts (i.e., Software, Project, Sprint, Phase, Lesson and Feature), six atomic roles (i.e., hasProject, hasSprint, hasPhase, hasFeature, hasLesson and belongsTo) and five inverse roles (i,e., isProjectOf, isSprintOf, isPhaseOf, isFeatureOf and isLessonOf) that develop essential concepts to be used by Hermit OWL reasoner. Each atomic concept represents a set and each role represents a binary relationship between two concepts. For example, 'hasProject' atomic role represents the binary relationship between a software and a project, 'hasSprint' represents the binary relationship between a project and a sprint, 'hasPhase' denotes the binary relation between a sprint and a phase, 'hasFeature' denotes the binary relationship between a sprint and a feature, 'hasLesson' denoted the binary relationship between a sprint and a lesson, and 'belongsTo' represents the binary relation between a lesson and a phase, respectively. The 'RelevantSoftware', 'RelevantProject' and 'RelevantSprint' concepts are derived through defined description logic used by the project agent to infer the sets of relevant softwares as per the given software domain or category. Further, project agent infers relevant projects that are already completed and their relevant sprints from the ontology knowledge-base through respective logic expressions. Afterwards, project agent infers count of relevant projects and their sprints using expressions for 'RelevantProjectCount' and 'RelevantSprintCount' that are used to get 'SprintsRequired'. The project agent finalizes its goals with 'EstimatedSprintLength' inferred from average days of all 'RelevantSprint' and submits goals to estimate agent. The feature agent infers relevant features from inferred relevant sprints by using 'RelevantFeature' description logic expression. It calculates the 'EstimatedFeatures' by averaging features of each relevant sprint and submits the result to the estimate agent.

Similarly, phase agent gets all the relevant phases involving analysis, design, coding and testing for all inferred relevant sprints by using respective description logic expressions for 'RelevantAnalysisPhase' ,'RelevantDesignPhase', 'RelevantCodingPhase' and 'RelevantTestingPhase', respectively. It suggests the team sizes of the four phases through logic expressions for 'AnalysisTeamSize', 'DesignTeamSize', 'CodingTeamSize' and 'TestingTeamSize' mentioned in Fig. 2. Phase agent also gets lengths of each phase in number of days through expression for 'AnalysisPhaseLength', 'DesignPhaseLength', 'CodingPhaseLength' and 'TestingPhaseLength'. When done, phase agent submits these eight results to the estimate agent.

The lesson agent serves to resolve the complex situations of upcoming project by providing a list of the lessons inferred from relevant sprints with given priority high, moderate or low by using the 'RelevantLesson' expression. It suggests the top-k priority based lessons from the relevant phases, separately for analysis, design, coding and testing by using 'Top(K)AnalysisLessons', 'Top(K)DesignLessons', 'Top(K)CodingLessons' and 'Top(K)TestingLessons' concepts defined in description logic. The values for K and lesson priority are specified by the scrum masters when they initiate the estimation process. The lesson agent also submits the inferred results to the estimate agent.

Once estimate agent receives all the inferred results from each agent, it gets 'Resources' by adding up 'AnalysisTeamSize', 'DesignTeamSize', 'CodingTeamSize' and 'TestingTeamSize' to further calculate the 'Estimated Effort (person-days)' by multiplying 'Resources', 'SprintsRequired' and 'EstimatedSprintLength'. Eventually, it compiles their results into an estimate report and presents it to the scrum master as per his/her given specifications.



Agents	Description Logic
Project agent	**Atomic concepts:** Software (the set of softwares) Project (the set of projects) Sprint (the set of sprints) Phase (the set of phases) Lesson (the set of lessons) Feature (the set of features) **Concept Description:** $RelevantSoftware \equiv Software\ (category)$ $RelevantProject \equiv Project \sqcap \exists isProjectOf.RelevantSoftware \sqcap Project\ (relevant\_project\_list)$ $RelevantSprint \equiv Sprint \sqcap \exists isSprintOf.RelevantProject$ $RelevantProjectCount \equiv Count[RelevantProject]$ $RelevantSprintCount \equiv Count[RelevantSprint]$ $\boldsymbol{SprintsRequired} \equiv \left\lfloor \frac{RelevantSprintCount}{RelevantProjectCount} \right\rfloor$ $\boldsymbol{Estimated\ SprintLength} \equiv \left\lfloor \frac{\sum_{i=1}^{\lvert RelevantSprint \rvert}(RelevantSprint_i.EndDateTime - RelevantSprint_i.StartDateTime)}{RelevantSprintCount} \right\rfloor$
Feature agent	$RelevantFeature \equiv Feature \sqcap \exists isFeatureOf.RelevantSprint$ $\boldsymbol{Estimated\ Features} \equiv \left\lfloor \frac{\sum_{i=1}^{\lvert RelevantSprint \rvert} \sum_{j=1}^{\lvert RelevantFeature \rvert}(RelevantFeature_{i,j})}{RelevantSprintCount} \right\rfloor$
Phase agent	$RelevantAnalysisPhase \equiv Phase \sqcap \exists isPhaseOf.RelevantSprint \sqcap Phase\ (\text{"Analysis"})$ $RelevantDesignPhase \equiv Phase \sqcap \exists isPhaseOf.RelevantSprint \sqcap Phase\ (\text{"Design"})$ $RelevantCodingPhase \equiv Phase \sqcap \exists isPhaseOf.RelevantSprint \sqcap Phase\ (\text{"Coding"})$ $RelevantTestingPhase \equiv Phase \sqcap \exists isPhaseOf.RelevantSprint \sqcap Phase\ (\text{"Testing"})$ $\boldsymbol{AnalysisTeamSize} \equiv \left\lfloor \frac{\sum_{i=1}^{\lvert RelevantAnalysisPhase \rvert}(RelevantAnalysisPhase_i.NumberOfTeamMembers)}{Count\ [RelevantAnalysisPhase]} \right\rfloor$ $\boldsymbol{AnalysisPhaseLength} \equiv \left\lfloor \frac{\sum_{i=1}^{\lvert RelevantAnalysisPhase \rvert}(RelevantAnalysisPhase_i.NumberOfDays)}{Count\ [RelevantAnalysisPhase]} \right\rfloor$ $\boldsymbol{DesignTeamSize} \equiv \left\lfloor \frac{\sum_{i=1}^{\lvert RelevantDesignPhase \rvert}(RelevantDesignPhase_i.NumberOfTeamMembers)}{Count\ [RelevantDesignPhase]} \right\rfloor$ $\boldsymbol{DesignPhaseLength} \equiv \left\lfloor \frac{\sum_{i=1}^{\lvert RelevantDesignPhase \rvert}(RelevantDesignPhase_i.NumberOfDays)}{Count\ [RelevantDesignPhase]} \right\rfloor$ $\boldsymbol{CodingTeamSize} \equiv \left\lfloor \frac{\sum_{i=1}^{\lvert RelevantCodingPhase \rvert}(RelevantCodingPhase_i.NumberOfTeamMembers)}{Count[RelevantCodingPhase]} \right\rfloor$ $\boldsymbol{CodingPhaseLength} \equiv \left\lfloor \frac{\sum_{i=1}^{\lvert RelevantCodingPhase \rvert}(RelevantCodingPhase_i.NumberOfDays)}{Count\ [RelevantCodingPhase]} \right\rfloor$ $\boldsymbol{TestingTeamSize} \equiv \left\lfloor \frac{\sum_{i=1}^{\lvert RelevantTestingPhase \rvert}(RelevantTestingPhase_i.NumberOfTeamMembers)}{Count\ [RelevantTestingPhase]} \right\rfloor$ $\boldsymbol{TestingPhaseLength} \equiv \left\lfloor \frac{\sum_{i=1}^{\lvert RelevantTestinPhase \rvert}(RelevantTestingPhase_i.NumberOfDays)}{Count\ [RelevantTestingPhase]} \right\rfloor$
Lesson agent	Selection of top-k (high/moderate/low) priority lessons learnt that belongs to the relevant phases. K value is inputted by scrum masters when they initiate their estimation requests. $RelevantLesson \equiv Lesson \sqcap \exists isLessonOf.RelevantSprint \sqcap Lesson(priority)$ $\boldsymbol{Top(K)AnalysisLessons} \equiv RelevantLesson \sqcap \exists belongsTo(\text{"Analysis"}) \sqcap \leq K.isLessonOf.Lesson)$ $\boldsymbol{Top(K)DesignLessons} \equiv RelevantLesson \sqcap \exists belongsTo(\text{"Design"}) \sqcap \leq K.isLessonOf.Lesson)$ $\boldsymbol{Top(K)CodingLessons} \equiv RelevantLesson \sqcap \exists belongsTo(\text{"Coding"}) \sqcap \leq K.isLessonOf.Lesson)$ $\boldsymbol{Top(K)TestingLessons} \equiv RelevantLesson \sqcap \exists belongsTo(\text{"Testing"}) \sqcap \leq K.isLessonOf.Lesson)$
Estimate agent	$\boldsymbol{Resources} \equiv AnalysisTeamSize + DesignTeamSize + CodingTeamSize + TestingTeamSize$ $\boldsymbol{Estimated\ Effort\ (person - days)} \equiv \lfloor Resources * SprintsRequired * EstimatedSprintLength \rfloor$

Atomic roles	Inverse roles
hasProject (the binary relation between software and project)	isProjectOf
hasSprint (the binary relation between project and sprint)	isSprintOf
hasPhase (the binary relation between sprint and phase)	isPhaseOf
hasFeature (the binary relation between sprint and feature)	isFeatureOf
hasLesson (the binary relation between sprint and lesson)	isLessonOf
belongsTo (the relation between lesson and phase)	

FIGURE 2. Description logic used by multiagents to produce an estimate.

B. ONTOLOGY KNOWLEDGE-BASE

Second component, the ontology knowledge-base, encompasses the knowledge classes of Scrum software process developed using OWL (Web Ontology Language). There are thirty-four classes in our Scrum ontology model as depicted in Fig. 3. This model is described in the form of 5-tuple, that are, $\langle C, I, P, R_H, R_C \rangle$. The symbol C represents a set of classes, I represent a set of Instances, P represents a set of Properties, R_H represents the class hierarchy and R_C represents the association among individual classes. Instance is an

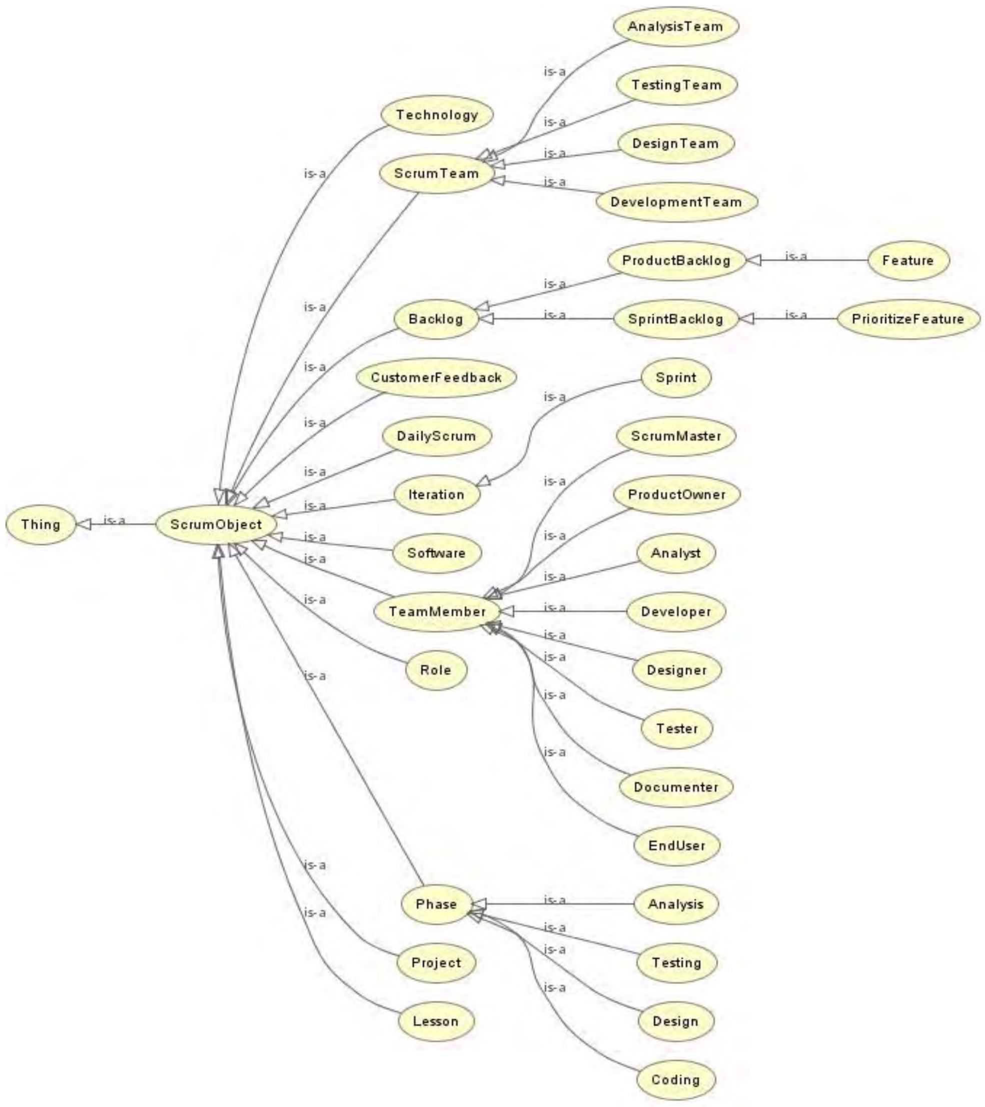


FIGURE 3. Scrum ontology model.

object that belongs to a particular class. Property represents Project, Sprint and concept name. Other tuples indicate the relationship among concepts. According to our Scrum ontology model, example components of 5-tuple is described as follows:

C: {ScrumObject, Project, Phase, Lesson, Role, TeamMember, Software, Sprint, DailyScrum CustomerFeedback, Backlog, ScrumTeam, Technology, Iteration, Analysis, Coding, Design, Testing, EndUser, Documenter, Tester, Designer, Developer, Analyst, ProductOwner, ScrumMaster, SprintBacklog, ProductBacklog, AnalysisTeam, DesignTeam, DevelopmentTeam, TestingTeam, Feature, PrioritizeFeature}

I: {i1,i2,i3,i4,i5,i6,i7,i8}

P: {(P1, spr1,i4),(P2, spr2,i5),...}

R_H: {i1(i2,i3),i2(i4,i5,i6),...}

R_C: {(i4,i7),...}

A case of Scrum ontology is shown in Fig. 4 for an Inventory Project. Classes used in the example are: Project (P), Sprint (Spr), Feature (F) and Lesson (L). I is the set of instances of these classes; for example, i1 is an instance of project class; i2 and i3 are instances of sprint class; i4, i5 and i7 are instances of feature class; similarly, i6 and i8 are instances of lesson class. P is the set of properties; according to the example, (P1, spr1, i4) means that instance i4 belongs to the sprint spr1 of project P1. R_H is a set of relations in the form of class hierarchy; for example, i1 (i2, i3) shows that i1 has two sub instances as i2, i3. R_C describes the set of similarities among features, lessons, sprints and projects.

Our developed ontology model's main purpose is to organize knowledge of current and previous projects in well-structured and semantic order. Ontology knowledge-base consists of Scrum repository that captures all the approved

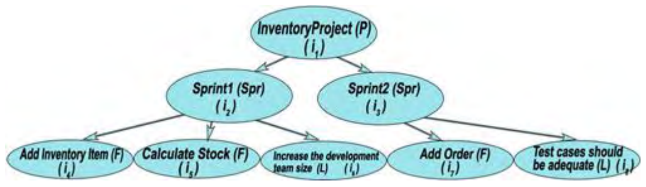

InventoryProject (P) (i_1)
Sprint1 (Spr) (i_2)
Sprint2 (Spr) (i_3)
Add Inventory Item (F) (i_4)
Calculate Stock (F) (i_5)
Increase the development team size (L) (i_6)
Add Order (F) (i_7)
Test cases should be adequate (L) (i_8)

FIGURE 4. Example of scrum process ontology.

knowledge instances and ontology structural changes regularly.

C. KNOWLEDGE CREATION COMPONENT

All project stakeholders including analysts, designers, software engineers and quality engineers of Scrum team add knowledge by creating instances of Scrum ontology regularly using knowledge creation component. Knowledge instances contain worthwhile facts of underdeveloped Scrum based projects which include sprints activities, product and sprint backlog information, development phases, prioritize features, assigned team members, time information, customer feedbacks and lessons learnt. Each new knowledge instance is then routed to the assigned Scrum master for a quick approval. Before finally approving, the Scrum master checks whether the new knowledge instances contain adequate information in relevance to the current projects. Scrum master validates whether newly created knowledge instances are rational, contain valuable piece of information that can be beneficial to the scrum teams or key stakeholders. Otherwise he or she discards that invalid knowledge instances. After validation Scrum master approves saving of that knowledge directly into the knowledge-base by using knowledge creation component.

D. KNOWLEDGE TREE MAINTENANCE COMPONENT

Due to dynamic needs of software companies, expert analysis is performed to update the knowledge classes and structure of Scrum ontology using knowledge tree maintenance component. This component is solely used by the Scrum masters of the underdeveloped projects to manage the structural changes of Scrum ontology.

E. MAPPING AND INTEGRATION

This component will facilitate software development companies that practice Scrum methodology to map and import their existing projects data to our proposed system in minimum time and effort. So, that they can easily integrate and manage their project knowledge in a semantic order. By using this component, software companies can shift their data easily after defining element to element mapping by following stepwise guidelines and can map their core elements to our existing knowledge classes of Scrum ontology model. Furthermore, they can use the facilities of multiagent estimation system and knowledge management components to get reliable estimates for their upcoming projects based on their own imported data. So far, our proposed system allows integration with two third party tools, namely, Taiga and OrangeScrum.

Moreover, the proposed approach enables the Scrum masters and other project stakeholders to save their experiences and lesson learnt with priority as high, low and moderate during the sprints after facing unique situations. This vital information really helps the scrum masters and other project stakeholders to resolve the upcoming complex situations of a project as well as aid to deliver the project successfully. Our approach also emphasize that software companies should use HermiT reasoner, OWL API, JADE and Java technologies in order to develop the proposed system to produce quality estimates and to manage their knowledge regularly. It takes approximately two months to build all the core components for those software companies that are currently developing small projects using Scrum methodology but development time can vary as it depends on the company size, project size and their dynamic needs.

IV. CASE STUDY OF SHOPPING CART PROJECT ESTIMATION

We developed an intelligent system, based on our proposed approach that computes a quality estimate of software effort required for a project. Tools and technologies we used were Java 7, JADE 4.3.3, OWL API 3.5.0 and HermiT Reasoner 1.3.8.3. As a case study, we present shopping cart project estimation produced by our system to successfully complete similar upcoming projects. In the study, scrum master wants to determine the estimate of upcoming shopping cart project that belongs to the category of E-commerce software and comprise of top-2 high priority lessons learnt in the specified category. So, all relevant agents perform inference through E-commerce shopping cart related project knowledge in the Scrum repository of our company as per their defined description logic against the scrum master requirements using HermiT reasoner. Fig. 5 and Fig. 6 depict the extracted knowledge of shopping cart project by the relevant agents in the form of ontology graphs which represent detailed and semantically well-structure knowledge of two distinct sprints. We have assigned colors to the nodes in the graphs to illustrate the inferred knowledge that belongs to a particular agent. Cyan color nodes belong to the project agent which represents sprints knowledge related to the shopping cart project. Similarly, brown nodes, orange nodes and yellow nodes respectively belong to the feature agent, phase agent and lesson agent. Brown nodes represent the features information; orange nodes signify phase as well as team information and yellow nodes indicate lesson information. All the inferred estimated results from each agent are collected and compiled by the estimate agent and presented to the scrum master in the form of a complete estimate. Fig. 7 represents a suitable estimate for this case study that is compiled by the estimate agent for an upcoming shopping cart project based on the current ontology graphs and scrum master requirements.

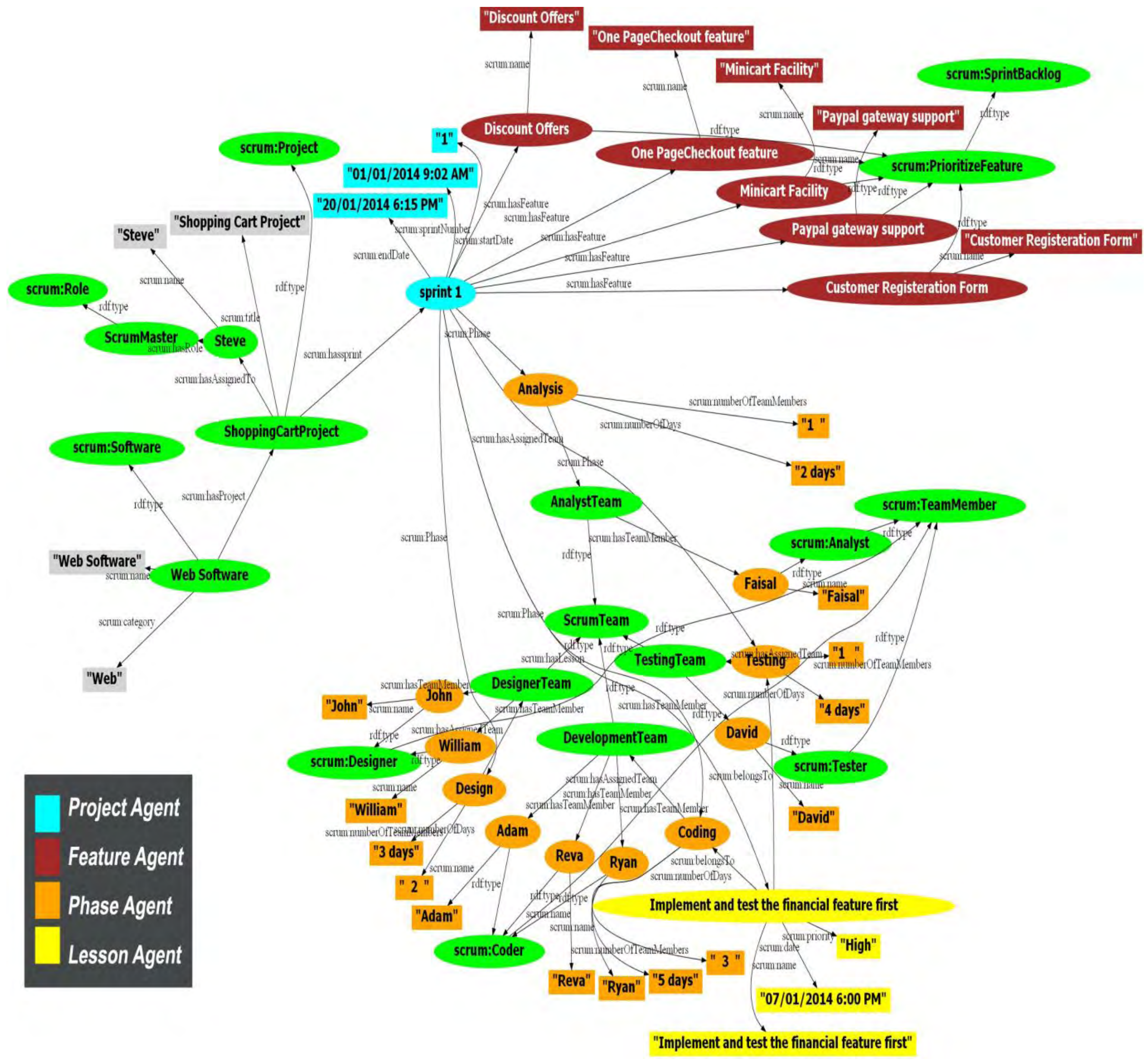


FIGURE 5. Ontology graph representing extracted knowledge of shopping cart project for Sprint1 in relevance to the multiagent.

V. EXPERIMENT AND RESULTS

An experiment was conducted to validate our approach by choosing a team comprising seven professionals consisting of one analyst, three developers, two designers and one quality engineer having more than two years of experience from a software house named as XcellentSoft that is currently practicing Scrum methodology. The experiment was based on twelve web projects related to e-commerce domain that were newly developed by this team. The said team was directed to consider the proposed approach estimations in order to develop their assigned web projects using PHP and MySQL technologies. Developed projects are labeled as P1 to P12 and containing the product backlog size of twenty five to thirty five features. In the product backlog planning activity of each e-commerce project, we had guided the said team to estimate their efforts in person-days unit by using our proposed approach and two other techniques, namely, delphi and planning poker. First of all, in the case of delphi estimation technique, we chose 3 to 4 most experienced individuals from the said team as experts and collected the required average estimates for the assigned e-commerce projects. Secondly, planner poker method was conducted by the said team in the presence of all project stakeholders including their respective product owners and carried out census based estimation by choosing card values ranging 0 to 200. Thirdly, proposed approach was applied by the same team to estimate the

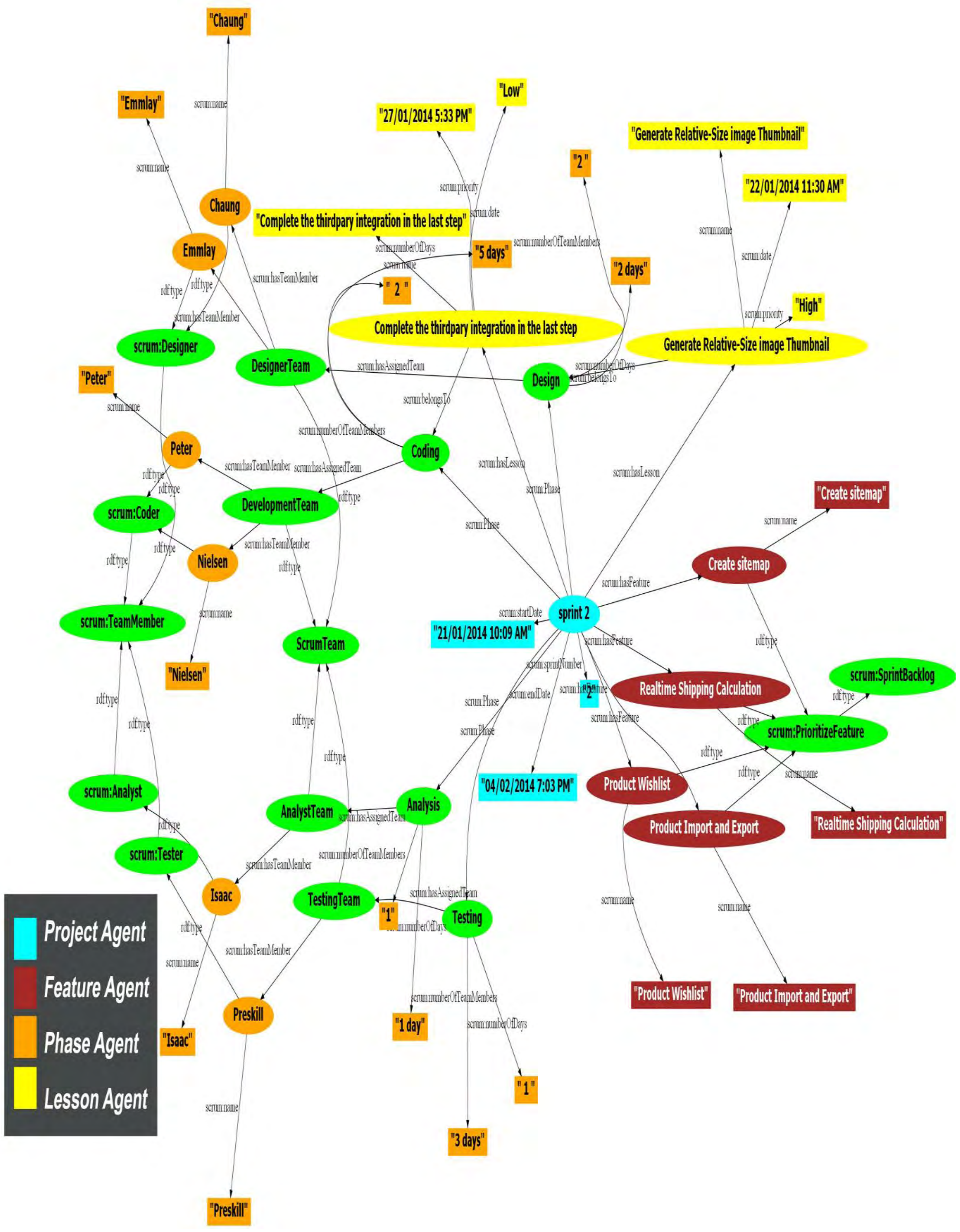
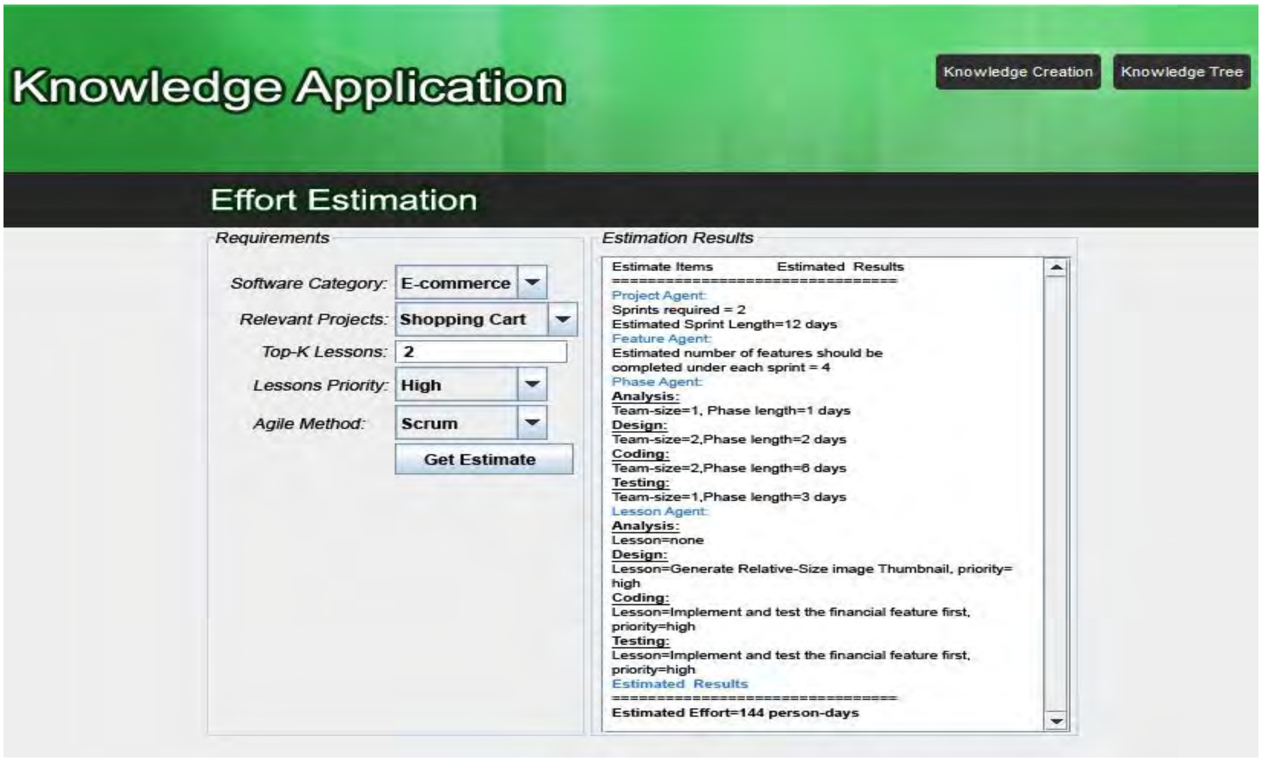


FIGURE 6. Ontology graph representing extracted knowledge of shopping cart project for Sprint2 in relevance to the multiagent.



Requirements

Software Category: E-commerce
Relevant Projects: Shopping Cart
Top-K Lessons: 2
Lessons Priority: High
Agile Method: Scrum

Estimation Results

```
Estimate Items          Estimated  Results
=================================
Project Agent:
Sprints required = 2
Estimated Sprint Length=12 days
Feature Agent:
Estimated number of features should be
completed under each sprint = 4
Phase Agent:
Analysis:
Team-size=1, Phase length=1 days
Design:
Team-size=2,Phase length=2 days
Coding:
Team-size=2,Phase length=6 days
Testing:
Team-size=1,Phase length=3 days
Lesson Agent:
Analysis:
Lesson=none
Design:
Lesson=Generate Relative-Size image Thumbnail, priority=
high
Coding:
Lesson=Implement and test the financial feature first,
priority=high
Testing:
Lesson=Implement and test the financial feature first,
priority=high
Estimated  Results
=================================
Estimated Effort=144 person-days
```

FIGURE 7. Screenshot showing input of scrum master to our implemented multiagent effort estimation system and its reliable estimate in response.

TABLE 1. Estimated efforts of twelve web projects by using delphi, planning poker and proposed approach along with their obtained actual effort(s) and compiled magnitude of relative error (MRE) values.

	Estimated Effort (person-days)				Magnitude of Relative Error (MRE)		
Project ID	Proposed Approach	Delphi	Planning Poker	Actual Effort	Proposed Approach	Delphi	Planning Poker
P1	144	96	148	136	0.058	0.294	0.088
P2	160	172	152	155	0.032	0.109	0.019
P3	120	136	128	112	0.071	0.214	0.142
P4	90	88	106	131	0.312	0.328	0.190
P5	126	110	145	122	0.032	0.098	0.188
P6	112	124	107	148	0.243	0.162	0.277
P7	180	160	190	174	0.034	0.080	0.091
P8	140	154	167	150	0.066	0.026	0.113
P9	100	128	120	122	0.180	0.049	0.016
P10	80	100	115	88	0.090	0.136	0.306
P11	132	102	140	139	0.050	0.266	0.007
P12	128	144	135	134	0.044	0.074	0.007

required efforts for all the web projects. We also provided the implemented effort estimation tool based on the proposed approach to the participating Scrum masters to further add up their new project instances into the knowledge-base to easily get the reliable estimates for the assigned projects. The detailed estimated efforts by using aforementioned estimation techniques along with actual efforts in person-days of all assigned web projects were collected and analyzed in the experiment. Table 1 shows comparison of estimated efforts collected by applying said estimation techniques

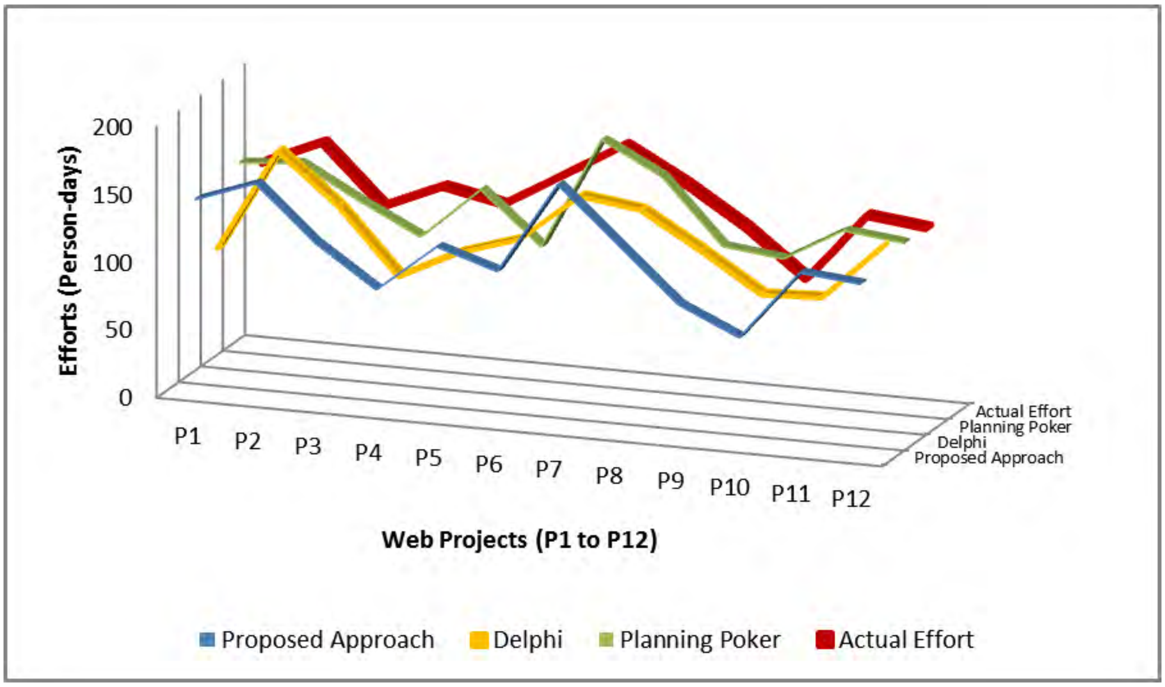


FIGURE 8. Comparison of actual efforts of twelve web projects with the estimated efforts of applied estimation methods that are proposed approach, delphi and planning poker.

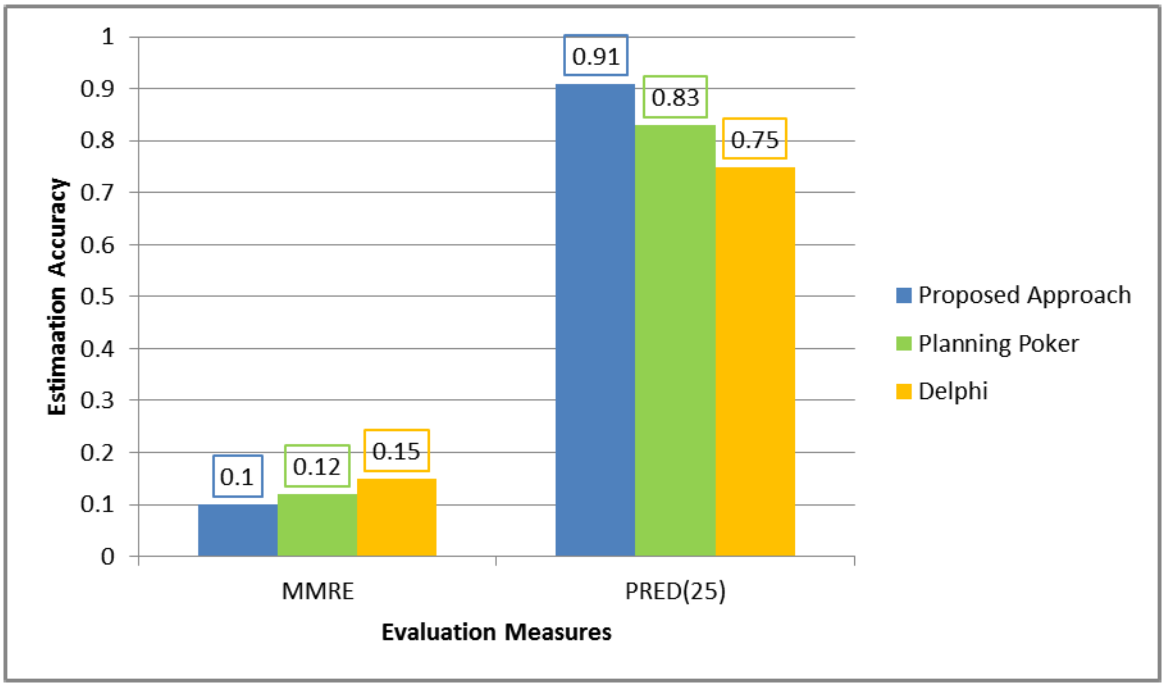


FIGURE 9. Comparison of obtained results after applying MMRE and PRED(25) shows that proposed approach provides very good estimation accuracy as compared to planning poker and delphi techniques.

with the actual efforts. Visual comparison is also depicted in Fig. 8.

In order to analyze and evaluate the results, we have used MMRE (Mean Magnitude of Relative Error) and PRED(x) (prediction at level x) measures. Both measures are defined as: $MRE = \frac{|(AE-EE)|}{AE}$, where terms 'AE' and 'EE' refer to the actual effort and estimated effort, respectively. $PRED(x) = \frac{L}{N}$, 'L' is the number of observations where MRE $\leq$ x and 'N' is total number of observations in the set [29], [30].

The obtained results show that in the case of proposed approach one value of MRE is greater than 0.25 against the project P4, whereas, in the case of planning poker two values

TABLE 2. Obtained results and estimation accuracy of all three estimation techniques by applying evaluation measures.

Methods	MMRE	PRED(25)	Estimation Accuracy
Proposed Approach	0.101	0.91	91%
Planning Poker	0.120	0.83	83%
Delphi	0.153	0.75	75%

of MRE are greater than 0.25 value against the projects P6 and P10. Also, in case of delphi technique three values of MRE are greater than 0.25 against the projects P1, P4 and P11. It is evident that proposed approach produced better prediction accuracy as compared to both planning poker and delphi methods as per the recommendation of PRED (25) measure. It is suggested that a good prediction model must have MRE $\leq$ 0.25 and PRED(25) $\geq$ 0.75. The obtained results are presented in Table 2 and their comparison is show cased in Fig. 9. Moreover, MMRE value of the proposed approach is also less than planning poker and delphi methods (i.e., $0.101 < 0.120$ and $0.101 < 0.153$, respectively). In addition, PRED(25) value of proposed approach is $0.91 > 0.75$ as recommended to be a good prediction model as per the PRED(25) measure. In the same way, 0.91 is greater than 0.83 and 0.75 which are PRED(25) values of planning poker and delphi methods, respectively. It indicates that proposed approach has 91% chance to accurately predict the effort estimations for the assigned web projects. Hence, the experiment results conclude that our proposed approach provides better prediction accuracy and provides more accurate estimates than planning poker as well as delphi methods.

VI. CONCLUSION AND FUTURE WORK

Focusing on Scrum, this paper draws attention on knowledge management for high quality estimation of software effort required for upcoming projects. It presents our multi-agent estimation system which is based on Scrum ontology model and description logic. We have applied MMRE and PRED(25) evaluation measures on the collected estimated efforts and actual efforts for twelve web projects in our experiment. The results show that proposed approach provides very good prediction accuracy. The proposed method produces more accurate estimates than planning poker and delphi methods. In the future, we will enhance our approach by including other flavors of agile methodologies to reliably estimate the required efforts for upcoming projects. These future enhancements will positively use this approach in the global software development context to integrate the distributed agile based project knowledge as well as to mitigate the coordination and communication challenges of geographically distributed development teams.

REFERENCES

[1] S. K. Sehra, J. Kaur, Y. S. Brar, and N. Kaur, "Analysis of data mining techniques for software effort estimation," in *Proc. 11th Int. Conf. Inf. Technol., New Generat. (ITNG)*, Las Vegas, NV, USA, Apr. 2014, pp. 7–9.

[2] H. Rastogi, S. Dhankhar, and M. Kakkar, "A survey on software effort estimation techniques," in *Proc. 5th Int. Conf. Confluence Next Generat. Inf. Technol. Summit*, Sep. 2014, pp. 25–26.

[3] BluePrint Software Systems Inc. (2014). *Blueprint IT Leadership Survey Finds Most IT Projects Delivered Late and Over Budget*. [Online]. Available: http://www.blueprintsys.com/news/blueprint-leadership-survey-findsprojects-delivered-late-budget/

[4] C. J. P. de Lucena, "The emergence of multiagent system software engineering," in *Proc. 25th Brazilian Symp. Softw. Eng. (SBES)*, Sep. 2011, pp. 28–30.

[5] S. Priyadarshini and S. Karthik, "Analysis of agent based system in agile methodology," in *Proc. Int. Conf. Pattern Recognit., Inf. Mobile Eng. (PRIME)*, Feb. 2013, pp. 21–22.

[6] H. Kniberg, *Scrum and XP From the Trenches* (InfoQ Enterprise Software Development Series), 2nd ed. Raleigh, NC, USA: C4Media, May 2015.

[7] M. Almseidin, K. Alrfou, N. Alnidami, and A. Tarawneh, "A comparative study of agile methods: XP versus SCRUM," *Int. J. Comput. Sci. Softw. Eng.*, vol. 4, no. 5, pp. 126–129, May 2015.

[8] D. Toka and O. Turetken, "Accuracy of contemporary parametric software estimation models: A comparative analysis," in *Proc. Euro Micro Conf. Ser. Softw. Eng. Adv. Appl.*, vol. 39. Sep. 2013, pp. 313–316.

[9] D. Manikavelan and R. Ponnusamy, "Software cost estimation by analogy using feed forward neural network," in *Proc. Int. Conf. Inf. Commun. Embedded Syst. (ICICES)*, Feb. 2014, pp. 27–28.

[10] V. Mahnic and T. Hovelja, "On using planning poker for estimating user stories," *Elsevier J. Syst. Softw.*, vol. 85, no. 9, pp. 2086–2095, Sep. 2012.

[11] J. López-Martínez, R. Juárez-Ramírez, A. Ramírez-Noriega, G. Licea, and R. Navarro-Almanza, "Estimating user stories' complexity and importance in scrum with Bayesian networks," in *Recent Advances in Information Systems and Technologies*, vol. 569. Cham, Switzerland: Springer, Mar. 2017.

[12] A. Rai, G. P. Gupta, and P. Kumar, "Estimation of software development efforts using improved delphi technique: A novel approach," *Int. J. Appl. Eng. Res.*, vol. 12, no. 12, pp. 3228–3236, 2017.

[13] M. Usman and R. Britto, "Effort estimation in co-located and globally distributed agile software development: A comparative study," in *Proc. IEEE Softw. Meas. Int. Conf. Softw. Process Product Meas. (IWSM-MENSURA)*, Oct. 2016, pp. 219–224.

[14] M. Jorgensen, "Unit effects in software project effort estimation: Workhours gives lower effort estimates than workdays," *Elsevier J. Syst. Softw.*, vol. 117, pp. 274–281, Jul. 2016.

[15] B. Tanveer, L. Guzmán, and U. M. Engel, "Effort estimation in agile software development: Case study and improvement framework," *J. Softw. Evol. Process*, vol. 29, no. 11, Nov. 2017, doi: 10.1002/smr.1862.

[16] C. Nagar and A. Shah, "Analogy-based estimation with three point technique," *Int. J. Modern Trends Sci. Technol.*, vol. 3, no. 2, p. 1, Feb. 2017.

[17] S. Dragicevica, S. Celarb, and M. Turicc, "Bayesian network model for task effort estimation in agile software development," *Elsevier J. Syst. Softw.*, vol. 127, pp. 109–119, May 2017.

[18] O. Akbari, "A survey of agent-oriented software engineering paradigm: Towards its industrial acceptance," *J. Comput. Eng. Res.*, vol. 1, pp. 14–28, Apr. 2010.

[19] M. Uschold and R. Jasper, "A Framework for Understanding and Classifying Ontology Applications," in *Proc. 12th Int. Workshop Knowl. Acquisition, Modelling, Manag. (KAW)*, vol. 99. 1999, pp. 16–21.

[20] M. Uschold and M. Grninger, "Ontologies: Principles, methods and applications," *Knowl. Eng. Rev.*, vol. 11, no. 2, pp. 93–136, 1996.

[21] A. Freitas and R. Vieira, "An ontology for guiding performance testing," in *Proc. IEEE/WIC/ACM Int. Joint Conf. Web Intell. (WI) Intell. Agent Technol. (IAT)*, Aug. 2014, pp. 11–14.

[22] R. G. C. Rocha, R. Azevedo, and S. Meira, "A proposal of an ontology-based system for distributed teams," in *Proc. 40th EUROMICRO Conf. Softw. Eng. Adv. Appl. (SEAA)*, Aug. 2014, pp. 27–29.
[23] S. Feldmann, S. Rosch, C. Legat, and B. Vogel-Heuser, "Keeping requirements and test cases consistent: Towards an ontology-based approach," in *Proc. 12th IEEE Int. Conf. Ind. Informat. (INDIN)*, Jul. 2014, pp. 27–30.
[24] Q. Wu, F. Ishikawa, Q. Zhu, and D.-H. Shin, "QoS-aware multigranularity service composition: Modeling and optimization," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 46, no. 11, pp. 1565–1577, Nov. 2016.
[25] G. L. Simmons and T. S. Dillon, "Towards an ontology for open source software development," in *Proc. 2nd Int. Conf. Open Sour. Syst.*, 2006, pp. 65–75.
[26] C. González-Pérez and B. Henderson-Sellers, "An ontology for software development methodologies and endeavours," in *Ontologies for Software Engineering and Software Technology*. Berlin, Germany: Springer, Jul. 2006, pp. 123–151.
[27] S. Sharma and M. Ingle, "Object Oriented versus Ontology Oriented software reliability development," in *Proc. 6th Int. Conf. Softw. Eng. (CONSEG)*, Sep. 2012, pp. 5–7.
[28] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau, *Extensible Markup Language (XML) 1.0*, 5th ed. [Online]. Available: https://www.w3.org/TR/xml/
[29] F. Ferrucci, C. Gravino, R. Oliveto, F. Sarro, and E. Mendes, "Investigating tabu search for Web effort estimation," in *Proc. IEEE 36th EUROMICRO Conf. Softw. Eng. Adv. Appl. (SEAA)*, Sep. 2010, pp. 350–357.
[30] S. Abraho, J. Gmez, and E. Insfran, "Validating a size measure for effort estimation in model-driven Web development," *Elsevier J. Inf. Sci.*, vol. 180, no. 20, pp. 3932–3954, Oct. 2010.

MUHAMMAD ADNAN received the M.S. degree in computer science from the National University of Computer and Emerging Sciences, Lahore, in 2008. He is currently pursuing the Ph.D. degree in computer science with the University of Engineering and Technology, Lahore. His research interest includes software design and development, programming languages, software engineering, web engineering, semantic web, and database systems.

MUHAMMAD AFZAL received the Ph.D. degree in computer science from the University of Engineering and Technology, Lahore, in 2012. He is currently an Associate Professor with the Department of Computer Science and Engineering, University of Engineering and Technology. His research interest includes natural language processing, algorithm analysis and design, speech processing, image processing, data structures, object oriented programming, artificial intelligence, and software engineering.