Advances in Science, Technology and Engineering Systems Journal
Vol. 3, No. 3, 80-91 (2018)
www.astesj.com
Special Issue on Multidisciplinary Sciences and Engineering

ASTES Journal
ISSN: 2415-6698

Towards Process Standardization for Requirements Analysis of Agent-Based Systems

Khaled Slhoub*, Marco Carvalho

Florida Institute of Technology, School of Computing, Florida, 32901, United States

ARTICLE INFO

Article history:
Received: 18 April, 2018
Accepted: 11 May, 2018
Online: 31 May, 2018

Keywords:
IEEE Std 830
Agent-Oriented Specification
Agent-Based Systems
Multi-Agent Systems
MAS
ABS
Software Engineering
Software Requirements Specification
SRS
Software Standards
Software Quality

ABSTRACT

The development of agent-based systems is negatively impacted by the lack of process standardization across the major development phases, such as the requirements analysis phase. This issue creates a key barrier for agent technology stockholders regarding comprehending and analyzing complexity associated with these systems specifications. Instead, such fundamental low-level infrastructure is loosely attended to in an ad-hoc fashion, and important aspects of requirements analysis are often neglected altogether. The IEEE Std 830 model is a recommended practice aimed at describing how to write better quality requirement specifications of conventional software. Knowing that agent-based computing is a natural and logical evolution of the conventional approaches to software development, we believe that the requirements phase in agent-based systems can benefit from applying the IEEE Std 830 model which results in high-quality and more accepted artifacts. This article provides a criteria-based evaluation that is derived from the software engineering body of knowledge guide to assessing the adoption degree of agent-oriented methodologies to software requirements standards. Then, it proposes a model-driven approach to restructuring and extending the IEEE Std 830-2009 standard model to specify requirements of agent-based systems. To evaluate the applicability and usefulness of the updated model, we design a research study that allows practicing the model with simple real-world problem scenarios and conducting a summative survey to solicit feedback on the model usages.

1 Introduction

This article is an extension of work initially presented in 2017 at the IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON) [1].

Agent-based systems (ABS) have proliferated in recent years into what is now one of the most active research areas in computing. This intensity of interest is increased not only because these complex systems impose a new promising means of developing software, but also because the agent community has become aware of the necessity to cast ABS as a software engineering paradigm (SE). This becomes important in order to assist ABS development to be more formal and efficient. Agent-Oriented Software Engineering (AOSE) is a new independent SE mainstream that aims to either extend or adapt existing SE methodologies to facilitate and improve the ABS complex development. SE practices have become a vital prerequisite of running successful software products and have been extensively used for decades to support the conventional ways of building software, such as object-oriented development which is currently the most popular programming paradigm [2].

Like other conventional software, software requirements specification (SRS) is an essential aspect in agent-based systems, and numerous agent-oriented methodologies demand it as an initial primary phase in their development process life-cycles. SRS describes how an agent system is expected to behave and extends the re-

*Corresponding Author: Khaled Slhoub, Email: kslhoub2014@my.fit.edu

https://dx.doi.org/10.25046/aj030311

quirements analysis phase to in-depth detail, detailed blueprints, used extensively by different stakeholders of agent systems. However, the complexity associated with ABS development and its target application domains often drives the complexity of ABS requirements phase. This results in poor quality SRS artifacts that have a genuine and significant negative impact on the entire ABS development life-cycle and create further technical problems to the stakeholders. With the emergence of industry willingness for agent systems, there is a real demand to develop a high-quality agent-oriented specification in such a way that makes ABS development easier, more formal, more disciplined, and more accepted industry-wide.

Over the past three decades, software standards have played a key role in improving quality of conventional software development. For instance, SE practices like the UML modeling language and IEEE/ISO standards have been widely used in the software industry to formalize the entire software development life-cycle. More specifically, the International Recommended Practice for Software Requirements Specifications model (IEEE Std 830-2009) has been commonly employed to help developers toward constructing high-quality and well-organized SRS artifacts [3]. Agent technology as another approach for software development can undoubtedly benefit from the IEEE Std 830-2009 model to standardize the SRS process which will mitigate the complexity of the requirements analysis and, as a result, produce more accepted SRS artifact. To do so, the IEEE Std 830-2009 model needs to be reconstructed and updated with new additional extensions to make it more suitable to handle the natural complexity of ABS and their target application domains.

In the original work, the intent was to restructure the IEEE Std 830-2009 model in such a way that makes it more suitable to handle ABS requirement specifications. We first explored the requirement phases in different well-known AOSE methodologies, described in [4], [5], and hence we identified five data models which need to be specified and incorporated into the ABS analysis process. Then, we checked the suitability of these data models according to the original IEEE Std 830-2009 sections and proposed rewriting some subsections, extending others, and adding more new extensions to the IEEE Std 830-2009 model to complement the process standardization in the ABS requirement phase. In this extended article, we explain in detail evaluation criteria used to determine the coverage degree of the agent-oriented methodologies with respect to the requirement standards. Also, we conduct an external evaluation study by asking participants to go through the updated model sections using a few simple real-world problem scenarios, and then take an online survey to provide feedback on their experience using the updated model.

The rest of this paper is organized as follows: section 2 gives the necessary overview of software agent concepts and goals, the agent-oriented specification process with some related work, and the structure of the original IEEE Std 830-2009 standard model along with some related work. Section 3 presents an evaluation matrix used to asses the coverage degree of agent-oriented methodologies with respect to requirement standards. Section 4 describes our proposed approach to restructuring the IEEE Std 830-2009 standard model in such a way that makes it more capable to address agent-oriented specification. Section 5 summarizes our evaluation method and results for the updated IEEE Std 830-200 model. 6 concludes.

2 Background and Related Work

2.1 Agent-Based Systems: What and Why

Over the past two decades, computing systems have become more complex. There are several reasons behind such complexity. First, numerous computing systems are now becoming concurrent and distributed. They are often configured and distributed over multiple locations with the use of multiple networks, tens of servers, and hundreds of different machines. Such systems also comprise a considerable number of software applications that extensively communicate with one another. Second, many computing systems operate in dynamic environments which are flexible enough to permit unpredictable changes to their contents and settings. Third, many computing systems now need to be continuously active and to provide services on an on-going basis [6]. Indeed, the explosive growth of the web and mobile technologies present a twofold problem in terms of the vast availability, wide variety, and heterogeneity of datasets [7]. Moreover, there has been a natural tendency to rely more on technology to resolve certain classes of complex real-world problems. People want computer systems to do more for them and, if possible, substitute them for performing costly and complex tasks. More than this, consumers demand systems to operate without their intervention and take the initiative when necessary. To do so, people build software components, known as agents, to act on behalf of users to achieve specific delegated goals. The intent is to develop sophisticated agents that are "capable of human-like reasoning and acting" [8], [9].

As a result of the above demands, new software characteristics and expectations have arisen and made current software development paradigms struggling to handle such complex development. These demands have also triggered a need to focus more on developing complex, real-time and distributed intelligent systems. Thus, this has prompted researchers to look for some effective alternative approaches to address the new demands of software development. Currently, agent-based computing is becoming the most promising development approach for handling such issues [5], [10], [11], [12], [13].

A software agent, another popular term used interchangeably with an agent in the agent community, could be defined in several different ways depending on its usage and dimensions, we came across at least 12 formal definitions for the software agent. To more clearly understand the software agent, we can think of it as a role-playing smart or active software object that can interact with other agents object despite the fact that "agents are not simply objects by another name" [14]. Wooldridge highlights that "agents are simply software components that must be designed and implemented in much the same way that other software components are. However, AI techniques are often the most appropriate way of building agents" [14]. Jennings also supports this by stating that "an agent is an encapsulated computer system that is situated in some environment, and that is capable of flexible, autonomous action in that environment in order to meet its designed objectives" [15]. In fact, researchers consider agent-oriented development as a natural and logical evolution of the current software development paradigms like structured and object-orientated approaches as it presents a higher-level of abstraction and encapsulation [11], [16].

Actually, agent orientation represents not only a higher-level of abstraction and encapsulation that is more flexible than the other prior programming paradigms but also offers some unique dimensions or characteristics of agenthood such as **autonomy**, capability of deciding for itself how best to achieve delegated goals; **intelligence**, capacity of learning, reasoning and making educated decisions; and **socialability**, capability of interacting and coordinating with other agents. Such characteristics allow software agents to simulate human-like reasoning capability and to act rationally and flexibly to attain delegated goals. Yu highlights that "Agent orientation is emerging as a powerful new paradigm for computing" [17]. Jennings also states that "agent-oriented approaches can significantly enhance our ability to model, design and build complex, distributed software systems" [15].

Despite the success of several single-agent systems, a software agent as a computational entity rarely functions in isolation [18]. Any individual agent is primarily restricted by the limitation of its knowledge, resources, and potential which make it often useless on its own [15], [19]. With the current complex real-world problems, it is necessary to develop capable, intelligent distributed systems that employ a group of individual agents to resolve a combination of issues. Multi-agent systems (MAS) are constructed around this concept of embedding groups of agents that collaborate with one another, and, to do so, these individual agents require the ability to cooperate, coordinate, and negotiate with each other, much as people do. One broad definition for MAS is "a collection of interacting agents which are highly autonomous, situated and interactive software components. They autonomously sense their environment and respond accordingly. Coordination and cooperation between agents that possess diverse knowledge and capabilities facilitate the achievement of global goals that cannot be otherwise achieved by a single agent working in isolation" [20]. As shown in Figure 1, software agents in MAS are organized into teams or groups, each side with specific tasks to perform. The agent in its squad strategically partners with other agents and efficiently cooperates, coordinates, negotiates, and competes to fulfill the best outcome for itself and its team. Such collective interactions among the groups result in pursuing the entire delegated system goals which are beyond the limits of a single agent to accomplish [15]. In a study carried out in 2014 [21], Muller and Fischer investigated and analyzed 152 agent applications all over the world created by parties from 21 countries. They were able to demonstrate that MAS have attracted much more attention than other agent type applications (125 MAS applications corresponding to 82 % of all agent applications in the study).

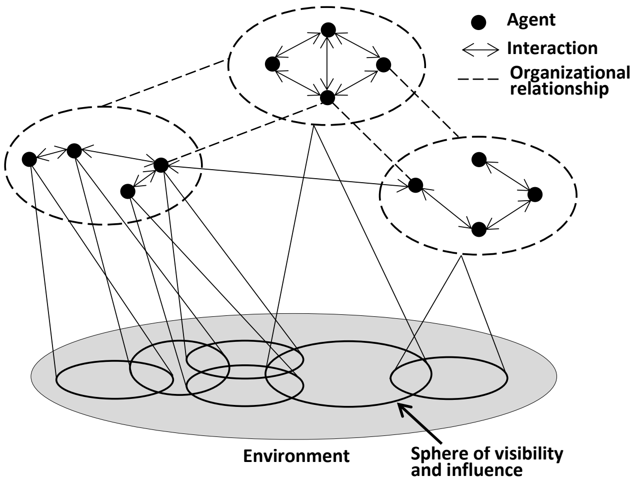
Agent
Interaction
Organizational relationship
Environment
Sphere of visibility and influence

Figure 1: High-level structure of agent-based systems

2.2 The Requirements Analysis Process in Agent-Based Systems

The requirements analysis process is a significant development phase in large and complex conventional software. It is a description that leads to an understanding of the system and its structure prior to implementation, provides the foundation for the rest of development phases like design and testing, and acts as a reference point for the system stakeholders. ABS, which are structured based on communicating software agents, are not an exception as various existing agent-oriented development methodologies identify the requirements analysis phase as the initial phase that developers of agent systems need to fulfill. However, unlike conventional software, the requirements phase in agent systems requires to cover more additional complex details of software agents, their roles, their dynamics interactions, their cooperation patterns, and internal behaviors combined with other information of their surrounded physical/virtual and operating environments, and the target application domain.

To deal with such complexity, Ferber et al. describe the ABS requirements from the organizational point of view by using the notions of agent, group, and role models. Based on this structure, requirements are classified into four types: single role behavior requirements, intragroup interaction requirements, intragroup communication successfulness requirements, and intergroup interaction requirements [22]. Similarly, Hilaire et al. aim to analyze complex agent communications by proposing an organizational model centered around agent interaction patterns that are composed of roles [23]. Instead, Herlea et al. introduce three levels of abstraction for the ABS requirement: requirements for the multi-agent system as a whole, requirements for an agent within the system, and requirements for a component within an agent [24]. Miller et al. indicate the need for adequate elicitation methods of agent-oriented requirements to mitigate complexity. They propose an elicitation approach that provides a list of questions to be answered by stakeholders and then map the answers to the ABS requirements [25].

Most of the existing agent-oriented development methodologies are derived originally from some software engineering processes models used in supporting the development of conventional software such as waterfall, evolutionary, and RUP [26], [27]. Such process models give significant attention to the requirements engineering phase to delineate the complexity issue and avoid situations like poor time and budget estimations, unpredictable outcomes, and customer dissatisfaction [28]. To do so, the phase is often supported by sets of standards, diagrammatic notations, and various CASE tools. Similarly, agent-based systems are basically software components that need to be developed in much the same ways of developing conventional software [15], [16]. Thus, these systems can also benefit from the existing SE practice in improving the requirements analysis phase. One essential improvement, for instance, is to apply different types of software documentation standards to increase the quality of software development. IEEE/ISO/IEC standards have been widely used to support conventional software development. For example, the IEEE Std 830 documentation model is widely used to support software requirements specification and can also play a key role not only in improving and formalizing the requirement analysis process in agent systems but also in giving a better signal to the industry to better understand the ABS concepts and realize their benefits.

2.3 The IEEE Std 830-2009 Standard Practice Model

Very little work has been reported in the literature to standardize and formalize agent-based systems development. For example, Foundation for Intelligent Physical Agents (FIPA) [29], accepted as a standard committee of the IEEE Computer Society in 2005, is proposed to facilitate agent aspects like agent platform, protocols, and communication languages. However, FIPA standard templates are large, complex, and hard to follow. Such templates are considered more appropriate to address the agent systems design phase but not the requirements phase [29], [30]. Agent modeling languages are another example proposed by extending the UML metamodel to meet the structural aspects of agent systems like TAO [31], AML [32] and Agent UML [33]. Such modeling languages are, however, restricted by object-oriented concepts and cannot tackle the required additional complex details of software agents mentioned in section 2.2.

Meanwhile, the International Recommended Practice for Software Requirements Specifications model (IEEE Std 830-2009) demonstrates a simple generic model to guide developers in the requirements analysis process [3]. IEEE Std 830 was specifically developed to standardize the process of software requirements specifications (SRS) in conventional software, also known as the high-level design of the system, and to drive developers towards producing high-quality and well-organized SRS artifacts. In addition, it aims to establish the basis for agreement between customers and suppliers of as to what the final software product should deliver; reduce the development time and effort; estimate efficiently project cost and scheduling; facilitate the smooth transactions between projects stages; ensure continuity of work; enhance software maintenance and reuse; and provide the baseline for testing [3]. The IEEE Std 830 model frequently serves as a reference for developers during software development processes and as a contract between project customers and suppliers. In fact, it has been widely used in software industry especially in supporting the object-oriented development, and it is recognized according to IEEE as the most popular and universal standard among all other IEEE standards with 50,925 full-text views in 20 years [3], [34]. As shown in Figure 2, the IEEE Std 830 model is divided into four main sections composed of subsections that support achieving specific objectives. Refer to [3] for more information.

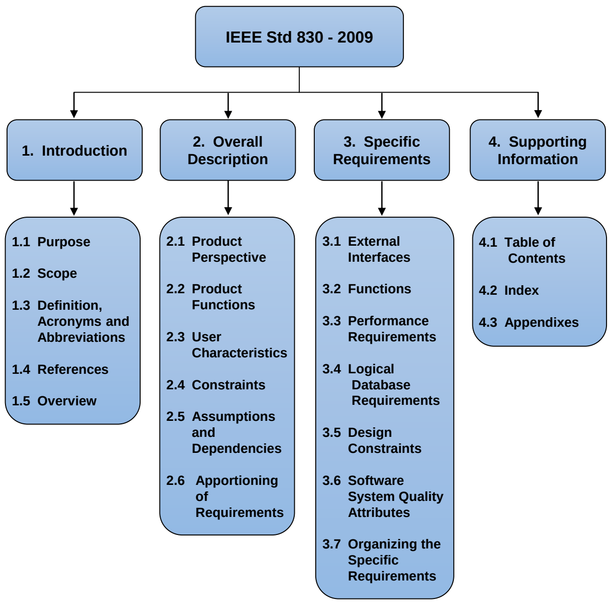

Figure 2: The IEEE Std 830-2009 model structure

3 Evaluating the Adoption of ABS Requirements to Software Standards

ABS development methodologies pay a great deal of attention to the requirements analysis process and identify it as an initial phase that ABS developers need to fulfill. We deeply explored the requirement phases in a large number of agent-oriented methodologies searching for evidence of implementing sound software standards. We wanted to determine how well the requirement phases adopt and practice software standards. To do so, we provided a simple criteria-based evaluation matrix that is basically derived from the software engineering body of knowledge (SWEBOK-V3.0) guide. SWEBOK-V3.0 is a well-defined, structured set of SE attributes used to deliver one of the essential benchmarks underlying the software engineering baseline [35]. Then, we used the evaluation matrix as a checklist to assess the coverage degree of an agent-oriented methodology with respect to requirement standards. Table 1 illustrates the evaluation matrix checklist combined with a set of agent-oriented development methodologies that were inspected for requirements standardization.

As shown in Table 1, the evaluation matrix was applied to more than 20 methodologies defined in [4], [5]. These methodologies were selected in light of the ongoing flow of publications, the remarkable impact on the AOSE community, the ease of understanding, and the profusion of supported guidance and tools. Despite the fact that all the inspected methodologies describe requirements analysis in sufficient detail, we found that the standards practice is often negated or rarely mentioned in these processes. In fact, to the best of our knowledge, no well-structured process standardization has been proposed and incorporated into any requirements analysis phase in the existing ABS development methodologies. We argue that the lack of process standardization in requirements analysis is linked to several AOSE challenges, such as immature ABS development methodologies, the absence of unified ABS methodologies and notation, and weak industrial acceptance of the AOSE paradigm.

4 Using the IEEE Std 830-2009 Model to Standardize Agent-Oriented Specification

Agent-based systems can be seen as a natural evolution from the object orientation paradigm in that they offer a higher-level of abstraction and encapsulation [11], [15], [16]. ABS are basically software components combined with additional dimensions or characteristics like autonomy, intelligence, and socialability that make ABS more flexible and suitable to address certain classes of complex real-world problems. ABS need to be developed in much the same ways as other conventional software components are developed. Thus, adopting the IEEE Std 830-2009 model, used with conventional software, in the requirement analysis phase of ABS development could become a core motif in promoting the entire ABS development quality and also map the way for the industry to accept and recognize the benefits of agent technology. To do so, The IEEE Std 830-2009 model needs to be restructured and extended to cover all agent-oriented specifications.

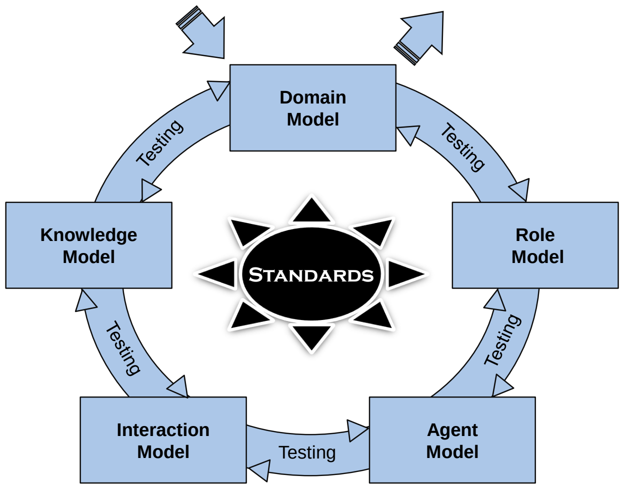

Figure 3: The five data sets required to meet agent-oriented specification [1]

As mentioned in section 2.3, the IEEE Std 830-2009 model was designed to provide recommended approaches for handling the software requirements specification. ABS can benefit from this standard model to

Evaluation Factor	Agent-Oriented Methodology
define clearly a process to elicitation, analysis, specification, and validation of requirements	PASSI, MaSE, Prometheus, Tropos, MAS-CommonKADS, Gaia, ADELFE, MESSAGE, INGENIAS, AOR, SODA, DESIRE, Agent Factory, MADE, AOSM, Agent OPEN, FIPA, ODAM, MASSIVE, Roadmap
monitor the quality and improvement of the requirements process like using quality standards and metrics	
establish software and system requirements documents	
use quality indicators to improve SRS	
provide an information basis for transferring work and software enhancement	
review the deviation from standard practice	

Table 1: The evaluation matrix used to determine the adoption degree of ABS requirements to standards

shape and formalize the requirements analysis process in such a way that results in more accepted specification document. Thus, we attempt in this section to specify ABS requirements by means of utilizing the IEEE Std 830-2009 model and proposing several new extensions to it. Accordingly, we first identified the main data models in the SRS process of ABS by investigating the analysis processes in several different AOSE development methodologies such as PASSI [36], ADELFE [37], MaSE [38], Gaia [39], and Tropos [40]. Then, we checked the fitness of these data models according to the IEEE Std 830-2009 model. Consequently, we were able to identify the data models that are not covered by the original IEEE Std 830-2009 model and proposed rewriting some subsections, extending others, and adding more new extensions to the model that can facilitate the agent-oriented specification. Figure 3 illustrates the identified 5 data models that carry the required information for specifying the ABS requirements. The 5 data models are described in the following list:

1. **The Domain model** is concerned with the analysis of a specific real world that an agent-based system is intended to work in and interact with. This model consists of three types of information:
   - *Operating Environment* includes descriptions of the general characteristics of the operating environment of ABS. Such environment provides the necessary infrastructure in which agents deployed and live. It also manages these agents while performing their tasks and delivers the most current information about them. Moreover, it is responsible for providing the structure of the agent communication language and the agent semantic language.
   - *Physical/Virtual Environment* includes descriptions of the world that software agents perceive and interact with. ABS are often executed in challenging environments that are dynamic, unpredictable, and unreliable. Such environments could be virtual like the web world or physical like a self-driving car and a train control system.
   - *Application Domain* includes information about the field in which the system operates. Such information may involve certain domain terminology or reference to domain concepts and policies.
2. **The Role model** is concerned with describing the ABS requirements in terms of agent tasks or services. Agents are the core actors that are responsible for performing tasks and achieving delegated goals. The descriptions may include information about the role responsibilities, prerequisites, and constraints.
3. **The Agent model** is concerned with identifying and classifying agent types that are instantiated later at ABS run-time, and it is directly linked to the Role model. The Agent model should also include descriptions of characteristics for every agent type which may include at least descriptions of the essential characteristics of agents such as autonomy, reactiveness, proactiveness, and socialability [15], [16].
4. **The Interaction model** is concerned with describing the agents behavior in terms of cooperating, collaborating, negotiating with users, system resources, and other agents. It should also specify the agents messaging protocols and interaction patterns along with the use of ontologies or conceptual means for describing the message contents. For instance, an agent can request another agent to perform some action, and the other agent may refuse or accept the request. It may also confirm to other agents that given proposition is true.
5. **The Knowledge model** is concerned with describing a repository of knowledge that agents may use to provide explanations, recovery information, or optimize their performance. For instance, some agent types, like reflex agents and learning agents, require some particular rules in order perform actions, so such rules need to be specified and stored in the system. The Knowledge model, also, should briefly describe agent architectures which underlying the concepts for building rational agents and their characteristics. Such information is important to design any agent-based system later. For instance, Belief-Desires-Intentions (BDI) architecture is probably the best-known agent architecture that agents rely on to reason about their actions [41].

After identifying the five main data models for agent-oriented specification, we carefully reviewed the four main sections of the original IEEE Std 830-2009 model to check the suitability of the 5 data models according to these sections. As a result, we were able to propose adding more new subsections, rewriting others, and extending other subsections to formulate the ABS requirements specification. Figure 4 illustrates the updated structure of the IEEE Std 830-2009 model by highlighting the new and edited subsections. Also, full descriptions are provided for all new information added to the updated model as follows:

1. The first section **"Introduction"** contains subsections that provide an overview of the entire SRS document, so we consider this information suitable for the five data models of ABS [3]. We only suggest the following change in the following subsection:
   - An existing subsection (1.5 Overview): this subsection should provide a brief description of the contents of the rest of SRS document, so it should also include a description

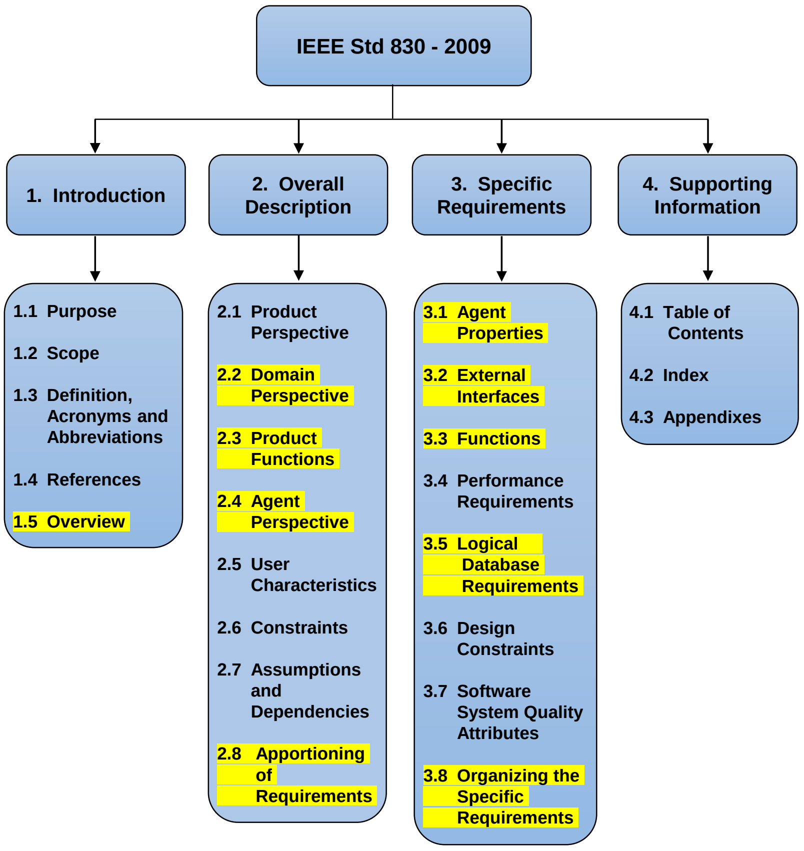

Figure 4: The updated structure of the IEEE Std 830-2009 model [1]

of how analyses of the five ABS data models are organized and presented in the SRS model.

2. The second section **"Overall Description"** contains subsections that describe the general factors that could affect the product and its requirements specifications [3]. We suggest the following changes to this section:

   - A new defined subsection (2.2 Domain Perspective): based on the provided description of this section, the *Domain model* should be described after (2.1 Product Perspective) for an organizational purpose. The domain application often has a great influence on ABS, ABS are considered as environment-driven business development. So, the potential features of the three sections of the *Domain model* are described in this subsection.
   - An existing subsection (2.2 Product Functions): this subsection is renumbered to subsection (2.3), and it is described in terms of both: the *Role model* and the user functions. Thus, it should provide a summary of the key roles that agents in ABS will perform. Notice that this only include a high-level summary of the *Role model* that defines the major functional roles for each listed agent.
   - A new defined subsection (2.4 Agent Perspective): based on the provided description of this section, the *Agent model* should be summarized in section 2. It should include a description that lists and defines the potential software agent categories and their main characteristics.
   - An existing subsection (2.6 Apportioning of Requirements): this subsection is renumbered to subsection (2.8), and it should identify functional agent roles and user functions that may be delayed until future versions of the agent-based system.

3. The third section **"Specific Requirement"** contains subsections that describe in sufficient technical details all ABS requirements. This includes all functional roles concerning all inputs and outputs to/from the agent-based system and all required nonfunctional requirements [3]. We suggest the following changes to this section as follows:

   - A new defined subsection (3.1 Agent Properties): this subsection should extend the subsection (2.4 Agent Perspective). It is concerned with agent properties/ dimensions to a level of sufficient detail that will help ABS designers to construct the internal states for every agent type in ABS.
   - An existing subsection (3.1 External Interfaces): this subsection is renumbered to subsection (3.2). One goal of the third section is

to provide detailed descriptions of software interfaces. So, the *Interaction model* should be described in this subsection in such a way that presents communication mechanisms and messages between all agent types.

- An existing subsection (3.2 Functions): this subsection is renumbered to subsection (3.3). It is concerned with all ABS requirements specifications to a level of detail. This should extend the subsection (2.2 Product Functions) and include sufficient detailed specifications for every functional agent role and user functions. Also, this subsection should extend the subsection (2.4 Agent Perspective) and include sufficient detailed specifications of software agents. It is critical also to explicitly link listed proposed agents to their functional roles. An agent-role list is an effective technique to use here.
- An existing subsection (3.4 Logical Database Requirements): this subsection is renumbered to subsection (3.5). It is concerned with any information that is to be stored in a database. Thus, the *knowledge model* should be described in such a way that shows how some types of agents, like the reflex and learning agents, can use it. Also, the agent architectures should be briefly outlined in this subsection.
- An existing subsection (3.7 Organizing the Specific Requirements): this subsection is renumbered to subsection (3.8). The agent-oriented specification is detailed and tends to be extensive. So, it is important to give special attention to organizing this information to ensure clarity and understandability. This section should include an additional organization which organizing by agent class. Associated with each agent class is a set of agent roles and characteristics.

4. The fourth section **"Supporting Information"** contains subsections that make the SRS document easier to use, for example, using the table of contents, index, and appendixes [3]. We suggest no change to this section and consider this information suitable for the five data models of ABS.

5 Evaluation Analysis

In the original published paper, we conducted two types of evaluation to assess the utility and effectiveness of the proposed restructured IEEE Std 830 model: 1) we specified requirements of a small multi-agent system by using both the new proposed model and the original model, and then we compared the outcomes of both models. 2) we used our proposed model to specify requirements of some previous agent systems that were already specified by using the original model, and we analyzed the outcomes of the different models based on the SRS quality attributes described in the IEEE Std 830-2009 model. Refer to [1] for more information about both assessments.

This article summarizes results from an independent and external research study that was conducted to assess the suitability and validity of the updated IEEE Std 830. The aim of the study was to determine whether the updated model satisfies ABS requirements specification and to solicit feedback on any possible missing details or weaknesses in the model structure. Several experts in software engineering and agent-oriented systems were invited to participate in the research study. They were asked to go through the proposed model sections with a few simple real-world problem scenarios (requirements specification of a simple small agent-based system: *Item-Trading Agent-Based System (ITABS)* that enables users to create and employ agents to act on behalf of them to trade items). Then, the participants conduct an online survey to provide feedback on their experience using the proposed model in specifying ITABS. The survey consisted of 12 statements as shown in Table 2. These statements were organized into 3 sets to determine participants' acceptance of the proposed model, their satisfaction with using the various added and updated agent information to the model, and the quality of the proposed model. The participants were asked to indicate their level of agreement or disagreement with each statement. We were able to collect and analyze 13 responses from the participants as shown in the following charts.

In Figure 5, our concern was focused on whether or not the participants think that the proposed standards model is capable of specifying ABS requirements. The participants were all in favor of the model as no one disagreed with the statements S1 and S2.

In Figure 6, the participants provided their feedback about the added and updated agent information to the original IEEE Std 830 model. We believe that the existence of such information makes the IEEE Std 830-2009 model more suitable to handle the agent-oriented specification. The positive responses actually confirmed our belief as all participants agreed with the statements S3 and S4, and only 4 persons disagreed with the statement S5. The proposed model is required to briefly outline agent architectures which support the concepts for building rational agents and their characteristics. This information is essential to start any ABS design later, but it seems that some participants disagree with referring to implementation details, like referring to agent architectures in the requirement phase.

In Figure 7, the participants gave their opinions about the quality of the proposed model after using it in specifying small agent-based system requirements. They rated the model based on the IEEE seven quality attributes of good software requirements, described in the original IEEE Std 830 model, by indicating how well they feel that the requirements meet the quality

S1	The updated IEEE 830 model helps the developers to identify an agent-based system to be developed along with its requirements.
S2	The updated IEEE 830 model describes what the agent-based system will do and how it will be expected to perform.
S3	Information regarding domain perspective which includes application domain, operating environment, and physical/virtual environment needs to be specified in the IEEE 830 model and will be used to build an agent-based system.
S4	Information regarding agent classification, characteristics, functions, and communications needs to be specified in the IEEE 830 model and will be used to build an agent-based system.
S5	Information regarding agent architectures and agent repository of knowledge, if there's any, needs to be briefly outlined in the IEEE 830 model and will be used to build an agent-based system.
S6	The functional requirements described in the model were **unambiguous**: there was only one interpretation for every stated requirement.
S7	The functional requirements described in the model were **complete**: all significant requirements concerning function, performance, interfaces, design constraints, and quality factors were included.
S8	The functional requirements described in the model were **consistent**: no subset of individual requirements characteristics, actions, or terms conflicted.
S9	The functional requirements described in the model were **verifiable** (Testable): requirements were stated in concrete terms that can be verified through testing or other objective means.
S10	The functional requirements described in the model were **modifiable**: the specification was structured and annotated so that changes may be made easily, completely, and consistently.
S11	The functional requirements described in the model were **traceable**: the origin of each requirement was clear and the format should aid in tracing forward and backward.
S12	The functional requirements described in the model were **usable**: the specifications should be useful during development and in later identifying maintenance requirements and constraints.

Table 2: The online survey statements given to the participants to scale

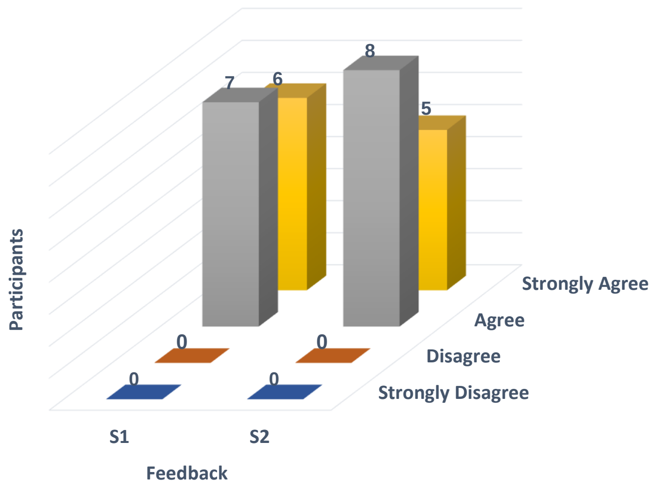

Figure 5: The degree of acceptance of the updated IEEE Std 830-2009 model

attributes on a scale from Strongly Agree to Strongly Disagree. Generally, the responses were positive with a range from 62% to 93% of the participants agreed or strongly agreed with each quality attribute applied to the updated model.

Figure 8 represents a straightforward view that only illustrates the summary of users' views who agreed with each quality attributes applied to the proposed standard model.

6 Conclusion

A strong demand desires to apply agent-based systems in complex and large real-world applications because of their capability to act rationally and flexibly to attain delegated goals. ABS development requires the inclusion of sound software engineering practices in order to be more formal, efficient, and adopted in the industry. Software standards as an SE key practice are widely applied in the software industry to support the entire software development life-cycle. Based on our evaluation applied to a large number of agent-oriented methodologies, we believe that no well-structured process standardization has been proposed and incorporated into any requirements analysis phase in the existing agent-oriented methodologies.This paper addresses the process standardization for requirements specification of agent-based systems by proposing a model-driven approach to restructure and update the IEEE Std 830-2009 model to make it more suitable to handle the ABS requirement specifications.

The updated IEEE Std 830-2009 model proposed adding new extensions and updating others in the orig-

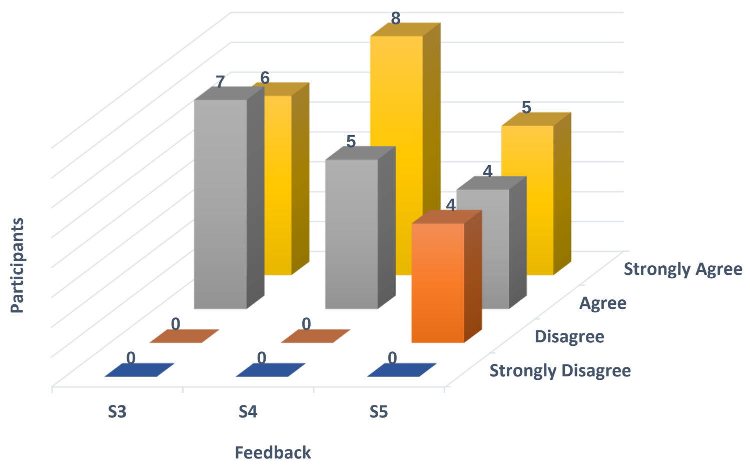

Figure 6: The degree of user satisfaction after applying the updated IEEE Std 830-2009 model

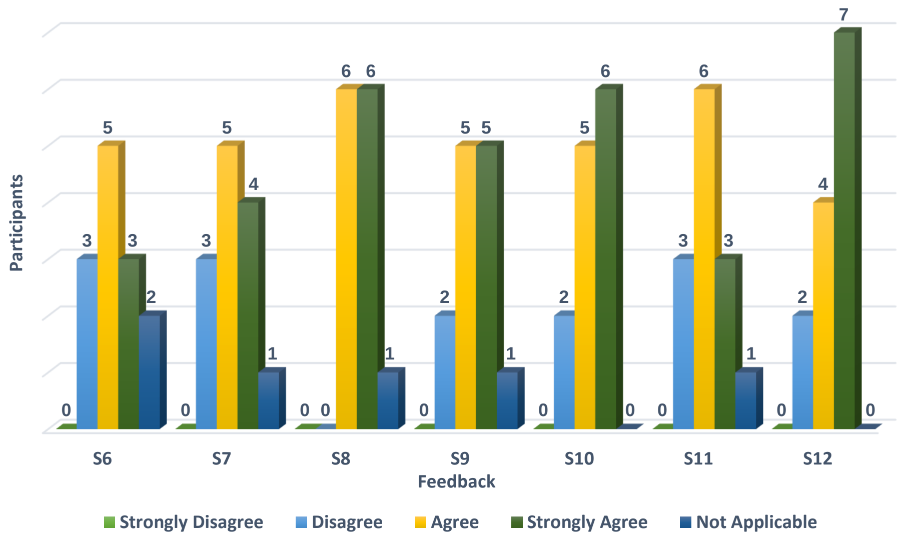

Figure 7: The summary of users' views regarding the given quality attributes as applied to the updated model

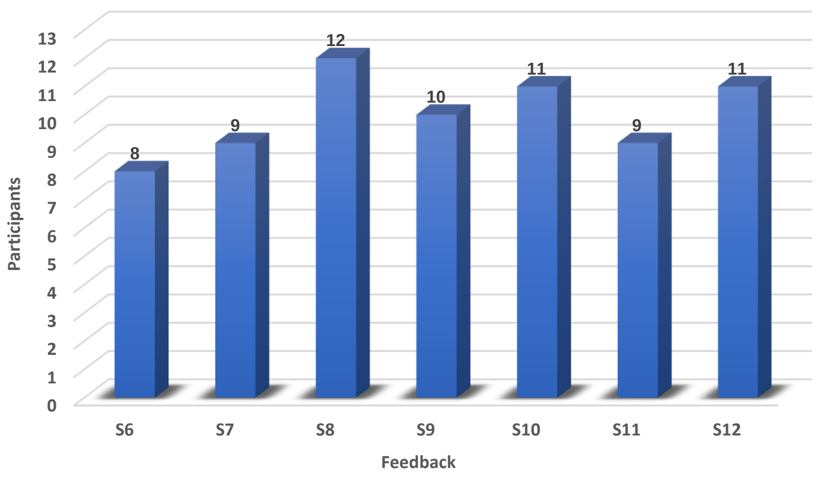

Figure 8: The summary of users' views who agreed with each quality attribute

inal model to complement the process standardization in ABS requirement specifications. We demonstrated the value of using the updated model by conducting several different types of evaluations. This paper describes an independent and external approach to evaluate the proposed model by asking expert participants to walk through the proposed model sections with a few simple real-world problem scenarios and then take an online survey to provide feedback on their experience using the model. The feedback was very encouraging as all participants agreed that the proposed model was capable of handling the ABS requirements, and were satisfied with using the model along with all updated and added agent information to it. Also, The summary of users' views with respect to the given quality attributes as applied to the updated model were positive and reflected how well the agent-oriented specification in the updated model met these quality attributes.

In the future, we plan to extend the updated IEEE Std 830-2009 model to cover more specific sub-standards for every ABS data model we identified and for every new central extension we added. This can be done by studying what sub-standard information is required to be specified in every extension. For instance, what ABS sub-standard information in the *section 2.2 domain perspective* needs to be defined, or what sub-standard information should be included in the *section 3.1 Agent properties*.

Conflict of Interest The authors declare no conflict of interest.

References

[1] K. Slhoub, M. Carvalho, and W. Bond, "Recommended practices for the specification of multi-agent systems requirements," in *Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON), 2017 IEEE 8th Annual*, pp. 179–185, IEEE, 2017. doi:10.1109/UEMCON.2017.8249021.

[2] B. Bauer and J. P. Müller, "Methodologies and modeling languages," *Agent-based Software Development*, pp. 77–131, 2004.

[3] IEEE, *IEEE Recommended Practice for Software Requirements Specifications*, vol. 2009. IEEE, 1998. doi:10.1109/IEEESTD.1998.88286.

[4] B. Henderson-Sellers, *Agent-oriented methodologies*. Idea Group Inc., 2005.

[5] O. Z. Akbari, "A survey of agent-oriented software engineering paradigm: Towards its industrial acceptance," *International Journal of Computer Engineering Research*, vol. 1, no. 2, pp. 14–28, 2010.

[6] F. Zambonelli, N. R. Jennings, and M. Wooldridge, "Developing multiagent systems: The Gaia methodology," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 12, no. 3, pp. 317–370, 2003. doi:10.1145/958961.958963.

[7] Q. H. Mahmoud, "Software Agents: Characteristics and Classification," *School of Computing Science, Simon Fraser University*, pp. 1–12, 2000.

[8] M. Wooldridge and N. R. Jennings, "Pitfalls of agent-oriented development," in *Proceedings of the second international conference on Autonomous agents*, pp. 385–391, ACM, 1998. doi:10.1145/280765.280867.

[9] A. Sloman, "What sort of architecture is required for a human-like agent?," in *Foundations of rational agency*, pp. 35–52, Springer, 1999. doi:10.1007/978-94-015-9204-8_3.

[10] F. Zambonelli and A. Omicini, "Challenges and research directions in agent-oriented software engineering," *Autonomous Agents and Multi-Agent Systems*, vol. 9, no. 3, pp. 253–283, 2004. doi:10.1023/B:AGNT.0000038028.66672.1e.

[11] L. Padgham and J. Thangarajah, "Agent Oriented Software Engineering: Why and How," *VNU Journal of Science: Natural Sciences and Technology*, vol. 27, no. 3, pp. 190–204, 2011.

[12] S. Maalal and M. Addou, "A new approach of designing Multi-Agent Systems With a practical sample," *CoRR*, vol. abs/1204.1, pp. 148–157, 2012.

[13] O. Shehory and A. Sturm, "A brief introduction to agents," in *Agent-Oriented Software Engineering: Reflections on Architectures, Methodologies, Languages, and Frameworks*, vol. 9783642544, pp. 3–11, Springer, 2014. doi:10.1007/978-3-642-54432-3_1.

[14] M. Wooldridge, "Agent-based software engineering," *IEEE Proceedings-software*, vol. 144, no. 1, pp. 26–37, 1997. doi:10.1049/ip-sen:19971026.

[15] N. R. Jennings, "On agent-based software engineering," *Artificial Intelligence*, vol. 117, no. 2, pp. 277–296, 2000. doi:10.1016/S0004-3702(99)00107-1.

[16] M. Wooldridge, *An Introduction to MultiAgent Systems*. Wiley, 2nd ed., 2009.

[17] E. Yu, "Agent-oriented modelling: Software versus the world," in *Artificial Intelligence and Bioinformatics*, vol. 2222, pp. 206–225, Springer, 2002. doi:10.1007/3-540-70657-7_14.

[18] Z. Ren and C. J. Anumba, "Multi-agent systems in construction-state of the art and prospects," *Automation in Construction*, vol. 13, no. 3, pp. 421–434, 2004. doi:10.1016/j.autcon.2003.12.002.

[19] P. Bogg, G. Beydoun, and G. Low, "When to use a multi-agent system?," vol. 5357 LNAI, pp. 98–108, Springer, 2008. doi:10.1007/978-3-540-89674-6_13.

[20] M. Dastani, "A survey of multi-agent programming languages and frameworks," in *Agent-Oriented Software Engineering*, vol. 9783642544, pp. 213–233, Springer, 2014. doi:10.1007/978-3-642-54432-3_11.

[21] J. P. Müller and K. Fischer, "Application impact of multi-agent systems and technologies: A survey," in *Agent-Oriented Software Engineering*, pp. 27–53, Springer, 2014. doi:10.1007/978-3-642-54432-3_3.

[22] J. Ferber, O. Gutknecht, C. M. Jonker, J. P. Müller, and J. Treur, "Organization models and behavioural requirements specification for multi-agent systems," in *Proceedings - 4th International Conference on MultiAgent Systems, ICMAS 2000*, pp. 387–388, IEEE, 2000. doi:10.1109/ICMAS.2000.858488.

[23] V. Hilaire, A. Koukam, P. Gruer, and J. J.-P. Müller, "Formal Specification and Prototyping of Multi-agent Systems," in *Engineering Societies in the Agents World SE - 9*, vol. 1972, pp. 114–127, Springer, 2000. doi:10.1007/3-540-44539-0_9.

[24] D. E. Herlea, C. M. Jonker, J. Treur, and N. J. E. Wijngaards, "Specification of bahavioural requirements within compositional multi-agent system design," vol. 1647, pp. 8–27, Springer, 1999. doi:10.1007/3-540-48437-X_2.

[25] T. Miller, B. Lu, L. Sterling, G. Beydoun, and K. Taveter, "Requirements elicitation and specification using the agent paradigm: The case study of an aircraft turnaround simulator," *IEEE Transactions on Software Engineering*, vol. 40, no. 10, pp. 1007–1024, 2014. doi:10.1109/TSE.2014.2339827.

[26] P. Giorgini and B. Henderson-Sellers, *Agent-Oriented Methodologies: an Introduction*. Idea Group Inc., 2005.

[27] L. Cernuzzi, M. Cossentino, and F. Zambonelli, "Process models for agent-based development," *Engineering Applications of Artificial Intelligence*, vol. 18, no. 2, pp. 205–222, 2005. doi:10.1016/j.engappai.2004.11.015.

[28] E. Ajith Jubilson, P. Joe Prathap, V. Vimal Khanna, P. Dhanavanthini, W. Vinil Dani, and A. Gunasekaran, "An empirical analysis of agent oriented methodologies by exploiting the lifecycle phases of each methodology," in *Advances in Intelligent Systems and Computing*, vol. 337, pp. 205–214, Springer, 2015.

[29] FIPA, "The standards organization for agents and multi-agent systems," 2005.

[30] P. Charlton, R. Cattoni, A. Potrich, and E. Mamdani, "Evaluating the FIPA standards and their role in achieving cooperation in multi-agent systems," in *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, vol. 00, pp. 1–10, IEEE, 2000. doi:10.1109/HICSS.2000.926996.

[31] V. T. Da Silva and C. J. P. De Lucena, "Modeling multi-agent systems," *Communications of the ACM*, vol. 50, no. 5, pp. 103–108, 2007. doi:10.1145/1230819.1241671.

[32] I. Trencansky and R. Cervenka, "Agent Modeling Language (AML): A comprehensive approach to modeling MAS," *INFORMATICA-LJUBLJANA-*, vol. 29, no. 4, pp. 391–400, 2005. doi:10.1.1.60.8902.

[33] B. Bauer, J. P. Müller, and J. Odell, "Agent UML: A formalism for specifying multiagent software systems," *International journal of software engineering and knowledge engineering*, vol. 11, no. 03, pp. 207–230, 2001. doi: 10.1142/S0218194001000517.

[34] A. Chikh and M. Aldayel, "Reengineering requirements specification based on IEEE 830 standard and traceability," in *Advances in Intelligent Systems and Computing*, vol. 275 AISC, pp. 211–227, Springer, 2014. doi:10.1007/978-3-319-05951-8_21.

[35] P. Bourque and R. E. Fairley, *Guide to the Software Engineering Body of Knowledge (SWEBOK(R)): Version 3.0*. Los Alamitos, CA, USA: IEEE Computer Society Press, 3rd ed., 2014.

[36] M. Cossentino and C. Potts, "PASSI: A process for specifying and implementing multi-agent systems using UML," *Retrieved October*, vol. 8, p. 2007, 2002.

[37] C. Bernon, M. Gleizes, S. Peyruqueou, and G. Picard, "ADELFE: A methodology for adaptive multi-agent systems engineering," in *International Workshop on Engineering Societies in the Agents World*, pp. 156–169, Springer Berlin Heidelberg, 2003. doi:10.1007/3-540-39173-8_12.

[38] M. Wood and S. Deloach, "An Overview of the Multiagent Systems Engineering Methodology," in *Proceedings - the First International Workshop on Agent-Oriented Software Engineering*, vol. 1957, pp. 207–221, Springer, 2001.

[39] M. Wooldridge, N. R. Jennings, and D. Kinny, "The Gaia methodology for agent-oriented analysis and design," *Autonomous Agents and multi-agent systems*, vol. 3, no. 3, pp. 285–312, 2000. doi:10.1023/A:1010071910869.

[40] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos, "Tropos: An agent-oriented software development methodology," *Autonomous Agents and Multi-Agent Systems*, vol. 8, no. 3, pp. 203–236, 2004. doi:10.1023/B:AGNT.0000018806.20944.ef.

[41] M. Georgeff, B. Pell, M. Pollack, M. Tambe, and M. Wooldridge, "The Belief-Desire-Intention Model of Agency," in *International Workshop on Agent Theories, Architectures, and Languages*, pp. 1–10, Springer, 1999. doi:10.1007/3-540-49057-4_1.