



Research Article

Agent-Oriented Software Engineering Methodologies: Analysis and Future Directions

Reem Abdalla ¹ and Alok Mishra ²

¹Benghazi University—Faculty of Education, Benghazi, Libya

²Informatics and Digitalization Group, Molde University College—Specialized University in Logistics, Molde, Norway

Correspondence should be addressed to Alok Mishra; alok.mishra@himolde.no

Received 18 October 2021; Revised 18 November 2021; Accepted 6 December 2021; Published 29 December 2021

Academic Editor: Saikou Diallo

Copyright © 2021 Reem Abdalla and Alok Mishra. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The Internet of Things (IoT) facilitates in building cyber-physical systems, which are significant for Industry 4.0. Agent-based computing represents effective modeling, programming, and simulation paradigm to develop IoT systems. Agent concepts, techniques, methods, and tools are being used in evolving IoT systems. Over the last years, in particular, there has been an increasing number of agent approaches proposed along with an ever-growing interest in their various implementations. Yet a comprehensive and full-fledged agent approach for developing related projects is still lacking despite the presence of agent-oriented software engineering (AOSE) methodologies. One of the moves towards compensating for this issue is to compile various available methodologies, ones that are comparable to the evolution of the unified modeling language (UML) in the domain of object-oriented analysis and design. These have become de facto standards in software development. In line with this objective, the present research attempts to comprehend the relationship among seven main AOSE methodologies. More specifically, we intend to assess and compare these seven approaches by conducting a feature analysis through examining the advantages and limitations of each competing process, structural analysis, and a case study evaluation method. This effort is made to address the significant characteristics of AOSE approaches. The main objective of this study is to conduct a comprehensive analysis of selected AOSE methodologies and provide a proposal of a draft unified approach that drives strengths (best) of these methodologies towards advancement in this area.

1. Introduction

Agent-based systems are one of the most vibrant and significant areas of research and development to have emerged in information technology in recent years [1]. Presently, the complexities in developing software are growing faster. Therefore, the software is failed to produce flexibility, robustness, efficiency, and reliability. To fulfill these demands, agent-oriented (AO) technique is evolved in the software engineering field [2]. AOSE supports the design of dynamically interacting components, each components with its own thread of control, and networking applications, includes ubiquitous computing, sensor networks, and intelligent cities, and can handle uncertainty [3–7].

The future IoT is expected to enable a new and wide range of decentralized systems (from small-scale smart homes to large-scale smart cities) in which “things” are

capable to sense/actuate, compute, and communicate and thus provides a central and crucial role. The growing importance of such a novel networked cyber-physical context demands suitable and effective computing paradigms to fulfill the various requirements of IoT systems engineering. The synergic meeting of agents with IoT makes it possible to develop smart and dynamic IoT systems of diverse scales [8].

The disruptive potentials of the IoT include complex requirements and development issues (large-scale deployments, heterogeneity, cyber-physicality, interoperability, distributed smartness, self-management, etc.). To appropriately address them and comprehensively support the development of the IoT ecosystem, agent-based computing represents proper and solid modeling, programming, and simulation paradigm [9]. Software agent-directed simulation and modeling can be undertaken on various devices, tiny notebooks to desktops, large-scale workstations, or powerful computing clusters and

supercomputers [10]. Software agents have been now integrated into many applications. Due to a move to mobile computing paradigms, lightweight platforms, mainly oriented for resource-constrained devices, have emerged [11]. However, resource-constrained devices lack the resources necessary to support a traditional runtime collection agent alongside the main application. Sherrell et al. [12] proposed an extensible, open, agent-based framework for the trustworthy collection and evaluation of runtime integrity evidence from resource-constrained platforms. Recently Savaglio et al. [9] supported that agent-based computing comprehensively supports the development of the IoT ecosystem, to represent proper and solid modeling, programming, and simulation paradigm. They further argued that agent-based computing facilitates the management of devices, their actions and reactions, and their data, as well as the interconnection of different embedded computing platforms and communication protocols. Savaglio et al. [13] showed as a case study that how agent-based methodologies can drive smart objects (SO)/IoT system reengineering, in order to highlight and enhance both functional and nonfunctional features (e.g., support to interoperability, attention for resource-constrained SOs, modularity, maintainability, and evolvability) that generate fundamental benefits for the complex, heterogeneous, and constantly evolving IoT scenario.

AO considerations are being continuously accepted into the various software design paradigms. Agents may work alone, but most commonly, they cooperate towards achieving some application goal(s). From a software engineering view, resolving a problem should include problem realization, requirements analysis, architecture design, and implementation [14]. Many AOSE methodologies have emerged in order to facilitate the construction of intelligent software [15]. However, the immaturity of this emerging technology can result in difficulties for a developer when determining which methodology can best fit a potential application. A few studies have been conducted to address the comparison and evaluation of AOSE methodologies [15]. AOSE is an efficient methodology for open, complex, and distributed systems in networking, academic, medical, industrial, and commercial areas [16].

In order to implement AOSE methodology for different problem domains, the developing system architecture has to be considered as MAS [16]. Many of the issues addressed with multiagent ways, for instance, distributed coordination and self-organization, are now becoming part of industrial and business systems. However, MASs are still not widely adopted in the industry owing to the lack of a connection between MAS and software engineering [17].

AO programming represents a novel programming paradigm that adopts concepts and technologies of MAS to implement software. It has gained great attention from researchers and practitioners from both artificial intelligence and software engineering fields [18]. In eHealth systems, progress depends on the interoperability of local healthcare software and is often hampered by ad hoc development methods leading to closed systems with a multitude of protocols, terminologies, and design approaches [19].

The eHealth domain, by requirements, includes autonomous organizations and individuals, for example, patients

and doctors, which would make AOSE a good way to develop systems that are potentially more open yet retain more local control and autonomy [19]. Recently Salazar and Li [20] observed that cyber-physical production system (CPPS) is one of the most significant concepts of Industry 4.0, which has attracted disseminated attention from both the academic community and the industry. They observed it is generally accepted that the IoT and advanced methodologies for manufacturing systems are essential factors to achieve CPPS. Fortino et al. [8] also supported that an agent-based computing paradigm is required to support IoT systems analysis, design, and implementation. They further observed that the agent-based computing paradigm provides developers with a high-quality set of metaphors, design methods, technology, and frameworks to simplify the modeling and programming of smart IoT systems as multiagent systems, generally by means of knowledge bases, conflict argumentation frameworks, and semantic annotation tools.

With the increasing significance of such critical systems in the industry, our need for employing agent technologies to promote commercial and industrial software systems is rising rapidly. Therefore, some promising methodologies are proposed, yet most of these methodologies need to be further evaluated [20]. For many years, the progress in AO development has focused on tools and methods for particular development phases, which is not enough for the industry to adopt agent technology [15]. There is still no standardization in AOSE, resulting in several methodologies, and another issue of this discipline is that there are very few tools that are able to automatically generate code [21].

Therefore, this work provides analyzing seven AOSE methodologies along with a proposed draft of unified methodology. The objective of this study is to analyze selected AOSE methodologies: Gaia, Tropos, PASSI, O-MaSE, ADELFE, ASEME, and Prometheus, in order to distinguish each one's strengths, weaknesses, commonalities, variations, and range of applicability. The aim of this paper is thus to move one step closer to a standard methodology by proposing a common process and set of models. We do not aim to propose a final standard methodology, but an exploratory proposal, which we hope will lead to discussion in the AOSE community and eventually this discussion and collaboration, might lead to a real unified AOSE methodology.

The rest of the paper is structured as follows. Section 2 is related work. In Section 3, selected methodologies are briefly presented. Section 4 describes the structural analysis of methodologies. Section 5 provides a comparative analysis of selected AOSE methodologies. Section 6 outlines a proposed novel unified AO methodology. Section 7 illustrates a case study of the proposed methodology. Section 8 contributes discussion. Finally, the paper concludes with the conclusion, limitations, and future research directions.

2. Related Work

There are multiple frameworks to compare proposed methodologies to the agent. Shoub et al. [22] discussed various agent-based systems classified in different application areas and an evaluation of three various AO

methodologies: PASSI, MaSE, and Prometheus. In the work by Sukhvir and Richa [23], there is a discussion regarding various agent-based systems classified in different application areas and an evaluation of five various AO methodologies, Gaia, O-MaSE, MESSAGE, Prometheus, and Tropos, through conducting a feature analysis to integrate their strong points to develop new extensions.

This work was achieved through a comparative study based on a features-analysis method. Elammari and Elsaeti [24] analyzed five agent-based methodologies in an attempt to identify a suitable AO methodology by performing structural analysis. Their evaluation assesses and understands the relationship among these AO methodologies [24]. Abdalla and Mishra [25] introduce a comparative study of agent approaches with the purpose to assess and compare the development lifecycle processes of four different well-known AOSE methodologies, ADELFE, PASSI, Gaia, and O-MaSE, to define the similarities and differences.

Zohreh [26] studied and surveyed AO methodologies towards its industrial acceptance. They proposed a solution to this problem by the usage of an assessment framework to provide a clear and basic definition of common characteristics and the main components used in MASs. Dam and Winikoff [27] provide an analysis study aimed to capture the relations among five prominent AO approaches, Gaia, O-MaSE, MESSAGE, Prometheus, and Tropos, in particular, through an analysis of their features by assessing the strengths and drawbacks within an attribute-based structure. Included in this framework are criteria such as concepts, modeling language, operation, and pragmatics.

The comparison can take a wider scope with a small experimental evaluation of methodologies, which makes this comparison more important as it includes inputs from methodological rulers using a questionnaire. A structural analysis is performed where main common factors and clear differences are identified for the five agent methodologies in terms of paradigms, processes, and programs. Also, several initial proposals are presented so as to integrate these AO approaches by combining their advantages and minimizing their drawbacks [28].

Given the main purpose as to apply AOSE methodology for constructing powerful, industrial-strength MAS as well as the current trends, the best way to develop agent programs is by obtaining further analysis and understanding of the proposed method. There are many attempts to evaluate and compare AO methodologies undertaken by many researchers in this field. Dam and Winikoff [27] state that it is time to work towards a new generation of AOSE to give us an ultimate and next-generation methodology.

3. Selecting Methodologies

We will adopt a multiphase selection method [29]. The primary step minimizes the set of approaches by utilizing the standard criteria that follow:

Documentation. The document of methodology must be clear and elaborate in characteristic, not just a presentation at a conference but also that cited and

included in books, journal papers, or other well-known technical reports.

Maturity. The methodology chosen has to have extensive use and improvement over time, involving the use by other nonauthors and their colleagues and students.

Tool support. The approach has to be preferably have backing tools rather than not. Only chosen methodologies are to be applied to develop a particular project [28]; for this reason, the presence of tool support is an effective attribute.

In further detail, the following are the methodologies selected for the present work.

Gaia [30] that is particularly designed for the development of agent-based systems; Tropos [31] that is a detailed AO software engineering methodology and that supports a broad domain of software lifecycle processes; PASSI [32] for planning and the evolution of multiagent communities to integrate design models; O-MaSE [33] that is a customizable AO approach based on compatible and well-established ideas strengthened by additional applications in industrial development environments; ADELFE [34] that is a multi-AO approach convenient for adaptive MAS; ASEME [35] that suggests a modular agent design approach and presents the notions of control within the agent; and finally, Prometheus that presents models and activities to construct smart agents using objectives, beliefs, plans, and events.

These approaches support all of the standards set earlier: all of their documentations have appeared in journal papers (see Tables 1–3) and have support tools (except for Gaia). In terms of maturity, O-MaSE, Prometheus, Tropos, Gaia, and PASSI stand out meeting the criteria, but the other two, ADELFE and ASEME, are a bit weaker on the maturity criteria, to be more precise and detailed.

3.1. Methodologies Overview. This short overview covers the processes, models, and techniques in the methodologies intended for our research.

3.1.1. Gaia Methodology. Gaia [36] was the first agent methodology, which was designed specifically for agent-based systems by dividing the activity of developing software into two main steps, analysis and design; it groups and arranges the specifications as the foundation for the second stage, which is a computational organization. Gaia determines both the objectives and the expected conduct of the organizations, which form the system in general. For this purpose, it breaks down the global organization into highly interrelated subsidiary institutions. The preliminary roles model may not be fully defined but can be determined without imposing a particular organizational architecture [36]. In this respect, the first step (i.e., preliminary interaction) describes the main interactions expected to achieve the elementary roles. The environmental model aims to represent the abstract and computational environment where MAS is to be developed. The organizational rule model provides a set of principles that the organization has to take into account and put into effect in its global conduct [37]. The design stage is based on the

TABLE 1: Comparative analysis of Gaia and Tropos methodologies.

Methodology	Strengths	Limitations	Application domain
Gaia	(i) Directs designers to a well-defined design for the MAS (ii) Model and copying with the features of complex systems (iii) Is easy to apply	(i) Agents cannot share common goals (ii) It does not handle systems while acting as an actor (iii) The coherence between the protocols is rather weak (iv) Components of the system may enter or leave at runtime in open or dynamic systems (v) It addresses goals, but verification of goals is still out of scope (vi) It does not UC scenarios [23]	(i) Application of agent methodology in healthcare information systems [46] (ii) Multiagent system vulnerability detector for a secured E-learning environment [66] (iii) Multiagent-based transformer condition monitoring [67]
	(i) Suggests methods and tools for automating models transformation (ii) Focuses on the preliminary stage of requirements (iii) Uses actors and goals as basic concepts [68].	(i) It is not designed to support a certain type of software. (ii) Developed software agents designed for accomplishing special goals are not Supported by Tropos methodology (iii) Does not UC scenarios [23].	(i) A process for developing adaptable and open service systems: Application in supply chain management [30] (ii) Using Tropos to model agent-based architectures for adaptive systems: a Case study in ambient intelligence [39]. (iii) Developing a decision support system for integrated production in agriculture [52].

TABLE 2: Comparative analysis of PASSI and O-MaSE methodologies.

Methodology	Strengths	Limitations	Application domain
PASSI	(i) Suggests a complete lifecycle methodology from requirement to code methodology (ii) Integrates design models and concepts from both object-oriented (OO) and MAS using UML notation (iii) Refers to the most diffused standards: UML, the foundation for intelligent physical agents (FIPA), JAVA, and Rational Rose [32]	(i) The need to refer simultaneously to various models in order to understand the system and the way it works and changes over time is a critical issue (ii) Every model offers its own set of notation and special concepts, resulting in an abnormal complexity in terms of vocabulary (iii) It does not support the environment model [32]	(i) A domain analysis approach for MASs product lines [58] (ii) The development of a multiagent-based middleware for RFID asset management system using the PASSI methodology [59] (iii) Patterns reuse in the PASSI methodology [32]
	(i) Is comprehensive for building MAS (ii) Is step-by-step lifecycle methodology MAS (iii) Provides guidance throughout the entire software development lifecycle (iv) Open systems are considered, thus agents can be created, deleted, or moved during implementation	(i) Some of the software applications are closed (ii) Management, product distribution, and testing and assessment have been absent (iii) There is the only one-to-one connection among agents in the system (iv) It does not clearly define the usage case model [23].	(i) Agent-based mixed-initiative collaboration project [60] (ii) O-MaSE: a customizable approach to developing multiagent development processes [61] (iii) Developing a multiagent conference management system using the O-MaSE process framework [42]

output of the analysis stage. The topology and control systems are defined through the architectural design step, which also employs catalogs of different institutions' models. Figure 1 shows the processes in Gaia stages.

3.1.2. Tropos Methodology. The methodology comprises five stages. In the first phase, the main system requirements are extracted. Later, most of the general features determined in the initial requirements analysis are used in the late

requirements analysis stage. Throughout both these phases, the same concepts and technical methods are applied, especially during the first phase as it is intended to extract the main system requirements [38].

In this phase, the primary function of the system actor is to deliver system-based services to actors based on services from the recent analysis stage [39]. In this respect, the architectural and detailed design stages concentrate mainly on the system specifications based on the requirements obtained from the previous stages [40].

TABLE 3: Comparative analysis of ADELFE, ASEME, and Prometheus methodologies.

Methodology	Strengths	Limitations	Application domain
ADELFE	(i) Can be used by a nonspecialist in agent systems	(i) It is specialized; thus, it cannot be used to design all existing applications or model all kinds of agents	(i) Tools for self-organizing applications engineering [49]
	(ii) Will also be easier to combine the parts from other approaches into ADELFE	(ii) Some definitions of work still do not exist	(ii) A sample application of ADELFE focusing on analysis and design of the mechanical synthesis problem [43]
	(iii) Use the cooperation rules [49]	(iii) Sometimes the developer may find the graphical modeling tool difficult to use [49]	(iii) A tool for generating model transformations by example in MASs [62]
	(iv) Allows different UML/AUML realizations		(iv) ADELFE 3.0 design, building adaptive multiagent systems based on simulation a case study [1]
ASEME	(i) Supports documentation of nonfunctional requirements	(i) The guidelines are missing because the authors rely only on the paradigm shifts	(i) Automated product pricing using argumentation [54]
	(ii) Provides the model transformations among the various development stages	(ii) The same thing happens with the tasks to be conducted in each stage of development	(ii) Using ASEME methodology for model-driven agent systems development [35]
		(iii) Some processes need to be improved	(iii) Engineering an agent-based system for product pricing and automation [35]
		(iv) In requirements, implementation, and design, there are no guidelines for the production of models	(iv) Using agent-based methodologies in healthcare information systems [47]
Prometheus	(i) Offers a complete lifecycle methodology from requirements specification to detailed design	(i) There is less focus on initial requirements	(i) An open meteorological alerting system: issues and solutions [63]
	(ii) Supports both dynamic and static models for individual agents	(ii) It does not deal at all with mobile agents	(ii) Tool support for agent development using the Prometheus methodology in quality software [64]
	(iii) Provides elaborated guidance on how to carry out the different stages	(iii) It cannot be established on UML	(iii) AO modeling and development of a person-following mobile robot [65]
	(iv) Gives clear support to the environment concept	(iv) Support for the socialite side of the agent focuses on the lowest common divisors: messages and protocols	

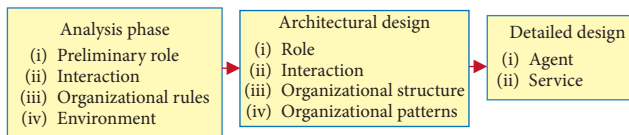


FIGURE 1: Gaia methodology process.

In the elaborate design step, the purpose is to define the agent abilities and reactions, by which point, the application platform is usually selected already and so can be utilized to an exhaustive design that will move directly to the code stage. The execution processes follow a step-by-step pattern of the exhaustive design characterizations based on the path among the execution platform constructed and the detailed design concepts [41] (see Figure 2).

3.1.3. PASSI Methodology. PASSI (Process for Agent Societies Specifications and Implementation) proposed by Cossention [32] is a step-by-step requirements to code methodology for designing and developing multiagent societies integrating design models and concepts from both object-oriented software engineering and artificial intelligence approaches using UML notations to code phase. It consists of five steps involving all the stages adopted in the UML modeling language. Since

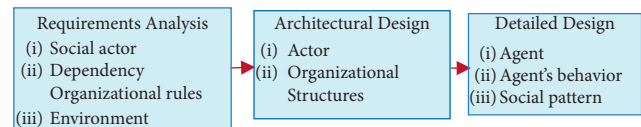


FIGURE 2: Tropos methodology process.

UML is highly accepted in both academia and the industry, PASSI uses UML as its language of modeling with expanded schemes to include certain notions left out from UML.

These are later updated so as to represent the best of what modeling should be in a particular artifact within the agent design community [32]. In PASSI, an agent is an important software entity, both at the abstract level and the design level. In light of this fact, an agent represents a class, thus making it an independent unit when designing software by reaching a goal through independent resolutions, procedures, and social relations: Figure 3 shows the processes in PASSI stages.

3.1.4. O-MaSE Methodology. The O-MaSE process structure [27, 42] assists processing engineers in order to determine custom MAS analysis activates. It offers the models, principles, approaches fragments, and guidelines necessary to collect O-MaSE-compatible operations with the main goal to

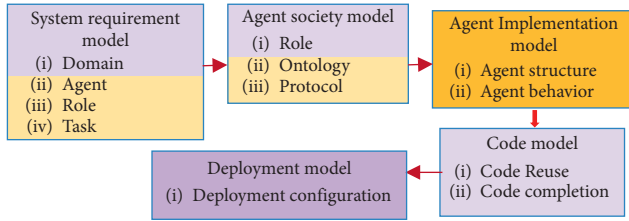


FIGURE 3: PASSI methodology process.

allow the construction of agent-software activities. O-MaSE is composed of three major models: (1) a metamodel, (2) a group of method fragments, and (3) a group of guidelines. The O-MaSE metamodel is regarded as the main factor needed to develop and execute MAS.

As for the method fragments, these are processes followed to provide a set of work products, possibly including diagrams, documents, or codes. The relationships among the method fragments are specified by the guidelines. Figure 4 shows the processes in O-MaSE stages.

3.1.5. ADELFE Methodology. The ADELFE methodology [43] has been designed to focus on several aspects left disregarded by current approaches. It unifies AMAS (adaptive multiagent systems) theory and offers a specific process derived from an interpretation of RUP (rational unified process). Several addendums have been proposed as AMAS theory so as to include features characterizing the environment in the system in order to identify certain shortcomings in the collaboration process. Each user and task deliverer has an individual goal to deal with the required request without the need to understand the entire system functions.

In this methodology, the engineer, AMAS specialist or not, is directed to use adaptive MASs in order to define the agents with the help of the environment models. ADELFE offers the analyst tool to evaluate the sufficiency of AMAS technology at two levels: global (system) and local (components). At the local level, three parameters are taken into account, whereas the global one involves eight. To demonstrate agent interaction protocol (AIP), the agent unified modeling language (AUML) [44] principle is used together with UML and RUP. Two other tools are incorporated into the framework (see Figure 5).

3.1.6. ASEME Methodology. ASEME (agent systems methodology), proposed by Spanoudakis and Moraitis [35], contains the requirements, design, and implementation stages. It backs modular agent design processes and offers the notions of an agent within control to determine the agent's conduct by formatting various modules that execute his ability and intraagent control (IAC) model that defines the protocols that control and coordinate the activities of the agent community. The requirements analysis stage identifies the actors, their goals and interactions, and use case (UC) diagrams so as to assign actors to roles and goals in accordance with the capabilities needed to fulfill those tasks.

The partake doers are recognized by the objectives assigned to each. Furthermore, data is collected as to the

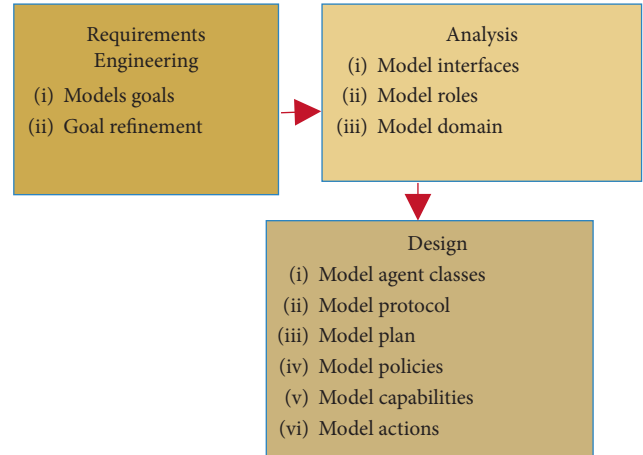


FIGURE 4: O-MaSE methodology process.

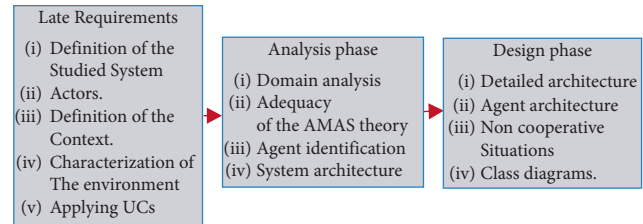


FIGURE 5: ADELFE methodology process.

specific requirements that constitute the anticipated tasks of the project. The analysis stage is based on the notions of ability and function, comprising two steps: the UC and the roles diagrams. Initially, the actors in the previous steps are converted to roles, which can be more abstract than the roles of specific actors depending on the scope of implementation. The design stage encompasses the functional and behavioral sides of the MAS, and the related models are the agent interactions protocol (IP) and IAC that carry out a certain IP by assuming the crucial roles and relations between them [35]. The execution phase is the programming language for different levels throughout the development process. Figure 6 shows the processes in ASEME stages.

3.1.7. Prometheus Methodology. The Prometheus methodology [45] is composed of three stages: first, at the specification stage, the system is defined by the objectives and UC scenarios; the system mediator to its surroundings is explained in terms of the events, concepts information, and tasks. Next comes structural design, through which the type of agents is determined. Here, the general structure of the system is described in the system overview model, while UC is incorporated into the communication protocols. Finally, the third stage involves detailed design used to develop and define the details of the internal agents such as capabilities, beliefs, plans, tasks, and events. Then, process models are applied as a starting point among the protocols and interaction plans. In general, each of these steps contains models that concentrate on the dynamics of the system in the form of schemas and models regarding system architecture or its

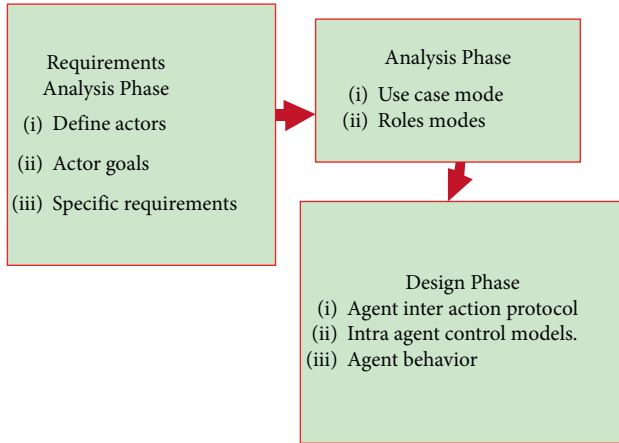


FIGURE 6: ASEM methodology process.

ingredient as well as textual depicter forms that deliver the details for individual elements. Figure 7 shows the processes in the Prometheus stages.

4. Structural Analysis of Methodologies

Based on our earlier work [25, 46, 47], the assessment of methodologies was performed by introducing a comparative study of agent approaches with the purpose to assess and compare the development lifecycle processes of different well-known AOSE methodologies, to define the similarities and differences between them [46] as well as a case study method that designed a target system using different methodologies [25, 47]. In this paper, we perform a structural comparison rather than comparing AOSE methodologies based on their features. We explore what models and processes these methodologies share and what are their distinguishing aspects.

In this section, methodology processes and models are addressed so as to identify the common cores and components of these methodologies. To do so, structural analysis is performed on each process and model, allowing us to determine the similarities between these methodologies as well as the benefits of each one.

4.1. Structural Analysis: The Commonalities

4.1.1. Initial Requirements. Of the seven selected AO methodologies, Tropos and ADELFE make use of this technique, which is an important stage in this methodology; Tropos focuses on the intentions of stakeholders. In addition, this model is concerned with key agent notions such as objectives, intents, plans, and so on, as used in AO programming. Through a variety of forms of objective-oriented analysis, these preliminary aims ultimately lead to the functional and nonfunctional demands of the target system.

Therefore, the difference among various development stages decreases. Furthermore, this can lead to the formation of unified and consistent techniques and tools for designing software. The goal-oriented requirements analysis approach was used in this model by Yu [48] to study an organizational

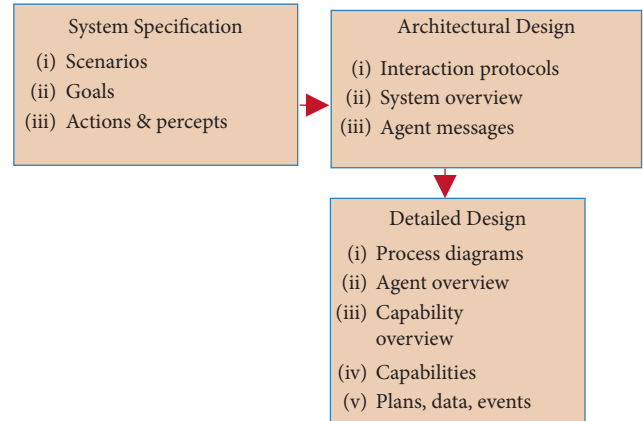


FIGURE 7: Prometheus methodology process.

base involving the users, their intentions, and relationship. ADELFE initial demand stage objective is to convert this seeing to a UC diagram and arrangement of the requirements (functional or not) at this step; the developer has to determine the task of the thoughtful system and model its environment [38].

4.1.2. Applying UC Requirement Analysis. This technique is applied in order to characterize each system's practical requirements. The model has been successful in object-oriented methods to gather system requirements apart from assisting developers to identify the main interactions among system entities. Of the selected methodologies, ADELFE, PASSI, ASEM, and Prometheus have applied UC models. In ADELFE, while complementing the workflow requirements, this operation consists of three steps: designing UC, clarifying the linked sequence diagrams, and determining collaboration failures. In these UC, only energetic components are implicit and appear as the products of an effective requirements group [49].

Exploring situation collaboration failure in the system and within its conditions is carried out so as to help developers in identifying problems and noncooperative items and incidents. This definition serves as a filter across the evolution activities to later define the agents in the process [49]. Instead of using the objectives in the requirements engineering stage, PASSI authors prefer the approach of Booch et al. [50] and use the UC diagrams to describe requirements. In this respect, the domain characterization stage practically depicts system components of a hierarchical set of UC diagrams. The sequence models explain graphical scenarios for the detailed UC, while agents are presented as a pack of UC in the functional analysis of the previous stage.

In ASEM, the purpose of using the UC model is to show that artificial agents are designed as components reactive with other artificial agents or human agents. No new elements other than those offered by UML are needed, but shifts in semantics are displayed. First, the actor interacts with the system and supposes a role. Then, agents are developed as roles inside the system boundary [35]. Similar to conventional UML, UC models, human

actors are demonstrated as roles outgoing the system box. Later, the various UC should be linked to at least one artificial agent role.

The UC diagram in ASEME also adds three new ideas regarding the actor diagram and offers actors designed inside the system boundary. The diagram can add abstraction roles to ensure that agent IP and goals are viewed from a developer standpoint by adding subgoals related to implementation in the shape of UC [51]. The third step of the system specification stage in the Prometheus approach is to set up UC as comprehensive descriptions of a series of events to fulfill specific goals or to react to a specific event [45].

4.1.3. The Concept of Environment. The concept of environment to MAS is essential as agents operate in an environment, and in order to support complex open systems, agent methodologies need evident model's characteristic of the domain as well as the environment of their implementation. Often, complicated open systems have very dynamic and diversified environments, with which defining, and familiarizing is necessary if the constant change is an issue.

As a result, an agent system needs to have models to accurately represent the environment in which it operates. Despite the importance of developing an environment model, ADELFE, Prometheus, and O-MaSE manage to specify such models. In ADELFE, before identifying UC and during the final requirements, the environment should be thoroughly planned by the developer. Afterward, one procedure is added to the RUP to describe the system environment. Specification starts by determining that elements interact with the system as well as the restrictions in these connections.

In UML, an entity represents an actor and can be characterized as effective or ineffective in ADELFE. In turn, an effective or active element may act independently and be capable of working dynamically with the system. Whereas an ineffective element can be taken as an exporter of the system and, as such, utilized or updated by an effective one, but it cannot be changed independently. This distinction between entities is necessary because the agent that makes up the system and is not yet recognized at this step will be set between the effective ones. A mediator model that includes a series of concepts and events conceivable in the surroundings of the agents is offered to showcase the environment.

Prometheus presents the environment from the system perspective. The motivation behind this showcase is that the system, in general, and agents, in particular, see the surroundings across a number of sensors. For this reason, it is essential to obtain obvious input/output specifications for the required system features. The model offered by Prometheus only covers this interface, and the main components of the surrounding in which agents will work are shown by the domain diagram in O-MaSE. These components are described in the form of objects from the surrounding, which contain agents, and interactions among those objects. It can also be used to show the general characteristics of the environment to see how the objects connect [42].

To show how an organization may conduct in a certain circumstance a developer uses the elements determined in the objectives, roles, and agents diagrams along with those elements determined in the domain model in order to define organizational policies. Nevertheless, most of the other seven approaches fail to properly consider this significant point. In fact, Gaia does not provide a comprehensive model as to the environment design for developers, and such data is encoded only in the authorizations and protocols of specific roles. This handicap makes Gaia unsuitable for designing implementations with active and diverse environments. In this respect, Tropos offers resources as an entity but no more.

4.1.4. Capturing Goals. In the domain of artificial intelligence, there are two forms of agents, weak and robust. The first group is distinguished from the next in terms of independence, reactivity, proactivity, and socialite capabilities [52]. The strong agency is defined by all such properties as those of weak agency, but sometimes, it also attributes to other features like mobility, veracity, and benevolence. In general, agents must have proactive abilities to accomplish their goals pursued over time. For this reason, all agent methodologies subject to the analysis here pay more intention to agent goals. The exceptions, in this case, are ADELFE and PASSI. Gaia models the goals in the form of roles and responsibilities. The other methodologies identify goals in the requirements analysis stage to be applied as a basis for identifying agents. In Tropos, setting goals is an important process, also known as goal modeling, and is done by setting both system and individual goals, thus resulting in more goal-oriented outcomes [53].

Goal diagrams are designed through the early requirements stage by utilizing initially specified actors and objectives and, thereafter, described during goal models [38]. These models operate identically while determining actor relationships in the late requirements stage and structural design [41]. The objectives and plan diagrams enable designers to understand the goals and plans by completing certain activities, namely means-end analysis, which converts a goal into subobjectives to determine schemas, resources, as well as soft goals, which offer a means for accomplishing the goal. Next comes contribution analysis, which enables developers to assess the objectives in terms of either positive or negative contribution.

While there are variations in terms of goal setting in agent societies, O-MaSE specifies a goal concept as a main objective of the institution and sets it whenever the situation calls for it. According to DeLoach and Garcia-Ojeda [33], the objectives of the organization are set in the form of a behavioral goal hierarchy and involve the goal characteristics and the relationship of precedence.

As to ASEME, the goal model is similar to the Tropos actor model but without the same diagrams. ASEME defines more concepts used by agent modeling language (AMOLA), which offers more appropriate models for system analysis, and uses the goal and actor concepts in the related diagram [54]. Determining objectives is also a vital ingredient in Prometheus and presents such models as an essential part of

the institution. The goal-capturing stage has two steps: identifying goals and structuring goals [45].

First, the objectives are specified by analyzing and understanding the group of requirements. Then, they are arranged in a shape that can be transferred to the design phase [45]. In general, each objective has a role (one to one). Also, in its analysis stage, Gaia defines the goals of the organizations alongside their predictable behavior. According to some studies, ADELFE and PASSI methodologies do not address the issue of goals well enough.

4.1.5. Social System Structure. Presently, computer system applications are becoming more and more complex. In this respect, the benefits associated with using MAS in all types of software engineering programming require strong support for engineering complex open systems that emphasize the social aspects of an agent system [55]. Among the most important features to add AO paradigms in software engineering is the ability to develop and implement complex systems [56]. Consequently, it is quite necessary for AOSE approaches to offer a clear way to comprehend the general framework of the system. Except for Gaia, most of the selected processes handle this matter fairly well. Between the seven approaches, only Gaia does not have a thorough vision when it comes to a social system [27].

O-MaSE provides a social overview by supporting the concepts of roles and objectives and represents the conversations among the main elements in the system. Tropos has extensive schemas, which offer the relations among actors, objectives, resources, and functions in the system. If the significance of a comprehensive look at the structure of an organizational system lies in its ability to show a fixed structure, by the same token, it is necessary to capture high-level system dynamics [27].

The conversations and connections taking place between agents involve characterization of the mechanism that agents employ to organize their other complicated social actions or interactions such as rivalry, discussion, and teamwork. Accordingly, O-MaSE and Tropos show conversations at two various standards of detail by including a group of high-level connections and more granularity performance in terms of IP [53].

From our observations, ADELFE, PASSI, Prometheus, Tropos, and O-MaSE share certain features as they model high-level interactions using sequence/interaction diagrams extracted from UML sequence models. Each methodology has different interaction diagrams; for example, Tropos and ADELFE depict conversations among agents. This is while the sequence models in O-MaSE show reactions among roles and actors. PASSI uses serial graphics to explore the tasks of each agent through role-specific screenplays [15].

The role model in ASEME is basically derived from the Gaia methodology since; in both cases, the conversations are explored and modeled at the role level instead of the agent level, and serial interactional schemas are not used for the objectives [51]. Getting into more IP detail, ASEME, ADELFE, and Tropos suggest the application of the AUML, IP model, with certain tasks added to the AIP to suit AMAS's demands in ADELFE.

4.1.6. Agent Acquaintance Model. In Prometheus and Gaia, the dependency and interaction among agents are illustrated in terms of agent acquaintance schemes in the form of guided diagrams including rectangles (referring to agents) and lines with arrows (referring to links, interactions, and relations) among different system elements. While Gaia realizes the connection, links are found among agents without determining the current properties of those links. Prometheus indicates the types of agents as well as the connections among them.

However, in the design stage of Prometheus, these connections appear in a more detailed manner. For this, agent knowledge diagrams in these two approaches are intended to help developers determine potential bottlenecks formed among the system elements.

Table 4 shows the results of this analysis.

4.2. Structural Analysis: The Differences

4.2.1. Model-Driven Engineering (MDE) Approach. In ASEME, the MDE method appears with special features concerning the AOSE community and can be applied in all stages of conventional software processes (from requirements to execution). It allows the transition from one stage to another through typical transformations. The three forms of conversion used for automation between the ASEME models are: model to model (M2M), text to model (T2M), and model to text (M2T).

Process designers simply enrich these models in each stage with specific information, thereby leading to execution progressively. Furthermore, the steps in the design stage within this approach is a state chart designing models known to developers that can be performed by either different programming languages or AO platforms [57]. Yet, unlike ASEME, the Tropos methodology nominates methods and tools to automate but only in several stages of software processes.

4.2.2. Documentation of Nonfunctional Requirements. One of the most significant differences between ASEME and the other participant methodologies is in its backing registration of nonfunctional requirements, in the requirements analysis step, where they are utilized to make management decisions and choose which technologies will be used for design and development [51].

4.2.3. Deployment Model. A major feature, which distinguishes PASSI from other participant methodologies, lies in its strong focus on the deployment model, regarded as one of the main models in UML [39]. There are substantial benefits to this model. For example, in the process of building the deployment model, developers can have a better grasp of the intricacies to operate the system. In addition, developing high-scale deployment offers a basis for evaluating the feasibility of executing the system and the use of the spread concept. It also provides an assessment of different other measures, such as costs. Using this activity, PASSI describes

TABLE 4: The summary of commonalities.

	Initial requirement	Using use cases	Capturing goal	Social structure	Acquaintance model
Gaia	X	X	√	X	√
Tropos	√	X	√	√	X
PASSI	X	√	X	√	X
O-MaSE	X	X	√	√	X
ADELFE	√	√	X	√	X
ASEME	X	√	√	√	X
Prometheus	X	√	√	√	√

the system based on agent classes and their position on the available processing units in the diagrams. While building the deployment diagrams, designers have the opportunity to accurately set the system in a design step by thinking about agent-linked data. PASSI offers elasticity in the system spread design by enabling the developers to create various system arrangements and to update them regularly.

4.2.4. Data Coupling Diagram. A key aspect that distinguishes Prometheus from other participant methodologies is data coupling as it offers evident processes to cope with agent characterizations. One of them is the use of data coupling models. The second is the model of agent acquaintance. These models are composed of system functions and external resources in terms of specified data. In this way, developers are able to assemble functionalities into agents by simply visually assessing the data coupling techniques and providing guidelines and processes. This depends on both minimizing coupling and increasing cohesion. It appears that placing agents that can read or write the same type of information with each other reduces the association of agents. The results are shown in Table 5.

5. Comparative Analysis of Selected AOSE Methodologies

After comprehensive analysis and understanding of these methodologies involved in the study, we can now extract the strengths and weaknesses (limitations), as well as the scope of the application of these methodologies. This will provide a foundation towards the next step of proposal that drives the strengths (best) of these methodologies. Tables 1–3 provide inclusive analyses at a glance of selected methodologies in the study.

6. Proposed Methodology

We work towards the unification of the strengths of selected agent methodologies. We applied the assessment technique (structural analysis) of the seven competing AO methodologies, yielding in-depth comprehension as an outcome of their comparative analysis. We assessed their advantages and disadvantages (limitations), and their resemblance and variations are also investigated as to processes and models. Following this analysis, we take the first step to combine their features with the purpose to construct a core approach and combine characteristics selected from various approaches.

In this respect, some preliminary suggestions are introduced for the design of a relatively complete MAS methodology based on the selected methodologies. The purpose of this step is to contribute to the combination of the best features selected for agent development methodologies. Figure 8 illustrates the proposed approach processes.

As in ASEME, the MDE method appears with special features concerning the AOSE community and can be applied in all stages of conventional software processes. It allows the transition from one model and stage to another through typical transformations. As shown in Figure 9, with some exceptions, we will utilize the common and essential notations as presented in [69].

6.1. Requirement Specification. The first stage proposed is requirements specification. This stage is significant in software development processes, where it enables the system analysts to obtain the requirements of the target system. From our perspective, the success of any system development depends on the in-depth study of the system requirements.

6.1.1. Initial Requirements. This stage provides the initial step of the requirements analysis towards identifying basic stakeholders and equipping designers with useful knowledge about the environment in which software operates and the type of interactions to take place among system agents. The early requirements phase of the Tropos includes a study of the organizational preparation, including stakeholders, their objectives, and relations.

In ADELFE the purpose of this step is to realize a convention on the initial demands related to the characterization of the system and the surroundings in which the system will be implemented. It is created to determine what the most suitable system can be for end-users. The unified approach should have an initial requirements stage, which can be adopted from Tropos and ADELFE.

6.1.2. Definitive Requirements. The aim of the definitive requirements is to convert this view to a UC diagram. To arrange the requirements (functional or not) in this step, the developer has to determine the task of the comprehensive system and to model its environment. The following steps, in our view, introduce strong support for gathering requirements.

(1) Distinguish UC. This technique is used to characterize the different system functional requirements. It has been demonstrated to be an efficient model in object-oriented (OO)

TABLE 5: The summary of differences.

	Model-driven	Documentation	Deployment model	Data coupling
Gaia	X	X	X	X
Tropos	X	X	X	X
PASSI	X	X	√	X
O-MaSE	X	X	X	X
ADELFE	X	X	X	X
ASEME	√	√	X	X
Prometheus	X	X	X	√

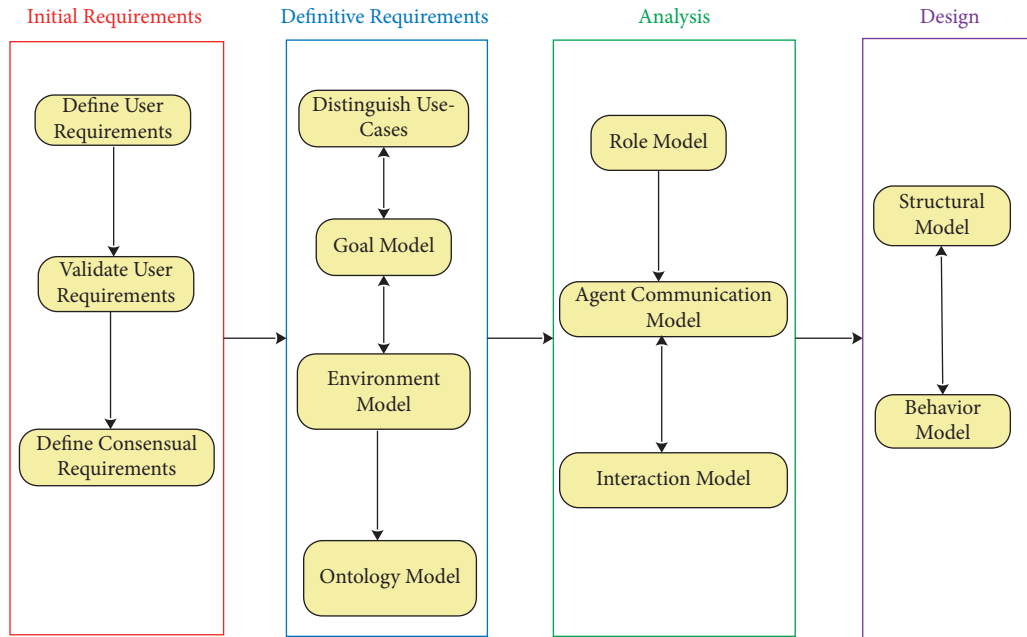


FIGURE 8: A proposed AOSE methodology.

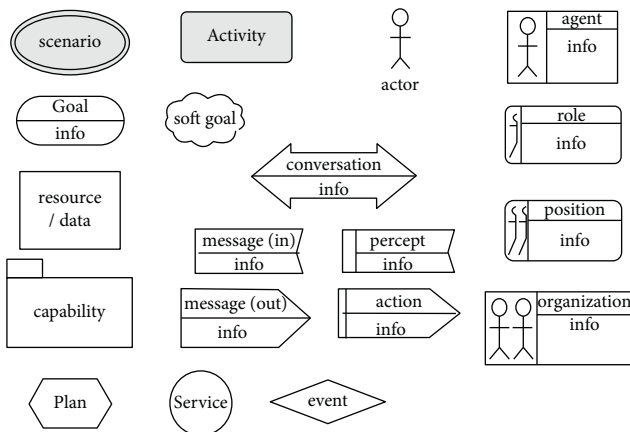


FIGURE 9: A unified notation set [69].

techniques in grouping system requirements. In addition, it assists the developers in deciding on the main interactions among system entities. Of the selected methodologies, ADELFE, PASSI, ASEME, and Prometheus have applied UC models. Particularly, Prometheus offers UCs, while PASSI proposes the use of UML-like UC diagrams, ADELFE utilizes this technique to clarify the linked sequence diagrams and to determine

collaboration failures. In these UC, only energetic components are implicit and appear as the products of an effective requirements group [49].

Exploring situation collaboration failure in the system and within its conditions is carried out so as to help developers in identifying problems and noncooperative items and incidents. In ASEME, some shifts in semantics are displayed. First, the actor interacts with the system and supposes a role. Then, agents are developed as roles either inside the system boundary, that is, for elements to be designed, or outside the system boundary, that is, for agents in the environment [35]. Similar to conventional UML UC models, human actors are demonstrated as roles outgoing the system boundary.

Later, the various UC should be linked to at least one agent role. The UC diagram in ASEME also adds three new ideas regarding the actor diagram and offers actors designed inside the system boundary. The diagram can include abstraction roles to ensure that agent IPs and goals are viewed from a developer standpoint by adding subgoals related to implementation in the shape of UC. To assist the analysts in defining the main connections/reactions among entities and adopt more with agent technology, the integrated methodology will combine the UC on ADELFE and ASEME approaches.

(2) *Goal Model*. It can be said that the agents' objectives are one of the most critical models of agency, which contribute to the strengthening of agents. Except for ADELFE and PASSI, most of the selected approaches concur on the significance of goal concepts and identify goals in the requirements analysis stage to be applied as a basis for identifying agents. Obtaining goals is one of the base processes in O-MaSE and is a significant modeling activity in Tropos and ASME. Gaia expresses goals in form of roles' responsibilities in a way that is more realistic than goals in other approaches.

Determining objectives is also a vital ingredient in Prometheus and presents such models as an essential part of the institution. The unified approach should support capturing objectives by defining goals and their architecture and representation. It is as well substantial to address stakeholders' purposes and their relations through the functions and resources utilized to fulfill goals as carried out in the requirements analysis stage in ASEME. Goals can be organized and presented as a hierarchy of objectives as applied in O-MaSE utilizing the suggested unified notation [69].

(3) *Environment Model*. The notion of the environment is basic for MAS since agents operate in an environment. To support complex open systems, agent methodologies need obvious diagrams to characterize the scope knowledge and the implementation environment. Complicated open systems generally have extremely dynamic and heterogeneous environments. By officially defining the environment, a knowledge base is created that constantly deals with environmental changes. As a result, an agent system needs to have models to represent the environment in which it operates.

Despite the significance of developing an environment model, only O-MaSE, ADELFE, and Prometheus manage to specify such models. In ADELFE, before identifying UC and during the final requirements, the environment should be thoroughly planned by the developer. Afterward, one procedure is added to the RUP to describe the system environment. Specification starts by determining which elements interact with the system as well as the restrictions in these connections.

The model offered by Prometheus is a view of the environment inside of actors, notions, and actions. The main components of the surrounding in which agents will work are shown by the domain diagram in O-MaSE. These components are described in the form of objects from the surrounding, which contain agents, and interactions among those objects. It can also be used to show the general characteristics of the environment to see how the objects connect. However, most competing approaches do not take into account this important matter thoroughly enough.

In fact, Gaia does not provide a comprehensive model of the implementation environment to the designers, and the environmental data is encoded in permissions and protocols for a specific role. This omission makes Gaia unsuitable for engineering implementations with effective and diverse environments. Tropos offers resources as an entity but no more [27]. At this stage of our proposed approach, the environment and the concepts and interactions should be taken into account. We may adopt Prometheus's analysis

overview model and characterize the environment activity in ADELFE to address this concept.

(4) *Ontology Model*. The ontology model provides the notions utilized by the agent system. Of the competing approaches, PASSI and O-MaSE offer the ontology diagram. Instead of this model, ADELFE adapts to the AMAS theory, which means that the agent is able to handle its environment and the other agents. PASSI has a domain ontology description model that involves notions as classes, elements of the domain, and predicates that emphasize characteristics of notions and tasks that agents can do in the scope. PASSI also has a connection ontology model that offers connection tracks among agent kinds. O-MaSE's domain model also represents the main elements. These are demonstrated within the scope of different types of objects that agents operate with. It also displays the interactions among those object kinds and among them and the agents. Object kinds are realized by a name and a group of features and are, then, utilized in other approach steps. Developing an ontology model has to be taken as a critical function in the activities of the proposed approach as the task includes specifying certain range notions, their properties, and relations. We suggest that the unified methodology utilizes PASSI's domain and communication ontology models to describe the ontology of the system scope in the best way due to its prominent advantage over the other methodologies in this regard.

6.2. *Analysis Phase*. The analysis stage intends to specify a comprehensive system's architecture and its conduct. These are obtained utilizing a role model as well as an interaction model. It seems to us that there are three major processes in the analysis stage: role model, agent communication, and interaction model. In respect of analysis activity, one critical stride is to determine the agents' roles. Based on the information captured from the requirements steps, the system analysts specify the number of roles functionalities existing in the system.

6.2.1. *Role Model*. Among the significant demands of AOSE approaches is to help designers distinguish the agents comprising the system; especially, agents are the main components in agent-based systems. A popular way applied in most selected approaches coping with agent role is to begin from the smallest elements of the agents and then group these elements to compose agents. In Prometheus, an agent kind is created by integrating one or more functionalities. Various sets of tasks present alternate designs that are assessed according to the coherence of the agent kinds and the range of connections among agents.

Tropos presents these elements as an ability. Gaia, O-MaSE, and ASEME specify the agent as a role. Agents are specified by gathering UC in PASSI. Tropos has extensive schemas, which offer relations among actors, objectives, resources, and functions in the system. As mentioned earlier, the depiction of ADELFE does not offer adequately elaborate

guidance to permit us to status whether it applied this technique or not.

6.2.2. Agent Communication Model. In Prometheus and Gaia, the dependency and interaction among agents are illustrated in terms of agent acquaintance schemes. While Gaia realizes the connection, links are found among agents without determining the current properties of those links. Prometheus indicates the types of agents as well as the connections among them. Agent acquaintance diagrams in these two approaches are intended to help developers determine potential bottlenecks formed among the system elements. The data coupling offers evident processes to cope with agent characterizations. One of them is the use of data coupling models. The second is the model of agent acquaintance.

These models are composed of system functions and external resources in terms of specified data. As noted earlier, using this technique, designers are able to assemble functionalities into agents by simply visually assessing the data coupling techniques and providing guidelines and processes. This depends on both minimizing coupling and increasing cohesion. It appears that placing agents together that can read or write the same type of information reduces the association of agents. These intermediate models are adopted in the unified approach.

6.2.3. Interaction Model. The prominence of this step lies in describing the agents in the system and their functions, responsibilities, and objectives. As mentioned in our structural analysis, ADELFE, PASSI, Prometheus, Tropos, and O-MaSE share certain features as they present high-level connections utilizing sequence/interaction models extracted from UML sequence models. Each methodology has different interaction diagrams; for example, Tropos and ADELFE depict conversations among agents. This is while the sequence models in O-MaSE show reactions among roles and actors. PASSI uses serial graphics to explore the tasks of each agent through role-specific [53].

The role model in ASEME is derived from the Gaia methodology since, in both cases, the conversations are explored and modeled at the role level instead of agent level, and serial/interactional schemas are not used for the objective [51]. Getting into more IP detail, ASEME, ADELFE, and Tropos suggest the application of the AUML IP model, with certain tasks added to the AIP to suit AMAS's demands in ADELFE. The interactions of the system can be obtained at a high-level utilizing AUML connection protocols for the proposed methodology.

6.3. Design Phase. This stage concentrates on determining agents' architecture and conduct via determining their elements and relations. It is quite necessary for AOSE approaches to offer a clear way to comprehend the general framework of the system. Not all current approaches present support for social structure and conduct (role, communication, and interactions). In O-MaSE, the process designer

needs to identify the specific set of stages and then identify the activities and tasks for each phase and reuse them again.

Since what has been said will be specific to every project underdeveloped, there are no strict bases on processes to be developed at any stage. It characterizes the microlevel of dynamics employing finite-state models. The design stage in ASEME encompasses the functional and behavioral sides of the MAS, and the related models are the agent IP and IAC that carry out a certain IP by assuming the crucial roles and relations between them. In Tropos, every agent's plan is depicted utilizing a UML process model.

As a result, in order to rethink beliefs and plans in an adaptive manner, it is difficult to think of a changing environment. Yet it proposes several potential notations that the developer could utilize to cover the agent's dynamic conduct. The service model in Gaia offers a description of the inputs, outputs, preconditions, and postconditions of each task offered by an agent. However, the models for showing the agent's organizing abilities are not depicted. ADELFE directs the designer to decide whether AMAS theory is ideal in the project being developed. This illustrates the significance of verifying the sufficiency of the analytical workflow and the sufficiency tool that analyzes the standards provided by the developer to determine whether this theory is beneficial.

If the system is not appropriate to AMAS technology, the developer could utilize another AO approach. Defining agent structures include determining their abilities and connections. The roles defined in the previous step can be useful in realizing such abilities. The conduct of agents can be getting by a set of potential current notations involving activity models, state charts, and finite state machines.

6.4. Implementation. Except for Gaia, All the participant methodologies provide backing tools that can produce code skeletons from design paradigms. ADELFE offers the analyst tools to evaluate the sufficiency of AMAS technology at two levels to demonstrate AIP. In this respect, the AUML principle is used together with UML and RUP. PASSI utilizes the UML deployment model and expands it with merits to permit the mobility of agents to be defined. For example, in the process of building the deployment model, developers can have a better grasp of the intricacies to operate the system. Also, developing high-scale deployment offers a basis for evaluating the feasibility of executing the system as well as the use of the spread concept. It also provides an assessment of different other measures, such as costs. Using this activity, PASSI describes the system based on agent classes and their position on the available processing units in the diagrams. O-MaSE also offers some limited backing for determining mobility processes in the case of concurrent functions. The execution phase in ASEME is the programming language for different levels throughout the development process.

Later, during the verification phase, the system functions are checked against the requirements, and the respective phase is completed in relation to three abstraction levels: societal, agent, and capability. Nevertheless, the preferable method is successive, that is, the software elements are checked for the effective running of algorithms, the agents are tested for the

effective execution of abilities, and the MAS is checked for its general valid procedures in the end, that is, all processes are carried out one after another. In general, this stage needs more improvement coinciding with the work on experimenting and addressing agent projects in recent years [15, 70].

7. Case Study

As mentioned previously, we do not aim to propose a definitive normative methodology, but rather a proposal draft, which we hope will lead to discussion in the relevant community, possibly resulting in aid towards a true standardized methodology for AOSE. To build an agent information project, we applied the health information system [49, 71] to the proposed methodology. The system is established to represent the hospital, family at home, and healthcare providers all in one integrated system.

7.1. Requirement Specification

7.1.1. Distinguishing UC. To assist the analysts in defining the main connections/associations among entities and adopt more with agent technology, the proposed methodology integrated the UC on ADELFE and ASEME approaches. Figure 10 shows an example of this combination by representing the UC for the example used in this study.

7.1.2. Environment Model. Figure 11 presents the general design of the system towards development of the agent overview scheme from the overall system model, which characterizes the four kinds of agents as specified and also displays the messages among them, that is, notions received by the user interface agent and the information read and written in this regard.

7.1.3. Goal Model. It is substantial to consider stakeholders' purposes and their interactions via the functions and resources utilized to fulfill goals. Goals can be organized and presented as a hierarchy of objectives, utilizing the suggested unified notation [69]. In the proposed methodology analysis phase, the significant step is in getting the objectives, which pick the initial system characterization and transform them into an arranged group of system objectives. It is shown in Figure 12.

7.2. Analysis Phase

7.2.1. Agent Role. A basic stage applied in our unified approach is coping with the agent role to begin from the smallest elements of the agents and then to group these elements to compose agents. Figure 13 displays obvious the roles and their goals according to the goal diagram in the previous phase. The goals linked with each role are represented under the role name.

7.2.2. Interaction Model. The AUML notation is used, but some specific functionality has been added to AIP diagrams to fit the AMAS theory requirements. Figure 14 displays the

interaction protocol for the system under study that uses serial graphics to explore the tasks of each agent through role-specific.

7.3. Design Phase. Depending on the analysis phase, Figure 15 shows a role model for the target system with more details. The lines among the roles in this model refer to possible connection paths. It also contains information on the interactions among role tasks. The goals related to each role are represented under the role name. Additionally, illustrated are the tasks related to each role, used to determine each role's behavior.

We adopted a case study approach in the form of feature analysis to develop the healthcare provider system, using the proposed approach. The goal of this study is to explore the potentials of the approach in presenting solutions for system problems. The system constructs software agents representing the hospital, the family members at home, the patient being monitored personal device assistant (PDA), and the healthcare providers.

The proposed approach is clearly agent-based and considers the social aspects. We applied the goal model covering the entire goals as classified into subgoals. It consists of a role hierarchy and a set of role schemas for each role in the hierarchy. The roles define the expected behaviors of the agents and provide services and goals to any agent. The role model allows us to build an abstract view of the system according to which its member agents behave in the system.

8. Discussion

The notation of the seven approaches is acceptable, and most have a modeling language in terms of fulfilling different aspects such as systematic transitions, modularity, and ease of comprehension. However, none of the approaches clearly offers mechanisms or models to back the design of reusable components. In addition, only ASEME provides techniques and adequate support for transition from one stage to another through typical transformations by three forms of conversion used for automation between the ASEME models. However, there are numerous aspects of agent-based systems that were not sufficiently handled in most of the approaches. For example, except ASEME, none of the seven approaches provides backing registration of non-functional requirements, in the requirements analysis step.

Regarding the environment percepts feature, ADELFE, Prometheus, and O-MaSE provide support for modeling the environment of agents. As to architecture, all seven methodologies provide support with different degrees to agent architecture, while Prometheus introduces distinctive backing for belief, goal, and plan; the other methodologies seem to be weak in this respect. Regarding the process lifecycle, all of the approaches reference the specification, analysis, design, and detailed design to a certain degree. Additionally, all of them offer instances and heuristics to help designers in all methodology stages. Tropos and ADELFE make use of initial and final requirements, which is an important stage in these approaches. Moreover, they suppose full specification of requirements and do treat the grouping of requirements. This

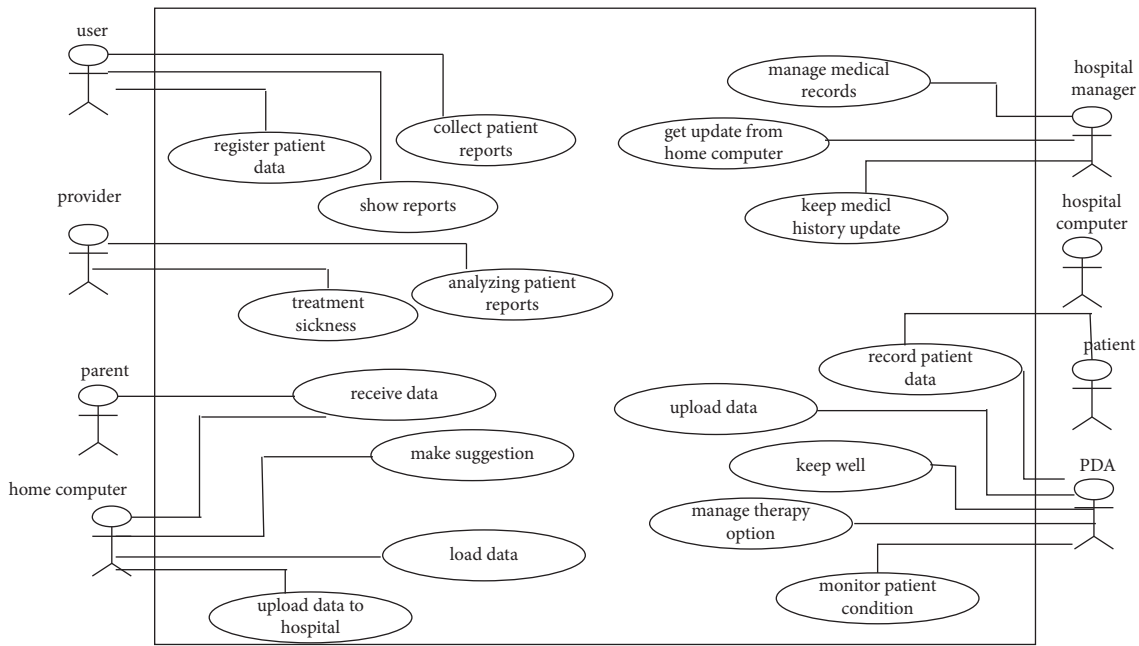


FIGURE 10: UC diagram.

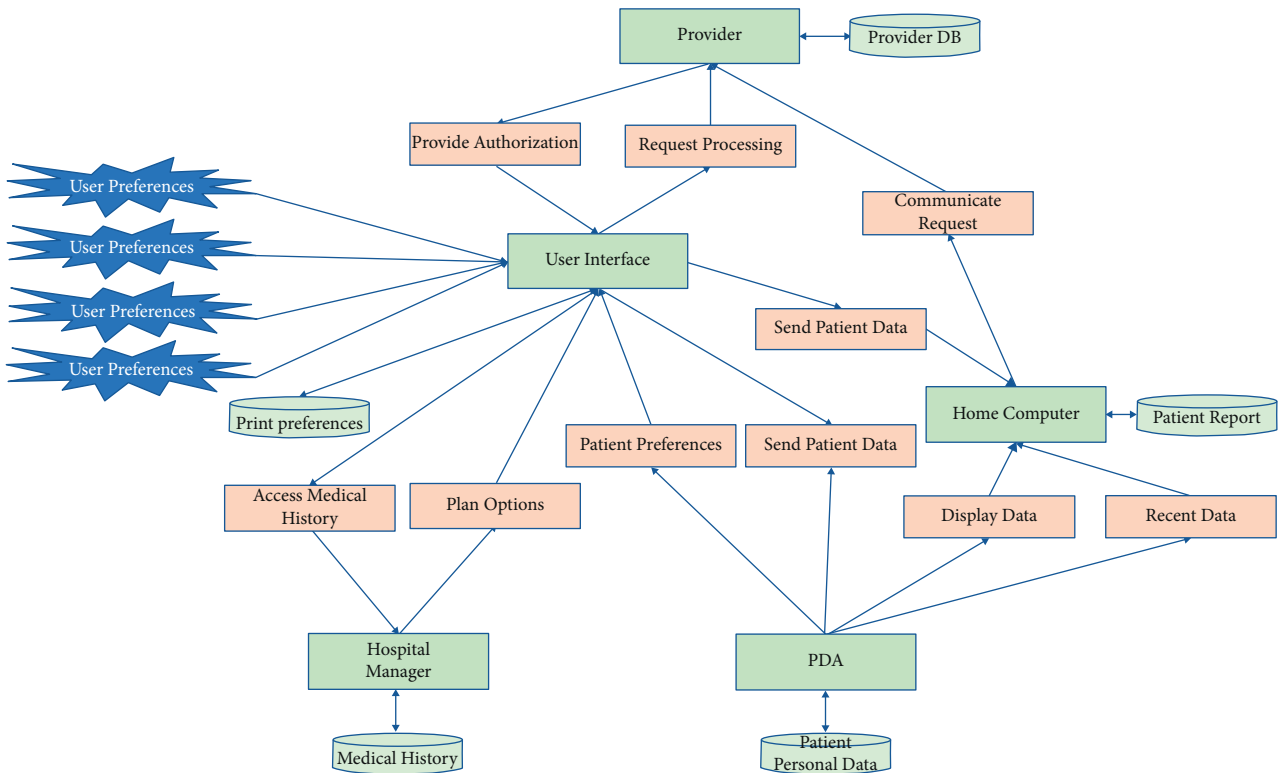


FIGURE 11: System overview diagram.

guides designers to the benefit of richer requirements produced by agent technologies. The seven methodologies also share several common features; for example, Prometheus and Gaia share the use of the acquaintance technique. In addition, all of these methodologies provide models to cover the social system structure.

In an environment that is generally constricted, the current agents and MASs provide solutions for complex applications. However, the present and future applications are convoluted and open to criticism. Like the Internet, these develop in an unpredictable environment and present other challenges to constructing software.

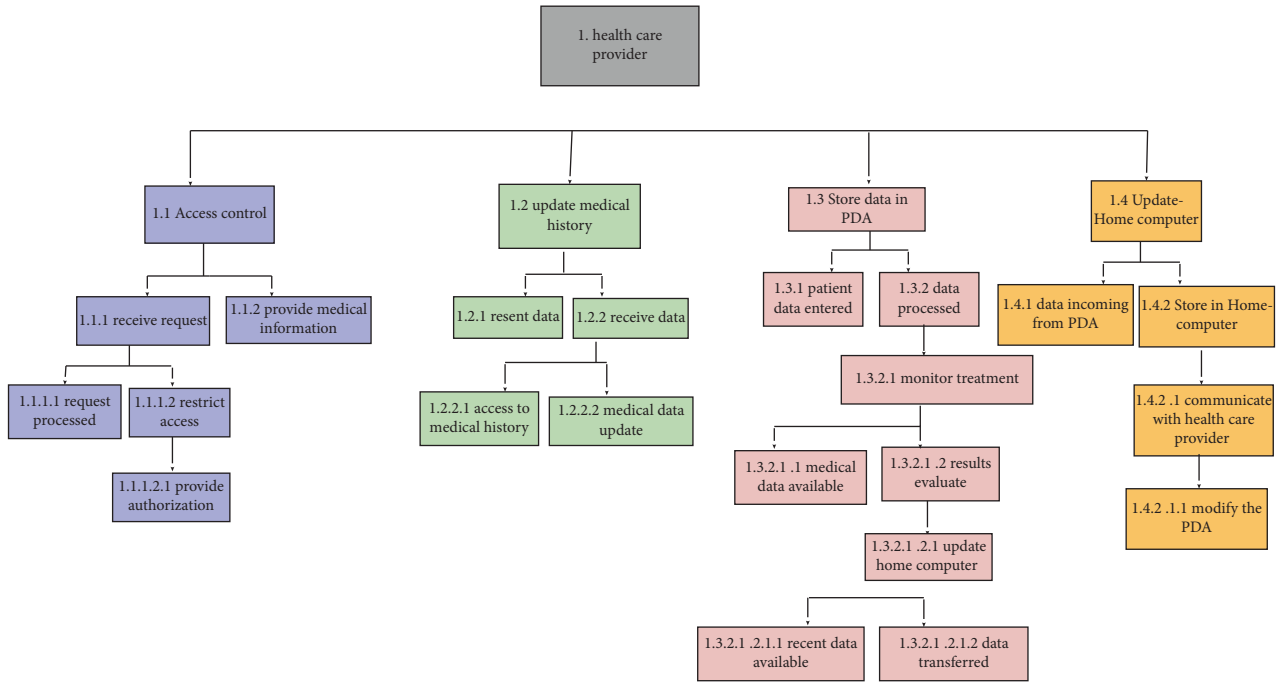


FIGURE 12: Goal diagram.

Role	Provider	Hospital	Home Compute	PDA
Goals	1.1,1.1.1, 1.1.2 1.1.1.2, 1.1.1.1 1.1.1.2.1	1.2,1.2.1, 1.2.2 1.2.2.2, 1.2.2.1	1.4,1.4.1, 1.4.2 1.4.2.1, 1.4.2.1.1	1.3,1.3.1, 1.3.2, 1.3.2.1 1.3.2.1.1, 1.3.2.1.2

FIGURE 13: Roles and their goals.

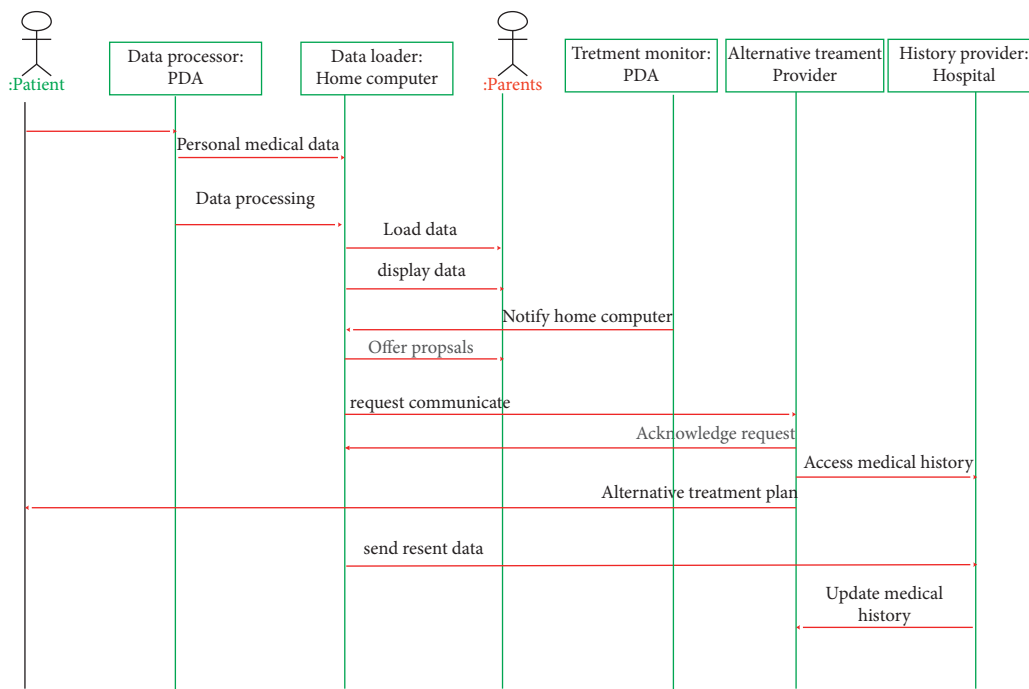


FIGURE 14: Interaction model.

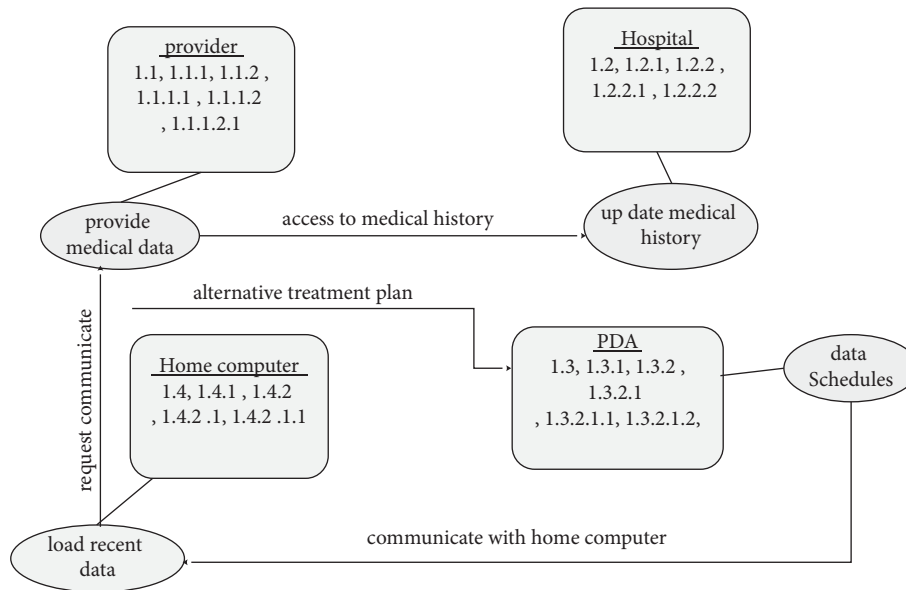


FIGURE 15: Role model.

As a result, it is significant to offer novel paradigms, tools, and approaches to meet these challenges. Many of the selected methodologies are picked according to “standards,” like the RUP, UML, and UML models, to improve AO technologies in the manufacturing world, where OO technology is the base. In terms of strengths, weaknesses, and application domains of seven methodologies, Tables 3–5 describe the results of this analysis study. Gaia is a general approach that is usable to a wide domain of MAS; furthermore, it addresses the social agent aspect of systems [37].

Gaia does not directly deal with special modeling technicalities. Also, it does not include an implementation stage. As we mentioned the difficulties are spotted when working on some design schemas and role models, with the underlying reason being that performing elaborated role models needs determining the liveness responsibilities of each role, decisive in plotting the protocols later. In addition, the authorizations and safety responsibilities mentioned in Gaia are not clear. As an outcome, in the paradigm of the role set up for the target system, this feature of roles is not well-described. Regarding the notation, which characterizes these roles’ characteristics, liveness responsibilities are rationally well-identified, while liability is not submitted [60].

As for Tropos, it was affected by the framework analysis of the initial demands of the intended system. It directs designers towards understanding an AO system as organizing of actors through plans that are dependent on other actors, which are seeking to accomplish goals. The Tropos methodology aims to cover all analyses and design processes in the software development lifecycle, starting with the analysis stage to system implementation.

It provides the early phase of the requirements analysis with identifying basic stakeholders and equips the designers with a satisfactory knowledge about the environment in which the software operates and the type of interactions that

should happen among system agents. In the requirements analysis, the main supposition that distinguishes Tropos from other methodologies is that actors and objectives are utilized as basic notions for modeling and analysis during all software lifecycle stages, not just during early requirements.

It concentrates on the preliminary stage of requirements engineering. The discovery of stakeholders and credits among them lies in the knowledge of the engineer. Based on various perspectives that could lead to different results, the achievement of objectives and their relations among stakeholders can be derived. The system actors are added in the midst of the analysis. To accommodate the new actor, the dependencies among the actors are rearranged. Throughout the entire development process, Tropos is based on the unified use of small groups of intentional symbols [52].

Nevertheless, in order to rethink beliefs and plans in an adaptive manner, it is difficult to think of a changing environment. O-MaSE is designed as a set of parts that developers combine to meet specific goals. In fact, they do not need a particular set of stages. They assume that they follow a traditional waterfall approach to allay this problem. Requirements analysis, design, and implementation are the three main phases with the main activities allocated as expected. It is the beginning point in O-MaSE [72]. In order to design MASs, there are several drawbacks. For example, O-MaSE presents MASs with a specific organization. The number of agents and their roles in O-MaSE is limited. Also, O-MaSE does not cover the concept of subteams and has no diagrams that represent interactions with the environment. In addition, there is the only one-to-one connection among agents in the system, and it does not clearly define the usage case model [23]. While many recurring problems have been processed in O-MaSE, there are some necessary missions for a mature agent approach such as management, product distribution, testing, and assessment, which are absent.

In PASSI, an agent is an essential part of the software at both low- and high-precision levels. The agent is an example of software execution of an independent element able to achieve a goal via its independent resolutions, activities, and social interactions based on this view [32]. While interacting with other agents to achieve their objectives, the agent can play many useful roles representing a set of functions carried out by the agent in pursuit of a subgoal. As a meaningful unit of individual or interactive behavior, the task is defined in PASSI. Distinguishing features also exist, such as in PASSI, from other participant methodologies. In the case of PASSI, this lies in its strong focus on the deployment model, which is regarded as one of the main models in UML.

Using this activity, PASSI describes the system based on agent classes and their position on the available processing units in the diagrams. ADELFE directs the designer in making the resolution as to if AMAS theory is desired in the project under development. This illustrates the necessity of verifying the sufficiency of the analytical workflow and the sufficiency tool that analyzes the standards provided by the developer to determine whether this technology is beneficial. It provides certain actions and standards to assist the developer to determine elements in the system that need execution as agents [49]. Their characteristics should be studied in addition to the relations and the cooperation failures that may determine if entities should be regarded as agents. This can be reutilized in other agent approaches, and the ADELFE processes can then be analyzed in the form of fragments.

It will also be easier to combine the parts from other approaches into ADELFE [68]. ADELFE's main strong point can also be a major constraint; it is highly specialized and cannot be used to develop all kinds of agents (e.g., BDI). As it permits the designer only to test the behavior of certain agents and verify them based on specifications, there is a need to improve many activities, especially fast prototyping [68]. To improve agent behavior by inserting or deleting portions of it, the designer will be able to react with the system as it is designed. Many work definitions still do not exist. Currently, for direct execution and testing, there is no running tool like the platform or a group of programming tools associated with this approach [68].

As mentioned earlier, ASEME, the model-driven engineering (MDE) method, appears with special features concerning the AOSE community and can be applied in all stages of conventional software processes (from requirements to execution). It allows the transition from one stage to another through typical transformations. In addition, it backs the registration of nonfunctional requirements in the requirements analysis step, where they are utilized to make management decisions and choose which technologies will be used for design and development.

Prometheus suggests complete lifecycle processes, from requirements specification to elaborated design, and supports the agent developer according to goals and tasks. It offers elaborated guidance on how to carry out the different stages. Prometheus has an environmental paradigm that describes the surroundings where agents work and provide support for this concept compared to other competitive

approaches. However, the Prometheus environment model is limited to clear input and output specifications with respect to the distinct needs and requirements of the system.

In this methodology, the formation of the kinds of agents can be oriented to data association. In addition, the agents are composed of other component roles and abilities. In addition, for individual agents, Prometheus supports both dynamic and static models [73].

Additionally, an important aspect, which distinguishes Prometheus from other participant methodologies, is data coupling as it offers evident processes to cope with agent characterizations. Developers can get the information that agents draw from the environment and the actions that the agents begin to respond to in these events using the system specification stage in Prometheus. Prometheus does not handle mobile agents at all and has less focus on initial requirements and business processes analysis than approaches, such as ADELFE and Tropos. Lastly, Prometheus is not established on UML.

9. Conclusion, Limitations, and Future Work

The main purpose of agent methodologies is to deliver the required processes, models, mechanisms, and tools so that agent-based systems can be developed in a more formal and organized manner. As pointed out previously, many AOSE methodologies with various backgrounds had been made available to experts in recent years. This study conducted an analysis and evaluation of key AOSE methodologies, focusing on their features and limitations and proposed unified AO methodology. With the increasing necessity for complex systems in the industry, the intention to employ agent technologies to promote commercial and industrial software systems is rising rapidly as well, moving the industry from a form of product for a user to a manner of delegation, where it can also involve the users in order to make decisions and take actions accordingly. Thus, the availability of the AO approach to support software engineers in designing agent-based projects is very significant.

To this end, the present research study carried out a comparison of seven prominent AO approaches to comprehend the interactions among them. Specifically, our major objectives observed were: first, to determine the resemblance and variations in terms of processes and models that were paramount in directing the development of agent-based projects and, second, to estimate each methodology's features, limitations, and range of usability.

However, assessment of these methodologies had continuously faced several difficulties; the completeness of various methodologies varies explicitly; and each methodology had different advantages and drawbacks with numerous individually oriented features to support various aspects of their proposed application scopes.

In our view, all of the seven AO methodologies were clearly agent-based; the structural is widely regarded as an agent methodology; and most of them consider the social aspects.

Generally, all these methodology approaches provided sufficient support for major AO properties. The most social

characteristics addressed in all offer support for most social standards as communication, communication language, and relationship. The assessment of the AOSE methodologies showed that most amount of attention had been paid to requirements, design, and implementation stages. However, progress is still needed in all stages of the software processes.

Based on this comparative analysis, we proposed an exploratory unification of the assessment AO approaches by integrating their strong characteristics. With regard to future work, it includes increasing the number of AO methodologies and adding other evaluation methods to the assessment. In doing so, the work can be improved by involving supplementary models and techniques. In addition, this paper had considered the commonalities and differences between participant methodologies in terms of process and models without the detailed techniques that are an important issue for future work.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors would like to thank the Molde University College—Specialized University in Logistics, Norway, for the support of the open access fund.

References

- [1] W. Mefteh, F. Migeon, M. P. Gleizes, and F. Gargouri, "ADELFE 3.0 design, building adaptive multi agent systems based on simulation a case study," in *Computational Collective Intelligence (ICCCI 2015), PT I Book Series: Lecture Notes in Artificial Intelligence*, P. Muller, Ed., vol. 9329, pp. 19–28, Instant UML, Wrox Press, Birmingham, UK, 2015.
- [2] R. Padmanaban, M. Thirumaran, K. Suganya, and R. V. Priya, "Aose methodologies and comparison of object oriented and agent-oriented software testing," in *Proceedings of the International Conference on Informatics and Analytics*, pp. 1–16, ACM, Pondicherry India, August 2016.
- [3] H. Mubarak, "Developing flexible software using agent-oriented software engineering," *IEEE Software*, vol. 25, no. 5, pp. 12–15, 2008.
- [4] O. Robert, P. Piotr, T. Agnieszka, K. Bogna, P. Tadeusz, and B. Marek, "Spatiotemporal modeling of the smart city Residents' activity with multi-agent systems," *Applied Sciences*, vol. 9, p. 2059, 2019.
- [5] B. Lorica, "How to think about AI and machine learning technologies, and their roles in automation: an overview and framework, including tools that can be used to enable automation," 2018, <https://www.oreilly.com/ideas/how-to-think-about-ai-and-machine-learning-technologies-and-their-roles-in-automation>.
- [6] A. Dey, A. Pal, and H. Long, "Fuzzy minimum spanning tree with interval type 2 fuzzy arc length: formulation and a new genetic algorithm," *Soft Computing*, vol. 24, no. 6, pp. 3963–3974, 2020.
- [7] A. Dey, A. Pal, and T. Pal, "Interval type 2 fuzzy set in fuzzy shortest path problem," *Mathematics*, vol. 4, no. 4, p. 62, 2016.
- [8] G. Fortino, W. Russo, C. Savaglio, W. Shen, and M. Zhou, "Agent-oriented cooperative smart objects: from IoT system design to implementation," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 11, pp. 1939–1956, 2017.
- [9] C. Savaglio, M. Ganzha, M. Paprzycki, C. Bădică, M. Ivanović, and G. Fortino, "Agent-based Internet of things: state-of-the-art and research challenges," *Future Generation Computer Systems*, vol. 102, pp. 1038–1053, 2020.
- [10] S. Abar, G. K. Theodoropoulos, P. Lemarinier, and G. M. P. O'Hare, "Agent based modelling and simulation tools: a review of the state-of-art software," *Computer Science Review*, vol. 24, pp. 13–33, 2017.
- [11] C. Skourlas, P. Belsis, S. Gritzalis, C. Lambrinouidakis, V. Tsoukalas, and D. Vassis, "An agent based architecture benchmark," *Procedia-Social and Behavioral Sciences*, vol. 147, pp. 429–435, 2014.
- [12] B. Sherrell, J. Clemens, and R. Pal, "Runtime state verification on resource-constrained platforms," in *Proceedings of the MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM)*, pp. 1–6, IEEE, Los Angeles, CA, USA, October 2018.
- [13] C. Savaglio, T. Leppänen, W. Russo, J. Riekkki, and G. Fortino, "Re-engineering IoT systems through ACOSO-meth: the IETF CoRE based agent framework case study," in *WOA*, pp. 81–89, 2018.
- [14] C. E. Lin, K. M. Kavi, F. T. Sheldon, K. M. Daley, and R. K. Abercrombie, "A methodology to evaluate agent-oriented software engineering techniques," in *Proceedings of the 2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07)*, p. 60, IEEE, Waikoloa, HI, USA, January 2007.
- [15] J. J. Gómez-Sanz and R. Fuentes-Fernández, "Understanding agent-oriented software engineering methodologies," *The Knowledge Engineering Review*, vol. 30, no. 4, pp. 375–393, 2015.
- [16] M. Habiba, "Metrics for evaluating agent-oriented software engineering model," in *Proceedings of the 2012 International Conference on Informatics, Electronics & Vision (ICIEV)*, pp. 17–22, IEEE, Dhaka, Bangladesh, May 2012.
- [17] C. Lucena and I. Nunes, "Contributions to the emergence and consolidation of agent-oriented software engineering," *Journal of Systems and Software*, vol. 86, no. 4, pp. 890–904, 2013.
- [18] X. Mao, Q. Wang, and S. Yang, "A survey of agent-oriented programming from software engineering perspective," *Web Intelligence*, vol. 15, no. 2, pp. 143–163, 2017.
- [19] A. Taweel, E. Garcia, S. Miles, and M. Luck, "Agent-oriented software engineering of distributed eHealth systems," in *Proceedings of the OTM Confederated International Conferences On the Move to Meaningful Internet Systems*, pp. 332–341, Springer, Graz, Austria, September 2013.
- [20] L. Salazar and H. Li, "Proportional reliability of agent-oriented software engineering for the application of cyber physical production systems," in *Service Orientation in Holonic and Multi-Agent Manufacturing*, pp. 139–156, Springer, Cham, Switzerland, 2018.
- [21] R. Cunha, C. Billa, and D. Adamatti, "Development of a graphical tool to integrate the Prometheus AEOLus methodology and jason platform," *International Journal of Natural Computing Research*, vol. 6, 2017.

- [22] K. Slhoub, M. Carvalho, and F. Nembhard, "Evaluation and comparison of agent-oriented methodologies: a software engineering viewpoint," in *Proceedings of the 2019 IEEE International Systems Conference (SysCon)*, April 2019.
- [23] S. Sukhvir and S. Richa, "Evaluation of agent oriented software engineering (AOSE) methodologies-A review," *International Journal of Latest Research in Science and Technology*, vol. 1, no. 2, pp. 94–97, 2012.
- [24] M. Elammari and R. Elsaeti, "Structural analysis of agent oriented methodologies," *Proceedings of International Journal of Information & Computation Technology*, vol. 4, no. 6, pp. 613–618, 2014.
- [25] R. Abdalla and Mishra, "Comparing the artifacts of agent methodologies," *TEM Journal*, vol. 7, no. 2, pp. 433–438, 2018.
- [26] O. Zohreh, "A survey of agent- oriented software engineering paradigm: towards its industrial acceptance," *Journal of Computer Engineering Research*, vol. 1, no. 2, pp. 14–28, 2010.
- [27] K. H. Dam and M. Winikoff, "Comparing agent-oriented methodologies," in *Agent-Oriented Information Systems, AOIS, in: Lecture Notes in Computer Science*, P. Giorgini, B. Henderson-Sellers, and M. Winikoff, Eds., vol. 3030, pp. 78–93, Springer, New York, NY, USA, 2004.
- [28] H. Dam and M. Winikoff, "Towards a next-generation AOSE methodology," *Science of Computer Programming*, vol. 78, pp. 684–694, 2013.
- [29] D. Law, "Methods for comparing methods: techniques in software development," in *Multi-Agent Systems the Knowledge Engineering Review*, vol. 30, pp. 394–418, , no. 4, Cambridge University Press, Cambridge, UK, 1988.
- [30] Y. Wautelet, Y. Achbany, J. Lange, and M Kolp, "A process for developing adaptable and open service systems: application in supply chain management," in *Enterprise Information Systems-BK. Lecture Notes in Business Information Processing*, vol. 24, pp. 564–576, Springer, Berlin, Germany, 2009.
- [31] P. Giorgini, M. Kolp, J. Mylopoulos, and J. Castro, "Tropos: a requirements-driven methodology for agent-oriented software," in *Agent-Oriented Methodologies*, B. Henderson-Sellers and P. Giorgini, Eds., Idea Group Publishing, Hershey, PA, USA, pp. 20–45, 2005.
- [32] M. Cossentino, "From requirements to code with the PASSI methodology," in *Agent-Oriented Methodologies IGI Global*, Hershey, PA, USA, 2005.
- [33] S. DeLoach and J. Garcia-Ojeda, "The O-MaSE Methodology," in *Chapter from Book Handbook on Agent-Oriented Design Processes*, pp. 253–285, Springer, Berlin, Germany, 2014.
- [34] C. M.-P. Bernon, G. Gleizes, and P. Glize, "The adelfe methodology for an intranet system design," in *Proceedings of the Fourth International Bi-Conference Workshop on Agent-Oriented Information Systems at CAiSE*, Springer-Verlag, Toronto, Canada, January 2002.
- [35] N. Spanoudakis and P. Moraitis, "Using ASEME methodology for model-driven agent systems development," in *LNCS, 6788, 106–127*, X. I. AOSE, D. Weyns, and M. P. Gleizes, Eds., Springer, Berlin, Germany, 2011.
- [36] F. Zambonelli, N. Jennings, and M. Wooldridge, "Multi-agent systems as computational organizations: the Gaia methodology," in *Agent-Oriented Methodologies*, B. Henderson-Sellers and P. Giorgini, Eds., Idea Group Publishing, Hershey, PA, USA, pp. 136–171, 2005.
- [37] M. Wooldridge, N. Jennings, and D. Kinny, "The Gaia methodology for agent- oriented analysis and design," *Autonomous Agents and Multi-Agent Systems*, vol. 3, no. 3, 2000.
- [38] 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.
- [39] L. Penserini, P. Bresciani, T. Kuflik, and P. Busetta, "Using tropos to model agent-based architectures for adaptive systems. a case study in ambient intelligence," in *Proceedings of the IEEE International Conference on Software - Science, Technology and Engineering*, pp. 37–46, IEEE, Herzlia, Israel, February 2005.
- [40] J. Mylopoulos, J. Castro, and M. Kolp, "Tropos: toward agent-oriented information systems engineering," in *Proceedings of the Second International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS2000)*, Springer, Stockholm, Sweden, June 2000.
- [41] F. Giunchiglia, J. Mylopoulos, and A. Perini, "The tropos software development methodology: processes, models and diagrams," in *Proceedings of the Third International Workshop on Agent-Oriented Software Engineering*, Springer, Bologna, Italy, July 2002.
- [42] S. DeLoach and J. Valenzuela, "An agent-environment interaction model," in *AOSE VII/AOSE 2006, LNCS, L. Padgham and F. Zambonelli, Eds., Vol. 4405*, Springer, Berlin, Germany, 2007.
- [43] D. Capera, G. Picard, M. P. Gleizes, and P. Glize, "A sample application of ADELFE focusing on analysis and design the mechanical synthesis problem. Engineering Societies," in *The Agents World Vbook Series: Lecture Notes in Artificial Intelligence*, vol. 3451, pp. 231–244, Springer, Berlin, Heidelberg, 2005.
- [44] J. Odell, H. Parunak, and B. Bauer, "Representing agent interaction protocols in UML," in *Proceedings of the 1st International Workshop on Agent-Oriented Software Engineering (AOSE 2000)*, pp. 121–140, Springer-Verlag, Limerick, Ireland, January 2000.
- [45] L. Padgham and M. Winikoff, "Prometheus: a practical agent-oriented methodology," in *Agent-Oriented Methodologies*, B. Henderson-Sellers and P. Giorgini, Eds., Idea Group Publishing, Hershey, PA, USA, pp. 107–135, 2005.
- [46] R. Abdalla and A. Mishra, "Application of agent methodology in healthcare information systems," *TEM Journal -Technology Education Management Informatics*, vol. 6, no. 1, pp. 147–152, 2017.
- [47] R. Abdalla and A. Mishra, "Using agent-based methodologies in healthcare information systems," *Cybernetics and Information Technologies*, vol. 18, no. 2, pp. 123–132, 2018.
- [48] E. Yu, "Modelling strategic relationships for process reengineering," Ph.D. thesis, University of Toronto, Department of Computer Science, Toronto, Canada, 1995.
- [49] K. Horvath, P. Sengstack, F. Opelka et al., "A vision for a person-centered health information system," *National academy of medicine*, 2018.
- [50] G. Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language User Guide*, Addison-Wesley, Boston, MA, USA, 1999.
- [51] N. Spanoudakis, "A method fragment for transforming Gaia or ASEME liveness formulas to BPMN models for simulation," in *Proceedings of the International Workshop on Engineering Multi-Agent Systems*, Springer, Turkey, May 2011.
- [52] A. Perini and A. Susi, "Developing a decision support system for integrated production," *Environmental Modelling & Software*, vol. 19, no. 9, pp. 821–829, 2004.
- [53] C. Giacomo, L. Letizia, and P. Mariachiarra, "Service-oriented agent methodologies," in *Proceedings of the 16th IEEE International Workshops on Enabling Technologies:*

- Infrastructure for Collaborative Enterprises (WETICE 2007)*, IEEE, Evry, France, June 2007.
- [54] N. Spanoudakis and P. Moraitis, "The agent systems methodology (ASEME): a preliminary report," in *The Agent Systems Engineering Methodology (ASEME)*, N. Spanoudakis, Ed., Paris Descartes University, Paris, France, 2009.
- [55] S. Mariani and A. Omicini, "Special issue "multi-agent systems"," *Applied Sciences*, vol. 10, p. 5329, 2020.
- [56] M. Dastani, "Programming multi-agent systems," *The Knowledge Engineering Review*, vol. 30, no. 4, pp. 394–418, 2015.
- [57] D. Harel and H. Kugler, "The rhapsody semantics of statecharts (or, on the executable core of the UML) - preliminary version," in *INT 2004. LNCS*, H. Ehrig, W. Damm, J. Desel et al., Eds., vol. 3147, pp. 325–354, Springer, Berlin, Germany, 2004.
- [58] I. Nunes, U. Kulesza, C. Nunes, C. deLucena, and E. Cirilo, "A domain analysis approach for multi-agent systems product lines," in *Enterprise Information Systems-BKBook Series: Lecture Notes in Business Information Processing* vol. 24, p. 716, Springer, Berlin, Germany, 2009.
- [59] L. V. Massawe, F. Aghdasi, and J. Kinyua, "The development of a multi-agent-based middleware for RFID asset management system using the PASSI methodology," vol. 1-3, pp. 1042–1048, in *Proceedings of the IEEE Computer Society Proceedings OF The 2009 Sixth International Conference ON Information Technology: New Generations*, vol. 1-3, pp. 1042–1048, IEEE, Las Vegas, NV, USA, April 2009.
- [60] M. Cox, B. Kerkez, C. Srinivas, G. Edwin, and W. Archer, "Toward agent-based mixed-initiative interfaces," in *Proceedings of the 2000 International Conference on Artificial Intelligence*, CSREA Press, Sydney, Australia, 2000.
- [61] S. DeLoach and J. Garcia-Ojeda, "O-MaSE: a customizable approach to developing multi-agent development processes," *International Journal of Agent-Oriented Software Engineering*, vol. 4, pp. 244–280, 2010.
- [62] I. Garcia-Magarino, S. Rougemaille, R. F. Fernandez, F. Migeon, M. P. Gleizes, and J. Gomez-Sanz, "A tool for generating model transformations by-example in multi-agent systems," in *Proceedings of the The International Conference ON Practical Applications OF Agents and Multi-Agent Systems (PAAMS 2009)*, vol. 55, University of Salamanca, Salamanca, Spain, April 2009.
- [63] I. Mathieson, S. Dance, L. Padgham, M. Gorman, and M. Winikoff, "An open meteorological alerting system: issues and solutions," in *Proceedings of the 27th Australasian Computer Science Conference*, V. Estivill-Castro, Ed., Australian Computer Society, Inc., Dunedin, New Zealand, pp. 351–358, January 2004.
- [64] L. Padgham, J. Thangarajah, and M. Winikoff, "Tool support for agent development using the Prometheus methodology," in *Proceedings of the Fifth International Conference on 23 January*, IEEE, Melbourne, Australia, September 2005.
- [65] M. Gascueña, Fernández, and A. Caballero, "Agent-oriented modeling and development of a person-following mobile robot," *Expert Systems with Applications*, vol. 38, no. 4, pp. 4280–4429, 2011.
- [66] F. El Hajj, A. El Hajj, and R. A. Chehade, "Multi-agent system vulnerability detector for a secured E-learning environment," in *Proceedings of the IEEE2016 SiXTH International Conference On Digital Information Processing and Communications*, pp. 113–118, IEEE, Beirut, Lebanon, April 2016.
- [67] F. Saminni and W. Tang, "Implementation of Gaia methodology for multi-agent based transformer condition monitoring," in *Proceedings of the IEEE PES Innovative Smart Grid Technologies Conference Europe*, IEEE, Berlin, Germany, October 2012.
- [68] B. Henderson-Sellers and P. Giorgini, *Agent Oriented Methodology*, Idea Group Publishing, Hershey, PA, USA, 2005.
- [69] L. Padgham, M. Winikoff, S. DeLoach, and M. Cossentino, "A unified graphical notation for AOSE," in *Proceedings of the Ninth International Workshop on Agent Oriented Software Engineering*, M. Luck and J. J. Gomez-Sanz, Eds., Springer, Estoril, Portugal, pp. 61–72, May 2008.
- [70] C. Nguyen, A. Perini, and P. Tonella, "Experimental evaluation of ontology-based test generation for multi-agent systems," in *Proceedings of the Ninth International Workshop on Agent-Oriented Software Engineering, AOSE*, J. J. Gomez-Sanz, M. ., and Luck, Eds., Springer, Estoril, Portugal, pp. 165–176, May 2008.
- [71] N. Ericsson, "Technical description," *Integrated Health Care Information System*, 2004.
- [72] S. DeLoach and M. Wood, "Multiagent systems engineering: the analysis phase," 2000. Technical Report, Air Force Institute of Technology, AFIT/EN-TR-00-02.
- [73] A. Bawa, S. Bhatia, and V. Kaur Attri, "A review on agent oriented software ENGINEERING," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, no. 4, 2015.