On the use of the analytic hierarchy process in the evaluation of domain-specific modeling languages for multi-agent systems

Tansu Zafer Asici, Baris Tekin Tezel, Geylani Kardas

PII: S2590-1184(20)30080-0
DOI: https://doi.org/10.1016/j.cola.2020.101020
Reference: COLA 101020

To appear in: *Journal of Computer Languages*

Received date : 8 September 2020
Revised date : 12 November 2020
Accepted date : 29 November 2020

On the use of the analytic hierarchy process in the evaluation of domain-specific modeling languages for multi-agent systems

Tansu Zafer Asici[a], Baris Tekin Tezel[a,b], Geylani Kardas[a]

[a]*International Computer Institute, Ege University, Izmir-Turkey*
[b]*Department of Computer Science, Dokuz Eylul University, Izmir-Turkey*

Abstract

Software agents and Multi-agent Systems (MAS) composed by these agents are used in the development of the complex intelligent systems. In order to facilitate MAS software development, various domain-specific modeling languages (DSMLs) exist. Unfortunately, the usability evaluation of these languages are mostly not considered or only a few assessments which cover one single MAS DSML are made. A comparative evaluation, which is missing in the existing studies, may help agent software developers to choose the MAS DSML which fits well into the system development requirements. Hence, in this paper, we introduce a comparative MAS DSML evaluation methodology based on the Analytical Hierarchy Process (AHP). A categorized set of MAS DSML criteria which can be used for the multi-criteria decision making is defined. These criteria can be prioritized by the developers according to their modeling language expectations and the application of the methodology allows the evaluation of DSML alternatives based on this prioritization. As the result of the automatic calculation of the importance distributions, the most appropriate DSML is determined. With the voluntarily participation of a group of agent software developers, the proposed methodology was applied for the comparative evaluation of four well-known MAS DSMLs. The conducted evaluation showed that the agent developers prioritized appropriateness, completeness and shortening the development time as the most significant criteria for the MAS DSML assessment while the attractiveness of the notations had a minimum effect on preferring a language. Favourite DSML for each comparison category and criteria was determined within this evaluation.

Keywords: Software agents, Multi-Agent System (MAS), Domain-specific Modeling Language (DSML), Agent-oriented Software Engineering (AOSE), Multi-criteria decision making (MCDM), Analytic Hierarchy Process (AHP)

1. Introduction

Software agents and Multi-agent Systems (MASs) composed by these agents are used effectively in the design and implementation of various autonomous and complex intelligent systems [1, 2, 3, 4]. However, autonomous, reactive and pro-active behaviours of agents make it more difficult to develop agent-based software systems than other systems. As also indicated in [5], software systems, made up of a possibly millions of agents need to be developed in near future where agents are able to behave in a reliable, intelligent, and trustworthy way at any scale of observation. Modeling both the open nature of MAS and its relation with the outer environment is still challenging. The implementation of the interactions between agents showing autonomous and proactive behavior require a more extensive and complicated program development including verification, data integrity, resource management and concurrency of these agents [6]. While programming agent plans or rules, the mechanisms for selecting and achieving goals as well as selecting and scheduling the agent intentions that are applicable in all situations of agent execution should be constructed properly. Besides, the communication of agents within MAS organizations and the composition of agent behaviors become even more complicated and challenging to implement while the varying needs of different agent environments are considered [7]. Hence, it is important to work at higher levels of abstraction prior to coding agent software during the development of MASs [8].

In order to solve the aforementioned problems of MAS development, the researchers working in the field of Agent-oriented Software Engineering (AOSE) define a variety of meta-models that incorporate the basic components and relationships of agent models [9]. By enriching these meta-models with specific syntactic and semantic definitions, the researchers also propose various domain-specific languages (DSLs) [10, 11] or domain-specific modeling languages (DSMLs) [12] to facilitate model-driven engineering (MDE) of agents (e.g.[8, 13, 14, 15, 16, 17, 18]). These modeling languages, which are shortly referred as MAS DSMLs hereinafter, may leverage the abstraction, expression power and ease of use in software development, since MAS models can be defined visually with these DSMLs first and then the software required for the related MAS can be automatically generated via a series of transformations defined on these models. In addition, most of these languages provide both static and dynamic modeling of agent software through various perspectives such as internal agent behavior, interactions with other agents and the use of other environment sources. Hence, the modeling domain of these DSMLs cover the whole MAS including all of the agent components indicated above. Due to the capacity of identifying the essence of a MAS using visual abstractions which are close to the agent domain elements, it is expected that new DSMLs will also be introduced in the near future to address a variety of concerns in MAS [19].

Although the descriptions of the MAS DSMLs are given in the studies with including some examples of how they can be utilized during MAS development, unfortunately many of these studies do not consider a through evaluation of the proposed language, e.g. evaluating the usability of the language and the efficiency of the generated artefacts. Remaining studies (e.g. [8, 16, 20, 21]) cover some sort of quantitative and/or qualitative language assessment in an idiosyncratic manner, i.e. they only aim at the evaluation of some specific features of one single MAS DSML and hence they lack providing a general evaluation method for MAS DSMLs. More importantly they do not support the comparative evaluation of many MAS DSMLs altogether. Such a comparative evaluation may help agent software developers to choose the MAS DSML which fits well into the requirements of their MAS software development processes.

Multiple-criteria Decision Making (MCDM) techniques help decision-makers to choose the best among many alternatives in environments with different competitors and often conflicting criteria. Over the past few years, the consistent growth has been observed in the literature on MCDM, while the Analytical Hierarchical Process (AHP) [22] keeps the position which is the most frequently used MCDM technique. The AHP has been proposed to build multiple-criteria decision support method via the prioritization of each alternative based on the pairwise comparisons of each object with all the others. The popularity of AHP in the decision-making approach derives from its ability to form MCDM that includes qualitative data. The method uses a decision matrix obtained by making pairwise comparisons of the linguistic or discrete presentation of the information. Generally, it seeks expert opinions to get the information about decision alternatives. The AHP, which is often used in solving complicated decision-making problems in other areas (e.g. [23, 24, 25]) as well as in software engineering (e.g. [26, 27, 28, 29, 30]) may also contribute to the assessment of MAS DSMLs. Hence, in this paper, we introduce an AHP-based methodology which provides the comparative evaluation of MAS DSMLs according to many language criteria prioritized by the agent software developers and leads the selection of the best MAS DSML among the alternatives. Main contributions of the study can be listed as follows:

* A well-defined and categorized set of criteria and sub-criteria is presented first time which can be used to create AHP hierarchy models for the comparative evaluation of various MAS DSMLs.

* With conforming to the AHP steps, a tool assisted evaluation methodology is constructed which consists of determining MAS DSML alternatives, prioritization of both evaluation criteria and MAS DSML alternatives according to these criteria, automatic calculation of the importance distributions and finally the suggestion of the best DSML among the alternatives.

* Based on the voluntarily participation of a group of agent software developers, the proposed methodology was applied for the comparative evaluation of four MAS DSMLs widely used in AOSE.

The rest of the paper is organized as follows: Section 2 gives the related work. Section 3 briefly describes the AHP for the background. Section 4 presents the AHP-based MAS DSML evaluation methodology. Section 5 discusses the experiment for comparing MAS DSML alternatives as well as the results achieved from applying the methodology. Section 6 concludes the paper.

2. Related Work

In the early 2000s, use of UML-based languages, such as AUML [31] and AML [32] became popular in MAS modeling. However, relying too much on UML which is originally designed for object-oriented system specification as well as lacking both formal and/or operational semantics and up-to-date tools led to the decline of agent developers' interest in these languages over time [33, 9]. Following these first efforts, the researchers have made and are still making significant contribution on the derivation of various new DSLs / DSMLs for facilitating the MDE of MAS [19], e.g. to support visual MAS modeling and code generation mostly. For instance, PDT [34] and Prometheus Graphical Editor (PGE) [35] enable modeling of agents and their interactions according to Prometheus AOSE methodology and lead the implementation of the modeled agents in JACK platform[1]. Similarly, Pavon et al. [33] suggest using a tool called IDK for agent software development by following the principles of INGENIAS MAS methodology. Fuentes-Fernandez et al. [36] discuss how an agent modeling environment can be generated to support both the lifecycle and the tasks of INGENIAS methodology again.

Hahn [13] introduces the DSML4MAS language, whose abstract syntax is structured into several aspects, each focusing on a specific viewpoint of a MAS. Graphical notations for the concepts and relations are defined to provide a visual concrete syntax. Furthermore, DSML4MAS supports the deployment of modeled MASs both in JACK and JADE[2] agent platforms by providing an operational semantics over model transformations.

Ciobanu and Juravle [37] define and implement a language for mobile agents. They generate a text editor with auto-completion and error signaling features and present a way of code generation for agent systems starting from their textual description. Likewise, SEA_L [38] and JADEL [21] are two agent DSLs both providing textual syntaxes based on Xtext specifications [39]. Agents and services used by the agents can be modeled with SEA_L and these models can be used to implement agents on JADEX agent platform which is a reasoning engine for executing Belief-Desire-Intention (BDI) [40] agents. JADEL is designed to support the effective implementation of JADE agents by natively supporting agent-oriented abstractions. Finally, Sredejovic et al. [17] introduce another agent DSL, called ALAS, to allow software developers to create intelligent agents having reasoning systems based on non-axiomatic logic. It is possible to convert ALAS code to Java code and hence execute agents.

The work conducted in [15] aims at creating a UML-based agent modeling language, called MAS-ML, which is able to graphically model various types of agent internal architectures. ERE-ML [18] extends MAS-ML according to the concepts of emergency response environments and presents a modeling environment to MDE of agents for disaster management. However, the language only provides static environment modeling and runtime dependent agent interactions and the coordination and collaboration strategies can not be modeled with ERE-ML.

SEA_ML language, introduced in [14], supports graphical modeling of

[1]https://aosgrp.com/products/jack/
[2]https://jade.tilab.com/

MAS and enables the construction of modeled agents over a series of model-to-code transformations. Specifically, it supports the detailed modeling of the interactions between agents and semantic web services to realize service discovery, agreement and execution dynamics. Wautelet and Kolp [41] investigate how a model-driven framework can be constructed to develop BDI agents by proposing strategic, tactical and operational views. Although it is possible to convert generated dependencies to BDI agents, the implementation of the required transformations and code generation are not included in the study. Based on this framework, MAS implementations can be built from high level analysis models, called the Rationale Trees [42].

A development method to design and implement agents via a transformation between agent models to platform-specific code is discussed in [16]. This method can be applied by using a modeling tool, called Sam, in which graphical modeling of agents and generating source code for BDI4JADE agent framework are possible. Sam delegates to the developers the task of implementing domain-specific code that cannot be represented using this tool. DSML4BDI [8] is another DSML proposed for creating agents conforming to BDI architecture. In addition to modeling internal structure of agents, their beliefs, goals, events and knowledgebase, DSML4BDI specifically allows modeling the difficult logical expressions, which might be used in any agent plan or rule. It is possible to generate agent descriptions from these models for implementing them on the open source Jason platform[3] which is an interpreter for an extended version of a Prolog-like logic programming language for BDI agents.

The vast majority of the above referenced studies do not include an evaluation of both the usability of the proposed MAS DSMLs and generated artefacts. Instead, they just exemplify how the new DSML and its supporting tools can be used to develop agent systems. Only a few of them [43, 16, 20, 21, 8, 44] can be said considering the evaluation of their proposed methods and/or agent DSMLs.

Challenger et al. [43] evaluate SEA_ML language's [14] performance on generating MAS artefacts and shortening the development time within the perspective of use cases. The same approach is applied in [8] to measure the code generation performance and time savings gained by using DSML4BDI language. Cost of both building model transformations between

[3]http://jason.sourceforge.net/wp/

MAS DSMLs and applying these transformations to extend the execution platform support of these DSMLs over language interoperability [20] is also analyzed with using the same approach. Bergenti et al. [21] show the effectiveness of using JADEL, a DSL for JADE agents, by analyzing the lines of code required to implement agent systems with this language. The evaluation of Sam tool in [16] consists of investigating whether the tool both facilitates the agent developers' understanding of an existing BDI agent project and/or improves the evolution of an existing BDI agent project. Moreover, Sam tool's language elements and code generation performance are evaluated in the same study according to Challenger et al.'s framework [43] again. Finally, Miranda et al. [44] investigate how the graphical syntax of SEA_ML can be improved based on developers' feedback. As can be seen, all of these MAS DSML evaluation studies take into account of evaluating the features of only one specific MAS DSML, which is also created by the same researchers and hence it is difficult to generalize these evaluations to apply for other MAS DSMLs.

In addition to being widely used in various domains for different purposes (e.g. military personnel assignment [45], oil-fields development [46], evaluation of clustering algorithms for financial risk analysis [24], supplier selection in automotive industry [25], evaluation of educational use simulation packages [47]), the software engineering research also benefits from AHP and other MCDM techniques during evaluation of software tools and methods. For instance, Zhu et al. [26] uses AHP in the evaluation of four software architecture design alternatives according to quality attributes including modifiability, scalability and performance. Huang et al. [27] aim at formulating an analysis model to express the security grades of software vulnerability and serve as a basis for evaluating danger level of information program. For this purpose, the crossover factors of the software blind spots are organized and an evaluation framework is built with AHP utilization. The selection method developed in [48] to choose the appropriate techniques for the software quality attributes, such as safety and performance, uses AHP to evaluate the candidate rankings according to cost/benefit criteria. Likewise, a systematic approach of modeling quality attributes in feature models based on domain experts' judgments using AHP is proposed in [49] to leverage the product configuration in software product lines. Asadi et al. [50] address the feature model configuration problem encountered in software product lines and propose a framework based on AHP to automatically select suitable features that satisfy both the functional and non-functional preferences and constraints of stakeholders. A fuzzy variant of AHP is used in [30] to provide a process selection framework to facilitate identifying the proper way to build software to run on a mobile, web, or desktop environment. Recently, MCDM and AHP-based approaches are also followed in the evaluation of cloud services software and mobile applications. Ma et al. [28] provide a user feature-aware trustworthiness measurement approach for cloud services in which the weights of user features are assigned appropriately by employing AHP. Similarly, QoS metrics of cloud services and quantifications are proposed and an evaluation scheme is developed in [51] for cloud service trustworthiness with an MCDM method. The systematic literature review given in [29] benefits from AHP to assign the weights to the identified mobile application characteristics to determine how these characteristics affect the test effort estimation of mobile applications. Finally, Akbar et al. [52] aim at developing a prioritization-based taxonomy of success factors which are critical for the software development activities performed within the cloud platforms. AHP is applied for the formalization of the required prioritization taxonomy.

Our study contributes these noteworthy efforts by providing an AHP-based methodology to be used in the comparative evaluation of the modeling languages for software agents. As far as we know, the proposed methodology also presents the first MCDM technique to be applied on selecting the most appropriate MAS DSML for the needs of the agent software developers. It is worth indicating that AHP was previously used in [53] within the context of AOSE. However, this work does not consider MAS DSMLs and it only covers the utilization of AHP for evaluating MAS architectures, namely the centralized auction, hierarchical auction-based, centralized leaky bucket and mobile broker which can be used specifically for load balancing control in intelligent networks.

3. The Analytic Hierarchy Process

One of the most frequently used approaches to MCDM problems in which factors are organized in a hierarchical structure is the Analytic Hierarchy Process (AHP) [22, 54]. The AHP derives priority scales through pairwise comparisons of preferences [55]. In this section, we briefly introduce the AHP on which our evaluation methodology is based. For readers who are not familiar with AHP, Table 1 describes each fundamental AHP matrix and vector. To solve a decision problem by AHP method, following steps need to be taken:

Notation	**Name**	**Description**
A	Pairwise criteria comparison matrix	This is the matrix where, once the hierarchy has been established, the criteria are evaluated in pairs to determine both the relative importance between them and their weights relative to the global goal.
C	Criteria importance distributions matrix	It consists of the normalized weight of each criterion in the comparison matrices which is obtained to interpret each criterion and give relative weight.
W	Priority Vector (Eigenvector)	It shows the relative weights between criteria.
K	Decision matrix	It expresses the percentage distribution of all criteria by alternatives.
L	Overall priority vector of the alternatives with respect to the criteria	It gives the distribution of the decision points.

Table 1: Descriptions of the fundamental matrices and vectors used in AHP steps.

Step 1: The decision-making problem is defined

Definition of the decision-making problem consists of two phases: determination of decision alternatives and criteria affecting to them. Decision alternatives refer to how many different conclusions of the decision will be evaluated. Based on this definition, let the number of decision alternatives are shown by m and the number of criteria affecting the decision alternatives are shown by n in the following example. Accurate determination of the criteria affects the decision while elaborating the definition of these criteria enables to make pair comparisons more consistent and logical.

Step 2: A pairwise criteria comparison matrix is created

Pairwise criteria comparison matrix is a $n \times n$ square matrix. The matrix components have to be 1 on the diagonal where i = j, since this means comparing the related criterion to itself. The matrix can be shown as follows:

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}$$

The comparison of the criteria with each other is performed one-to-one and mutual basis by considering the degree of the importance they have according to each other. The importance scale given in Table 2 is used for this comparison process.

Importance Values	Value Definitions
1	Both criteria have equal importance
3	1st criterion has moderate importance in comparison with the 2nd criterion
5	1st criterion has essential or strong importance in comparison with the 2nd criterion
7	1st criterion has very strong importance in comparison with the 2nd criterion
9	1st criterion has extreme importance in comparison with the 2nd criterion
2, 4, 6, 8	Intermediate values between the criteria

Table 2: Importance scale table

For example, if the second criterion has moderate when we compare it with the fourth criterion, then a_{24} element in the comparison matrix should be 3. Otherwise, a_{24} element should be 1/3 and a_{42} should be 3.

It is enough that comparisons can be made for the values of pairwise criteria comparison matrix which are above the diagonal. Inherently, the formula (1) is used for the components under the diagonal.

$$a_{ji} = \frac{1}{a_{ij}} \tag{1}$$

Step 3: Importance distributions of criteria are determined as percentages

Column vectors are used to determine the weighting of the criteria, i.e. to determine the importance distributions. This vector is shown as:

$$B_j = \begin{bmatrix} b_{1j} \\ b_{2j} \\ \vdots \\ b_{nj} \end{bmatrix}$$

The formula (2) is used when calculating B-column vectors:

$$b_{ij} = \frac{a_{ij}}{\sum_{i=1}^{n} a_{ij}} \tag{2}$$

For example, following matrix A shows the comparison of a set of criteria with each other and let the vector B_1 be calculated.

$$A = \begin{bmatrix} 1 & 1/3 & 5 \\ 3 & 1 & 4 \\ 1/5 & 1/4 & 1 \end{bmatrix}$$

In this case, b_{11} element of the vector B_1 will be calculated as:

$$b_{11} = \frac{1}{1+3+0.2}$$

Similarly, when the other elements of B_1 vector are calculated, a column vector will be obtained as follows. The sum of the elements of this vector should be 1.

$$B_1 = \begin{bmatrix} 0.238 \\ 0.714 \\ 0.048 \end{bmatrix}$$

When n B-column vectors are obtained, they are combined in a matrix structure, each corresponding to a column, i.e. a C matrix is created:

$$C = \begin{bmatrix} c_{11} & c_{12} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ c_{n1} & c_{n2} & \dots & c_{nn} \end{bmatrix}$$

Consider the given example above, C matrix will be as follows:

$$C = \begin{bmatrix} 0.238 & 0.210 & 0.500 \\ 0.714 & 0.632 & 0.400 \\ 0.048 & 0.158 & 0.100 \end{bmatrix}$$

By using the C matrix, distributions showing the importance values of criteria relative to each can be obtained. In order to obtain these distributions, as shown in formula (3), arithmetic average of the row components of the C matrix is calculated and the W-column vector called Priority Vector (Pr) is computed.

$$w_i = \frac{\sum_{j=1}^{n} c_{ij}}{n} \tag{3}$$

The W-vector is expressed as follows:

$$W = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$$

It is worth indicating that both the sum of the elements in the columns of the C matrix and the sum of the elements of the W vector must be 1. Considering the above example, elements of the priority vector will be calculated as follows. When the three criteria are evaluated together, the first criterion gets approximately 32%, the second criterion gets approximately 58% and the third criterion gets approximately 10% importance values.

$$W = \begin{bmatrix} \frac{0{,}238+0{,}210+0{,}500}{3} \\ \frac{0{,}714+0{,}632+0{,}400}{3} \\ \frac{0{,}048+0{,}1580+0{,}100}{3} \end{bmatrix} \cong \begin{bmatrix} 0,32 \\ 0,58 \\ 0,10 \end{bmatrix}$$

Step 4: Consistency in criteria comparisons is measured

The consistency of the results depends on the consistency between criteria which is provided by a decision-maker as one-to-one matching. There is a mechanism in AHP that measures the consistency of these comparisons. This mechanism is called Consistency Ratio (CR) and it can be used to test the consistency of the priority vector, namely comparison between the criteria. CR calculation in AHP is based on the comparison between the number of criteria and a coefficient called Principal Eigenvalue (λ). For calculation of the λ, first, the D-column vector $[d_1,...,d_n]$ is obtained by matrix multiplication of the comparison matrix A and the W-priority vector.

$$D = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$$

By using the formula 4, E_i values related to each evaluation criterion are obtained from the division of the mutual elements of the D-column vector and the W-column vector.

$$E_i = \frac{d_i}{w_i} \quad (i = 1, 2, \dots, n) \tag{4}$$

Arithmetic mean calculated from the formula 4 gives the Principal Eigenvalue (λ) for the comparison.

$$\lambda = \frac{\sum_{i=1}^{n} E_i}{n} \tag{5}$$

After finding the λ, the Consistency Index (CI) can be calculated using the formula 6.

$$CI = \frac{\lambda - n}{n - 1} \tag{6}$$

Finally, CI is divided by the standard correction value called Random Index (RI) in the formula 7 to obtain the Consistency Ratio (CR). From Table 3, N value, which corresponds to the number of criteria, is selected. For example, RI value would be 0.58 in 3-criteria comparison.

$$CR = \frac{CI}{RI} \tag{7}$$

N	**RI**	**N**	**RI**
1	0	6	1,24
2	0	7	1,32
3	0,58	8	1,41
4	0,90	9	1,45
5	1,12	10	1,49

Table 3: RI values

If the calculated CR value is less than 0.10, it indicates that the comparisons are consistent. Otherwise, there are some errors in the transactions, or the decision maker acted inconsistent in the comparisons.

Step 5: Importance distributions of the criteria in m decision alternatives are found as percentages

At this step, one-to-one comparisons and matrix operations are repeated n times or the number of criteria. But this time, the size of G comparison matrices will be $m \times m$. Because, the comparison matrices are obtained by comparing the alternatives with each other based on the related criteria, not by comparing the criteria with each other. After each comparison, $m \times 1$ sized S column vectors, which show the percentage distribution of the corresponding criterion by alternatives, are obtained. These column vectors are as follows:

$$S_j = \begin{bmatrix} S_{1j} \\ S_{2j} \\ \vdots \\ S_{mj} \end{bmatrix}$$

Step 6: Result distribution in decision points is found

At this step, firstly, the $m \times n$ sized K decision matrix, composed of the n number $m \times 1$ sized S-column vector is created:

$$K = \begin{bmatrix} S_{11} & S_{12} & \dots & S_{1n} \\ S_{21} & S_{22} & \dots & S_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ S_{m1} & S_{m2} & \dots & S_{mn} \end{bmatrix}$$

As a result, when the decision matrix is multiplied by the W column vector (the priority vector), the m sized L column vector is obtained (see below). The L column vector gives the distribution of the decision points as percentages. The sum of the elements of the vector is equal to 1. This distribution also shows the importance of the decision points.

$$L = \begin{bmatrix} s_{11} & s_{12} & \dots & s_{1n} \\ s_{21} & s_{22} & \dots & s_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ s_{m1} & s_{m2} & \dots & s_{mn} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} l_{11} \\ l_{21} \\ \vdots \\ l_{m1} \end{bmatrix}$$

4. The AHP-based Evaluation Methodology for MAS DSMLs

This section introduces the methodology which we propose for the comparative evaluation of MAS DSMLs. As conforming to the AHP steps described in the previous section, the application of this evaluation methodology consists of defining MAS DSML evaluation criteria and determining MAS DSML alternatives, collecting and preparing data for the evaluation, calculating weights, and finally analyzing the achieved results. Figure 1 illustrates the overview of this methodology. Definition of the evaluation criteria and the general process of how these criteria can be used to formalize and evaluating the related hierarchy are discussed in the following subsections. Execution of the remaining steps of the methodology for conducting the experiment covering data collection and preparation and finally analyzing the achieved results are all elaborated in the next section of the paper.

4.1. Defining Criteria and Sub-criteria

According to the AHP Step 1 (see Section 3), the problem, criteria and alternatives should be determined first. The problem presented here is how to facilitate the quality evaluation of MAS DSMLs and guide software developers to determine the most suitable DSML(s) for their MAS implementations. After the problem is determined, MAS DSML evaluation criteria need to be defined. During derivation of these criteria that will be used in AHP-based MAS DSML evaluations, we benefited from our previous work [43], [20], [56], [8] on the quantitative analysis and qualitative assessment of MAS DSMLs, called SEA_ML and DSML4BDI. Although just one specific MAS DSML was evaluated in each of these studies without considering any comparison, these prior experiences enabled us to define new criteria especially on the usability and the productivity features of MAS DSMLs in this study. Finally, we also adopted the characteristics of the Framework for Qualitative Assessment of DSLs (FQAD) [57]. However, both these characteristics and their definitions which are originally given to evaluate DSLs in general terms,

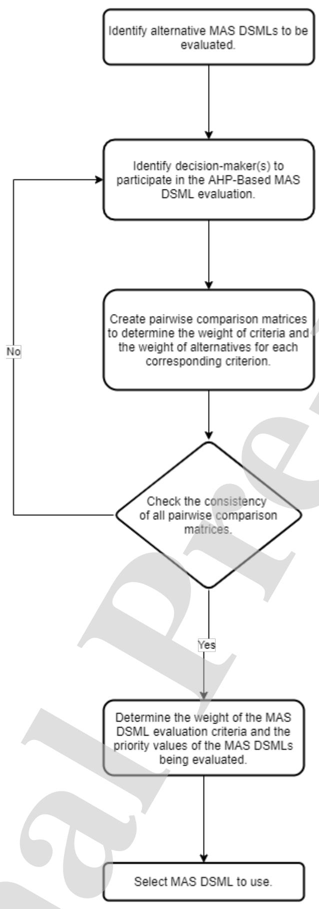

Identify alternative MAS DSMLs to be evaluated.
Identify decision-maker(s) to participate in the AHP-Based MAS DSML evaluation.
Create pairwise comparison matrices to determine the weight of criteria and the weight of alternatives for each corresponding criterion.
No
Check the consistency of all pairwise comparison matrices.
Yes
Determine the weight of the MAS DSML evaluation criteria and the priority values of the MAS DSMLs being evaluated.
Select MAS DSML to use.

Figure 1: Steps of the AHP-based evaluation methodology for MAS DSMLs

are re-engineered and specialized in our work for the needs of assessing MAS DSMLs. In addition, we defined new categories e.g. on the compatibility of MAS DSMLs and new criteria for constructing agents which are not included in FQAD definitions.

Table 4 lists the categories and the criteria which we defined to create the AHP hierarchy model to be used in MAS DSML evaluations. Since each criterion and sub-criterion are just related with one category, which refers to the first level criterion from the AHP point of view, it also enables to

No	Category	Criterion	Sub-criterion
1	Functional Suitability	Completeness	
		Appropriateness	
2	Usability	Comprehensibility	
		Learnability	
		Number of Activities	
		User Perception	
		Operability	
		Attractiveness	
		Compactness	
		Ease of Use	Ease of Use of the DSML
			Ease of Use of the Tools
3	Reliability	Model Checking	
		Correctness	
4	Expressiveness	Uniqueness	
		Orthogonality	
		Correspondence to Important Domain Concepts	
		Conflicting Elements	
		Right Abstraction Level	
5	Compatibility	Compatibility with the Domain	
		Compatibility with the Development Process	
6	Maintainability		
7	Productivity	The Development Time	
		The Amount of Human Resource	
8	Extensibility		
9	Reusability		
10	Integrability		

Table 4: Criteria and sub-criteria for AHP hierarchy model to be used in MAS DSML evaluation

make category based evaluation for MAS DSMLs. There are 10 categories having various criteria. These categories are *Functional Suitability*, *Usability*, *Reliability*, *Expressiveness*, *Compatibility*, *Maintainability*, *Productivity*, *Extensibility*, *Reusability* and *Integrability*.

The first category, *Functional Suitability* represents a MAS DSML's ability to meet the requirements for agent software. This category has two criteria: *Completeness* and *Appropriateness*. *Completeness* means that all concepts and scenarios of an agent domain can be expressed with a MAS DSML. *Appropriateness* indicates whether a MAS DSML can be used for the design and the implementation of different agent models and architectures such as behavioral models or BDI [40] agent architectures.

The second category, *Usability*, aims at evaluating whether a MAS DSML can be used to achieve the goals specified for the construction of agent systems. Usability category contains 9 criteria which are *Comprehensibility*, *Learnability*, *Number of Activities*, *User Perception*, *Operability*, *Attractiveness*, *Compactness* and *Ease of Use*. *Comprehensibility* means that the agent modeling elements of a MAS DSML are easily understandable by the developers. Moreover, the concepts and symbols provided by a MAS DSML should be both learnable and recallable for the language's users, i.e. concepts given for the design of agent plans and inner tasks need to be catchy and good to follow for further MAS implementations. This feature is evaluated in the *Learnability* criterion. On the other hand, the *Number of Activities* criterion evaluates whether MAS DSMLs minimize the steps and the number of development tasks required during MAS design and implementation.

With *User Perception* criterion of the Usability category, our intent is to determine the recognition of agent developers on the appropriateness of the DSMLs for their needs. Based on their knowledge and experience on MAS development, agent developers (MAS DSML users), can define and choose the MAS DSMLs which are most suitable for agent development. *Operability* criterion considers whether a MAS DSML's elements can be operated easily and provide managing the language features with little effort. In addition, this criterion also aims at identifying the functionality of both MAS DSMLs and their CASE tools since many DSMLs in AOSE field only emerge from the academic studies and the continuous support on improving especially the language constructs and the related software tools is often unavailable contrary to the commercial products. Hence, agent developers may encounter difficulties in the operation of these languages and their tools. Furthermore, agent developers may easily understand what the symbols or notations mean and they may not have any conceptual confusion when the language elements of a MAS DSML are pleasing to the eye. Notations, symbols and other graphical elements presented by MAS DSMLs are evaluated according to the *Attractiveness* criterion in our study. A recent example of how improving the attractiveness of language elements may contribute to the usability of a MAS DSML can be found in [44]. MAS DSMLs have several background mechanisms, e.g. model transformations and code generations which provide both the power and the representation of the language. *Compactness* of them also leverages the usability of these languages. Finally, with its two sub-criteria, *Easy of Use* criterion evaluates the convenience of using both the language itself and modeling and implementation tools come along with the language.

As is the case in other programming and modeling languages, MAS DSMLs are expected to provide the correct construction of agent models and bring features that will minimize developer errors. *Reliability* category in the proposed AHP hierarchy model includes two criteria to evaluate MAS DSMLs for this purpose. *Model Checking* criteria is used to evaluate whether a MAS DSML is able to reduce errors which may be encountered during MAS modeling. *Correctness* means a MAS DSML has sufficient controls preventing agent developers from connecting incorrect language components e.g. during the construction of agent internals or messaging between agents. The language is expected to prevent such kind of improper MAS modeling automatically or at least, it is required to process user models, detect modeling errors and notify the users.

Expressiveness can be defined as the degree to which a problem solving strategy can be mapped into programs easily [57]. A MAS DSML should provide a convenient way of modeling agents and their artifacts which conforms to the design specified by the agent developers. Related capabilities of MAS DSMLs are evaluated in our AHP hierarchy model with the *Expressiveness* category which consists of 5 different criteria: Uniqueness, Orthogonality, Correspondence to Important Domain Concepts, Conflicting Elements, and Right Abstraction Level. *Uniqueness* denotes a MAS DSML has only one way to express the concepts of agent domain. In addition, the *Orthogonality* is supported when each MAS DSML construct is used to represent one distinct concept in the domain. For instance, it would be better for a language to provide only a single form of modeling to implement each plan of an agent as a composition of inner tasks. *Correspondence to Important Domain Concepts* criteria can be used to assess whether the abstract syntax of each MAS DSML provides all significant concepts of agent domain for the system design. Similarly, the syntax needs to be as free from *Conflicting Elements* as possible to eliminate confusion during MAS modeling. That may also guide to remove unnecessary elements. The metamodel of a MAS DSML should also be at the *Right Abstraction Level*. The meta-entities of the language's metamodel and their relations should not be too complicated, i.e. they should not include unnecessary details of physical agent platforms. On the other hand, these elements also should not be too abstract from the exact MAS implementations since the models created by the language can now be trivial and they will need more intervention than necessary. For instance, a very abstract model of agent plans causes the generation of light code templates for agent tasks and hence these structures need to be completed very deeply before executing the related plans. An extensive discussion of how critical to achieve right abstraction level in the metamodels of MAS DSMLs can be found in [20].

To increase the adoption of agent developers, a MAS DSML needs to be compatible both with the MAS domain and the software development methodologies applied to implement agent systems. These features of MAS DSMLs are evaluated within the *Compatibility* category which includes two criteria. As its name suggests, the *Compatibility with the Domain* measures the MAS domain coverage of a DSML and evaluates the language's capability to operate with remaining domain elements. *Compatibility with the Development Process* means models and any other artifacts of a MAS DSML can be used inside existing AOSE methodologies [58] and/or modeling with the DSML can be integrated into the MAS development process, e.g. it can be used as part (or perhaps one of the steps) of the development process.

The ability to modify the existing functions in a MAS DSML is evaluated in the *Maintainability* category. Some features can be added, deleted or changed by developers or support communities without decreasing the usability of the MAS DSML.

The seventh category is *Productivity*. It evaluates whether a MAS DSML uses as little resources as possible to achieve the goals. Resource utilization is evaluated with two criteria. The first one is *the Development Time*. A MAS DSML is expected to shorten the development time required to design and implement MAS. Users naturally do not prefer a MAS DSML extending system development time. The second criteria is *the Amount of Human Resource*. Similar to the development time, use of a MAS DSML may decrease the amount of human resources required to construct agent systems in comparison with the conventional approaches followed to implement same systems.

Inside the *Extensibility* category, adding new features into an existing MAS DSML is evaluated. If the DSML supports extensibility, it will be possible to add new modeling elements, new model transformations and code generation mechanisms in a convenient way to support the implementation of the MAS modeled with this DSML into various agent execution platforms, e.g. JACK, JADE or Jason. Similar to Maintability, being extensible also facilitates keeping a MAS DSML up-to-date and hence adoption by the users for a long time.

Another desirable feature is that the syntax and semantics definitions of a MAS DSML can be used to create other languages or agent development platforms. That feature is evaluated inside the *Reusability* category. In addition, it is possible to integrate existing MAS DSMLs with other DSMLs and/or agent programming languages for more enhanced agent development environments. The last category, *Integrability*, is aimed to assess a MAS DSML within this context. For instance, a MAS DSML can be integrated with the application programming interfaces (APIs) of various agent languages or frameworks to formalize a complete toolset for the MDE of autonomous agents. Also, it can be possible to integrate many MAS DSMLs together to facilitate implementing a modeled MAS in various agent execution platforms without creating new transformations for code generation required for each specific agent execution platform as discussed in [20].

These criteria, brought for the evaluation of MAS DSMLs, can also be closely related to the most of the cognitive dimensions defined in the well-known Cognitive Dimensions Framework (CDF) [59, 60]. CDF enables the usability assessment of visual programming languages as well as DSLs (e.g. [61]) by taking into consideration a set of cognitive dimensions including e.g. Abstraction gradient, Closeness of mapping, Consistency, Diffuseness, Error-proneness, Hard mental operations, Role-expressiveness and Viscosity. For instance, having a *Right Abstraction Level* for a MAS DSML may also lead the language *abstraction gradient* as defined in CDF; so modeling both agents, their interactions and environments in different levels from overall MAS perspective to the programming constructs of the underlying MAS platform is possible. Moreover, *Completeness* and *Appropriateness* criteria in our AHP model aim at evaluating whether a MAS DSML minimizes the semantic gap between MAS modeling and actual system implementation which is similar to the context of CDF's *Closeness of mapping* dimension. Likewise, *User Perception* criterion may refer to the CDF's *Consistency* dimension since it helps evaluating the agent developers' expectations on the features of the DSML in question. *Uniqueness* and *Orthogonality* criteria consider the number of the syntactic elements required for expressing relations in a MAS model as is the case in CDF's *Diffuseness* dimension. *Criteria for Correspondence to Important Domain*, *Conflicting Elements*, *Model Checking* and *Correctness* criteria in our AHP model support the evaluation of whether a

MAS DSML's notation catches mistakes and eliminates errors, i.e. the language is evaluated within CDF's *Error-proneness* dimension. *Compactness* of the model transformations and code generation features of a MAS DSML minimizes *hard mental operations* while assessing the *Operability* and *Comprehensibility* of the MAS modeling elements also provides examining the relations of these elements with each other, i.e. they are inspected within the *Role-expressiveness* dimension of CDF. We evaluate the ability of modifying the features of a MAS DSML without losing the usability by means of the *Maintainability* criteria which is very similar to the assessments made under CDF's *Viscosity* dimension. Finally, evaluating the *Attractiveness* and *Easy of Use* of a MAS DSML's graphical syntax also supports measuring both the appearance of the model and the traces between modeling elements for agents, goals, plans, etc. as considered in *Visibility* dimension of CDF.

4.2. Formalizing and Evaluating the Hierarchy

After creating the evaluation criteria, a set of MAS DSMLs is required to be selected as the alternatives to be used in our AHP-based methodology. These criteria and MAS DSML alternatives construct the required hierarchy. The priority of the criteria should be determined as first. For this purpose, participants (decision-makers) must perform a series of pair-wise comparisons deriving numerical measurement scales for each node in the hierarchical structure. Criteria are compared in pairs according to their level of priority. So, decision-makers determine the weight to be given to each criterion when choosing the most appropriate MAS DSML from their point of view. Alternatives are compared in pairs according to each of the preference criteria, too. By this way, the weight to be given to each alternative is determined according to each of the criteria. Thus, while generating the global priority for the alternatives, the priorities of the alternatives are formed on the basis of each criterion. Next section discusses how these steps are applied to evaluate four MAS DSMLs within a conducted experiment.

5. Experiment and Discussion

The methodology was used for the comparative evaluation of four different MAS DSMLs. In the following subsections, determining these MAS DSML alternatives, collecting and preparing data, calculating the weights and finally analyzing the results are discussed.

5.1. Determining MAS DSML Alternatives

Four different MAS DSMLs were determined here, namely JACK, PDT, SAM, and SEA_ML. In addition to being widely used in the AOSE domain, these DSMLs were selected for this evaluation by taking into account various features and advantages they brought for the development of agent internals, agent interactions and other artifacts required in a MAS. All of them provide a convenient IDE for MAS modeling, a well-defined set of graphical syntax and an executable semantics for the implementation of the modeled systems. Another reason of choosing these DSMLs is all of them are fully-functional with their modeling and implementation tools which are accessible online and running at the time of conducting this evaluation.

Although *JACK* [62] is, in fact, a commercial MAS programming framework with an API especially dedicated to implement BDI agents, it also provides a convenient IDE to MDE of MAS. It presents graphical agent modeling and then auto-generation of JACK codes. These features caused us to consider JACK as one of the DSMLs in this study. *PDT* [63] is the software tool, developed for supporting the well-known Prometheus AOSE methodology. It also provides a DSML whose graphical concrete syntax enables users to visually model e.g. agents, goals, plans and communications based on the descriptions of the Prometheus diagram types. In addition to support fundamental process of Prometheus, PDT also includes a code generator to implement the modeled agents and plans for an agent execution platform. *SAM* [16] is one of the newest DSMLs which also supports BDI agent design and implementation. It enables both the graphical design of BDI plans and implementation of these plans for BDI4JADE agent execution platform. Finally, *SEA_ML* language [14] provides the graphical modeling of MAS and enables the construction of modeled agents over a series of model-to-code transformations. Specifically, it supports the detailed modeling of the interactions between agents and semantic web services to realize service discovery, agreement and execution dynamics. The overview of the created AHP model with including both these MAS DSML alternatives and evaluation categories and criteria previously introduced in Section 4.1 is shown in Figure 2.

5.2. Data Collection and Preparation

Seven agent developers were voluntarily participated in this study. All participants had sufficient knowledge and experience on the computer related fields and passed graduate courses called Advanced Software Engineering, Agent-oriented Software Development and Multi-agent Systems, taught in

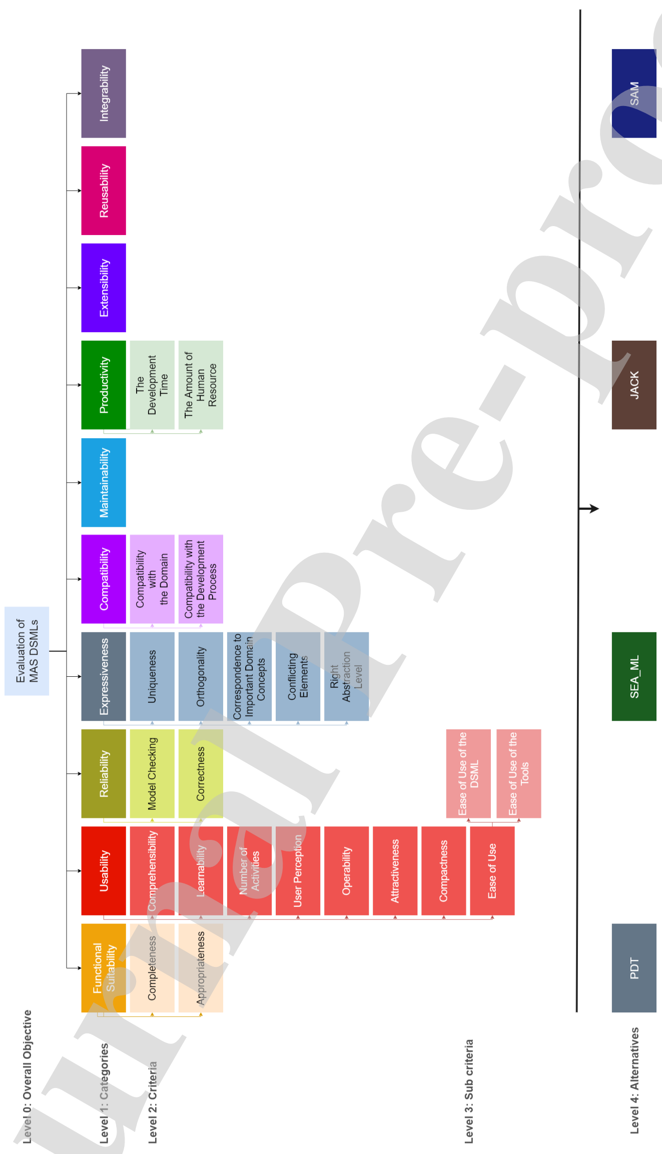
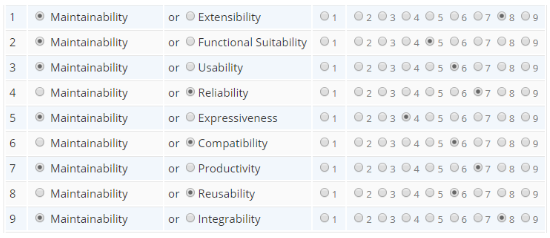


Figure 2: AHP hierarchy model for the MAS DSML evaluation

Computer Engineering Department and International Computer Institute (ICI) of Ege University. One participant had Ph.D. degree while two of them were Ph.D. candidates and the remaining were M.Sc. students. On the average, the participants had at least 2 years MAS design and implementation experience covering the application of AOSE methodologies and using some agent development APIs like JADE and JACK. They also had almost the same level of experience on using all MAS DSMLs being evaluated in this study since all these DSMLs were previously used by all participants while taking the above courses related to MAS. They used all these DSMLs in the same software development projects aiming to design and implement MAS for various domains including hotel reservation, garbage collection, online scientific conference management and stock trading. The participants were familiar with software engineering methodologies, having at least 4 years' experience on using IDEs such as Eclipse, NetBeans, IntelliJ and Visual Studio. Four participants were also working in the industry at the time of this evaluation was performed and they possessed the experience of developing software in industrial scale (2.5 years on average).

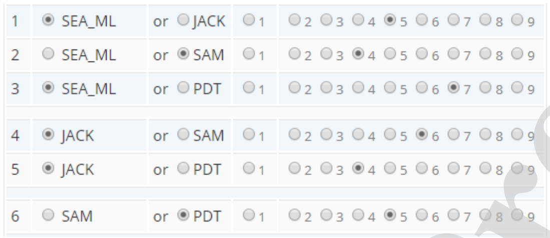
Additionally, during our experimental study, we benefited from the features of AHP-OS online tool, available in [64], which is widely used in AHP studies in order to create the decision hierarchy, determine inconsistent decisions, make sensitivity analysis and evaluate the alternatives [65]. With this tool, we managed to obtain the participant choices with group decision support features. All evaluators easily participated in our online AHP group sessions via this tool. The AHP for the evaluation of the above MAS DSMLs was created inside AHP-OS. Evaluation of the priorities and MAS DSML alternatives as well as calculating the priorities based on pairwise comparisons were all performed during our experiment again with using AHP-OS. The screenshots given in Figure 3 show how the online forms and tables of this tool are used during our AHP-based MAS DSML evaluation experiment.

During the study, two kinds of questionnaires were created and presented to the participants. In the first questionnaire, the participants prioritized the criteria introduced above in Sect. 4.1. Each criterion was compared with each other respectively and the participants gave scores between $1 - 9$ to decide which criterion had priority over them. Scoring with 1 means that the criterion has no priority and 9 indicates that it has the highest priority for the participant. As a result, it was aimed to obtain the general priority score of each criterion. After the participants completed this questionnaire, they were given the second questionnaire. This questionnaire provided the



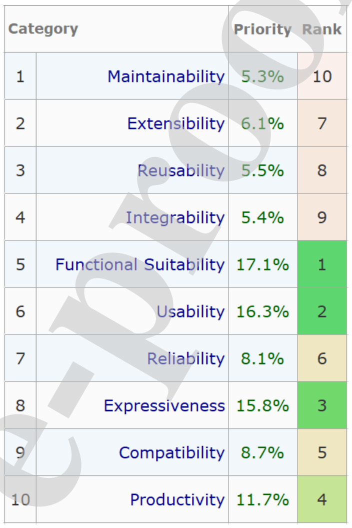
1	[✓ Maintainability]	or Extensibility	1	2 3 4 5 6 7 [✓ 8] 9
2	[✓ Maintainability]	or Functional Suitability	1	2 3 4 [✓ 5] 6 7 8 9
3	[✓ Maintainability]	or Usability	1	2 3 4 5 [✓ 6] 7 8 9
4	Maintainability	or [✓ Reliability]	1	2 3 4 5 6 [✓ 7] 8 9
5	[✓ Maintainability]	or Expressiveness	1	2 3 [✓ 4] 5 6 7 8 9
6	Maintainability	or [✓ Compatibility]	1	2 3 4 5 [✓ 6] 7 8 9
7	[✓ Maintainability]	or Productivity	1	2 3 4 5 6 [✓ 7] 8 9
8	Maintainability	or [✓ Reusability]	1	2 3 4 5 [✓ 6] 7 8 9
9	[✓ Maintainability]	or Integrability	1	2 3 4 5 6 7 [✓ 8] 9

(a) Online form for comparing the evaluation criteria to obtain the pair-wise comparison matrix.



1	[✓ SEA_ML]	or JACK	1	2 3 4 [✓ 5] 6 7 8 9
2	SEA_ML	or [✓ SAM]	1	2 3 [✓ 4] 5 6 7 8 9
3	[✓ SEA_ML]	or PDT	1	2 3 4 5 6 [✓ 7] 8 9
4	[✓ JACK]	or SAM	1	2 3 4 5 [✓ 6] 7 8 9
5	[✓ JACK]	or PDT	1	2 3 [✓ 4] 5 6 7 8 9
6	SAM	or [✓ PDT]	1	2 3 4 [✓ 5] 6 7 8 9

(b) Online form for comparing the MAS DSML alternatives with each other to obtain the pair-wise comparison matrix.



	Category	Priority	Rank
1	Maintainability	5.3%	10
2	Extensibility	6.1%	7
3	Reusability	5.5%	8
4	Integrability	5.4%	9
5	Functional Suitability	17.1%	1
6	Usability	16.3%	2
7	Reliability	8.1%	6
8	Expressiveness	15.8%	3
9	Compatibility	8.7%	5
10	Productivity	11.7%	4

(c) The criteria priority table automatically generated according to the obtained pair-wise comparison matrix.

Figure 3: Some screenshots taken from the AHP-OS tool

comparative evaluation of MAS DSMLs according to the defined criteria, i.e. each participant evaluated MAS DSMLs with each other according to each criterion separately. Here again, the participants gave scores between $1-9$. Hence, this questionnaire led to see which MAS DSML comes into prominence more when considering each criterion. When these questionnaire were completed, we applied Step 2 of AHP and hence criteria comparison matrix was formed. Thus, the weights of the criteria were determined as described in Step 3 of AHP. After the weight distribution was created, the consistency of the answers were also checked according to Step 4 of AHP. As indicated in Sect. 3 of this paper, when the Consistency Ratio is greater than 10%, it shows that given answers are inconsistent with the process. In our study, when this situation occurred, the participants were asked to review the relevant part of the survey again. A participant was not allowed to proceed to the next stage until the Consistency Ratio of the relevant part decreased below 10%.

5.3. Calculation of the Weights

After the questionnaire process was completed, we applied Step 5 and Step 6 of AHP one after another to obtain both the importance percentages of the MAS DSML evaluation criteria and the priority values of the MAS DSMLs being evaluated. In other words, the distribution of the scores given to the criteria was determined according to the MAS DSML alternatives. Here, the MAS DSMLs were evaluated in general manner as well as according to each evaluation criterion. Therefore, apart from the overall assessment, one MAS DSML can be evaluated according to some criteria or sub-criteria which come more relevant to the related agent developer (in here the participant).

5.4. Analysis and Discussion

At first, the evaluation scores for the MAS DSML evaluation criteria, listed in Table 5, were gained when the participants compared these criteria with each other based on their own priorities and expectations from any MAS DSML. After each participant scored each MAS DSML according to the evaluation criteria, the results for MAS DSML evaluations were obtained (see Table 6) at last.

In Table 5, levels 1, 2 and 3 contain the categories, the criteria, and the sub-criteria for MAS DSML evaluation respectively. The numbers next to each item indicate the priority scores calculated based on the participants' choices. These participant choices were obtained by the first questionnaire made during executing the second step of our methodology (see Sect. 4.2). The priorities are between $0-1$ where 0 represents that a criterion has no priority for the participants, and 1 represents that the criterion has the highest priority for these participants. Global Priorities (Glb. Prio.) column in Table 5 shows the ratio-scale weights (in percentages) of any criterion or sub-criterion with respect to their importance on evaluating MAS DSMLs, i.e. the order of the significances of each MAS DSML evaluation criterion according to the participants' prioritizations. These percentages were calculated by creating the importance distributions of the criteria as discussed in Sect. 3, AHP Step 3. After calculating the weights for each binary comparison matrix, the weight of each sub-criterion at the bottom of the hierarchy in the calculation of global priorities is achieved by multiplying the local weights starting from each sub-criterion upwards. Thus, with the help of these calculated weights, global priorities are found.

Level 0	Level 1	Level 2	Level 3	Glb Prio.
Evaluation of MAS DSML Criteria	Maintainability 0.053			5.3%
	Extensibility 0.061			6.1%
	Reusability 0.055			5.5%
	Integrability 0.054			5.4%
	Functional Suitability 0.171	Completeness 0.490		8.4%
		Appropriateness 0.510		8.7%
	Usability 0.163	Comprehensibility 0.263		4.3%
		Learnability 0.151		2.5%
		Number of Activities 0.073		1.2%
		User Perception 0.062		1.0%
		Operability 0.190		3.1%
		Attractiveness 0.046		0.8%
		Compactness 0.074		1.2%
		Ease of Use 0.140	Ease of Use of the DSML 0.549	1.3%
			Ease of Use of the Tools 0.451	1.0%
	Reliability 0.081	Model Checking 0.366		3.0%
		Correctness 0.634		5.1%
	Expressiveness 0.0158	Uniqueness 0.183		2.9%
		Orthogonality 0.261		4.1%
		Correspondence to Important Domain Concepts 0.187		2.9%
		Conflicting Elements 0.217		3.4%
		Right Abstraction Level 0.152		2.4%
	Compatibility 0.087	Compatibility with The Domain 0.780		6.8%
		Compatibility with The Development Process 0.220		1.9%
	Productivity 0.117	The Development Time 0.630		7.4%
		The Amount of Human Resource 0.370		4.3%

Table 5: Evaluation scores of MAS DSML criteria

According to global priority percentages shown in Table 5, the Appropriateness was selected by the participants as the most significant sub-criterion for MAS DSML evaluation with 8.7%. It can be deduced that the participants want to develop different agent models and architectures with using the same MAS DSML. We can interpret that the agent developers do not want to lose time and decrease the MAS development efficiency by dealing with different MAS DSMLs, e.g. in case they are forced to use one MAS DSML only eligible for reactive agent planning and use another DSML for BDI modeling. The second highest priority percentage was given to Completeness with 8.4%. That means the participants expect a MAS DSML to cover the whole MAS domain with all required agent concepts and their relations and hence it enables to realize any kind of MAS execution scenarios. Moreover, these two sub-criteria with the highest-percentages belong to the same criterion, Functional Suitability and so, it stands out as the highest priority criterion among all MAS DSML evaluation criteria. That result can be estimated since the functional suitability of a MAS DSML is defined in our categorization as the ability to meet agent software requirements and naturally the agent developers (the participants in our evaluation) preferred using a MAS DSML which is capable of providing various modeling features to design and implement any kind of MAS with changing requirements. The Development Time sub-criterion of the Productivity criterion got the third highest percentage with 7.4%. That shows the participants mostly agreed on a MAS DSML is expected to contribute shortening the overall MAS development time, i.e. design and implementation with a MAS DSML will take less time in comparison with using the conventional tools and approaches which are mostly just based on utilizing general purpose programming languages. Finally, the lowest priority criterion was found to be the Attractiveness with 0.8% which is a little bit surprising since this means the good appearance of a MAS DSML's notations, symbols or other graphical interface components has minimum priority for the participants. The global priority percentage distribution of all these criteria and sub-criteria which effects the selection of MAS DSML alternatives is also shown in the bar chart given in Figure 4.

The second questionnaire, made in the second step of the applied methodology, enabled to achieve the comparative evaluation results for the four MAS DSMLs. As discussed in Sect. 5.2, each participant scored these MAS DSMLs with comparing each other by taking into consideration the evaluation criteria. Table 6 shows these results reflecting which MAS DSML is found better according to which criterion. The subsequent columns after Glb. Prio. col-

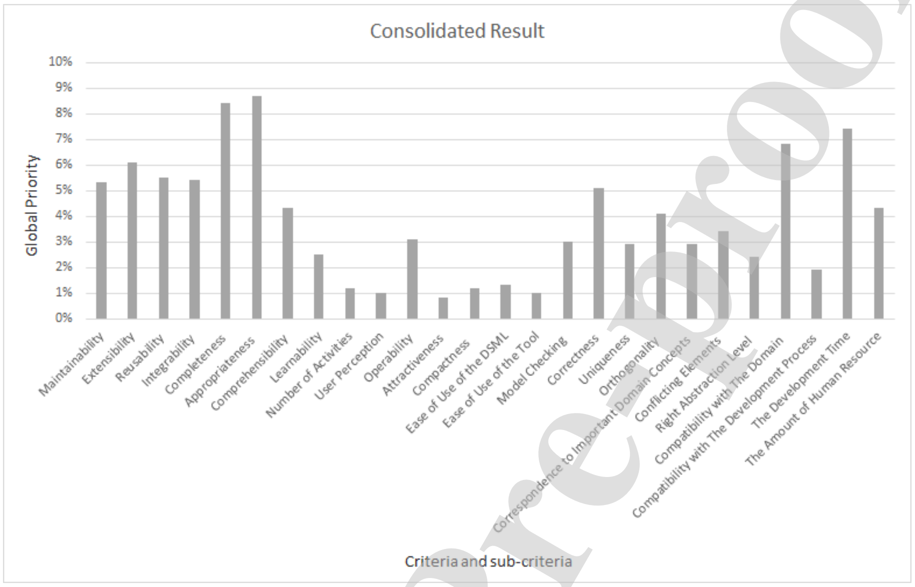


Figure 4: Effects of criteria and sub-criteria in AHP as percentage

umn show the priority scores of all MAS DSMLs for each criterion. For example, in terms of the Maintainability criterion, SEA_ML was identified as the most successful language with 0.389 priority score. Moreover, SEA_ML stands out in comparison with the other alternative MAS DSMLs when we take into account Extensibility, Reusability, Completeness, User Perception, Model Checking, Compatibility with the Domain and Development Time criteria. On the other hand, JACK received the highest priority scores for the following evaluation criteria: Integrability, Appropriateness, Comprehensibility, Learnability, Operability, Attractiveness, Compactness, Ease of Use of the DSML, Ease of Use of the Tools, Correctness, Uniqueness, Orthogonality, Correspondence to Important Domain Concepts, Conflicting Elements, Right Abstraction Level, Compatibility with the Development Process and The Amount of Human Resource. Neither SAM nor PDT got the highest priority scores in any of the existing criteria.

Finally, the bottom row in Table 6 shows the overall priority percentages for the alternatives and indicates which MAS DSML comes to the fore as the result of the comparative evaluation. It is worth indicating that these

Level 0	Level 1	Level 2	Level 3	Glb Prio.	SEA_ML	JACK	SAM	PDT
Evaluation of MAS DSML Alternatives	Maintainability			5.3%	0.389	0.265	0.213	0.134
	Extensibility			6.1%	0.442	0.187	0.214	0.157
	Reusability			5.5%	0.308	0.301	0.187	0.204
	Integrability			5.4%	0.302	0.332	0.188	0.178
	Functional Suitability	Completeness		8.4%	0.398	0.278	0.130	0.193
		Appropriateness		8.7%	0.326	0.371	0.138	0.165
	Usability	Comprehensibility		4.3%	0.245	0.431	0.151	0.172
		Learnability		2.5%	0.219	0.449	0.138	0.194
		Number of Activities		1.2%	0.238	0.478	0.128	0.155
		User Perception		1.0%	0.366	0.300	0.183	0.151
		Operability		3.1%	0.183	0.517	0.148	0.151
		Attractiveness		0.8%	0.333	0.359	0.154	0.154
		Compactness		1.2%	0.257	0.433	0.156	0.154
		Ease of Use	Ease of Use of the DSML	1.3%	0.227	0.368	0.162	0.243
			Ease of Use of the Tools	1.0%	0.227	0.415	0.160	0.198
	Reliability	Model Checking		3.0%	0.343	0.335	0.165	0.158
		Correctness		5.1%	0.250	0.371	0.195	0.184
	Expressiveness	Uniqueness		2.9%	0.217	0.462	0.152	0.169
		Orthogonality		4.1%	0.204	0.409	0.167	0.220
		Correspondence to Important Domain Concepts		2.9%	0.318	0.356	0.158	0.168
		Conflicting Elements		3.4%	0.273	0.345	0.188	0.194
		Right Abstraction Level		2.4%	0.267	0.400	0.161	0.172
	Compatibility	Compatibility with The Domain		6.8%	0.367	0.332	0.165	0.136
		Compatibility with The Development Process		1.9%	0.313	0.417	0.117	0.153
	Productivity	The Development Time		7.4%	0.313	0.283	0.185	0.219
		The Amount of Human Resource		4.3%	0.264	0.310	0.233	0.193
	Overall priority percentages			**100.0%**	**30.98%**	**34.27%**	**17.06%**	**17.69%**

Table 6: Evaluation scores of MAS DSML alternatives (SEA_ML, JACK, SAM, PDT)

percentages were calculated as discussed in Sect. 3 (see AHP Step 6). The MAS DSML which has the highest priority level within the scope of all these evaluation criteria is JACK with 34.27%. In other words, when a developer needs to choose among these DSML alternatives in order to design and implement a MAS, these results suggest preferring JACK. JACK was favoured by the participants and selected as the most successful language considering the majority of the criteria. Especially, the functional suitability and ease of use features brought by the language as well as being compatible with existing AOSE methodologies and rapid integration into MAS development steps make JACK as the first choice of the agent developers participated in this evaluation. The fact that JACK is a commercial product that is actively used in the industry and has continuous professional support and development also contributed this result. The second most preferred MAS DSML is SEA_ML with 30.98%. SEA_ML was found easy to maintain and add new modeling features and MAS implementation mechanisms to support various agent execution platforms. Furthermore, the participants agreed that SEA_ML's all-embracing model of fundamental agent components enables the system models compatible with the MAS domain and reusable for various applications. Validation of the MAS models before model-to-code transformations and shortening the development time were also acknowledged by the participants.

To conclude, the proposed AHP enabled the comparative evaluation of four MAS DSML alternatives according to the defined set of criteria and sub-criteria. Both the introduced AHP itself and the defined MAS DSML evaluation criteria are generic and can be directly used for the comparative evaluation of DSMLs different from the language group evaluated in this study. According to the evaluators in this study, being appropriate for the MDE of various agent models and MAS architectures is the most desirable feature for a MAS DSML. Moreover, having a complete notation to realize modeling a wide range of autonomous agents, their internal structures and relations with each other is another important property which is favoured by the agent developers to select a MAS DSML. Shortening the MAS development time came third in the prioritization list of the participants. Among the four MAS DSML alternatives, JACK was selected as the best language. SEA_ML was the second most preferred DSML while SAM and PDT did not favored by the developers in the overall assessment. However, these rankings naturally pertain only to this comparison set of the languages and hence they can not be generalized, i.e. within another group of MAS DSML alternatives, maybe JACK can not be the most appropriate one while PDT or SAM can step forth inside another language set.

5.5. Threats to the Validity

As it is the case in any evaluation study, there are also some threats to the validity of the conducted evaluation. First, a relatively limited number of evaluators could participate in the assessment of the MAS DSMLs. Compared to the many other computer science and software engineering disciplines, AOSE is a young research field and hence the number of developers familiar with MAS implementations is relatively low. We observed that recent studies [43], [20], [16], [8] on evaluating model-driven MAS development approaches and/or using MAS DSMLs assigned the number of participants close to the number of participants used in our study. Moreover, various AHP-based software tool and methodology evaluation studies (e.g. [48],[49],[30]) were realized with even smaller number of participants. Hence, the number of the participants in our study can be considered acceptable for the related research field. We paid attention on conducting this evaluation only with the software developers who possess knowledge and experience on programming agents and actively used all four MAS DSMLs being evaluated during the construction of MAS for various domains. It is worth indicating that the majority of the participants had also significant experience in developing large-scale software systems in the industry which contributed to the usability evaluation of all MAS DSML alternatives. Nevertheless, the number of our participants is satisfying considering Nielsen's scale [66] for usability studies.

Second, the participants' previous knowledge and experience on the MAS DSMLs being evaluated here, may affect prioritizing the evaluation criteria and scoring these MAS DSMLs. A developer, who possesses deeper knowledge and more experience in a specific MAS DSML, would most likely favoured (or conversely does not preferred) this DSML. That may cause a rather subjective formalization of the final hierarchy model. For an evaluator group with changing levels of experience, the second questionnaire used in the study can be extended with additional questions enabling the participants evaluating themselves about their prior experience with each of these MAS DSMLs and then a correlation analysis can be performed between these responses and already obtained responses on the selected DSMLs. However, in our case, this risk is already mitigated with creating an evaluator group with almost the same level of knowledge and experience on using all MAS DSMLs being evaluated in this study. As also discussed in Sect. 5.2, all participants previously used all these MAS DSMLs in the same software development projects aiming to design and implement MAS for various domains and scale during the AOSE-related graduate courses they took recently.

Third, the selection of the MAS DSML alternatives may have an influence on the results. As indicated in Sect. 4.1, MAS DSMLs evaluated in this study are well-known in the AOSE community and are being widely used both in academic research and industrial agent system development. The evaluators participated in our study had experience on using all these DSMLs which is also another reason of selecting this alternative set. Moreover, the selected MAS DSMLs are perhaps the most available ones whose IDEs and tools were fully-functional at the time of conducting this study. Other agent modeling languages such as AUML[31] and AML[32] could also be included into this set of DSMLs. However, their tools were not available or working when this study was conducted. Hence, the set of DSMLs selected in our study can be considered a good sample for the evaluation. Finally, we can state that the applied methodology provides the comparative evaluation of the MAS DSMLs, i.e. DSMLs are evaluated inside the selected set and their priorities (showing their adoption level by the evaluators) are determined only according to the set of the current alternatives. Since both the evaluation criteria and the AHP-based evaluation methodology introduced here are independent from the selection of the MAS DSMLs, they can be directly used for the comparative evaluation of any other group of MAS DSMLs.

6. Conclusion

An AHP-based evaluation methodology for the comparative evaluation of MAS DSMLs has been introduced in this paper. For this purpose, a categorized set of criteria which can be used for the multi-criteria decision making has been defined. These criteria can be prioritized by the agent developers according to their modeling language expectations and the application of the methodology allows the evaluation of MAS DSML alternatives based on this prioritization. As the result of the automatic calculation of the importance distributions, the best MAS DSML is determined. All stages of the proposed methodology are supported with an online tool which enables collecting evaluation data via questionnaires, calculating the distributions and finally analyzing and displaying the results. The comparative evaluation of four widely used MAS DSMLs, JACK, SAM, SEA_ML and PDT was performed by using the proposed methodology. The conducted evaluation showed that the agent developers prioritized appropriateness, completeness and shortening the development time as the most significant criteria for the MAS DSML assessment while the attractiveness of the notations had a minimum effect on preferring a language.

Favourite DSML for each comparison category and criteria was determined. In the overall assessment, JACK was selected as the best language especially taking into account the functional suitability and ease of use features. In practice, achieved results suggest that a developer can choose JACK when there is a need to design and implement a MAS with varying agent interaction and execution scenarios in addition to the different agent architectures and behavioral models (such as reactive or BDI) should be taken into consideration during the specification and realization of the agent internals. Developers who use different AOSE methodologies may also benefit from the IDE provided by JACK since our evaluation showed that its IDE is more compatible with the development process of the many existing MAS development methodologies and more importantly modeling with JACK can be easily added as one of the main parts of these AOSE methodologies. Being a commercial product and hence having both professional support and an extensive set of developer resources, examples and user manuals as well as providing continuous updates may also lead selecting JACK during MAS development. Finally, the results also showed that the developers may choose SEA_ML among these DSML alternatives, e.g. when adding new modeling elements or code generation mechanisms is required to extend the support for various MAS implementation platforms. SEA_ML can also be used when the developers prefer working with a rich set of agent concepts with appropriate notations mainly exist for modeling agent plan structures and knowledge-bases, agent and service interactions, overall MAS organizations and agent environments. SEA_ML seems also a good choice for model checking since it provides various automatic controls to eliminate the errors in instance MAS models.

While applying the methodology, there is the possibility of an overall assessment, as well as an assessment based on certain categories or criteria which will be tailored to the needs of the users, e.g. agent software developers. Thus, it is possible to formalize a hierarchical structure which can compare more than one MAS DSML according to all or subset of the defined language criteria. Both the set of DSML evaluation criteria and the proposed methodology are generic and can naturally be used during comparative evaluation of other MAS DSMLs. We plan to evaluate the MDE features of other MAS DSMLs by applying the same methodology and integrate these new results with the existing ones to extend the pool of alternative MAS DSMLs. These new evaluations may also pave the way for enriching the criteria definition and prioritization stages of the methodology, i.e. various AHP hierarchy models can be derived for the MAS DSML assessments.

Finally, additional MCDM techniques can also be applied for comparing MAS DSMLs again with using the same or the extended set of the evaluation criteria we introduce in this paper. For instance, in our another future work, we aim at using the Logic Scoring of Preference (LSP) [67, 68] which is a general decision method for evaluating and selecting various hardware and software systems. Especially, it can be possible to improve the specification of the preference aggregation structure currently created in our methodology with using LSP's stepwise aggregation technique that covers the formalization of the global preference by systematically aggregating the elementary preferences e.g. for MAS DSML selection.

Acknowledgments

This study was funded by the Scientific and Technological Research Council of Turkey (TUBITAK) under grant 115E591.

Conflict of interest

The authors declare no potential conflict of interests.

[1] G. Weiss, Multiagent Systems (Intelligent Robotics and Autonomous Agents series) 2nd edition, The MIT Press, The MIT Press, 2016.

[2] J. Qin, Q. Ma, Y. Shi, L. Wang, Recent Advances in Consensus of Multi-Agent Systems: A Brief Survey, IEEE Transactions on Industrial Electronics 64 (2017) 4972–4983.

[3] B. Lejdel, Negotiation and cooperation between agents for generalizing geographic objects, Journal of Computer Languages 51 (2019) 15–27.

[4] K. Tazi, F. M. Abbou, F. Abdi, Multi-agent system for microgrids: design, optimization and performance, Artificial Intelligence Review 53 (2020) 1233–1292.

[5] F. Zambonelli, A. Omicini, Challenges and Research Directions in Agent-Oriented Software Engineering, Autonomous Agents and Multi-Agent Systems 9 (2004) 253–283.

[6] V. J. Koeman, K. V. Hindriks, C. M. Jonker, Designing a source-level debugger for cognitive agent programs, Autonomous Agents and Multi-Agent Systems 31 (2017) 941–970.

[7] V. Mascardi, D. Weyns, A. Ricci, Engineering Multi-Agent Systems: State of Affairs and the Road Ahead, ACM SIGSOFT Software Engineering Notes 44 (2019) 18–28.

[8] G. Kardas, B. T. Tezel, M. Challenger, Domain-specific modelling language for belief–desire–intention software agents, IET Software 12 (2018) 356–364.

[9] G. Kardas, Model-driven development of multi-agent systems: a survey and evaluation, The Knowledge Engineering Review 28 (2013) 479–503.

[10] M. Mernik, J. Heering, A. M. Sloane, When and how to develop domain-specific languages, ACM Computing Surveys 37 (2005) 316–344.

[11] T. Kosar, S. Bohra, M. Mernik, Domain-Specific Languages: A Systematic Mapping Study, Information and Software Technology 71 (2016) 77–91.

[12] U. Frank, Domain-Specific Modeling Languages: Requirements Analysis and Design Guidelines, in: I. Reinhartz-Berger, A. Sturm, T. Clark, S. Cohen, J. Bettin (Eds.), Domain Engineering, Springer, 2013, pp. 133–157.

[13] C. Hahn, A domain specific modeling language for multiagent systems, in: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems (AAMAS 2008) -Volume 1, International Foundation for Autonomous Agents and Multiagent Systems, 2008, pp. 233–240.

[14] M. Challenger, S. Demirkol, S. Getir, M. Mernik, G. Kardas, T. Kosar, On the use of a domain-specific modeling language in the development of multiagent systems, Engineering Applications of Artificial Intelligence 28 (2014) 111–141.

[15] E. J. T. Gonçalves, M. I. Cortes, G. A. L. Campos, Y. S. Lopes, E. S. Freire, V. T. da Silva, K. S. F. de Oliveira, M. A. de Oliveira, MAS-ML 2.0: Supporting the modelling of multi-agent systems with different agent architectures, Journal of Systems and Software 108 (2015) 77–109.

[16] J. Faccin, I. Nunes, Sam: a Tool to Ease the Development of Intelligent Agents, Engineering Applications of Artificial Intelligence 62 (2017) 195–213.

[17] D. Sredejovic, M. Vidakovic, M. Ivanovic, ALAS: agent-oriented domain-specific language for the development of intelligent distributed non-axiomatic reasoning agents, Enterprise Information Systems 12 (2018) 1058–1082.

[18] S. HoseinDoost, T. Adamzadeh, B. Zamani, A. Fatemi, A model-driven framework for developing multi agent systems in emergency response environments, Software and Systems Modeling 18 (2019) 1985–2012.

[19] G. Kardas, J. Gomez-Sanz, Special issue on model-driven engineering of multi-agent systems in theory and practice, Computer Languages, Systems & Structures 50 (2017) 140–141.

[20] G. Kardas, E. Bircan, M. Challenger, Supporting the platform extensibility for the model-driven development of agent systems by the interoperability between domain-specific modeling languages of multi-agent systems, Computer Science and Information Systems 14 (2017) 875–912.

[21] F. Bergenti, E. Iotti, S. Monica, A. Poggi, Agent-oriented model-driven development for JADE with the JADEL programming language, Computer Languages, Systems & Structures 50 (2017) 142–158.

[22] T. L. Saaty, The Analytic Hierarchy Process, McGraw-Hill, 1980.

[23] O. S. Vaidya, S. Kumar, Analytic hierarchy process: An overview of applications, European Journal of operational research 169 (2006) 1–29.

[24] G. Kou, Y. Peng, G. Wang, Evaluation of clustering algorithms for financial risk analysis using MCDM methods, Information Sciences 275 (2014) 1–12.

[25] F. Dweiri, S. Kumar, S. A. Khan, V. Jain, Designing an integrated AHP based decision support system for supplier selection in automotive industry, Expert Systems with Applications 62 (2016) 273–283.

[26] L. Zhu, A. Aurum, I. Gorton, R. Jeffery, Tradeoff and Sensitivity Analysis in Software Architecture Evaluation Using Analytic Hierarchy Process, Software Quality Journal 13 (2005) 357–375.

[27] C.-C. Huang, F.-Y. Lin, F. Y.-S. Lin, Y. S. Sun, A novel approach to evaluate software vulnerability prioritization, Journal of Systems and Software 86 (2013) 2822–2840.

[28] H. Ma, Z. Hu, L. Yang, T. Song, User feature-aware trustworthiness measurement of cloud services via evidence synthesis for potential users, Journal of Visual Languages & Computing 25 (2014) 791–799.

[29] A. Kaur, K. Kaur, Investigation on test effort estimation of mobile applications: Systematic literature review and survey, Information and Software technology 110 (2019) 56–77.

[30] P. Pandey, R. Litoriya, Software process selection system based on multicriteria decision making, Journal of Software: Evolution and Process (2020). doi:10.1002/smr.2305.

[31] B. Bauer, J. P. Muller, J. Odell, Agent UML: A formalism for specifying multiagent software systems, International Journal of Software Engineering and Knowledge Engineering 11 (2001) 207–230.

[32] R. Cervenka, I. Trencansky, M. Calisti, D. Greenwood, AML: Agent Modeling Language Toward Industry-Grade Agent-Based Modeling, Lecture Notes in Computer Science 3382 (2005) 31–46.

[33] J. Pavón, J. Gómez-Sanz, R. Fuentes, Model driven development of multi-agent systems, in: European Conference on Model Driven Architecture-Foundations and Applications (ECMDA-FA 2006), Springer, 2006, pp. 284–298.

[34] J. Thangarajah, L. Padgham, M. Winikoff, Prometheus design tool, in: Proceedings of the fourth international joint conference on Autonomous

agents and multiagent systems (AAMAS 2005), International Foundation for Autonomous Agents and Multiagent Systems, 2005, pp. 127–128.

[35] J. M. Gascueña, E. Navarro, A. Fernández-Caballero, Model-driven engineering techniques for the development of multi-agent systems, Engineering Applications of Artificial Intelligence 25 (2012) 159–173.

[36] R. Fuentes-Fernández, I. García-Magariño, A. M. Gómez-Rodríguez, J. C. González-Moreno, A technique for defining agent-oriented engineering processes with tool support, Engineering Applications of Artificial Intelligence 23 (2010) 432–444.

[37] G. Ciobanu, C. Juravle, Flexible software architecture and language for mobile agents, Concurrency and computation: practice and experience 24 (2012) 559–571.

[38] S. Demirkol, M. Challenger, S. Getir, T. Kosar, G. Kardas, M. Mernik, A DSL for the development of software agents working within a semantic web environment, Computer Science and Information Systems 10 (2013) 1525–1556.

[39] M. Eysholdt, H. Behrens, Xtext: implement your language faster than the quick and dirty way, in: Proceedings of the 25th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (SPLASH/OOPSLA 2010), ACM, 2010, pp. 307–309.

[40] A. S. Rao, M. P. Georgeff, Decision procedures for BDI logics, Journal of Logic and Computation 8 (1998) 293–343.

[41] Y. Wautelet, M. Kolp, Business and model-driven development of BDI multi-agent systems, Neurocomputing 182 (2016) 304–321.

[42] Y. Wautelet, S. Heng, S. Kiv, M. Kolp, User-story driven development of multi-agent systems: A process fragment for agile methods, Computer Languages, Systems & Structures 50 (2017) 159–176.

[43] M. Challenger, G. Kardas, B. Tekinerdogan, A systematic approach to evaluating domain-specific modeling language environments for multi-agent systems, Software Quality Journal 24 (2016) 755–795.

[44] T. Miranda, M. Challenger, B. T. Tezel, O. F. Alaca, A. Barisic, V. Amaral, M. Goulao, G. Kardas, Improving the Usability of a MAS DSML, in: 6th International Workshop on Engineering Multi-Agent Systems (EMAS 2018), Lecture Notes in Artificial Intelligence, vol. 11375, Springer, 2019, pp. 55–75.

[45] I. Korkmaz, H. Gokcen, T. Cetinyokus, An analytic hierarchy process and two-sided matching based decision support system for military personnel assignment, Information Sciences 178 (2008) 2915–2927.

[46] M. P. Amiri, Project selection for oil-fields development by using the AHP and fuzzy TOPSIS methods, Expert systems with applications 37 (2010) 6218–6224.

[47] F. Samanlioglu, Z. Ayağ, A fuzzy AHP-VIKOR approach for evaluation of educational use simulation software packages, Journal of Intelligent & Fuzzy Systems 37 (2019) 7699–7710.

[48] Y. K. Chiam, M. Staples, X. Ye, L. Zhu, Applying a selection method to choose Quality Attribute Techniques, Information and Software technology 55 (2013) 1419–1436.

[49] G. Zhang, H. Ye, Y. Lin, Quality attribute modeling and quality aware product configuration in software product lines, Software Quality Journal 22 (2014) 365–401.

[50] M. Asadi, S. Soltani, D. Gasevic, M. Hatala, E. Bagheri, Toward automated feature model configuration with optimizing non-functional requirements, Information and Software Technology 56 (2014) 1144–1165.

[51] L. Lu, Y. Yuan, A novel TOPSIS evaluation scheme for cloud service trustworthiness combining objective and subjective aspects, Journal of Systems and Software 143 (2018) 71–86.

[52] A. M. Akbar, A. A. Khan, S. Mahmood, A. Alsanad, A. Guamei, A robust framework for cloud-based software development outsourcing factors using analytical hierarchy process, Journal of Software: Evolution and Process (2020). doi:10.1002/smr.2275.

[53] P. Davidsson, S. Johansson, M. Svahnberg, Using the analytic hierarchy process for evaluating multi-agent system architecture candidates,

in: International Workshop on Agent-Oriented Software Engineering (AOSE 2005), Springer, 2005, pp. 205–217.

[54] T. L. Saaty, How to make a decision: the analytic hierarchy process, European journal of operational research 48 (1990) 9–26.

[55] T. L. Saaty, Fundamentals of the analytic hierarchy process, in: D. L. Schmoldt, J. Kangas, G. A. Mendoza, M. Pesonen (Eds.), The analytic hierarchy process in natural resource and environmental decision making, Springer, 2001, pp. 15–35.

[56] M. Challenger, B. T. Tezel, O. F. Alaca, B. Tekinerdogan, G. Kardas, Development of semantic web-enabled BDI multi-agent systems using SEA_ML: an electronic bartering case study, Applied Sciences 8 (2018) 1–32.

[57] G. Kahraman, S. Bilgen, A framework for qualitative assessment of domain-specific languages, Software & Systems Modeling 14 (2015) 1505–1526.

[58] J. J. Gomez-Sanz, R. Fuentes-Fernandez, Understanding Agent-Oriented Software Engineering methodologies, The Knowledge Engineering Review 30 (2015) 375–393.

[59] T. R. G. Green, M. Petre, Usability Analysis of Visual Programming Environments: A 'Cognitive Dimensions' Framework, Journal of Visual Languages and Computing 7 (1996) 131–174.

[60] T. R. G. Green, A. E. Blandford, L. Church, C. R. Roast, S. Clarke, Cognitive dimensions: Achievements, new directions, and open questions, Journal of Visual Languages and Computing 17 (2006) 328–365.

[61] T. Kosar, N. Oliveira, M. Mernik, M. J. Varanda Pereira, M. Crepinsek, D. da Cruz, P. R. Henriques, Comparing General-Purpose and Domain-Specific Languages: An Empirical Study, Computer Science and Information Systems 7 (2010) 247–264.

[62] N. Howden, R. Rönnquist, A. Hodgson, A. Lucas, JACK intelligent agents-summary of an agent infrastructure, in: 5th International Conference on Autonomous Agents (AGENTS 2001), volume 162, International Foundation for Autonomous Agents and Multiagent Systems, 2001.

[63] L. Padgham, M. Winikoff, Prometheus: A practical agent-oriented methodology, in: B. Henderson-Sellers (Ed.), Agent-oriented methodologies, IGI Global, 2005, pp. 107–135.

[64] K. D. Goepel, AHP Online System - AHP-OS, `https://bpmsg.com/ahp/ahp.php`, 2019. Accessed: 2020-10-30.

[65] K. D. Goepel, Implementation of an online software tool for the analytic hierarchy process (AHP-OS), International Journal of the Analytic Hierarchy Process 10 (2018).

[66] J. Nielsen, How many test users in a usability study, Nielsen Norman Group 4 (2012). URL: `https://www.nngroup.com/articles/how-many-test-users/`.

[67] J. J. Dujmović, A method for evaluation and selection of complex hardware and software systems, in: Proceedings of the 22nd International Conference for the Resource Management & Performance Evaluation of Enterprise Computing Systems (CMG96), 1996, pp. 368–378.

[68] J. J. Dujmović, Continuous Preference Logic for System Evaluation, IEEE Transactions on Fuzzy Systems 15 (2007) 1082–1099.

Tansu Zafer Asici: Investigation, Software, Methodology, Visualization, Writing - Original Draft

Baris Tekin Tezel: Conceptualization, Methodology, Validation, Writing - Original Draft

Geylani Kardas: Conceptualization, Methodology, Supervision, Writing - Original Draft, Writing - Review & Editing

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: