Evaluating and Comparing some Agent-Oriented Software Engineering Methodologies

Rana Khadier Abbass

Abstract

Along with the growing interest in agent applications, there has been an increasing number of agent-oriented software engineering methodologies proposed in recent years. These methodologies were developed and specially tailored to the characteristics of agents. The roles of these methodologies can provide methods, models, techniques, and tools so that the development of agent-based systems can be carried out in a formal and systematic way.

The goal of this paper is to understand the relationship between five key agent-oriented methodologies: Gaia, MaSE, MESSAGE, Prometheus and Tropos. More specially, we evaluate and compare these five...
methodologies by performing a feature analysis, on them, which is
carried out by evaluating the strengths and weaknesses of each
participating methodology using an attribute-based evaluation
framework. This evaluation framework addresses some areas of an agent-
oriented methodology: concepts, modeling language, process and
pragmatics.

1.1 Introduction
The role of software engineering is to provide methodologies (set of
methods, models and techniques) that make it easier to handle the
complexity of the software development process increasing the quality of
the resulting systems. Thus, the role of agent-oriented methodologies is
to assist an agent-based application in all of its life cycle phases [1].
Agent Oriented (AO) methodology aims to prescribe all the elements
necessary for the development of a software system, especially in the
context of commercial applications. Most AO methodologies are in an
early stage and still in the first context of mostly “academic”
methodologies for agent-oriented systems development, although many
of these methodologies have been tested in small, industrial applications
[2].

1.2 Agent Background
An Agent is an entity that is not only designed to run routine tasks
commanded by its users, but also to achieve a proposed setting (or goal)
within the context of a specific environment. The difference between an
Agent and a traditional software entity is that the latter just follows its
designed functions, procedures or macros to run deterministic codes. The
former incorporates the ability to practice intelligence by making
(autonomous/semiautonomous) Decisions based on dynamic runtime
situations [3].

1.3 Agent-Oriented Software Engineering (AOSE)
Software engineering is the application of a systematic, disciplined,
quantifiable approach to the development, operation, and maintenance of
software; that is, the application of engineering to software [4].
By definition, AOSE is the application of agents to software engineering
in terms of providing a means of analyzing, designing, and building
software systems [5].

1.4 Agent-Oriented (AO) Methodologies
Methodologies are the means provided by software engineering
to facilitate the process of developing software and, as a result, to
increase the quality of Software products. By definition, a software
engineering methodology is a structured set of concepts, Guidelines or activities to assist people in undertaking software development [6].

Object-oriented methodologies generally do not provide techniques and model to the intelligent behavior of agents [7]. Therefore, there need to be software engineering methodologies, which are specially tailored to the development of agent-based systems. In answering that demand, there have been an increasing number of agent-oriented methodologies proposed in recent years.

We tend to focus on several prominent agent-oriented methodologies to examine them in depth in order to identify their strengths, weaknesses, domains of applicability as well as commonalities and differences between them. The selection of AO methodologies were based on several factors such as the methodology's significance and relevance with respect to the field of agents, and its available resource such as documentation, tool support, etc.

The five methodologies which were chosen in this research are: Gaia, MaSE, MESSAGE, Prometheus, and Tropos.

1.4.1 Gaia
Gaia is one of the first methodologies which is specially tailored to the analysis and design of agent-based systems. Its main purpose is to provide the designers with a modeling framework and several associated techniques to design agent-oriented systems [8].

1.4.2 Multiagent Systems Engineering (MaSE)
Multiagent Systems Engineering (MaSE) [9] is an agent-oriented software engineering methodology which is an extension of the object-oriented approach.

As a software engineering methodology, the main goal of MaSE is to provide a complete-lifecycle methodology to assist system developers to design and develop a multi-agent system. Similar to Gaia, it also assumes the availability of an initial requirements prior specification to the start of software development under the methodology process [10].

1.4.3 MESSAGE
MESSAGE is the end product of a two-year project hosted by the European Institute for Research and Strategic Studies in Telecommunications (EURESCOM) Methodology as an extension to existing methodologies to allow them to support agent-oriented software engineering [10]. Consequently, the life-cycle model of the Rational Unified Process (RUP) for software development, which provides a
generic software engineering project life-cycle framework, is adapted to MESSAGE [8].

**1.4.4 Prometheus**

The Prometheus methodology is a detailed AOSE methodology, which aims to cover all of the major activities required in the developing agent systems. The aim of Prometheus is to be usable by expert and non-expert users [11].

**1.4.5 Tropos**

Tropos is an agent-oriented software development methodology created by a group of authors from various universities in Canada and Italy. The methodology is designed specifically for agent-based system development. Similar to the other AOSE methodologies we described above, agent-related concepts such as goals, plans, tasks, etc. are included in all the development phases [12].

**2.1 Software Engineering Methodology Evaluation**

Several approaches have been applied to review and classify a large range of agent-oriented methodologies evaluation or to perform comparisons on a small number of methodologies [13]. Unfortunately, such evaluations or comparisons are mostly subjective and are solely based on inputs from a single assessor (or group of assessors).

This research compares and evaluates five AO methodologies (GAIA, MaSE, MESSAGE, Prometheus and Tropos). This section briefly discusses various key methods, techniques and frameworks of the evaluated methodologies. Since object-oriented methodologies are considered as the “predecessor” of agent-oriented methodologies, we also look back at work on evaluating and comparing a number of object-oriented methodologies.

**2.1.1 Methods for Evaluating Methodologies**

We classify the major approaches to methodology valuation into four main groups: Feature-based evaluation, Quantitative evaluation, NIMSAD framework and other evaluation approaches.

**a. Feature-based evaluation**

Feature-based evaluation (also often called Feature Analysis) is the most prominent and popular comparison approach (it has been chosen in this work). It is regarded as a qualitative method [14]. It involves building an evaluation framework that can be represented in terms of a set of properties, qualities, attributes or characteristics [15].

**b. Quantitative evaluation approaches**
Quantitative evaluations assess a methodology according to some measurable results produced by its use. These can be the software applications produced or the changes in the development process [16].

c. The NIMSAD framework
Being an evaluation framework, NIMSAD is not in fact a method for practically and efficiently comparing methodologies. It provides an alternative way of understanding and evaluating methodologies on the basis of the models and epistemology of systems thinking perspective [17].

2.1.2 Comparisons of Object-Oriented Methodologies
Significant work in the area of OO methodology comparisons can be classified into three styles: comparison against a framework (feature analysis), comparison by meta-modeling and comparison by outcome (quantitative evaluation) [18].

3.1 Evaluation Methods
Before proceeding with assessment, one needs to decide what evaluation methods should be used.

3.1.1 The Purpose of Evaluation
1. Getting a better understanding of the nature of AOSE methodologies, including their philosophies, objectives, features, etc.
2. Identify the strengths and weaknesses of the methodologies as well as their commonalities and differences in order to perform classifications and to improve future agent-oriented software system development.

Furthermore, it is emphasized that we are not trying to search for, in an isolation, for a best methodology. We believe that it is not always the case that all the AOSE methodologies are mutually exclusive. In fact, different methodologies may be appropriate to different situations, thus a methodology should be selected on the basis of considering different issues. These influencing factors can be the context of the problem being addressed, the domain, and the organization and its culture. However, we also expect that the evaluation would help in practical choices such as identifying the domains of applicability of each evaluated methodology.

3.1.2 The Evaluation Type and Procedure
There are several factors that may affect the decision of choosing an appropriate type of evaluation and procedure to carry out the evaluation.
These are the available time, the level of confidence we need to obtain in the results of the evaluation, and the cost of the evaluation [16].

The chosen programming language for the evaluation was visual java and the main evaluation procedures that we performed for all the comparisons are described in the algorithm below:

![Evaluation Algorithm Diagram]

Figure 1: Main Evaluation Procedures

### 3.2 The Evaluation Framework

In this section, we describe a methodology evaluation framework within which the feature-based comparison is conducted. The framework consists of a set of criteria which addresses not only classical software engineering attributes but also properties which are uniquely found in AOSE.

In order to avoid using an inappropriate comparison framework, the properties in our framework were derived from a survey of work on comparing AOSE methodologies and on comparing OOSE.
methodologies. The evaluation framework covers four major aspects of each AOSE methodology: Concepts, Modeling language, Process, and Pragmatics. This framework is adapted from various frameworks [17] for comparing Object-Oriented methodologies as shown in figure 1.

![Evaluation Framework](image)

A methodology appears to provide for a feature, i.e. what degree of support seems to be present. For this type of evaluation criteria, we use a judgment scale ranging from 1 to 5, where 1 indicates a low level of support and 5, implies that the methodology provides a high level of support. The other type of evaluation feature indicates what is supported by a methodology. These criteria are marked with the text "Narrative" next to them.

### 3.2.1 Concepts
The concepts related to agents distinguish an agent-based system from other types of systems. Hence, one of the important facets of evaluating agent-oriented methodologies is an examination of the methodologies' support for agent-based systems' characteristics such as autonomy, pro-activeness, reactivity, etc [17].

We divide the agents' properties into two groups: internal features and cooperation features. The former addresses the characteristics that are internal to agents, whereas the latter are concerned with the cooperation process between agents.

### 3.2.2 Modeling Language
The modeling language, also called model or notation, of a methodology provides the foundation of the methodology's view of the world [18].

Based on its constituted components and purposes, we categorize the criteria assessing the modeling language of each methodology into two
groups: usability and technical criteria. Usability criteria reflect the first aim of a modeling language, i.e. providing the way for users to exchange thoughts and ideas. On the other hand, technical criteria aim at the second and third purposes.

3.2.3 Process
In constructing a software system, software engineering also emphasizes the series of activities and steps performed as part of the software life cycle. These activities and steps form the process which assists system analysts, developers and managers in developing software [10].

3.2.4 Pragmatics
In addition to issues relating to notation and process, the choice of an agent-oriented software engineering (AOSE) methodology depends on the pragmatics of the methodology. This can be assessed based on two aspects: management and technical issues.

Management Criteria consider the support that a methodology provides to management when adopting it. They include the cost involved in selecting the new methodology, its maturity and its effects on the current organization business practices [19]. Technical criteria look at a methodology from another angle. We use the following criteria, which are discussed in [20], to evaluate the technical aspect of the methodology's pragmatics.

4.1 Results and Discussion of methodologies concepts evaluation
The results of the evaluation of the five methodologies with respect to their concepts are shown in Table 1. For each methodology, there are various columns. The columns named $A$ contain the responses of the authors of the methodology.

The column named $U$ contains the responses of the user (i.e. the student) of the methodology. The final column of each methodology, named $W$ shows our own assessment. The assessment scale has six possible responses. High" ($H$), Medium" ($M$), Low" ($L$) and None" ($N$) responses indicate the level of support of the methodology for a particular concept. Don't Know" ($DK$) means that the responder is not aware of the methodology's support for this particular concepts, whereas Not Applicable" ($DK$) implies a particular concept is not relevant with respect to the assessed methodology.
Table 1: Comparing methodology's concepts.

<table>
<thead>
<tr>
<th>Concept &amp; properties</th>
<th>MaSE</th>
<th>Prometheus</th>
<th>Tropos</th>
<th>MESSAGE</th>
<th>Gaia</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td>A</td>
<td>W</td>
<td>A</td>
<td>U</td>
</tr>
<tr>
<td>Autonomy</td>
<td>H</td>
<td>M</td>
<td>DK</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>Mental attitudes</td>
<td>L</td>
<td>M</td>
<td>H</td>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td></td>
<td>M</td>
<td>H</td>
<td>M</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td></td>
<td>M</td>
<td>H</td>
<td>M</td>
<td>L</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>L</td>
<td>N</td>
<td>L</td>
<td>M</td>
<td>L</td>
</tr>
<tr>
<td>Proactive</td>
<td>M</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>Reactive</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td></td>
<td>M</td>
<td>H</td>
<td>M</td>
<td>L</td>
<td>M</td>
</tr>
<tr>
<td></td>
<td>L</td>
<td>DK</td>
<td>L</td>
<td>M</td>
<td>L</td>
</tr>
<tr>
<td></td>
<td>M</td>
<td>H</td>
<td>L</td>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td></td>
<td>L</td>
<td>M</td>
<td>L</td>
<td>M</td>
<td>L</td>
</tr>
<tr>
<td>Concurrency</td>
<td>M</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>Situated</td>
<td>M</td>
<td>L</td>
<td>H</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>Framework</td>
<td>M</td>
<td>H</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>L</td>
<td>NA</td>
<td>N</td>
<td>H</td>
<td>M</td>
</tr>
<tr>
<td></td>
<td>L</td>
<td>NA</td>
<td>N</td>
<td>M</td>
<td>L</td>
</tr>
<tr>
<td>Protocols</td>
<td>A</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td></td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td></td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>M</td>
<td>H</td>
</tr>
<tr>
<td></td>
<td>H</td>
<td>M</td>
<td>H</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td></td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td></td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td></td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td></td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>M</td>
<td>H</td>
</tr>
<tr>
<td></td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>L</td>
<td>M</td>
</tr>
</tbody>
</table>

The table shows that; for Internal properties:

1. **Autonomy:** As mentioned earlier, autonomy is commonly regarded as one of the key properties of agents. It differentiates agents from other existing entities such as objects. According to the responses from the survey and our assessment, all of the five agent-oriented methodologies recognize that importance.

2. **Mental attitudes:** Prometheus and Tropos support well (medium to high) the use of mental attitudes (such as beliefs, desires, intentions) in modeling agents' internals.

3. **Pro-activeness and reactivity:** Based on the results, it seems that these two attributes are difficult to measure we received highly varying responses. They seem to be fairly well supported by some of the five methodologies (medium-high for MaSE and Prometheus, mostly high for Tropos). Similarly to mental attitudes, this can be explained by the fact that in these three methodologies agents' goals are captured and so are the execution of plans (i.e. actions or tasks) to achieve these goals.

4. **Concurrency:** In terms of support for concurrency, although the ratings are mostly medium-high and vary considerably, MaSE is probably best with its concurrent task diagrams and communication class diagrams.

5. **Situatedness:** Even though the responses to this criteria with respect to the five methodologies range from medium to high, in our view only Prometheus provides a clear support for modeling the situatedness of
agents. Prometheus has an environment model which represents the environment in which the agents operate.

Also, the table shows that; for social features:

1. **Method for co operations**: The responses from both Tropos and MaSE authors are that the agent cooperation's which these two methodologies support are general and that any type of cooperation's can be captured. However, what we are interested in, in this criterion, is the cooperation modes that are clearly supported by the methodology via provided techniques or models. Only negotiation (i.e. how to manage an acceptable agreement for all agents concerned) and task delegation are directly supported.

2. **Teamwork**: the five methodologies provide support for designing teamwork.

3. **Communication modes**: According to the result of the survey, there was a strong agreement regarding this feature. All of the methodologies provide a wide range of communication modes. More specifically, they support both direct/indirect and synchronous asynchronous communication.

4. **Protocols**: Regarding this criterion, based on the respondents including ourselves, MaSE is clearly a leader with its protocol analyzer.

5. **Communication language**: Since interactions between agents take place at the knowledge level, all of the five agent-oriented methodologies target speech act as a primary communication language. Tropos is even more flexible; according to one of its authors, the methodology does not use any particular communication language.

### 4.2 Results and Discussion of Modeling Language evaluation

The results of the evaluation of the five methodologies with respect to their modeling language similar to the Concept table, shown in table 2, each methodology has a number of columns. Columns named **A** contain the responses of the authors of the methodology. Column **U** contain the responses of the user (i.e. the student) of the methodology. The final column of each methodology, named **W**, shows our own assessment.

If traceability exists between the models provided by a methodology, we stated that The modeling language supports traceability, which is the ability to track dependencies between different models and between models and code.
There are five levels of legitimate answers. Strongly Agree (SA) is the highest positive level, indicating that the respondent is very confident that the methodology supports traceability. An Agree” (A) answer signifies a less confident answer, whereas Neutral” (N) means the judgment does not have an adequate basis. Similarly, Strongly Disagree” (SDA) and Disagree” (DA) are the opposites of Strongly Agree” and Agree” answers respectively as follows.

Table 2: Comparing methodology's modeling language.

<table>
<thead>
<tr>
<th>Modeling &amp; Notation</th>
<th>MaSE</th>
<th>Prometheus</th>
<th>Tropos</th>
<th>MESSAGE</th>
<th>Gaia</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clear notation</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>N</td>
</tr>
<tr>
<td>Semantics well-defined</td>
<td>A</td>
<td>SA</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>Adequate &amp; Expressive</td>
<td>SA</td>
<td>N</td>
<td>A</td>
<td>A</td>
<td>N</td>
</tr>
<tr>
<td>Different views</td>
<td>N</td>
<td>N</td>
<td>A</td>
<td>A</td>
<td>N</td>
</tr>
<tr>
<td>Easy to learn</td>
<td>N</td>
<td>N</td>
<td>A</td>
<td>A</td>
<td>N</td>
</tr>
<tr>
<td>Consistency checking</td>
<td>SA</td>
<td>SA</td>
<td>SA</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>Traceability</td>
<td>SA</td>
<td>SA</td>
<td>SA</td>
<td>A</td>
<td>N</td>
</tr>
<tr>
<td>Reusability</td>
<td>S</td>
<td>SA</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
</tbody>
</table>

The table shows that; for Usability criteria:

1. **Clarity and understandability**: the notations provided by all of the five methodologies are fairly clear and understandable.

2. **Adequacy and expressiveness**: The number of static and dynamic models are a good indicator of this criterion and so is the number of different views represented the target system. Both MaSE and Prometheus model the dynamic aspects of the system and handle protocols well.

3. **Ease of use**: the notations of the five methodologies are easy to learn and use. This also relates to the agreement of the understandability and clarity of the notation as discussed above.

Also, The table shows that; for Technical criteria:

1. **Unambiguity**: Semantics are also well-defined by all the methodologies.

2. **Consistency**: In terms of consistency checking, the level of support differs between methodologies. MaSE and Prometheus support it well whereas MESSAGE, Tropos and Gaia do not appear to support it.

3. **Traceability**: Likewise to consistency, MaSE and Prometheus appear to be the leader in terms of supporting this feature.
4. **Refinement**: Refinement is generally well-supported by all five methodologies.

5. **Reusability**: None of the methodologies explicitly provide techniques, guidelines or models to encourage the design of reusable components. Also, reuse of existing components is also not addressed in any methodology.

### 4.3 Results and Discussion of process component evaluation

The assessment's results of the Process component of the five selected methodologies are summarized in Table 3 and part of Table 4, (for estimating and quality assurance, management decision making guidelines criteria).

The table shows that, for Development principles:

**1. Architectural design and detailed design**: To some extent. Implementation is also discussed in all of them except Gaia. Testing/Debugging is only included in Prometheus and MaSE.
2. **Process steps:** The process steps described in the requirements analysis and design phases are also addressed well in most of the five methodologies.

3. **Supporting development context:** there are several main development contexts such as Green field, Prototyping, Reusing, Reengineering, etc. In addition, taking into account that one of the key issues which determines whether the agent-oriented paradigm can be popular is the degree to which existing software can be.

4. **Estimating and quality assurance guidelines:** Because of the immaturity of agent-oriented methodologies, issues relating to cost estimating or quality assurance are not addressed in all five methodologies. They probably rely on the current software engineering practice of these matters, see Table 4.

4.4 **Results and Discussion of Pragmatics evaluation**

The results of the evaluation of the five methodologies with respect to their pragmatics are shown in table 4. Similar to the Concept table shown in table 1, each methodology has a number of columns. Columns named A contain the responses of the authors of the methodology. Column U contains the responses of the user (i.e. the student) of the methodology. The final column of each methodology (W) shows our own assessment. SA for Strongly Agree, A for Agree, N for Neutral, DA for Disagree, SDA for Strongly Disagree, "-" for no response. The columns named A is the developers of the methodology, the column U is the student, and the column W is our own assessment.

Table 4: Comparing methodology's pragmatics.

<table>
<thead>
<tr>
<th>Pragmatics</th>
<th>MaSE</th>
<th>Prometheus</th>
<th>Tropos</th>
<th>MESSAGE</th>
<th>Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quality guidelines</td>
<td>N</td>
<td>DA</td>
<td>A</td>
<td>DA</td>
<td>DA</td>
</tr>
<tr>
<td>Cost estimation</td>
<td>DA</td>
<td>SA</td>
<td>N</td>
<td>DA</td>
<td>DA</td>
</tr>
<tr>
<td>Management decisions</td>
<td>DA</td>
<td>SA</td>
<td>DA</td>
<td>SDA</td>
<td>DA</td>
</tr>
<tr>
<td>Number apps</td>
<td>21+</td>
<td>21+</td>
<td>-</td>
<td>1-5</td>
<td>1-5</td>
</tr>
<tr>
<td>Real apps</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Used by non-creators</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Domain specific</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>
The table shows that; for Management criteria:

1. **Maturity:** To our knowledge, MaSE was used to design a team of autonomous, heterogeneous search and rescue robots [19]. Tropos was used to develop a web-based broker of cultural information and services for the government of Trentino, Italy [6] and an electronic system called Single Assessment Process to deliver an integrated assessment of health and social care needs for older people [21]. Prometheus was used to design an agent system to perform Holonic Manufacturing [22]. There also has been one application developed using MESSAGE.

2. **Cost:** Regarding the cost of acquiring methodology and tool support, to our knowledge, all of the methodologies are free to access. In our view, the potential cost of training is not large because the methodologies are aim at different levels of expertise.

Also, The table shows that; for Technical criteria:

1. **Domain applicability:** There is no limitation to the application domains where one of the five agent-oriented methodologies can be applied. These domains are suitable to agent-based system, promising to deliver robust, reliable and autonomous software.

2. **Dynamic structure and scalability:** The methodologies do not tell how to deal with the introduction of new components or modules in an existing system. Furthermore, none of the methodologies, as we mentioned earlier, currently support design of open systems.

3. **Distribution:** Overall, all of the methodologies implicitly support distribution.

### Conclusion

1. This paper carries out an evaluation of five prominent agent-oriented methodologies to understand the relationship between them. In particular, our main purposes are: (a) Assessing each methodology’s strengths, weaknesses, and domain of applicability, and (b) Identifying the similarities and differences among them in terms of techniques and models that are necessary or useful in guiding the developing of agent-based systems. Following this comparative analysis, we proposed an initial unification scheme for five key methodologies.

2. All five methodologies provide a reasonable support for basic agent-oriented concepts such as autonomy, mental attitudes, pro-activeness, and
reactiveness. However, there are several characteristics of agent-based systems that are not addressed or sufficiently addressed in most of the methodologies. For instance, none of the five methodologies provide explicit support for designing teamwork in agents.

3. The notation of the five methodologies is generally good. Most of them have a strong modeling language in terms of satisfying various criteria such as clarity and understandability, adequacy and expressiveness, ease of use, and unambiguity. However, there are several exceptions. Tropos was not perceived as being easy to use whilst MESSAGE and GAIA were both ranked weakly on adequacy and expressiveness. In addition, only Prometheus and MaSE provide techniques and tools for maintaining the consistency and traceability between models. For the other three methodologies, there is still more room for improvement with respect to these issues. It is also emphasized that none of the evaluated methodologies explicitly provide techniques, guidelines, or models to encourage the design of reusable components or the reuse of existing components.

References