A Critical Review of the Evaluation of Framework for Agent Oriented Methodologies

BY
1 Omankwu, Obinnaya Chinecherem
   Department of Computer Science,
   Michael Okpara University of Agriculture,
   Umudike Umuahia, Abia State, Nigeria.
   saintbeloved@yahoo.com

2 Anigbogu, S.O.
   Department of Computer Science,
   Nnamdi Azikiwe University,
   Awka Anambra State, Nigeria.
   dranigbogu@yahoo.com

3 Nwagu, Kenneth Chikezie
   Department of Computer Science,
   Nnamdi Azikiwe University,
   Awka Anambra State, Nigeria.
   Nwaguchikeziekenneth@hotmail.com

4 Anigbogu, Kenechukwu,
   Department of Computer Science,
   Anambra State Polytechnic
   Mgbakwu Awka, Nigeria.

Abstract
Multiple agent-oriented methodologies were introduced in recent years. However no systematic evaluation of these was offered. In this work we presented an evaluation framework for agent-oriented methodologies: The review of this evaluation framework focused on four major facets of a methodology, namely: Concepts and Properties, Notations and Modeling techniques, Development Process, and Pragmatics. In analyzing the results, the author recognized that the mentioned facets of methodology need further improvements within the existing agent-oriented methodologies.

1.0 Introduction
In the last decade, many methodologies for developing agent-based systems have been developed. A methodology is the set of guidelines for covering the whole lifecycle of system development both technically and managerially. A methodology, according to Graham et al., (2017), should provide the following: a full lifecycle process; a comprehensive set of concepts and models; a full set of techniques (rules, guidelines, heuristics); a fully delineated set of deliverables; a
modeling language; a set of metrics; quality assurance; coding (and other) standards; reuse advice; and guidelines for project management. The relationships around these components are shown in Figure 1.1. In figure 1.1, the UML notations are used to depict the relationships around the components. As depicted in the said figure, a methodology consists of a set of techniques, a modeling language and a lifecycle process. The set of techniques consists of metrics, quality assurance (QA) activities, a set of standards and tools. The modeling language comprises notations and a meta model. The lifecycle process consists of project management, a number of roles (e.g., an analyst or a designer), a number of procedures (e.g., how to move between development stages), and a number of deliverables (e.g., a design document, source code). In addition, Figure 1.1 also shows that the tools can be based on the Meta model of the modeling technique and they represent the modeling technique’s notations. The deliverables use the modeling technique.

![Figure 1.1. The components of a methodology and the relationships among them](image)

2.0 Literature Review.

At present, more than twenty four agent-oriented methodologies exist. The multiplicity and variety of methodologies result in the following problems (Sturm and Shehory, 2003).

(i) Industrial problem: selecting a methodology for developing an agent-based system/application becomes a non-trivial task, in particular for industrial developers which hold specific requirements and constraints (Cernuzzi and Rossi, 2002);

(ii) Standards problem: multiple different methodologies are counter-productive for arriving at a standard. With no standard available, potential industrial adopters of agent technology refrain from using it (Sturm and Shehory, 2003).
Research problems: excessive efforts are spent on developing agent-oriented methodologies, in times producing overlapping results.

Additionally, as a result of allocating resources to multiple methodologies, no methodology is allocated sufficient research resources to enable addressing all facets and providing full-fledged agent-based methodology (Cernuzzi and Rossi, 2002).

A few evaluation of agent-oriented methodologies have been suggested. In (Yu and Cysneiros, 2002), the authors set a list of questions that a methodology should address. However, neither evaluation nor a comparison has been performed using that set. Another study (Cernuzzi and Rossi, 2002) suggested a framework for evaluating agent-oriented methodologies. That framework uses a set of evaluation criteria to examine methodologies’ expressiveness. However, it does not examine other properties encompassed within the methodology definition. In (Kumar, 2002), the author performed an evaluation of five agent-oriented methodologies, but, referred only to some supported concepts such as organization design and cooperation. He did not refer to the broad set of attributes that constitute a complete methodology. In (Shehory and Sturm, 2001), the authors performed an evaluation of the modeling part within a methodology, while other parts which are concept and property, and pragmatics not evaluated.

In Dam and Winikoff, (2003), three methodologies were compared: MaSE Prometheus and ROADMAP. The comparison was performed by gathering feedback regarding the properties of the methodologies from students that used them, and from the methodologies’ developers. The gathered feedback included several inconsistent answers. This resulted in difficulty in analyzing methodology properties. Many studies that dealt with evaluating agent-oriented methodologies compared two or three methodologies, mainly with respect to the expressiveness of the methodologies and their supported concepts, and not with respect to other software engineering criteria.

The evaluation framework used in this study was based on a feature analysis technique. In other word, the features of each of the examined methodologies were evaluated. The evaluation was performed based on information regarding the examined methodologies available in publications. The framework’s four facets were: concepts and properties, Notations and Modeling Techniques, Development Process, and Pragmatics. These facets, and the metric used in conjunction with them, are introduced below.
3.0 Evaluation of The different Agent Oriented Framework

The evaluation framework used in this work is based on a feature analysis technique. That is, the features of each of the examined methodologies are evaluated. The evaluation is performed based on information regarding the examined methodologies available. The framework’s four facets are: Concepts and Properties, Notations and Modeling Techniques, Development Process, and Pragmatics.

Concepts and Properties

A concept is an abstraction or a notion inferred or derived from specific instances within a problem domain. A property is a special capability or a characteristic (Cernuzzi and Rossi, 2002). This section deals with the question of whether a methodology addresses the basic notions (concepts and properties) of agents and Multi Agent System. The following are the concepts according to which an agent-oriented methodology should be evaluated:

1. Autonomy: is the ability of an agent to operate without supervision;
2. Reactiveness: is the ability of an agent to respond in a timely manner to changes in the environment;
3. Proactiveness: is the ability of an agent to pursue new goals; and
4. Sociality: is the ability of an agent to interact with other agents by sending and receiving messages, routing these messages, and understanding them.

Meanwhile, the following are the building blocks that encompass the basic components of Multi Agent System (MAS). These building blocks are based on the work of Sturm and Shehory, (2003).

1. Agent: is a computer program that can accept tasks, can figure out which actions to execute in order to perform these tasks and can actually execute these actions without supervision. It is capable of performing a set of tasks and providing a set of services.
2. Belief: is a fact that is believed to be true about the world.
3. Desire: is a fact of which the current value is false and the agent (that owns the desire) would prefer that it be true. Desires within an agent may be contradictory. A widely used specialization of a desire is a goal. The set of goals within an agent should be consistent.
4. Intention: is a fact that represents the way of realizing a desire. Some-times referred to as a plan.
5. Message: is a means of exchanging facts or objects between entities.

6. Norm: is a guideline that characterizes a society. An agent that wishes to be a member of the society is required to follow all of the norms within. A norm can be referred to as a rule.

7. Organization: is a group of agents working together to achieve a common purpose. An organization consists of roles that characterize the agents, which are members of the organization.

8. Protocol: is an ordered set of messages that define the admissible patterns of a particular type of interaction between entities.

9. Role: is an abstract representation of an agent’s function, service, or identification within a group.

10. Society: is a collection of agents and organizations that collaborate to promote their individual goals.

11. Task: is a piece of work that can be assigned to an agent or performed by it. It may be a function to be performed and may have time constraints.

**Notations and Modeling Techniques**

Notations are technical system of symbols used to represent elements within a system. A modeling technique is a set of models that depict system at different levels of abstraction and different system’s facets including structural and behavioral facets as stated by Shehory and Sturm, (2001). This section deals with the properties to which methodology’s notations and modeling techniques should adhere. The list of these properties is adopted from Shehory and Sturm, (2001).

1. Accessibility: is an attribute that refers to the ease, or the simplicity, of understanding and using a method. It enhances both experts and novices capabilities of using a new concept.

2. Analyzability: is a capability to check the internal consistency or implications of models, or to identify aspects that seem to be unclear, such as the interrelations among seemingly unrelated operations. This capability is usually supported by automatic tools.

3. Complexity management (abstraction): is an ability to deal with various levels of abstraction (i.e., various levels of detail). Sometimes, high-level requirements are needed, while in other situations, more detail is required. For example, examining the top level design of a Multi Agent System (MAS), one would like to understand which agents are within the system, but not necessarily what their attributes and characterizations are. However, when
concentrating on a specific task of an agent, the details are much more important than the system architecture.

4. Executability (and testability): is a capability of performing a simulation or generating a prototype of at least some aspects of a specification. These would demonstrate possible behaviors of the system being modeled, and help developers determine whether the intended requirements have been expressed.

5. Expressiveness (and applicability to multiple domains): is a capability of presenting system concepts that refers to:
   - The structure of the system;
   - The knowledge encapsulated within the system;
   - The system’s ontology;
   - The data flow within the system;
   - The control flow within the system;
   - The concurrent activities within the system (and the agents);
   - The resource constraints within the system (e.g., time, CPU and memory);
   - The system’s physical architecture;
   - The agents’ mobility;
   - The interaction of the system with external systems; and
   - The user interface specification.

6. Modularity (incrementality): is the ability to specify a system in an iterative incremental manner. That is, when new requirements are added it should not affect the existing specifications, but may use them.

7 Preciseness: is an attribute of disambiguity. It allows users to avoid misinterpretation of the existing models.

A development process is a series of actions that, when performed, result in a working computerized system. This section deals with the process development facet of a methodology. This facet is evaluated by examining the following:

1 Development context: specifies whether a methodology can be used in creating new software, reengineering or reverse engineering existing software, prototyping, or designing for or with reuse components.
2 Lifecycle coverage: specifies what elements of software development are dealt with within the methodology. Each methodology may have elements that are useful in several stages of the development lifecycle. Here, the lifecycle stages are defined as follows: requirements’ gathering, analysis, design, implementation, and testing.

Again having the development stages defined is not sufficient to render a methodology usable. A methodology should further elaborate the activities within the development lifecycle. Providing a detailed description of the activities included in the development lifecycle would enhance the appropriate use of a methodology and increase its acceptability as a well-formed engineering approach and to verify that a methodology provides detailed activity descriptions, we need to examine the details of the development process. This verification can be performed by answering the following questions regarding an evaluated methodology:

1. What are the activities within each stage of a methodology? For example, an activity can be the identification of a role, a task, etc. The methodology may consist of heuristics or guidelines helping the developer to achieve his/her system development goals.

2. Does the process provide for verification? This question checks whether a methodology has rules for verifying adherence of its deliverables to the requirements.

3. Does the process provide for validation? This question checks whether a methodology has rules for validating that the deliverables of one stage are consistent with its preceding stage.

4. Are quality assurance guidelines supplied?

5. Are there guidelines for project management?

Pragmatics
Pragmatics refers to dealing with practical aspects of using a methodology. This section deals with pragmatics of adopting the methodology for a project or within an organization. In particular, the framework suggests examining the following:

1. Resources: These are the (publicly available) publications describing in detail the methodology (e.g., textbooks and papers), users’ groups, training and consulting services offered by third parties and automated tools (CASE tools) available in support of the methodology (e.g., graphical editors, code generators, and checkers).

2. Required expertise: This is the required background of those learning in agent oriented methodology. A distinguishing characteristic of many
methodologies is the level of mathematical sophistication required to fully exploit the methodology. A criterion within the required expertise may check the required knowledge in some discipline (Ardis, et.al. 2012).

3. Language (paradigm and architecture) suitability: This is the level to which the methodology is coupled with a particular implementation language (e.g., object oriented programming language) or a specific architecture (e.g., BDI).

4. Domain applicability: This indicates the level of suitability of a methodology to a variety of domains (e.g., information systems, real-time systems).

5. Scalability: This is the ability of the methodology to be adjusted to handle various application sizes. For example, can it provide a lightweight version for simple problems.

And to enable ranking of the properties examined in the evaluation process, the framework proposes a scale of 1 to 7 with the following interpretations:

1. An indication that the methodology does not address the property.

2. An indication that the methodology refers to the property but no details are provided.

3. An indication that the methodology addresses the property to a limited extent. That is, many issues that are related to the specific property are not addressed.

4. An indication that the methodology addresses the property, yet some major issues are lacking.

5. An indication that the methodology addresses the property, however, it lacks one or two major issues related to the specific property.

6. An indication that the methodology addresses the property with minor deficiencies.

7. An indication that the methodology fully addresses the property.

Thus far, we have described the evaluation framework, its evaluation criteria, and its metric.

**4.0 Summary and Conclusion**

We have reviewed the evaluation of Feature Based Analysis as a framework that examines the various facets of an Agent Oriented methodology. The results of that reviewed showed that Feature based Analysis framework covered the four major Agent Oriented Methodology facets. We summarize the evaluation thus (Ardis, et.al. 2012).
1. Concepts and properties: the autonomy and proactiveness criteria are properly addressed by the framework. The reactivity is lacking in the sense that the connection between events and responses to them is not well specified. The sociality deals with organization rules, but not with multiple organizations or societies structure, nor with role hierarchy. The building blocks coverage is good; but none of the frameworks covers all of them. This coverage varies among the frameworks as a result of their different goals (Ardis, et.al. 2012).

2. Notations and modeling techniques: this facet was addressed to a limited extent. The accessibility of the framework was good even though it requires further enhancements. Current limitations resulted from the multiplicity of models and the use of logic within the specification stages (Ardis, et.al. 2012).

3. Pragmatics: Frameworks were not coupled to a specific programming language or an agent architecture, and therefore can be used for multiple domains. Scalability (i.e., the ability to be adjusted according to a specific project needs) was not supported by the methodologies (Ardis, et.al. 2012).

In conclusion, the examined framework provided an appropriate infrastructure. The framework considered four major aspects of methodologies namely: Concepts, Notations, Process, and Pragmatics. Each of these areas defined proper evaluation criteria regarding the methodology aspects in general and the agent-orientation concepts in particular. This framework can be utilized for identifying the strengths and weaknesses of agent-oriented methodologies, that for selecting methodologies for application development. It can also be used for promoting existing methodologies which may advance the acceptability of agent technology by introducing a mature, well-structured engineering approach.

Again the outcome of this review showed a need for further research and improvements. The agent-oriented methodologies evaluation frameworks, in order to promote and help in arriving at industry-grade methodologies. The evaluation performed in this work provided researchers and practitioners with a detailed framework among other agent-oriented methodology frameworks. The framework used in this study may be utilized by others to evaluate and compare other methodologies as needed.
References


Burmeister, B.,(2010). *Models and Methodology for Agent-Oriented Analysis and Design,* In Working Notes of the KL’96 Workshop on
Agent-Oriented Programming and Distributed Systems, Saarbrücken, Germany.


Cossentino, M., & Potts, C. (2012). A case tool supported methodology for the design of multi-agent systems. In The 2002 International Conference on Software Engineering Research and Practice (SERP’02), Las Vegas (NV), USA.


