Evaluating Agent-Oriented Software Engineering Methodologies

Abdulsalam Alarabeyyat
Information Technology Department (IT)
Faculty of Prince Abdullah Bin Ghazi of Science and Information Technology
Al-Balqa Applied University - Jordan
Email: aw_arabiat@hotmail.com

***Abstract*—This paper contributes to the evolution of Agent Oriented Software Engineering Approach (AOSE) by proposing a new framework called Multilevel Features Analysis Framework (MFAF) for software evaluation in general and Role-Based AOSE methodologies evaluation in particular. MFAF is a general-purpose framework that can be adopted and adapted to evaluate software-related products (e.g., programming languages, software engineering methodologies, software development tools, Interaction communications protocols, etc.). This evaluation framework is applied to a number of role-based AOSE methodologies. In this paper we also document the results of applying MFAF on the CASSIOPEIA method.**

Keywords- AOSE Methodologies Evaluation , Comparing Agent Development Methodologies, AOSE.

I. Introduction

Agent-based computing technology has become one of the most important emerging technologies of the decade [15]. The concept of an agent can be traced back to the early days of research into Distributed AI in the 1970s - indeed, to Carl Hewitts concurrent Actor model [6] and it has been accepted that it came originally from the Artificial Intelligence Community (AI), where an Agent has been defined as an integrated system performing some tasks on behalf of a user. Agents are highly situated, autonomous, interactive software entities that have been introduced as the next significant breakthrough in software development, the new revolution in software [4]. Software agents (often simply termed agents) are critically needed to meet the dynamic changes in information technology, which inhabits vast amounts of discrete information that require complex processing under risky and uncertain conditions in order to extract knowledge and make decisions on behalf of their users. As a consequence, Agent-Oriented Software Engineering (AOSE) methodologies have been developed in order to assist in modeling and developing agent-based systems and applications. Many diverse Agent Oriented Software Engineering (AOSE) approaches and methodologies have been proposed. Each of the methodologies has different strengths and weaknesses, and different specialized features to support different aspects of their intended application domains. The usage of this technology in industry has demonstrated that agent oriented techniques lead to improve in distributed complex system developments. However, the benefits promised by agents technology cannot be fully achieved yet because although many AOSE methodologies have been developed, there is no universally accepted definition of what exactly determines agent systems requirement [16]. Therefore, there is no standardized development method to build agent oriented applications. As a result, it is clear that appropriate Evaluation Framework is needed to evaluate and standardize the common elements of the existing methodologies. As a step towards evaluating the current agent-based methodologies, we propose a new framework called Multilevel Features Analysis Framework (MFAF) for software evaluation in general and Role-Based AOSE methodologies evaluation in particular.

The paper is organized as follows. The following section presents an overview of the current evaluation frameworks. Section 3 and 4 present the Multilevel Features Analysis Framework. Section 5 presents the main features of MFAF. Finally, the paper ends with the conclusions and pointers to future work.

II. Related Work

Q. Tran, G. Low, and M. Williams [14] present a preliminary comparative feature analysis framework, which they used to compare ten AOSE methodologies. They actually considered the three major components of a system development methodology (*process, models* and *techniques*) as defined by one of the most leading public-domain and process-focused full lifecycle methodologies (Object-oriented Process, Environment and Notation (OPEN)). OPEN was mainly designed for the development of software intensive applications, especially object oriented and component-based developments [5].

O. Shehory and A. Sturm [11] carried out an experimental study to compare and evaluate the modeling technique existing in three agent-based methodologies: AOM , ADEPT and DESIRE . Their evaluation criteria assess both generic software engineering features (such as, ease of use and understanding, expressiveness, modularity, analyzeability, complexity management, testability, refinability and open system architecture) and specific agent-oriented features of an AOSE methodology (such as autonomy, complexity and communication). This framework offers a well-defined, structured set of features that an agent-oriented methodology should include. They examined the evaluation criteria via applying a case study that utilizes a single-agent, auction agent, which participates and bids in web-based auctions on behalf of its user to perform a number of tasks to purchase specific items based on some given parameters.

Silva et al. [13] proposed a Non-Functional Requirements (NFR) framework to describe the internal properties of agent-oriented systems and to evaluate the agent-based methodologies based on these properties. The NFR framework is derived from studying and identifying the key properties of agents, such as autonomy, deliberativity, reactivity, sociability, organization and negotiation. Such properties characterize the agent-oriented paradigm, which provides a higher level of abstraction to handle the major features and behaviors of complex software systems.

Dam and Winikoff [2] proposed an attribute-based framework to evaluate a number of agent-based methodologies(MESSAGE, Gaia, MaSE, Tropos and Prometheus). However, they selected only the most prominent three AOSE methodologies to carry out their task, MaSE, Prometheus and Tropos. These methodologies have been chosen since they were presented in the literature with sufficient details, having significant impact on the agent community, and have been developed and used over an extended period of time. After that, the comparison has been made based on four major criteria: *concepts*, *modeling language*, *process* and *pragmatics*. In addition, some business-related and software engineering issues have been considered and added to the evaluation criteria. In fact, the evaluation framework of Dam et al. was mainly based on the properties derived from a number of surveys on comparing Object Oriented (OO) and AOSE methodologies.

O'Malley and DeLoach's [8] framework evaluates both the project requirements and management features of an AOSE methodology. They provide some technique to assist software engineers with the selection of a software engineering methodology based on the criteria provided by their framework.

Sabas et al. [10] presented a framework called MUCCMAS (MUltidimensional framework of Criteria for the Comparison of MAS methodologies) for the comparative analysis of AOSE methodologies. The MUCCMAS framework is defined in terms of six dimensions:

- *The Methodology Dimension* involves a number of evaluating criteria, including models, process phases, development approach, user implication and the availability of the tools that support the methodology.
- *The Representation Dimension* describes the formalisms and principles used during the methodology's modeling phase, such as abstractions levels.
- *The Agent Dimension* characterizes the agents' main properties that determine their social and cooperative behaviors.
- *The Organization Dimension* involves the structure describing how individual agents in MAS are in relation with one another and how they interact to achieve common goals.
- *The Cooperation Dimension* involves a number of evaluating criteria that determine the influence of the agents' social behavior on the overall system performance.
- *The Technology Dimension* describes the characteristics of the potential MAS on which the methodology can be applied, such as application type.

A. General Limitation & Drawbacks of the Current AOSE and OO Evaluation frameworks

Although Agent-Oriented Programming was proposed as early as in 1993 [12], agent abstraction is not clear even today. Part of the reason is attributed to the fact that an agent is far more complicated than an object, which forms the center of object-oriented programming (OOP). So, it is important to recognize and understand the relationship between agent's technology and Object Oriented technology because of the importance and motivation of both technologies in modeling and developing software systems. In fact, there is a real need for both approaches to become integrated, so that agents and objects interact with each other [9]. The general limitations of the current AOSE and OO Evaluation frameworks are as follows:

- Most of the current evaluation frameworks are limited to a specific number of features.
- Most of the evaluation frameworks provide partial feedback when selecting a limited number of evaluation criteria to compare a small number of AOSE methodologies.
- The current evaluation frameworks lack the practical extent to which many other aspects and features must be considered and addressed, such as the software engineering aspects that describe and assess the development process of the individual agents and their environment, features that describe and examine the possibility of future expansions and upgrade-ability, features that assess field history, domain applicability and maturity of the evaluated AOSE methodology.

III. Multilevel Features Analysis Framework(MFAF)

In this section we present a new framework called Multilevel Features Analysis Framework (MFAF) for software evaluation in general and Role-based AOSE methodologies evaluation in particular. MFAF is a general-purpose framework that can be adopted and adapted to evaluate software-related products (e.g. programming languages, software engineering methodologies, software development tools, Interaction communications protocols, Modeling notations, etc.). The proposed evaluation framework addresses some of the problems and disadvantages of the previous approaches that were discussed in the related work section.

A. Fundamental components of MFAF

The MFAF consists of three basic components *levels* , *attributes* and *Metrics*. The framework contains a number of levels, each of which represents one of the major criteria that will be considered when evaluating software. Attributes are the different features pertaining to each criterion to best describe it in terms of definite questions. Metrics are the values that are given to measure the attributes. When applying MFAF, the data upon which we carry out our evaluation are collected through the available documents about each methodology. To enable ranking the properties examined in the evaluation process, we use the same scale in [11] as follows:

- "1" Indicates that the methodology does not address the property.
- "2" Indicates that the methodology refers to the property but no details are provided.
- "3" Indicates 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" Indicates that the methodology addresses the property, yet some major issues are lacking.
- "5" Indicates that the methodology addresses the property, however, it lacks one or two major issues related to the specific property.
- "6" Indicates that the methodology addresses the property with minor deficiencies.
- "7" Indicates that the methodology fully addresses the property.

B. Identifying levels

In this step, we will implement our Framework (MFAF) by first identifying the appropriate levels that describe the major evaluation criteria, as well as their descended attributes. For better explanation, we will utilize a set of multi-level diagrams to model this framework. In this sense, we studied the current role-based methodologies comprehensively to identify the most important and common dimensional characteristics that will be used as evaluation criteria, and we came up with eight major criteria or measures that will be expressed as levels. As a consequence of this step, we came up with the following eight levels:

- L1: Role-related features
- L2: Agency-related features
- L3: Modeling-related features
- L4: Interaction-related features
- L5: Process-related features
- L6: Upgrade-related features
- L7: Application-related features
- L8: Supporting features

IV. EVALUATING CASSIOPEIA

In order to demonstrate the MFAF framework in the following sections we will undertake the evaluation of the CASSIOPEIA methodology. The CASSIOPEIA method is a way to address a type of problem-solving where collective behaviors are put into operation through a set of agents. It is a methodological (bottom-up) approach that distinguishes three main steps for designing a MAS, elementary, relational, and organizational. It is not targeted at a specific type of application nor does it require a given architecture of agents [1], [3].

- The elemental agent behaviors are listed using functional or object oriented techniques. The goal of this step is to identify all the individual roles that are required to achieve the collective task, by grouping together the elementary behavior needed to fulfil the task, and thus determines the individual roles that the agents can play. Agents classes are subsequently defined as sets of the identified roles. Each agent may assign a particular role to act as its "active" role at a given time while other roles are "idle".
- Then the relational behaviors are analyzed, that is, the dependencies between the agents classes are analyzed by using a coupling graph. This step also consists in analyzing the structure of the organization based on the dependencies between the individual roles being considered.
- Finally, the dynamics of the organization structure are described by analyzing the coupling graph. It consists in specifying the organizational roles that will enable the agents to manage the formation and dissolution of the defined groups.

A. Level-1: Role-related features

This level contains attributes that address features involving the internal properties and basic architecture of Roles. The hierarchical structure of this level is shown in Figure 1

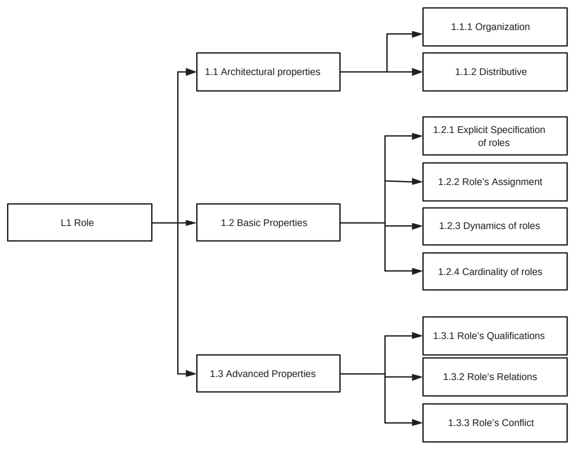

Fig. 1. **Hierarchical structure of level 1**

1) Architecture properties
   - *Role Organization*: Jennings [7] defined organization as the process of identifying and managing the inter-relationships between the various problem-solving components. CASSIOPEIA supports role organization implicitly through the "Definition of Organization Roles" step, which addresses the dynamics of MAS organization by assigning the organizational roles of "group initiator" and "group participant" to different agents. The ranking grade is 5.
   - *Role Distributivity*: Roles in dynamic and open environments could be located in different platforms.

CASSIOPEIA does not support distributive role modeling since they don't offer open system support and they intend to support local and closed MASs. The ranking grade is 1.

2) Basic Properties
   - *Explicit Specification of roles*: CASSIOPEIA supports "The Explicit Specification of roles" via the modeling of agents as entities with purpose (represented as roles, goals, tasks and capabilities). The ranking grade is 4.
   - *Roles Assignment*: CASSIOPEIA supports role assignment but it does't dedicate a special model to represent this process. it does not provide a technique to group roles to identify agent classes and it rely on the developer's intuition and experience.
   - *Dynamic of roles*: It is outside the scope of CASSIOPEIA. The ranking grade is 1.
   - *Cardinality of roles*: CASSIOPEIA supports "Cardinality of roles" via the specification of roles in the interaction model provided at the design phase. The ranking grade is 7.
3) Advanced Properties
   - *Role Qualifications*: CASSIOPEIA provides the basic element to model roles during the development phases. It does not model pre/post conditions of roles in the analysis phase. The ranking grade is 5.
   - *Role Relationships*: CASSIOPEIA supports this attribute via the modeling of roles as the basic element to form the organization or the agent architecture in the target system. The ranking grade is 7.
   - *Roles Conflict*: CASSIOPEIA does not provide any technique to prevent role-tasks conflict. The ranking grade is 1.
   - Roles States: CASSIOPEIA does not provide any method to model roles states such as active roles ,occupied roles and suspended roles. The ranking grade is 1.
   - *Role Operations*: CASSIOPEIA does not provide any method to model roles operations such as classify, declassify, activate, reclassify and shift operations. The ranking grade is 1.

B. Level-2: Agency-related features

This level contains attributes that address features involving the internal properties and basic architecture of agents. A concept is an abstraction or a notation inferred or derived from specific instances, within a process. A property is a special capability or characteristic. With respect to a methodology, the main properties of agency will be considered: autonomy, reactiveness, proactiveness and sociability. The following concepts were evaluated: agent, belief, desire, intention, message, norm, organization, protocol, role, service, society and task. The evaluation will check whether the methodology under study defines the concepts and properties above according with their theoretical meaning, and if not, on what extent these items are fulfilled.

1) Architecture properties
   - *Agents Organization*: CASSIOPEIA were found to support Agent organization through the support of role organization as the building block of the agent organization of the target system via the specification of explicit Architectural /Organizational/Agent models. The ranking grade is 7.
   - *Agents Mobility*: The distribution and Mobility of agents is not supported by CASSIOPEIA. The ranking grade is 1.
2) Basic Properties
   - *Autonomy*: CASSIOPEIA were found to support this attribute via the modeling of agent classes as entities with purpose (represented as roles, goals/tasks and capabilities) or entities with internal control (represented as knowledge/belief, plans and problem solving methods). The ranking grade is 7.
   - *Reactivity*: In CASSIOPEIA, the reactiveness is expressed by the agent's actions in responding to the environments events. The ranking grade is 5.
   - *Reasoning*: Agent reasoning cannot be realized in CASSIOPEIA, since CASSIOPEIA does not address how agents' knowledge (such as agents' plans and actions) relates to agents' ontology knowledge model. The ranking grade is 1.
   - *Life Span*: CASSIOPEIA does not model the agents' life span and it does not specify the active agents nor the active roles during the interaction process. The ranking grade is 1.
   - *Cooperative Behavior*: CASSIOPEIA were found to support this feature via the specification of agent classes interaction in the interaction model. The ranking grade is 7.
3) Advanced Properties
   - *knowledge*: CASSIOPEIA were found to support "Agent's Knowledge" via the specification of agent beliefs/knowledge, Agents plans, goal achievement methods and agent behavioral knowledge. The ranking grade is 7.
   - *Goals/Tasks*. CASSIOPEIA were found to support this feature via the specification of role tasks at the analysis phase. The ranking grade is 7.
   - *Actions*: CASSIOPEIA were found to support this feature via the specification of agent classes actions at the design phase. The ranking grade is 7.

C. Level-3: Modeling-related features

This level includes attributes that address and examine specific features to describe the most common and important aspects in modeling agents. A modeling technique is a set of models that depict a system at different levels of abstraction and different system's aspects. Regarding with notations and models, evaluation will focus on the following aspects:

- *Notation*: CASSIOPEIA has its own notation to model the target system and it provides a clear definition of the semantics and syntax of the notation.
- *Ease of Use and Understanding*: CASSIOPEIA's notation is not a standard notation but it is not hard to understand. The ranking grade is 6.
- *Expressiveness/Completeness*: CASSIOPEIA's notation is not expressive and can't handle a large variety of agent systems due to its domain applicability. The ranking grade is 1.
- *Level of Abstraction*: CASSIOPEIA doesn't offer a level of abstraction. It is considered as step-by-step methodology. These types of methodologies merely present guidelines on what to be modeled and not how these can be represented. The ranking grade is 1.
- *Derivation and Reusability*: CASSIOPEIA has the ability to transfer models into other models and to make reuse of the created models. The ranking grade is 7.
- *Complexity Management*: the modeling notation of CASSIOPEIA contains a techniques that facilitate the decomposition, assignment and management of tasks among agent classes. The ranking grade is 5.

D. Level-4: role interaction-related features

This level addresses features that are related to different possible interactions and interfacing of role classes.

1) Local Interaction
    - *Cooperation, Coordination, Competition* and *Negotiation*: CASSIOPEIA supports these features via the modeling of the interacted agent classes in the interaction model not on the role level. The ranking grade is 5.
    - *Multiple Role Interaction*: CASSIOPEIA doesn't model Multiple Role Interaction since it model only the interacted agent classes. The ranking grade is 1.
2) External Interaction
    - *Interaction with the external environment*, *Sub system Interaction* and *User Interaction*: CASSIOPEIA support these features but at the agents level not at roles level. The ranking grade is 5.
3) Protocol-related features
   CASSIOPEIA does not model *Message Multiplicity, End/Start of Role, Role Status, Role Instances, Roles Constraints* or *Input/Output of roles* in its interaction model. The ranking grade is 1.

E. Level-5: Process-related features

This Level encompasses attributes that address and examine a number of important issues that identify the development process of agents and MAS.

- *Interaction with the external environment*: CASSIOPEIA doesn't Support this attributes since it does not contain an ontology model that facilitate the process of external interactions with agent and non-agent entities.
- *Support for verification and validation*: CASSIOPEIA doesn't provide any method for verification and validation. The ranking grade is 1.
- *Specification of steps for the development process*: CASSIOPEIA provides a full Specification of steps for the development process. The ranking grade is 7.
- *Specification of model types and/or notational components*: CASSIOPEIA were found to support this feature via the specification of the model definition produced at the analysis and design phases. The ranking grade is 7.
- *Definition of inputs and outputs for steps*: CASSIOPEIA provides examples to most of the development steps. The ranking grade is 6.
- *Ease of understanding of techniques and Usability of techniques*: CASSIOPEIA does not use any kind of formal methods to present role and agents task. So it considered as it is easy to follow since the designer does not need to know any formal languages. The ranking grade is 7.
- *Usability of the development process*: CASSIOPEIA's support for reusability is considered to be limited support, because it can't show how heterogeneous components (Agent and Non-Agent) can be reused. The ranking grade is 5.
- *Support for refinability*: CASSIOPEIA supports for refinability via the specification of the notation produced to represent its model types. The ranking grade is 7.

F. Level-6: Upgrading-related features

This level includes attributes that examine some features involving upgrading software agents in order to meet the future expansion needed for potential MAS.

- *Modifiability*: CASSIOPEIA has a limited support for this feature due to its limited support for ontology model and Ontology-Agent Role. The ranking grade is 5.
- *Scalability*: CASSIOPEIA does't support scalability due to its lake in supporting level of abstraction. This methodology merely present guidelines on what to be modeled and not how these can be represented. But it does't present any method for subsystems development. The ranking grade is 1.
- *Dynamic System Support*: CASSIOPEIA has the ability to support dynamic systems since it defines agent's behavior in dynamically manner in forming, joining and dissolving of agent groups. The ranking grade is 5.

G. Level-7: Application-related features

This level includes attributes that address and assess some aspects involving the methodology's applicability in practice, and examine some factors that affect the decision of recommending and adopting a role-based AOSE methodology.

- *Applicability*: CASSIOPEIA is a general-purpose methodology for developing and constructing MAS. The ranking grade is 5.
- *Maturity*: CASSIOPEIA lakes an implementation phase and early requirement phase. the methodology is merely

developed for the analysis and design of MAS. The ranking grade is 5.

- *Field history*: CASSIOPEIA is illustrated by the design of a RoboCup soccer team. The ranking grade is 7.

H. Level-8: Supporting features

This level encompasses three attributes (ontology, security, and collaborative services) that describe additional features for an AOSE methodology.

- *Ontology-support*: In CASSIOPEIA, There is a need for an ontology model. The ranking grade is 1.
- *Security-aspects*: In CASSIOPEIA, there is no security model or even a method and they leave the security aspects for the developer at the implementation phase. The ranking grade is 1.
- *Collaborative Services*: CASSIOPEIA supports the well known FIPA architecture. It can be used to model Fipa-BASED Systems. The ranking grade is 7.

V. Main Features and Advantages of MFAF

The Main Features and Advantages of MFAF over the existing evaluation frameworks presented in the related work section are as follow:

- Compatibility: MFAF completes, integrates and overcomes drawbacks existing in other frameworks. This is because it has been built upon recognizing the most important features of other frameworks, completes any obvious deficiencies, and adopts new features that generalize and extend its usability. As a consequence, this framework is also capable to adopt similar evaluation studies to many cases presented in specialized literature, such as [14], [13], [11], [10], and provide relief for such drawbacks discussed in previous sections.
- Structure: the framework can be represented by an effective hierarchical structure, which derives its power from the principle of 'divide and conquer' that contributes to successfully analyzing a complete taxonomy of evaluation attributes. Moreover, the structure of MFAF allows for computational processing by converting it to any formats, such as, tree structures or Graph structure or Analytical Hierarchy Process structures.
- Scalability: the framework is flexible to scaling up or down in order to expand or reduce its levels and/or attributes.

VI. Conclusion

This paper has proposed a new framework called *Multilevel Features Analysis Framework (MFAF)* for software evaluation in general and Role-based AOSE methodologies evaluation in particular. MFAF is a general-purpose framework that can be adopted and adapted to evaluate software-related products (e.g. programming languages, software engineering methodologies, software development tools, Interaction communications protocols, Modeling notations, etc.). This paper has also documented the process of evaluating CASSIOPEIA , by applying the Multilevel Feature Analysis Framework(MFAF). The results showed that the methodology is lacking in one or more of the following areas of MAS development:

- The CASSIOPEIA's elementary step only specifies roles for each agent, without supporting the agent internal design. The methodology also does not provide any formal set of model types, except for the Coupling Graph which captures agents roles and agents relationships. Moreover, CASSIOPEIA doesn't provide a graphical or formal notation to model MAS within a changing environment.
- There is no systematic method for identifying Role in agent-oriented methodologies.
- Most of the role based methodologies for MAS development doesn't support the requirement elicitation process.
- In the design phase for role based methodologies there is no defined method for allocates roles to agents, especially when there is more than one role to allocate to the same agents.
- There is no method to identify agent type and how to use the identified roles to determine the agent-type.

References

[1] Aose methodologies. Available at http://www.science.unitn.it/ recla/aose/ (accessed March 06, 2008).

[2] K. Dam and M. Winikoff. Comparing agent-oriented methodologies. In *Proceedingsof the 5th Int'l Bi-Conference Workshop on AgentOriented Information Systems (AOIS), Melbourne, Australia, 2003.*, 2003.

[3] A. Drogoul and J. Zucker. Methodological issues for designing multi-agent systems with machine learning techniques: Capitalizing experiences from the robocup challenge, 1998.

[4] C. Guilfoyle and E. Warner. Intelligent agents: The new revolution in software. Technical report, OVUM, 1994.

[5] Henderson-Sellers, A. Simons, and H. Younessi. *The OPEN toolbox of techniques*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1998.

[6] C. Hewitt. Viewing control structures as patterns of passing messages. *Artif. Intell.*, 8(3):323–364, 1977.

[7] N. R. Jennings. On agent-based software engineering. *Artificial Intelligence*, 177(2):277–296, 2000.

[8] S. A. O'Malley and S. A. DeLoach. Determining when to use an agent-oriented software engineering paradigm.

[9] S. Poslad, P. Buckle, and R. Hadingham. Fipa-os agent platform: Open source for open standards. In *the Practical Application of Intelligent Agents and Multi- Agent Systems (PAAM2000)*, pages 355–368, 2000.

[10] A. Sabas, M. Badri, and S. Delisle. A multidimentional framework for the evaluation of multiagent system methodologies. In *the 6 World Multiconference on Systemics, Cybernetics and Informatics (SCI-2002)*, pages 211–216, 2002.

[11] O. Shehory and A. Sturm. Evaluation of modeling techniques for agent-based systems. In *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*, pages 624–631, New York, NY, USA, 2001. ACM.

[12] Y. Shoham. Agent-oriented programming, 1993.

[13] C. Silva, P. Tedesco, J. Castro, and R. Pinto. Comparing agent-oriented methodologies using NFR approach, 2004.

[14] Q.-N. N. Tran, G. Low, and M.-A. Williams. A preliminary comparative feature analysis of multi-agent systems development methodologies. In *AOIS*, Lecture Notes in Computer Science, pages 157–168. Springer, 2004.

[15] M. Wooldridge and P. Ciancarini. Agent-Oriented Software Engineering: The State of the Art. In P. Ciancarini and M. Wooldridge, editors, *First Int. Workshop on Agent-Oriented Software Engineering*, volume 1957, pages 1–28. Springer-Verlag, Berlin, 2000.

[16] F. Zambonelli, N. Jennings, and M. Wooldridge. Organisational abstractions for the analysis and design of multi-agent systems, 2000.