Evaluation of Agent-Oriented Methodologies

written by Arnon Sturm (joint work with Onn Shehory and Dov Dori)

1. INTRODCUTION

The need for methodologies in the domain of agent-based system has been discussed by several studies [2, 3, 13, 14, 15]. For example, according to [13] AOSE is a key factor for introducing agent-based systems to the industry as an engineering approach. During the last decade, there was a blooming of agent-oriented methodologies. More than two dozen methodologies (see Appendix) were developed based on various theoretical grounds, such as, object-orientation and knowledge representation. Yet, only in the last few year the evaluation of these methodologies gains the research community attention, deriving by standardization efforts.

Examining the various studies done on evaluation of agent-oriented methods, we found that there they are lacking in a systematic way of evaluating the methodologies. For example, a comparison among agent-oriented methodologies appears in [2]. That work examines the similarity between the models of the GAIA [23] and MAS-CommonKADS [12] methodologies. Yet, it does not explicitly evaluate these methodologies or provide techniques for doing so. A similar comparison is presented in [8], in which the authors compare between GAIA and MaSE [10] and concludes that MaSE is much more detailed than GAIA. However, they do not address drawbacks of MaSE nor do they provide guidelines for making a comparison between methodologies. The studies presented in [2] and [8] focus on the expressiveness (in terms of agent systems) of the methodologies they examined. However, those studies do not examine the software engineering criteria of the studied methodologies in the comparison they perform. Another effort for setting a basis for evaluating agent-oriented methodologies is presented in [7]. That work checks the level to which some specific qualitative features are present within a methodology using decision trees, it computes the level to which the evaluated method meets the needs of its users. It does so by allocating a weight to each feature of the method, grading each feature, and calculating a weighted average grade. However, the features according to which a methodology should be evaluated are not determined. In [24], the authors suggest an exemplar case study according to which the various methodologies could be evaluated. In addition to the case study, they list a set of questions to be asked about an agent-oriented methodology. However, the questions are somewhat vague and answering these questions may not lead to an understanding of what the right methodology is for a specific project, i.e., there is no framework for evaluating agent-oriented methodologies within that study. Another work on the comparison among agent-oriented methodologies [13] summarizes the advantages and drawbacks of several streams (such as software engineering, formal methods, and knowledge engineering) within the

domain of agent-oriented methodologies. However, it overlooked some of the software engineering aspects of MAS and agent application properties. Additionally, that work did not provide evaluation criteria for assessing advantages and drawbacks of various modeling methods within a specific stream.

In this work, we present our experience using various techniques for evaluating agent-oriented methodologies. We focus on the evaluation techniques rather on their results. We first present a framework for evaluating and comparing agent-oriented methodologies which is based on a set of pre-defined criteria (i.e., features). A second approach we present is a survey which introduces a well-formed questionnaire to undergraduate students and enables analyzing their opinion/experience/understanding of four agent-oriented methodologies. The third approach we present is an experiment done with students, comparing their understanding of MaSE and OPM/MAS [18] using deviation analysis.

The paper is organized as follows. In section 2, we introduce techniques for evaluating methodologies. Section 3 presents a feature-based evaluation framework. In Section 4 we present a survey we performed. Section 5 presents an experiment we conducted and Section 6 concludes.

2. EVALUATION TECHNIQUES

In this section, we present evaluation techniques that can be used for the evaluation of agent-oriented methodologies. Based on [16], the following are the major evaluation techniques for methodologies:

1. **Feature-based comparison** - a comparison that is based on a set of features to be examined. The major shortcoming of this technique is its subjectivity in the selection of features and in the evaluation itself. The advantage of this technique is that it has been applied successfully in many cases and that it can be performed independently of external resources such as an active industry partner.
2. **Meta-modeling** - the meta-modeling approach for evaluating analysis and design methodologies is used as a formal approach to compare methodologies [4]. Such an evaluation in done by (1) building a general meta- model for the ultimate methodology; (2) building a meta-model of the evaluated methodology (or using an exiting meta-model thereof); and (3) comparing the aforementioned meta-models. This technique is similar to the feature-based comparison; however, because the comparison is done in a formal way (i.e., using the meta-model), the comparison itself is objective.
3. **Metrics** - the metrics approach analyzes the complexity of a method. This may consist of analyzing the number of constructs as proposed by [17]. A construct is a building block of the methodology (for example, in UML a class, an object, an association link, a message,

a state and an activity are constructs). This technique is subjective in the decision upon the constructs to be counted and their granularity. In addition, a lot of empirical work is needed to validate the metrics [16].

4. **Ontological evaluation** - the ontological evaluation approach [21, 22] proposes to use ontological concepts to evaluate methodologies. Using the ontological model of Bunge [5, 6], the constructs within a modeling method are mapped to the ontological constructs. The evaluation checks whether the one-to-one mapping between the method and the ontology exists. If there is no such mapping, the method might consists of problems such as construct overload, construct redundancy, construct excess or construct deficit. The advantage of this technique is that an external, objective body defines the required constructs and that the evaluation is required to identify the constructs within the evaluated modeling method. However, doubts were raised whether the choice of the Bunge model for describing a system's ontology is justified [16]. Moreover, there are no guidelines for identifying constructs within a methodology.
5. **Empirical evaluation techniques** - include surveys, laboratory and field experiments, and case studies. These techniques usually require the participation of organizations and practitioners. This is very difficult to achieve and requires many resources. As stated in [16], when applying these techniques, only minor cooperation of the organization and the practitioners was afforded, and this placed a major obstacle.

3. THE EVALUATION FRAMEWORK

In this paper, we refer to a methodology as the entire set of guidelines and activities: 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; guidelines for project management. These are each associated with one of four major divisions: concepts and properties, notations and modeling techniques, process, and pragmatics.

3.1 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. This section deals with the question whether a methodology adheres to the basic notions (concepts and properties) of agents and multi-agent systems. In order to perform such an evaluation we need to define these concepts, since there is no agreement (yet) within the agent community regarding the basic concepts of agent-based systems. In this paper, we leverage on previous studies and utilize notions defined there as a basis for our set of concepts. The following are the properties 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.
4. **Sociality:** is the ability of an agent to interact with other agents by sending and receiving messages, routing these messages and understanding them. In addition, sociality refers to the ability of agents to form societies and be part of an organization.

In addition to the above properties, an evaluation of an agent-oriented methodology should examine whether the methodology supports the following building blocks along their notations and to what extent it does so.

1. **Agent** - A computer program that can accept tasks, can figure out which actions to perform in order to perform these tasks and can actually perform these actions without supervision. It is capable of performing a set of tasks and providing a set of services.
2. **Belief** - A fact that is believed to be true about the world.
3. **Desire** - 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 entity 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** - A fact that represents the way of realizing a desire.
5. **Message** - A means of exchanging facts or objects between entities.
6. **Norm** - 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** - 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** - An ordered set of messages that together define the admissible patterns of a particular type of interaction between entities.
9. **Role** - An abstract representation of an agent's function, service, or identification within a group.
10. **Service**: An interface that is supplied by an agent to the external world. It is a set of tasks that together offer some functional operation. A service may consist of other services.
11. **Society** - A collection of agents and organizations that collaborate to promote their individual goals.
12. **Task** - 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. Sometimes referred to as an action.

3.2 Notations and Modeling Techniques

Notations are a technical system of symbols used to represent elements within a system. A modeling technique is a set of models that depict a system at different levels of abstraction and different system's aspects. This section deals with the properties to which methodology's notations and modeling techniques should adhere. The list of these properties is borrowed and adjusted from [1].

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, 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 definitions.

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.

3.3 Process

A development process is a series of actions, changes, and functions that, when performed, result in a working computerized system. This section deals with the process development aspect of a methodology. This aspect is evaluated using the following issues:

1. Development context: specifies whether a methodology is useful in creating new software, reengineering or reverse engineering existing software, prototyping, or designing for or with reuse components.
2. Lifecycle coverage: Lifecycle coverage of a particular methodology involves ascertaining what elements of software development are dealt with within the methodology. Each methodology may have elements that are useful to several stages of the development life cycle. In this paper, the lifecycle stages are defined as follows:
    - Requirements' gathering is the stage of the lifecycle in which the specification (usually in free text) of the necessities from the system, is done.
    - Analysis is the stage of the lifecycle that describes the outwardly observable characteristics of the system, e.g., functionality, performance, and capacity.
    - Design is the stage of the lifecycle that defines the way in which the system will accomplish its requirements. The models defined in the analysis stage are either refined, or transformed, into design models that depict the logical and the physical nature of the software product.
    - Implementation is the stage of the lifecycle that converts the developed design models into software executable within the system environment. This either involves the hand coding of program units, the automated generation of such code, or the assembly of already built and tested reusable code components from an in-house reusability library.
    - Testing focuses on ensuring that each deliverable from each stage conforms to, and addresses, the stated user requirements.

Having the development stages defined is not sufficient for using a methodology. A methodology should further elaborate the activities within the development lifecycle, in order to provide the user of the methodology with the means of using it properly and efficiently. Providing a detailed description of the various activities during the development lifecycle would enhance the appropriate use a methodology and increase its acceptability as a well-

formed engineering approach. Hence, we suggest to examine the process in a more detailed way. These details can be provided by answering the following questions regarding the 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. It may consist of heuristics or guidelines helping the developer to achieve his/her goals (in developing the system).
2. What deliverables are generated by the process? This question refers mainly to the documentations. For example, what models are specified and can be delivered to the customer. Another example is whether an acceptance testing plan is required and when it is required.
3. Does the process provide for verification? This question checks whether a methodology has rules for verifying adherence of its deliverables to the requirements.
4. 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.
5. Are quality assurance guidelines supplied?
6. Are there guidelines for project management?

3.4 Pragmatics

Pragmatics refers to dealing with practical aspects of deploying and 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 issues:

1. **Resources:** What resources are available in order to support the methodology? Is a textbook available? Are users' groups established? Is training and consulting offered by the vendor and/or third parties? In addition, are automated tools (CASE tools) available in support of the methodology (e.g., graphical editors, code generators, and checkers)? This issue should be examined in order to enable a project/organization aiming at adopting a methodology to check the resources (in terms of training and budget) required and the alternatives for acquiring these.
2. **Required expertise:** What is the required background of those learning the methodology? A distinguishing characteristic of many methodologies is the level of mathematical sophistication required to fully exploit the methodology. Does the methodology assume knowledge in some discipline? This issue should be examined in order to enable a project/organization aiming at adopting a methodology to check whether the qualifications required for using the methodology are met by the candidate users.
3. Language (paradigm and architecture) suitability: Is the methodology targeted at a particular implementation language? That is, is the methodology based on concepts from

a specific architecture or a programming language? For example, a methodology may be limited to BDI-based applications; it may be oriented towards a specific object-oriented language. This issue should be examined to check whether a methodology adheres to the organization/project infrastructure and knowledge.

4. **Domain applicability:** Is the use of the methodology suitable for a particular application domain (e.g., real-time and information systems)? This issue should be examined to check whether the methodology adheres to the intended problem domain.
5. **Scalability:** Can the methodology, or subsets thereof, be used to handle various application sizes? For example, can it provide a lightweight version for simpler problems? This issue should be examined to check whether the methodology is appropriate for handling the intended scale of applications within the project/organization.

It should be noted that the structure of this framework is reinforced by similar research done by Hoa-Dam and Winikoff [9]and Trans et al. [20].

4. THE SURVEY APPROACH

In this section we introduce an application of the survey approach. The purpose of the survey was to analyze users' understandings of the four selected methodologies, namely GAIA, MaSE, Tropos[3] , and OPM/MAS.

4.1 Population Background and Training

The surveyed population consisted of senior undergraduate as well as graduate information systems engineering students. The survey was carried out in the 2003 academic year as an assignment in an advanced course on Agent-Oriented Software Engineering. Fifteen students participated in the course. The students had some prior knowledge on specification and modeling, some of them had prior knowledge on OPM and all of them were familiar with UML, but none of the students had prior knowledge or experience in agents and multi-agent systems.

During the course, which was taught by the author of this paper, the students were exposed to the basics of multi-agent systems. The course included the following topics: introduction to agents and multi-agent systems, agent-oriented software engineering, agent communication and knowledge sharing, agents and MAS architectures, and FIPA specifications. Additionally, each student had to learn, via self-learning, a specific methodology, and then design a system using that methodology. The students also presented in class the methodology assigned to them and their system design. Based on the presentations, each student had to evaluate all of the methodologies and the design documents.

4.2 Survey Design

The survey was based on the assignments given during the course. The students carried out the assignments in couples (though several assignments were submitted by a single student). The survey examined four methodologies: (1) GAIA; (2) MaSE; (3) OPM/MAS; and (4) Tropos. It was based on two case studies: (1) the Personal Itinerary Planning System (PIPS) [11]; and (2) the Guardian Angel System (GAS) [19]. The students were divided into seven groups as follows: GAIA-PIPS, GAIA-GAS, MaSE-PIPS, MaSE-GAS, OPM/MAS-PIPS, OPM/MAS-GAS, Tropos-PIPS.

The first assignment was to analyze a case study and produce a requirements document. Upon completing the requirements analysis, a session that unified the requirements for all groups was performed. The next assignment was self-learning of a specific methodology and designing a MAS system using it. After submitting the design documents, the students were requested to fill in an evaluation form for each one of the four methodologies. They were technically instructed how to fill the form but were not given any guidance as to what answers are expected. The evaluation was done based on the students' class presentations and the design documents they submitted. As part of the evaluation, the students were asked to write a short summary on each methodology. The evaluations were done before the design grades were published, in order to avoid an influence of the grades on the survey results.

4.3 Survey Limitation

The survey we performed may suffer from the following limitations. First, the students' capabilities across the groups were heterogeneous. This diversity was manifested during the presentations and within the submitted design documents, however the assignments and the course grades' standard deviation were low. Second, some of the students were familiar with OPM, but they were not the ones who were assigned to learn and present OPM/MAS. In addition, all of the students were familiar with UML. Since UML was utilized (within the detailed design stage) by the group that was assigned to the Tropos method, and MaSE also borrowed some UML concepts, it could have positively affected the comprehension level of Tropos and MaSE. Third, the survey was conducted by the authors of this paper who developed OPM/MAS, and while all attempts were made to maintain as unbiased an opinion as possible on the capabilities and qualities of the four agent-oriented methodologies examined, there is always the possibility that the students might have received some hidden message that could have affected their judgment and opinion. Fourth, the number of the survey's participants was low, thus statistical analysis is lacking. It would be beneficial to repeat such an experiment by a non-involved third party and with a larger number of subjects.

5. THE EXPERIMENT APPROACH

In this section we introduce an application of the experiment approach. The purpose of the experiment was to analyze student's comprehension of MaSE and OPM/MAS design outcomes and their ability to adjust existing models in both methods.

5.1 Population Background and Training

The experiment population consisted of senior undergraduate in information systems engineering students. The survey was carried out in the 2004 academic year as an exam in an advanced course on Agent-Oriented Software Engineering. Twenty students participated in the course. The students had some prior knowledge on specification and modeling (basically, on UML), but none of the students had prior knowledge or experience in agents and multi-agent systems.

During the course, which was taught by the author of this paper, the students were exposed to the basics of multi-agent systems. The course included the following topics: Introduction to agents and multi-agent systems, agent-oriented software engineering methodologies, agent communication and knowledge sharing, and agents and MAS architectures. In addition, the students were taught MaSE and OPM/MAS and were given home assignments in which they build a system model following each of the methods' guidelines. All in all, MaSE and OPM/MAS were taught for 7 academic hours each and an additional hour (for each method) for feedback on their homework assignments.

5.2 Experiment Design

The experiment was performed during the course exam in which the students were divided into two groups. Each group got the same two case studies, each related to one agent-oriented method, but in opposite order. The case studies were the conference management system and an emergency room system.

Each part (i.e., case study) consists of thirteen right/wrong questions related to the communication, structure, and behavioral aspect of the system, four open question related to the same aspects and two question related to adjusting the models (one related for user intervention, and the other to system behavior).

5.3 Experiment Limitation

The experiment results may suffer from the following limitations. First, the students were familiar with UML which might lead to better understanding of MaSE due to its use of similar ideas. Second, the survey was conducted by the author of this paper who developed OPM/MAS, and while all attempts were made to maintain as unbiased an opinion as possible on the capabilities and qualities of the two methods examined, there is always the possibility that the students might have received some hidden message that could have affected them.

Third, the number of the experiment participants was low, thus statistical analysis might be lacking.

6. SUMMARY

In this paper we present a few techniques for evaluating agent-oriented methodologies. We do not provide guidelines for selecting one of these techniques but present their settings, possible results, and limitation. While considering to perform an evaluation of agent-oriented methodologies there is a need to think about the goal of the evaluation, the methods' suitability for performing it, and take into account the allocated resources for the evaluation.

APPENDIX A – AGENT-ORIENTED METHDOLOGIES

Table 1 lists the most familiar agent-oriented methodologies along with their authors and their category. Possible categories are software engineering (SE) and the knowledge engineering (KE).

Table 1. Agent-oriented methodologies

Method Name	**Authors**	**Category**
AAII	Kinny et al.	KE
Adelfe	Bernon et al.	SE
ADEPT	Jennings et al.	SE
AO	Burmeister	SE
AOR	Wagner	SE
AUML	Odell et al.	SE
Cassiopeia	Collinot et al.	SE
CoMoMas	Glaser	KE
DESIRE	Treur et al.	KE
FAF	d' inverno, Luck	SE
GAIA	Wooldridge et al.	SE
INGENIAS	Pavon et al.	SE
MaSE	Deloach	SE
MAS-CommonKADS	Iglesias et al.	KE
MASSIVE	Lind	SE
MESSAGE	Caire et al.	SE
Nemo	Huget	SE
ODAC	Gervis	SE
OPEN for MAS	Debenham, Henderson	SE
PASSI	Cossentino, Poggi	SE
Prometheus	Padgham, Winikoff	SE
Roadmap	Juan et al.	SE
SADDE	Siera et al.	SE
SODA	Omicini	SE
Styx	Bush et al.	SE
Tropos	Mylopoulos et al.	KE

REFERENCES

[1] M. A. Ardis, J. A. Chaves, L. J. Jagadeesan, P. Mataga, C. Puchol, M. G. Staskauskas and J. Von Olnhausen, "A Framework for Evaluating Specification Methods for Reactive Systems, Experience Report", IEEE Trans. Software Engineering, Vol. 22, No. 6, pp 378-389, June 1996.

[2] P.Bayer, Marcus Svantesson, "Comparison of Agent-Oriented Methodologies Analysis and Design, MAS-CommonKADS versus Gaia", Blekinge Institute of Technology, Student Workshop on Agent Programming, 2001.

[3] Bresciani P., Giorgini P., Giunchiglia F., Mylopoulos J., and Perini A.. "TROPOS: An Agent-Oriented Software Development Methodology". Accepted to the Journal of Autonomous Agents and Multi-Agent Systems, 2003

[4] S. Brinkkemper, S. Hong, and G. van den Goor, "A formal approach to the comparison of object-oriented analysis and design methodologies", in Proc.of the Twenty-Sixth Hawaii Intl. Conf. on , Vol. 4 , pp. 689-698, Jan 1993.

[5] Bunge, M., "Treatise on Basic Philosophy: Vol. 3, Ontology I: The Furniture of the World", Reidel, Boston, June 1977.

[6] Bunge, M., "Treatise on Basic Philosophy: Vol. 4, Ontology II: A World of Systems", Reidel, Boston, April 1979.

[7] L. Cernuzzi and G. Rossi, "On the Evaluation of Agent Oriented Methodologies", in Proc. of the OOPSLA 2002 Workshop on Agent-Oriented Methodologies, November 2002.

[8] M. T. Cox and S. DeLoach, Multiagent systems and mixed initiative planning course, Wright State University, http://www.cs.wright.edu/people/faculty/mcox/Teaching/Cs790/, April 2000.

[9] Dam K. H. and Winikoff M., "Comparing Agent-Oriented Methodologies", in the proceedings of the Fifth International Bi-Conference Workshop on Agent-Oriented Information Systems, 2003.

[10] S. A. DeLoach, M. F. Wood and C. H. Sparkman, "Multiagent Systems Engineering", The Intl. Jour. of SE and KE, Vol. 11, No. 3, June 2001.

[11] FIPA, FIPA Personal Travel Assistance Specification, http://www.fipa.org/specs/fipa00080/XC00080B.pdf, 2001.

[12] Iglesias C. A., Garrijo M., Gonzalez J., and Velasco J. R., "Analysis and Design of multiagent systems using MAS-CommonKADS", Proceedings of the Fourth International

Workshop on Agent Theories, Architectures and Languages (ATAL), LNCS 1365, Springer-Verlag, pp. 313-328, 1998.

[13] Iglesias C. A., Garijo M., and Gonzalez J. C.. "A survey of agent-oriented methodologies". Intelligent Agent V, Proc. of ATAL-98, LNAI 1555, pp. 317-330, Springer, July 1999.

[14] N. R. Jennings and M. Wooldridge, "Agent-Oriented Software Engineering", Handbook of Agent Technology (ed. J. Bradshaw) AAAI/MIT Press, 2000.

[15] N. R. Jennings, K. Sycara and M. J. Wooldridge, "A Roadmap of Agent Research and Development", Jour. of Autonomous Agents and Multi-Agent Systems Vol.1, No .1, pp.7-38, 1998.

[16] K.Siau and M. Rossi, "Evaluation of Information Modeling Methods – A Review", in Proc. 31 Annual Hawaii International Conference on System Science, pp. 314-322, January 1998.

[17] K. Siau and Q. Cao., "Unified Modeling Language: A Complexity Analysis, Journal of Database Management", Vol. 12, No. 1, pp. 26-34, March 2001.

[18] Sturm A., Dori D., and Shehory O., "Single-Model Method for Specifying Multi-Agent Systems", Proceeding of Second International Joint Conference on Autonomous Agents and Multi Agent Systems, pp. 121-128, 2003.

[19] Szolovits P., Doyle J., and Long W.J., "Guardian angel: Patient-centered health information systems", Technical Report TR-604, MIT/LCS, 1994.

[20] Tran Q. N, Low G., and Williams M. A., "A Feature Analysis Framework for Evaluating Multi-agent System Development Methodologies", ISMIS 2003, pp. 613-617. 2003.

[21] Y. Wand and R. Weber, "An Ontological Model of an Information System," IEEE Trans. on Software Engineering, Vol. 16, No. 11, pp. 1282-1292, November 1990.

[22] Y. Wand and R. Weber, "On the Ontological Expressiveness of Information Systems Analysis and Design Grammars," Journal of Information Systems, pp. 217-237, October 1993.

[23] M. Wooldridge, N. R. Jennings, and D. Kinny, "The Gaia Methodology for Agent-Oriented Analysis and Design", Journal of Autonomous Agents and Multi Agent Systems, Vol. 3, No. 3, pp. 285-312, March 2000.

[24] E. Yu and L.M. Cysneiros, "Agent-Oriented Methodologies – Towards A Challenge Exemplar", 4th Intl. Workshop on Agent-Oriented Information Systems (AOIS'02), May 2002.