World Academy of Science, Engineering and Technology
International Journal of Economics and Management Engineering
Vol:2, No:9, 2008

Evaluation Framework for Agent-Oriented Methodologies

Zohreh O. Akbari, and Ahmad Faraahi

***Abstract*—**Many agent-oriented software engineering methodologies have been proposed for software developing; however their application is still limited due to their lack of maturity. Evaluating the strengths and weaknesses of these methodologies plays an important role in improving them and in developing new stronger methodologies. This paper presents an evaluation framework for agent-oriented methodologies, which addresses six major areas: concepts, notation, process, pragmatics, support for software engineering and marketability. The framework is then used to evaluate the Gaia methodology to identify its strengths and weaknesses, and to prove the ability of the framework for promoting the agent-oriented methodologies by detecting their weaknesses in detail.

***Keywords*—**Agent-Oriented Software Engineering, Evaluation Framework, Methodology.

I. Introduction

DUE to the complexity of software development process, wide range of software engineering paradigms have been devised (e.g. structured programming, object-oriented programming, procedural programming and declarative programming)[1]. But recently, with the high rate of increase in complexity of projects associated with software engineering, agent concepts which originated from artificial intelligence have been considered to devise a new paradigm for handling complex systems [1]-[7]. Considering its basic concepts, agent-orientation seems to be a potentially powerful new paradigm. But regarding its newness and complexity and even though many agent-oriented software engineering methodologies have been proposed, few are mature or described in sufficient detail to be of real use.

The best solution to profit from the abilities of agent-orientation is to study the overall of issues on subject, consider each strengths and weaknesses, and derive a powerful methodology. Thus the first step would be to present and make available a powerful evaluation framework.

In this paper methodology is referred to as an economical process of developing software equipped with distinct concepts and modeling tools. In this regard methodologies can be considered in six major aspects: concepts, notation, process, pragmatics, support for software engineering and marketability (this classification had been also proposed previously in [8] to compare object-oriented methodologies). Some previous attempts on this subject have also followed this classification but have ignored the two last areas for different reasons (the overlap of evaluation criteria presented in these two areas with those considered in the other four, the inapplicability of evaluating the marketability of agent-oriented methodologies since they are still in its research process, etc.) [9]-[11]. But regardless of these reasons, the framework presented in this paper will cover the support for software engineering and the marketability areas, to enable the users of the framework to evaluate methodologies from those points of view quite easily and clearly. This complete coverage of these areas somehow looks necessary regarding their interrelated to or trade off factors with the others presented in the four first areas.

Zohreh O. Akbari is a post graduate student at Payame Noor University, Tehran, Iran (e-mail: z.o.akbari@ gmail.com).

Ahmad Faraahi is a faculty member of Payame Noor University, Tehran, Iran (e-mail: afaraahi@ pnu.ac.ir).

This paper has been prepared as an academic analysis submitted under the guidance of Dr. Ahmad Faraahi, professor at Payame Noor University where the author is presently enrolled.

II. The Evaluation Framework

As mentioned in the introduction section, the framework proposed in this paper addresses six major areas of the methodologies:

- Concepts
- Notation
- Process
- Pragmatics
- Support for Software Engineering
- Marketability

The framework defines evaluation criteria for each area regarding the methodological aspects in general and the agent-orientation concepts in particular. Actually each area could be considered by its different general aspects presented by criteria categories then each category broken down to its detailed criteria to enable precise ranking.

A. Concepts

The framework will recognize the concepts of agent-oriented paradigm as the criteria to evaluate methodologies regarding [1], [6], [7] and [9]-[12]. Studying this paradigm, two main groups of concepts is considered: The agent's general concepts and the agent's lateral concepts. The lateral concepts are those sub-concepts which are implemented to derive the general concepts of agents. Fig. 1 describes the complete evaluation criteria categories for concepts aspect.

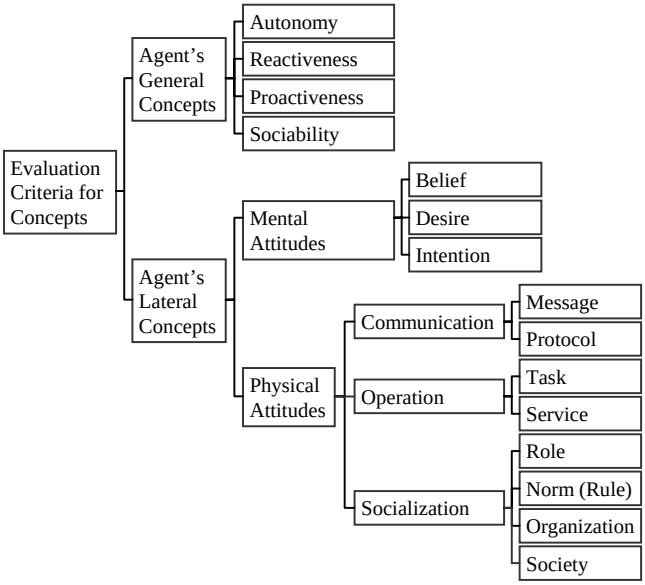

Fig. 1 Evaluation Criteria for Concepts

The followings are the description of the concepts evaluation criteria:

- **Autonomy:** The ability of an agent to operate without supervision.
- **Reactiveness:** The ability of an agent to respond in a timely manner to changes in the environment.
- **Proactiveness:** The ability of an agent to pursue new goals.
- **Sociability:** The ability of an agent to interact with other agents by sending and receiving messages, routing these messages, and understanding them.
- **Belief:** A fact that is believed to be true about the world.
- **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.
- **Intention:** A fact that represents the way of realizing a desire, sometimes referred to as a plan.
- **Message:** A means of exchanging facts or objects between entities.
- **Protocol:** An ordered set of messages that together define the admissible patterns of a particular type of interaction between entities.
- **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.
- **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.
- **Role:** An abstract representation of an agent's function, service, or identification within a group.
- **Norm (Rule):** 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.
- **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.
- **Society:** A collection of agents and organizations that collaborate to promote their individual goals.

B. Notation

Since agent-oriented concepts are the basis for any agent-oriented methodology, modeling techniques which are used for representing these concepts should be also considered in methodology evaluation framework. Fig. 2 describes the complete evaluation criteria categories for notation aspect regarding [8]-[11] and [13].

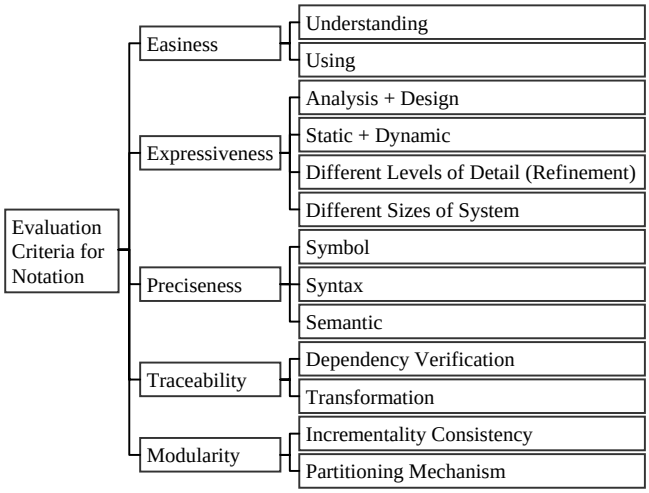

Fig. 2 Evaluation Criteria for Notation

The followings are the description of the notation evaluation criteria:

- **Understanding:** The ease of understanding a modeling technique.
- **Using:** The ease of using a modeling technique.
- **Analysis + Design:** The capability of expressing both analysis and design concepts.
- **Static + Dynamic:** The capability of expressing both static and dynamic concepts.
- **Different Levels of Detail (Refinement):** The ability to deal with various detail levels.
- **Different Sizes of System:** The ability to deal with various system sizes.
- **Symbol:** The preciseness of symbol definition.
- **Syntax:** The preciseness of syntax definition.
- **Semantic:** The preciseness of semantic definition.
- **Dependency Verification:** The ability to trace and verify the dependency between models or between models and code.
- **Transformation:** The ability to transform a model into another model or to transform a model to code.
- **Incrementality Consistency:** 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.

- **Partitioning Mechanism:** The ability to limit the visible information to objects of interest, at a particular time.

C. Process

In software development, software engineering also emphasizes the series of activities and steps performed as part of the software life cycle. To evaluate and compare methodologies, which are derived from life cycles, it is obviously necessary to consider life cycle standard and life cycle coverage of methodology.

According to ESA[1] software engineering standards [14] are divided into three parts:

- **Product Standards**: Contains standards, recommendations and guidelines concerning the product, i.e. the software to be defined, implemented, operated and maintained.
- **Procedure Standards**: Describes the procedures which are used to manage a software project.
- **Appendices:** Contains summaries, tables, forms and checklists of mandatory practices.

In this regard the framework will consider the product standards which actually refer to the standard of life cycle as the life cycle criteria category, and the procedure standards which actually refer to the management plans as management plans criteria category. Fig. 3 describes the complete evaluation criteria categories for process aspect regarding [8]-[11].

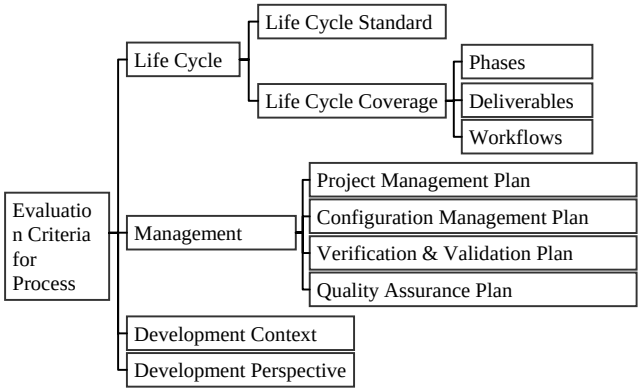

Fig. 3 Evaluation Criteria for Process

The followings are the description of the process evaluation criteria:

- **Life Cycle Standard:** The life cycle standard of the methodology.
- **Phases:** The phases of the life cycle covered by the methodology.
- **Deliverables:** The deliverables of the life cycle covered by the methodology.
- **Workflows:** The workflows of the life cycle covered by the methodology.
- **Project Management Plan:** The plan for software project management.
- **Configuration Management Plan:** The plan for software configuration management.
- **Verification & Validation Plan:** The plan for software verification & validation.
- **Quality Assurance Plan:** The plan for software quality assurance.
- **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.
- **Development Perspective:** Specifies the development perspective of the methodology (Top-Down, Bottom-Up, Iterative, etc.).

D. 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. Fig. 4 describes the complete evaluation criteria categories for pragmatics aspect regarding [8]-[11].

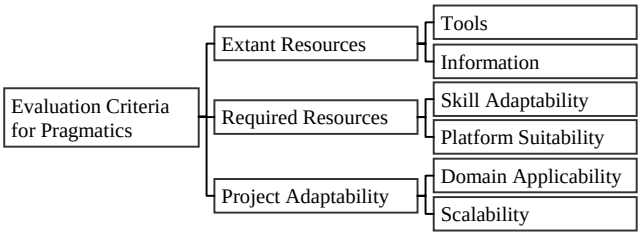

Fig. 4 Evaluation Criteria for Pragmatics

The followings are the description of the pragmatics evaluation criteria:

- **Tools:** The tools available to support the methodology (CASE-tools).
- **Information:** The information available to support the methodology (documents, trainings, consulting services).
- **Skill Adaptability:** The new skills required to learn the methodology.
- **Platform suitability:** The platform suitability (computer architecture, OS, programming language, runtime libraries, graphical user interface, etc.).
- **Domain Applicability:** The usability of the methodology for different application domains (e.g. real-time and information systems).
- **Scalability:** The usability of the methodology for different application sizes.

E. Support for Software Engineering

This area considers those factors presented by a methodology to support software engineering itself. Fig. 5 describes the complete evaluation criteria categories for support for software engineering aspect regarding [8].

[1] European Space Agency

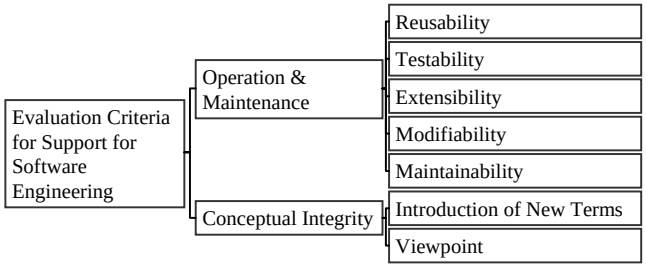

Fig. 5 Evaluation Criteria for Support for Software Engineering

The followings are the description of the support for software engineering evaluation criteria:

- **Reusability:** The extent to which a module can be used in multiple applications.[2]
- **Testability:** The degree of ease with which a software product can be examined with the intention of finding errors.
- **Extensibility:** The attribute of something that allows it to last or continue, or to be expanded in range or scope.
- **Modifiability:** The extent to which something facilitates its changes.
- **Maintainability:** The measure of the ease with which changes necessitated by new requirements, error corrections, new environments, and enhancements, may be introduced into a product**.**
- **Introduction of New Terms:** The extent to which an introduction of term not commonly shared with other methodologies can happen.
- **Viewpoint:** Determining the viewpoint is a little like conducting a psychological profile on a methodology to determine the attitude of the methodologist.

F. Marketability

The marketability of something is a measure of how easily it can be sold, introduced, and adopted by an organization. In this area criteria can be categorized considering the individuals involved with the project. Fig. 6 describes the complete evaluation criteria categories for marketability aspect regarding [8].

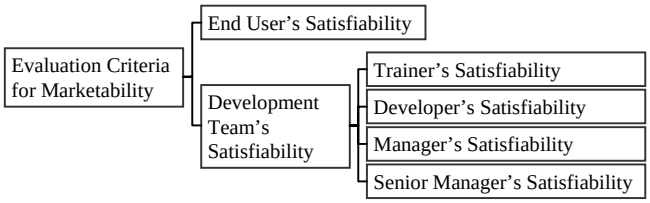

Fig. 6 Evaluation Criteria for Marketability

The followings are the description of the marketability evaluation criteria:

- **End User's Satisfiability:** How quickly can this approach deliver end user's application or system and how much will it cost? In this case the process elements of the method are important.
- **Trainer's Satisfiability:** How difficult will it be to train staff in using this approach, and what kind of support is available? For trainers, the pragmatics is important.
- **Developer's Satisfiability:** How will developers' existing skills be affected? Will developers need to learn a few, or many new skills and tools? Will developers' position become obsolete? What will be the impact on developers' status in the organization? Developers will be interested in the notation, the concepts, and the process elements primarily.
- **Manager's Satisfiability:** What is the impact on the project plans, and how to recruit or train people to use this approach? The manager is also interested in the process element, as well as the software engineering attributes.
- **Senior Manager's Satisfiability:** How much risk is there in adopting this approach? Can the project leverage investments in existing skills and tools, or must the senior manager start from scratch? What is the payback of the adoption of this approach? The senior manager will be concerned with the process and pragmatics elements. Process elements provide visibility into the development effort, while pragmatics elements provide outside support.

III. Metric

Proposing the framework's different areas and criteria, to enable ranking the properties examined in the evaluation process, the framework proposes a scale of 0 to 6 as follows according to [8] and [10]:

0: Indicates that the methodology does not address the property.

1: Indicates that the methodology refers to the property but no details are provided.

2: 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.

3: Indicates that the methodology addresses the property, yet some major issues are lacking.

4: Indicates that the methodology addresses the property, however, it lacks one or two major issues related to the specific property.

5: Indicates that the methodology addresses the property with minor deficiencies.

6: Indicates that the methodology fully addresses the property.

IV. Evaluation of Gaia

This section will demonstrate the use of the evaluation framework presented in Section II, by performing an evaluation on Gaia as an agent-oriented methodology. The evaluation is performed refer to [15] and [16], written by the methodology designers.

A. Concepts

Table I describes the results of evaluating Gaia with the evaluation criteria for concepts.

[2] defined by the IEEE Standard Glossary

TABLE I
EVALUATION OF GAIA CONCEPTS

Criteria	Grade	Justification
Autonomy	6	Expressed by functionality encapsulation of role and autonomy in making decisions.
Reactiveness	2	Expressed by liveness property of role's responsibility (but this does not specify the occurrence of events and role's reactions).
Proactiveness	6	Expressed by role's responsibility.
Sociability	6	Expressed by organization and rule.
Belief	0	Belief is not presented and its influences are implemented by other concepts.
Desire	0	Desire is not presented and its influences are implemented by other concepts.
Intention	0	Intention is not presented and its influences are implemented by other concepts.
Message	6	Expressed by protocol.
Protocol	6	Expressed by protocol.
Task	6	Expressed by activity and responsibility.
Service	6	Expressed by activity and responsibility.
Role	6	Expressed by role.
Norm (Rule)	6	Expressed by following concepts in Gaia: Safety (Static), Liveness (Dynamic)
Organization	6	Expressed by organization.
Society	6	Expressed by several organizations coexisting in a system which have autonomous interactions.

B. Notation

Table II describes the results of evaluating Gaia with the evaluation criteria for notation.

TABLE II
EVALUATION OF GAIA NOTATION

Criteria	Grade	Justification
Understanding	4	Models are easy to understand but the logical expressions can be confusing.
Using	4	Models are easy to use but the logical expressions can be confusing.
Analysis + Design	5	Phases are concentrated on analysis and design, and have distinct models to express their own concepts.
Static + Dynamic	5	Expressed by safety and liveness properties of roles and rules.
Refinement	4	Gaia refines analysis models at design phase.
Different Sizes of System	3	Gaia is more suitable for small and medium systems.
Symbol	4	Modeling provides precise symbols, yet there are still improvement possibilities.
Syntax	4	Modeling provides precise syntax, yet there are still improvement possibilities.
Semantic	4	Modeling provides precise semantics, yet there are still improvement possibilities.
Dependency Verification	0	Gaia does not perform any operations for dependency verification.
Transformation	4	Models delivered from analysis phase are transformed to low level models in design phase, to be implemented.
Incrementality Consistency	3	Gaia is mostly modular due to its design in building blocks (assigning new roles to agent has no effect on internal model but changes within protocol might cause changes in the internal structure of the role).
Partitioning Mechanism	3	Gaia supports different levels of detail and design in building blocks.

C. Process

Table III describes the results of evaluating Gaia with the evaluation criteria for process.

TABLE III
EVALUATION OF GAIA PROCESS

Criteria	Grade	Justification
Life Cycle Standard	6	Gaia could be mapped to the life cycle introduced by ESA.
Phases	2	Gaia covers three phases of the life cycle introduced by ESA: Software Requirements, Architectural Design and Detailed Design (with no reference to implementation).
Deliverables	6	Both phases of Gaia (analysis and design) have deliverables (models).
Workflows	3	Each phase in Gaia has distinct activities but limited guidelines.
Project Management Plan	0	Gaia does not provide any management plans.
Configuration Management Plan	0	Gaia does not provide any management plans.
Verification & Validation Plan	0	Gaia does not provide any management plans.
Quality Assurance Plan	0	Gaia does not provide any management plans.
Development Context	5	Gaia can be used in creating new software, reengineering and designing systems with reuse components; however it does not support classical reverse engineering (from code to model) and prototyping, since it does not address implementation aspects
Development Perspective	3	Gaia is suitable for top-down designs but not for iterative ones.

D. Pragmatics

Table IV describes the results of evaluating Gaia with the evaluation criteria for pragmatics.

TABLE IV
EVALUATION OF GAIA PRAGMATICS

Criteria	Grade	Justification
Tools	0	Gaia does not provide automated tools.
Information	2	Gaia information resources are very limited.
Skill Adaptability	1	Gaia requires solid background and new concepts (logic) also need to be learnt. This causes a reduction in its accessibility since many developers do not know or do not want to get familiar with logic.
Platform	6	Gaia does not refer to the implementation issues, thus is not limited to specific language of platform.
Domain Applicability	3	Gaia is not suitable for developing applications with dynamic characteristics.
Scalability	3	Gaia does not support the use of subsets, yet it may fit different application sizes due to its simple structure.

E. Support for Software Engineering

Table V describes the results of evaluating Gaia with the evaluation criteria for support for software engineering.

TABLE V
EVALUATION OF GAIA SUPPORT FOR SOFTWARE ENGINEERING

Criteria	Grade	Justification
Reusability	3	Gaia supports reusing components.
Testability	0	Gaia does not perform any operations for testing.
Extensibility	3	Gaia support for changes is limited to specific changes and extensions.
Modifiability	2	Gaia does not fully support changes (changes within the protocol might cause changes within the internal structure of the role).
Maintainability	1	Gaia does not fully support changes and is not suitable for dynamic systems.
Introduction of new terms	4	Terminology and notation are almost close to that of object-oriented.
Viewpoint	3	In general Gaia implements strong agent-orientation concepts and such a powerful agent-oriented methodology.

F. Marketability

Table XII describes the results of evaluating Gaia with the evaluation criteria for marketability.

TABLE VI
EVALUATION OF GAIA MARKETABILITY

Criteria	Grade	Justification
End User's Satisfiability	2	Gaia is not suitably introduced yet, thus it has not won the full trust of the end users.
Trainer's Satisfiability	2	Gaia information resources are very limited and new concepts (logic) should be taught.
Developer's Satisfiability	3	Gaia requires solid background and new concepts also need to be learnt, however developers would be profited by the agent-orientation concepts and Gaia agilities in solving the problems.
Manager's Satisfiability	1	Gaia does not provide any project management plans.
Senior Manager's Satisfiability	1	Gaia does not provide any project management plans and still has a very limited support.

V. CONCLUSION

In this paper, an evaluation framework is proposed for agent-oriented methodologies. The framework considers six major aspects of methodologies: concepts, notations, process, pragmatics, support for software engineering and marketability. Each of these areas defines 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, thus can be utilized for selecting methodologies for application developing or promoting existing methodologies which may advance the acceptability of agent technology by introducing a mature, well-structured engineering approach.

This paper also demonstrates the use of the evaluation framework by performing an evaluation on Gaia. Gaia methodology is justifiably considered as an advanced agent-oriented methodology, but due to the results of evaluation performed in section IV, there are several aspects in which the Gaia methodology can be improved, to provide a more powerful methodology. The parallel result of this evaluation is that the framework presented in this paper can improve existing agent-oriented methodologies by detecting their weaknesses in detail.

Although the ranking grades of the framework may vary across evaluators, the overall evaluations will be similar due to the well-defined criteria and the ranking scale.

REFERENCES

[1] N. R. Jennings and M. Wooldridge, "Agent-Oriented Software Engineering", In Handbook of Agent Technology (ed. J. Bradshaw) AAAI/MIT Press, 2000.
[2] Y. Shoham, "Agent-Oriented Programming", Technical Report STAN-CS-1335-90, Computer Science Department, Stanford University, Stanford, CA 94305, 1990.
[3] Y. Shoham, "Agent-oriented programming", Artificial Intelligence, 60(1):51-92, 1993.
[4] M. R. Genesereth and S. P. Ketchpel, "Software agents", Communications of the ACM", Vol. 37, NO 7, 48-53, July 1994.
[5] N. R. Jennings and M. Wooldridge, "Agent technology: foundations, applications and markets", Spring Verlag, 1998.
[6] N. R. Jennings and M. Wooldridge, "Software agents", IEE Review, pp 17-20, January 1996.
[7] M. Wooldridge, "Agent-based Software Engineering", *IEE Proceedings on Software Engineering*, 144(1):26–37, February 1997.
[8] E. V. Berard, "A comparison of object-oriented methodologies", Technical Report, Object Agency Inc., 1995.
[9] K.H. Dam and M. Winikoff, "Comparing agent-oriented methodologies", Proceedings of the 5th Int'l Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS), Melbourne, Australia, 2003.
[10] A. Sturm and O. Shehory, "A Framework for evaluating agent-oriented methodologies", Workshop on Agent-Oriented Information System (AOIS), Melbourne, Australia, 2003.
[11] Jan Sudeikat, Lars Braubach, Alexander Pokahr, and Winfried Lamersdorf. "Evaluation of agent-oriented software methodologies: Examination of the gap between modeling and platform", Proceedings of the Workshop on Agent-Oriented Software Engineering (AOSE), New York, USA, July, 2004
[12] M. Wooldridge and N. R. Jennings, "Intelligent agents: Theory and practice", *The Knowledge Engineering Review*, 10(2):115–152, 1995.
[13] U. Frank, "Evaluating modeling languages: relevant issues, epistemological challenges and a preliminary research framework, Technical Report 15, Arbetsberichte des Instituts fuer Wirtshaftsinformatik (Universitt Koblenz-Landau), 1998.
[14] ESA PSS-05-0 Issue 2, February 1991.
[15] 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.
[16] F. Zambonelli, N. R. Jennings, M. Wooldridge, "Organizational rules as an abstraction for the analysis and design of multi-agent systems", Intl. Jour. of SE and KE, Vol. 11, No. 4, pp. 303-328, April 2001.