A Preliminary Comparative Feature Analysis of Multi-agent Systems Development Methodologies

Quynh-Nhu Numi Tran[1], Graham Low[1], Mary-Anne Williams[2]

[1] School of Information Systems, Technology and Management
The University of New South Wales
New South Wales, Australia
{numitran, g.low}@unsw.edu.au
[2] Innovation and Technology Research Laboratory
Faculty of Information Technology, University of Technology Sydney
New South Wales, Australia
Mary-Anne@it.uts.edu.au

Abstract. While there are a considerable number of software engineering methodologies for developing multi-agent systems, not much work has been reported on the evaluation and comparison of these methodologies. This paper presents a comparative analysis of five well-known MAS-development methodologies. The comparison is based on a feature analysis framework published previously [1]. This framework allows the comparative analysis to be made on a variety of evaluation criteria, covering both agent-oriented aspects and system engineering dimensions. The analysis also compares the methodologies in terms of their support for the steps in the development process, and for agent-oriented concept modeling.

1 Introduction

Compared to the preceding efforts in system engineering such as object-oriented (OO) paradigm, the work in agent-oriented (AO) system engineering is still underdeveloped. However, with the rapid growth and promise of the agent technology, a number of methodologies for developing MAS (denoted as "MAS methodologies") have been proposed in recent years. This has in turn led to the need to evaluate and compare them, thereby noting their strengths and weaknesses, and determining which methodology to use in a particular application.

In a previous publication, we have proposed an evaluation framework for assessing MAS methodologies [1]. Based on the feature analysis approach, this framework provides a list of evaluation criteria or methodological features to be used as yardsticks to assess MAS methodologies from different dimensions and aspects. This paper presents an application of this framework to five well-known MAS methodologies: MASE [2], GAIA [3][4], methodology for systems of BDI agents [5], Prometheus [6],

In Proceedings of the 6th International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS-2004)

and MAS-CommonKADS [7]. The objective is to obtain a comparative analysis of the five methodologies, rather than a detailed analysis of each.

The remainder of the paper is organized as follows: section 2 provides an overview of the feature analysis framework while section 3 gives a summarized description of the five MAS methodologies. We present the comparative analysis in section 4, and some conclusions and perspectives in section 5.

2 Feature Analysis Framework

The framework proposed in [1] was developed from the synthesis of various existing feature analysis frameworks, including those for evaluating conventional system development methodologies – namely [8], [9], [10] and [11], and those for evaluating MAS methodologies – namely [12], [13], [14] and [15]. The framework therefore improves on the existing work by extensively assessing both agent-specific (or MAS specific) and generic system engineering dimensions. It also pays attention to all three major components of a system development methodology – i.e. process, techniques and models. The framework's evaluation criteria are considered representative, case-generic, and centered on the capabilities and usefulness of a MAS methodology.

The structure of the framework is shown in figure 1. It is comprised of four components [1]:

- *Process Related Criteria*: 15 criteria that assess a methodology's support for the MAS-development process.
- *Technique Related Criteria:* 5 criteria that examine the methodology's techniques to develop MAS.
- *Model Related Criteria*: 22 criteria that evaluate the capabilities of the methodology's models.
- *Supportive Feature Criteria*: 8 criteria that evaluate various high-level methodological capabilities.

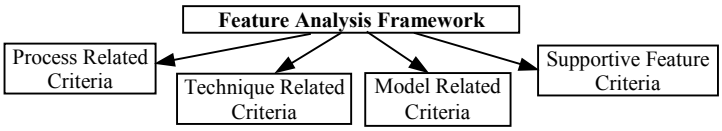

Fig. 1. Structure of the adopted feature analysis framework

Each criterion in the framework is accompanied by an evaluation question (Table 1). Two criteria, "*Steps in the development process"* (in Process Related Criteria) and "*Concepts"* (in Model Related Criteria), which respectively examine the development steps supported by a MAS methodology, and the concepts that the methodology's models are capable of expressing, require a more comprehensive assessment. Tran et al. [1] proposed a list of "standard" process steps and concepts that serve as a checklist for this assessment (Tables 3 and 4).

Table 1. Feature analysis framework for evaluating MAS-development methodologies [1]

Process Related Criteria
1. **Development lifecycle:** What development lifecycle best describes the methodology (e.g. waterfall)?
2. **Coverage of the lifecycle:** What phases of the lifecycle are covered by the methodology (e.g. analysis, design, implementation…)?
3. **Development perspective:** What development perspective is supported (i.e. top-down, bottom-up, or hybrid)?
4. **Application domain:** Is the methodology applicable to a specific or multiple application domains?
5. **Size of MAS:** What size of MAS is the methodology suited for?
6. **Agent nature:** Does the methodology support agents of any type (i.e. heterogeneous agents), or of a particular type (i.e. homogeneous agents)?
7. **Support for verification:** Does the methodology contain rules to allow for the verification and validation of correctness of developed models and specifications?
8. **Steps in the development process:** What development steps are supported by the methodology?
9. **Notational components:** What models and diagrams are generated from each process step?
10. **Comments on the overall strengths/weaknesses of each step:** This criterion allows the evaluator to record any comments on a process step that cannot be recorded anywhere else.
11. **Ease of understanding of the process steps:** Are the process steps easy to understand?
12. **Usability of the methodology:** Are the process steps easy to follow?
13. **Definition of inputs and outputs:** Are inputs and outputs to each process step defined, with possible examples?
14. **Refinability:** Do the process steps provide a clear path for refining the methodology's models through gradual stages to reach an implementation, or at least for clearly connecting the implementation level to the design specification?
15. **Approach towards MAS development:** what is the methodology's a. Generic MAS development approach (e.g. OO-based or knowledge-engineering based)? b. Approach towards using "role" in MAS development? c. Approach in role identification, if the methodology uses "role" in MAS development?
Technique Related Criteria
16. **Availability of techniques and heuristics:** a. What are the techniques to perform each process step? b. What are the techniques to produce each notational component (i.e. modeling techniques)?
17. **Comments on the strengths/weaknesses of the techniques:** This criterion allows the evaluator to record any comments on the techniques to perform each step or to produce each model.
18. **Ease of understanding of techniques:** Are the techniques easy to understand?
19. **Usability of techniques:** Are the techniques easy to follow?
20. **Provision of examples and heuristics:** Are examples and heuristics of the techniques provided?
Model Related Criteria
21. **Concepts:** What concepts are the methodology's models capable of expressing?
22. **Expressiveness:** How well can each model express these concepts? (e.g. is each model capable of capturing the concept at a great level of detail, or from different angles?)
23. **Completeness:** Are all necessary agent-oriented concepts that describe the target MAS captured by the methodology's models?
24. **Formalization/Preciseness of models:** Are notation (syntax) and semantics of models clearly defined?
25. **Model derivation:** Does there exist explicit process/logic and guidelines for transforming models into other models, or partially creating a model from information present in another?
26. **Consistency:** a. Are there rules and guidelines to ensure consistency between levels of abstractions within each model (i.e. internal consistency), and between different models? b. Are representations expressed in a manner that allows for consistency checking between them?
27. **Complexity:** is there a manageable number of concepts expressed in each model/diagram?
28. **Ease of understanding of models:** Are the models easy to understand?
29. **Modularity:** Does the methodology and its models provide support for modularity of agents?
30. **Abstraction:** Does the methodology allow for producing models at various levels of detail and abstraction?
31. **Autonomy:** Can the models support and represent the autonomous feature of agents (i.e. the ability to

act without direct intervention of humans or others, and to control their own states and behaviours)?
32. **Adaptability:** Can the models support and represent the adaptability feature of agents (i.e. the ability to learn and improve with experience)?
33. **Cooperative behavior:** Can the models support and represent the cooperative behavior of agents (i.e. the ability to work together with other agents to achieve a common goal)?
34. **Inferential capability:** Can the models support and represent the inferential capability feature of agents (i.e. the ability to act on abstract task specifications)?
35. **Communication ability:** Can the models support and represent "knowledge-level" communication ability (i.e. the ability to communicate with other agents using language resembling human-like speech acts)?
36. **Personality:** Can the models support and represent the personality of agents (i.e. the ability to manifest attributes of a "believable" human character)?
37. **Reactivity:** Can the models support and represent reactivity of agents (i.e. the ability to selectively sense and act)?
38. **Temporal continuity:** Can the models support and represent temporal continuity of agents (i.e. persistence of identity and state over long periods of time)?
39. **Deliberative behavior:** Can the models support and represent deliberative behavior of agents (i.e. the ability to decide in a deliberation, or proactiveness)?
40. **Concurrency:** Does the methodology allow for producing models to capture concurrency (e.g. representation of concurrent processes and synchronization of concurrent processes)?
41. **Human Computer Interaction:** Do the models represent human users and the user interface?
42. **Models Reuse:** Does the methodology provide, or make it possible to use, a library of reusable models?

Supportive Feature Criteria

43. **Software and methodological support:** Is the methodology supported by tools and libraries (e.g. libraries of agents, agent components, organizations, architectures and technical support)?
44. **Open systems and scalability**: Does the methodology provide support for open systems and scalability (e.g. the methodology allows for dynamic integration/removal of new agents/resources)?
45. **Dynamic structure**: Does the methodology provide support for dynamic structure? (i.e. the methodology allows for dynamic reconfiguration of the system)?
46. **Agility and robustness**: Does the methodology provide support for agility and robustness (e.g. the methodology captures normal processing and exception processing, provides techniques to analyze system performance for all configurations, or provides techniques to detect/recover from failures)?
47. **Support for conventional objects**: Does the methodology cater for the use/integration of ordinary objects in MAS (e.g. the methodology models the agents' interfaces with objects)?
48. **Support for mobile agents**: Does the methodology cater for the use/integration of mobile agents in MAS (e.g. the methodology models which/when/how agent should be mobile)?
49. **Support for self-interested agents**: Does the methodology provide support for MAS with self-interest agents (whose goals may be independent or enter in conflict with other agents' goals)?
50. **Support for ontology**: Does the methodology cater for the use/integration of ontology in MAS (i.e. ontology-driven agent systems)?

3 MAS Development Methodologies

The five MAS methodologies selected for the comparative analysis are considered the most comprehensive, widely referenced, and well documented AO software engineering methodologies compared to other existing work. Each of these methodologies offers a set of steps, techniques, and/or models for the analysis and design of MAS.

Multiagent Systems Engineering - MaSE [2]
This methodology has been applied to numerous graduate-level and research projects. Its process steps include identifying and organizing system goals, distilling use cases and elaborating them into sequence diagrams, identifying roles, identifying agent

classes from roles, defining inter-agent conversations, designing agent internals, and specifying MAS deployment details.

The *GAIA* methodology [3][4]
GAIA adopts an organization-oriented approach towards MAS development. Its Analysis phase develops four major models: Preliminary Role Model, Preliminary Interaction Model, Environment Model (which describes MAS environment in terms of abstract resources), and Organizational Rule Model (which specifies rules that affect the whole MAS).

The design phase then transforms these models into sufficiently low-level abstractions, including Complete Role and Interaction Models, Organizational Structure Model, Agent Model, Service Model (which specifies the services offered by each agent), and Acquaintance Model.

Methodology for BDI agents – BDIM [5]
This methodology classifies models into *external* or *internal* levels. External models represent a system-level view of the system, and include Agent and Interaction Models. At the internal level, each model describes an abstract internal component of the agent, including Belief Model, Goal Model, and Plan Model.

The *Prometheus* methodology [6]
Prometheus aims to provide a detailed, complete methodology for developing MAS with BDI-like agents. It consists of three phases:

- *System specification*: identifies the basic functionalities, percepts, actions, and use case scenarios of the target MAS;
- *Architectural design*: identifies agents, events, interactions, and shared data objects; and
- *Detailed design:* designs the internals of each agent. Each agent is composed of "capabilities", which are in turn made up of lower-level capabilities, plans, internal events, and data.

The *MAS-CommonKADS* methodology [7]
This methodology also extends from CommonKADS, although it takes advantage of many OO techniques. The guidelines for constructing each model are summarized as follows:

- *Agent Model*: Agents are identified using use cases, problem statements, RDD and CRC techniques.
- *Task Model*: Tasks are identified and decomposed as in CoMoMAS functional analysis.
- *Coordination Model*: Agent interactions are identified from use case scenarios. Coordination protocols are described by Event Flow Diagrams, State Transition Diagrams, and Message Sequence Charts.
- *Expertise Model*: Different types of agent knowledge (e.g. domain knowledge, task knowledge, inference knowledge, and problem-solving methods) are specified.

- *Organization Model*: MAS organization is described in terms of agent aggregation and inheritance.
- *Design Model*: Infrastructure facilities, agent architecture, software and hardware required for MAS implementation are specified.

4 Comparative Analysis

Using the feature analysis framework of Tran et al. [1], the comparative analysis of the above MAS methodologies was performed for Process Related, Model Related, and Supportive Feature Criteria. The evaluation of Technique Related Criteria is not presented in this paper, as it entails an in-depth analytical discussion of each methodology, which is most relevant when the developer has decided on which particular methodology to use, or is choosing between a small number of methodologies that provide the same or similar process steps (thus requiring an investigation of techniques to determine which method is the best in performing these common steps for a particular application). An in-depth comparison of MAS methodologies' techniques will be presented in a future paper. Criteria 9, 10, 13, 22, and 27 are also not presented in this paper for the same reason. Criteria "Steps in the development process" and "Concepts represented by MAS models" will each be analyzed separately because each requires an extensive assessment.

Process-Related Analysis (Table 2)
Apart from BDIM which does not explicitly specify its lifecycle model, the other MAS methodologies adopt an iterative, incremental SDLC for their MAS development. The documentation of BDIM [5] actually does reveal the need for iterative refinements for its models (specifically, the refinement of internal models like Belief, Goal, and Plan Models feeds back to the external models such as Agent and Interaction Models, and vice versa). All methodologies cover only the Analysis (A) and Design (D) phases of SDLC, except for MAS-CommonKADS that touches on the issues of conceptualization (C) phases.

With regard to the development perspective, GAIA and MASE are top-down (TD), Prometheus is bottom-up (BU), while BDIM and MAS-CommonKADS are hybrid (H). We define an AOSE methodology as top down if it starts from the analysis of high-level elements such as system goals, major functionality, problem statement, and organizational structure and proceeds to identifying and designing agents as system components that realize these elements. In contrast, a bottom-up AOSE methodology begins by analyzing low-level behaviours or tasks of the system, which are then packaged to compose agents. A hybrid (H) approach integrates both approaches by identifying agents from the consideration of both high-level system goals/organization, and low-level system tasks and responsibilities.

Most MAS methodologies are suitable to all types of application domains and heterogeneous agents, except for BDIM and Prometheus which target BDI-like agents. MaSE and Prometheus are considered supportive of the verification and validation process, since they provide rules or guidelines to assist the system developers in verifying and validating the developed models. For example, Prometheus suggests that a

good MAS design will have a minimal number of shared data objects captured in its System Overview Diagram. MaSE, Prometheus and MAS-CommonKADS are also perceived to be easier to understand and to follow than GAIA and BDIM, thanks to their detailed instructions on the development process and on each process step. All methodologies provide a clear path for refining their models through gradual stages to reach an implementation (or at least for clearly connecting the implementation level to the design specification).

With regard to the approaches towards MAS development, our assessment is performed on three categories of approaches:

- *Generic approach*: including OO-based approach and Knowledge-Engineering (KE) based approach. The former either adapts or extends OO models and techniques, while the latter builds upon techniques from knowledge engineering [22].
- *The use of "role"*: A MAS methodology can be role-oriented (RO), i.e. using "roles" as the main abstraction for MAS analysis and design, or non-role-oriented (NRO), i.e. relying on other constructs such as use cases, enterprise/workflow models, and interactions to develop agents and MAS.
- *Approach in role identification:* If a methodology is role-oriented, it can identify roles in the system by following a goal-oriented analysis approach (GO), behavior-oriented analysis approach (BO), or organisation-oriented analysis approach.

The five investigated MAS methodologies can demonstrate the adoption of all of the above approaches, except for the behavior-oriented analysis approach for role identification.

Model-Related Analysis (Table 2)

Compared to other MAS methodologies, MAS-CommonKADS can capture and represent the highest number and the most diverse AO concepts (i.e. criterion "Completeness") thanks to its comprehensive set of models. All five methodologies offer detailed explanations on their models' notation and semantics, except for MAS-CommonKADS which does not provide any notation for its Design model (i.e. criterion "Formalization/Preciseness"). All methodologies, except for MAS-CommonKADs, offer steps and related techniques to support the transforming of models into other models (i.e. "Model Derivation" criterion).

"Consistency" criterion is assessed in terms of two questions:

- whether there are rules and guidelines to ensure consistency between levels of abstractions of a model/diagram or between different models/diagrams; and
- whether the models/diagrams are expressed in a manner that allows for consistency checking between them

As shown in Table 2, methodologies that offer the highest support for consistency assurance are MaSE and Prometheus. All methodologies however encourage their models to be developed at various levels of details and abstractions (i.e. "Abstraction" criterion).

Agent characteristics that all five MAS methodologies can support and model are modularity, autonomy, agent cooperative behavior, "knowledge-level" communication ability, reactivity, and deliberative behavior. This finding is desirable, considering the significance of these constructs in MAS analysis and design. Constructs that

most methodologies overlook are agent adaptability, agent personality, agent temporal continuity, concurrency, and sub-system interactions.

All five methodologies make it possible to reuse the developed models, e.g. Expertise Models of MAS-CommonKADS can be reused by agents with similar task inference requirements [7].

Supportive Feature Analysis (Table 2)
The five investigated methodologies appear to focus merely on the development of typical, simple MASs, without paying much attention to add-on capabilities of MAS such as openness/scalability, software tool, agility and robustness. No methodologies address the use of mobile agents in MAS. Only GAIA explicitly supports the development of MASs with self-interested agents[1]. Despite of its significance in MAS design and operation, ontology is not supported nor used by most MAS methodologies. Only MAS-CommonKADS briefly involves ontology in its development process, particularly in the modeling of agent "domain knowledge". It also acknowledges that ontology servers should be part of the infrastructure facilities to be designed for the agent network.

Support for Steps in the Development Process (Table 3)
The list of standard MAS-development steps proposed by [1] is used as a checklist to compare the five MAS methodologies. The support of each methodology for each step is assessed on a 4-point scale:

0: no support is provided
1: the step is included but no techniques or examples are provided
2A: the methodology provides techniques for performing the step
2B: the methodology provides examples of how the step can be performed
3: the step is discussed with techniques and examples

This scheme of rating allows us to indirectly assess and compare the provision of techniques and heuristics by the methodologies. Methodologies that are most complete in terms of their support for the development steps are MaSE, Prometheus, and MAS-CommonKADS.

Support for Concepts of MAS Models (Table 4)
We will use the list of standard MAS concepts proposed by [1] to compare the five MAS methodologies. If a MAS methodology can represent or capture a concept in its models, we can simply give it a tick ✓.

Most concepts in the categories of "problem domain", "agent properties", "agent relationships", and "agent interactions" are supported by most MAS methodologies. However, "deployment" concepts are overlooked by most methodologies, indicating their lack of support for MAS deployment issues.

[1] This issue is addressed in the updated version of GAIA [4]

Table 2. Comparative analysis results

Evaluation Criteria	MaSE	GAIA	BDIM	Prometheus	MAS-CommonKADS
Process Related Criteria					
Development Lifecycle	Iterative across all phases	Iterative across all phases	Not specified	Iterative across all phases	Risk-driven & component-based
Coverage	A & D	A & D	A & D	A & D	C, A & D
Development approach	TD	TD	H	BU	H
Application Domain	Any	Any	Any	Any	Any
Size of MAS	≤ 10 agents	≤ 100 agents	Not specified	Not specified	Not specified
Agent Nature	Hete.	Hete.	BDI agents	BDI agents	Hete.
Support for verification	Yes	No	No	Yes	Briefly mentioned
Ease of understanding of process steps	High	High	High	High	High
Usability of the methodology	High, except for internal agent modeling.	Medium. Missing many important steps	Medium, Lack of detailed instructions for each step	High	High
Refinability	Yes	Yes	Yes	Yes	Yes
Approach towards MAS development	• OO • RO • GO	• OO • RO • OO	• OO • RO • N/A	• OO • NRO • N/A	• KE • NRO • N/A
Model Related Criteria					
Completeness	High	Medium	Medium	High	High
Formalization/ Preciseness	High	High	High	High	Low
Model derivation	Yes	Yes	Yes	Yes	No
Ease of understanding	High	High	High	High	Medium
Consistency	• Yes • Yes	• Yes • Yes	• No • Yes	• Yes • Yes	• No • Yes
Modularity	Yes	Yes	Yes	Yes	Yes
Abstraction	Yes	Yes	Yes	Yes	Yes
Autonomy	Yes	Yes	Yes	Yes	Yes
Adaptability	No	No	No	No	No
Cooperative behaviour	Yes	Yes	Yes	Yes	Yes
Inferential capability	Yes	No	Yes	Yes	Yes
Communication ability	Yes	No	Yes	Yes	Yes
Personality	No	No	No	No	No
Reactivity	Yes	Yes	Yes	Yes	Yes
Deliberative behavior	Yes	Yes	Yes	Yes	Yes
Temporal continuity	No	No	No	No	No
Concurrency	Yes	No	No	No	No
Human Computer Interaction	No	No	No	Yes	Yes
Models Reuse	Yes	Yes	Yes	Yes	Yes
Supportive Feature Criteria					
Software and methodological support	Yes	No	No	Yes	No
Open systems and scalability	No	Yes	No	No	No
Dynamic structure	No	Yes	No	No	No
Agility and robustness	No	No	No	Yes	No
Support for conventional objects	No	No	No	Yes	No
Support for mobile agents	No	No	No	No	No
Support for self-interested agents	No	Yes	No	No	No
Support for ontology	No	No	No	No	Yes

Table 3. Comparative analysis on support for steps in the development process

Steps	MaSE	GAIA	BDIM	Prometheus	MAS-CommonKADS
Problem Domain Analysis					
Identify system goals	3	0	0	0	0
Identify system roles	3	3	2A	0	0
Identify system functionality/tasks	3	3	1	3	2A
Develop use cases/scenarios	3	0	0	3	2B
Produce sequence diagrams	3	0	0	0	2B
Identify design requirements	0	0	0	0	0
Identify agent classes	3	3	3	3	3
Agent Interaction Design					
Specify agent interaction pathways	3	3	2A	3	3
Define exchanged messages	3	0	0	1	2B
Specify interaction protocols	3	0	0	3	3
Specify contracts/commitments	0	0	0	0	0
Specify conflict resolution mechanisms	0	0	0	0	0
Specify coordination/control regime (e.g. centralized or hierarchical)	1	0	0	0	0
Specify agent communication language	0	0	0	0	0
Agent Internal Design					
Define agent architecture	3	0	0	0	1
Define agent mental attributes (e.g. goals, beliefs, plans…)	0	0	3	3	3
Define agent behavioral interface (e.g. capabilities, services)	0	3	3	3	0
System/Environment Design					
Define system architecture/organisational structure	0	0	0	0	0
Specify dynamic agent group formulation / dissolution	0	0	0	0	0
Specify agent relationships (e.g. inheritance, aggregation & association)	0	3	3	0	2B
Specify co-existing non-agent entities	0	3	0	2A	0
Specify infrastructure/environment facilities	0	0	0	0	1
Specify agent-environment interaction mechanism	0	0	0	3	1
Instantiate agent classes	3	1	3	0	0
Specify agent instances location	3	0	0	0	0

Table 4. Comparative analysis on support for concepts of MAS models

Concepts	MaSE	GAIA	BDIM	Prometheus	MAS-CommonKADS
Problem Domain					
System goals	✓				
System roles	✓	✓	✓		
System functionality/Tasks	✓	✓		✓	✓
Task responsibilities/Procedures	✓	✓	✓		
Design requirements					
Use case/Scenarios	✓			✓	✓
Agent Properties					
Agent classes	✓	✓	✓	✓	✓
Agent instances (including cardinality)	✓	✓	✓	✓	✓
Agent's roles	✓	✓	✓		✓
Agent's functionality	✓	✓	✓	✓	✓
Agent's knowledge/Beliefs			✓	✓	✓
Agent's plans			✓	✓	✓
Agent's goals	✓		✓		✓
Agent's capabilities			✓	✓	✓
Agent Mobility					
Agent Interaction					
Interaction pathways	✓	✓	✓	✓	✓
Exchanged messages	✓		✓	✓	✓
Interaction protocols	✓			✓	✓
Interaction constraints					
Conflict resolution mechanisms					
Contracts/commitments					
Ontology					
Agent Relationships					
Inheritance			✓		✓
Aggregation		✓	✓		✓
Association		✓	✓		✓
System/Environment					
Co-existing non-agent entities				✓	
Infrastructure/environment facilities		✓			
Organisational Structure					
Agent-environment interaction				✓	✓
Environment characteristics					
Deployment					
Agent architecture	✓				✓
System architecture					
Location of agent instances	✓				
Sources of agent instances					

5 Conclusions

In this paper, we have compared five well-known MAS methodologies using the feature analysis framework proposed in [1]. The comparison takes into account a variety of evaluation criteria and methodological features, covering from process related and model related aspects to high-level MAS capabilities. We also assessed the capability of methodologies in terms of their support for steps in the development process,

and for AO concept modeling. This assessment will help developers to decide on the most appropriate methodology to use in a specific application. However, it should be noted that while this paper examines the features (and steps and concepts) of a methodology as independent from each other, some methodologies may offer the features (or steps or concepts) in combination. Thus the developer may need to assess these constructs as a group rather than as independent entities. Future work includes extending the comparative analysis to many other existing MAS methodologies, in order to obtain an overall assessment of the current work in AO software engineering.

References

1. Tran, Q.N., Low, G., Williams, M.A.: A Feature Analysis Framework for Evaluating Multiagent System Development Methodologies. In Zhong, N., Ras, Z.W., Tsumoto, S., Suzuki, E. (eds): Foundations of Intelligent Systems – Proc. of the 14th Int. Symposium on Methodologies for Intelligent Systems ISMIS'03 (2003) 613-617.
2. Wood, M.: Multiagent Systems Engineering: A Methodology for Analysis and Design of Multiagent Systems. MS Thesis, Air Force Institute of Technology, USA (2000).
3. Wooldridge, M., Jennings, N.R. and Kinny, D.: The Gaia methodology for agent-oriented analysis and design. Journal of Autonomous Agents and Multi-Agent Systems, 3 (2000) 285-312
4. Zambonelli, F., Jennings, N., Wooldridge, M.: Developing multiagent systems: the Gaia methodology. ACM Transaction on Software Engineering and Methodology (in press)
5. Kinny, D., Georgeff, M., Rao, A.: A Methodology and Modelling Technique for Systems of BDI Agents. Proc. of the 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World (1996) 56-71
6. Padgham, L., Winikoff, M.: Prometheus: a methodology for developing intelligent agents. Proc. of the 1st Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems (2002).
7. Iglesias, C. A., Garijo, M., Gonzalez, J.C., Velasco, J.R.: Analysis and Design of Multiagent Systems Using MAS-CommonKADS. In Singh, M.P., Rao, A., Wooldridge, M.J. (eds.). Intelligent Agents IV (*ATAL'97)*. Springer-Verlag, Berlin (1998)
8. Wood, B., Pethia, R., Gold, L.R., Firth, R.: A Guide to the Assessment of Software Development Methods. Technical Report CMUSEI-88-TR-8, SEI, Software Engineering Institute, Carnegie Mellon University (1988)
9. Jayaratna, N.: Understanding and Evaluating Methodologies - NIMSAD A Systematic Framework. McGraw-Hill, England (1994)
10. Olle, T.W., Sol, H.G., Tully, C.J. (eds.): Information Systems Design Methodologies - A Feature Analysis. Elsevier Science Publishers, Amsterdam (1983)
11. The Object Agency Inc.: A Comparison of Object-Oriented Development methodologies. http://www.toa.com/smnn?mcr.html (1995)
12. Shehory, O., Sturm, A.: Evaluation of modeling techniques for agent-based systems. Proc. of the 5th Int. Conf. on Autonomous agents (2001) 624-631.
13. O'Malley, S.A., DeLoach, S.A.: Determining When to Use an Agent-Oriented Software Engineering Paradigm. Proc. of the 2nd Int. Workshop on Agent-Oriented Software Engineering (AOSE) (2001).
14. Cernuzzi, L., Rossi, G.: On the Evaluation of Agent-Oriented Modelling Methods. Proc. of the OOPSLA Workshop on Agent-Oriented Methodologies (2002)
15. Sabas, A., Badri, M., Delisle, S.: A Multidimentional Framework for the Evaluation of Multiagent System Methodologies. Proc. of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI-2002), 211-216.

16. Iglesias, C.A., Garijo, M., & Gonzalez, J.C.: A survey of agent-oriented methodologies. Proc. of the 5th Int. Workshop on Intelligent Agents V: Agent Theories, Architectures, and Languages (1999)