A Feature Analysis Framework for Evaluating Multi-agent System 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. This paper proposes a comprehensive and multi-dimensional feature analysis framework for evaluating and comparing methodologies for developing multi-agent systems (MAS). Developed from a synthesis of various existing evaluation frameworks, the novelty of our framework lies in the high degree of its completeness and the relevance of its evaluation criteria. The paper also presents a pioneering effort in identifying the standard steps and concepts to be supported by a MAS-development process and models.

1 Introduction

Today, with the availability of numerous methodologies for analyzing and designing multi-agent systems (MAS), MAS-developers have to deal with a difficulty that has plagued object-oriented (OO) developers, i.e. comparing the available MAS-development methodologies, thereafter deciding on the most appropriate methodology to use in a specific application. Unfortunately, the numerous feature analysis frameworks for evaluating conventional system development methodologies do not assess the agent-oriented aspects of MAS development. On the other hand, due to the recent emergence of MAS and their inherent complexity, few frameworks exist for evaluating MAS-development methodologies ([5], [6], [7], [8]). These frameworks mainly examine MAS-specific characteristics without adequately considering the system engineering dimensions. We fill the current void by proposing a comprehensive, multi-dimensional framework that evaluates a MAS-development methodology from both the dimensions of system engineering and those specific to MAS engineering.

Instead of developing the framework from scratch, we built on the established work in the literature by, firstly, selecting the relevant evaluation criteria from the various existing feature analysis frameworks, thereafter synthesizing these criteria into a new comprehensive framework. Our pool of resources consists of a) the most outstanding and well-documented evaluation frameworks for conventional system development methodologies including OO methodologies – namely [1], [2], [3] and [4],

and b) all the identified frameworks for MAS-methodology evaluation – namely [5], [6], [7] and [8]. The former provides a well-established account of the generic system engineering features to be subject to methodological evaluation, while the latter presents various agent-oriented and MAS-specific aspects for assessment.

To promote the relevance of our framework, we adopted the evaluation criteria that are representative, case-generic, and centered on the capabilities and usefulness of a methodology. We also added several evaluation criteria that are not yet accounted for in the existing evaluation frameworks, e.g. a methodology's approach towards MAS development, support for mobile agents, and support for ontology.

2 Specification of the New Feature Analysis Framework

Our evaluation framework is comprised of four components (Table 1):

- *Process Related Criteria*: evaluating a methodology's support for the MAS-development process
- *Technique Related Criteria:* assessing the methodology's techniques to develop MAS
- *Model Related Criteria*: examining the capabilities of the methodology's models
- *Supportive Feature Criteria*: evaluating a variety of high-level methodological capabilities

This structure highlights the completeness of our framework, as it targets at all three major components of a system development methodology – process, models, and techniques – as defined by OPEN [9]. Full details of how the framework was specified are not presented due to space constraints.

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

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, and implementation)?
3. **Development approach:** What development approach is supported (i.e. top-down or bottom-up)?
4. **Application domain:** Is the methodology applicable to a specific or multiple application domains?
5. **Scope of effort:** What size of MAS is the methodology suited for (i.e. small, medium, or large)?
6. **Agent nature:** Does the methodology support only homogeneous agents, or heterogeneous agents?
7. **Support for verification and validation:** 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:** Is the methodology a. OO-based or knowledge-engineering based?

b. Role-oriented or non-role-oriented regarding its approach towards agent identification?
c. Goal-oriented, behavior-oriented, or organization-oriented in the identification of roles (if a role-oriented approach in b. applies)?
d. Architecture-independent or architecture-dependent?

Technique Related Criteria

1. **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)?
2. **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.
3. **Ease of understanding of techniques:** Are the techniques easy to understand?
4. **Usability of techniques:** Are the techniques easy to follow?
5. **Provision of examples and heuristics:** Are examples and heuristics of the techniques provided?

Model Related Criteria

1. **Concepts:** What concepts are the methodology's models capable of expressing?
2. **Expressiveness:** How well can the models express these concepts and relationships between concepts? (e.g. are the models capable of capturing each concept at a great level of detail, or from different angles?)
3. **Completeness:** Are all necessary agent-oriented concepts that describe the target MAS captured?
4. **Formalization/Preciseness of models:**
   a. Are notation (syntax) and semantics of the models clearly defined?
   b. Are examples of the models presented?
5. **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?
6. **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?
7. **Complexity:**
   a. Is there a manageable number of concepts expressed in a single model/diagram?
   b. Is notation semantically and syntactically simple across models?
8. **Ease of understanding of models:** Are the models easy to understand?
9. **Modularity:** Does the methodology and its models provide support for modularity of agents?
10. **Abstraction:** Does the methodology allow for producing models at various levels of detail and abstraction?
11. **Autonomy:** Can the models support and represent the autonomous feature of agents?
12. **Adaptability:** Can the models support and represent the adaptability feature of agents (i.e. the ability to learn and improve with experience)?
13. **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)?
14. **Inferential capability:** Can the models support and represent the inferential capability feature of agents (i.e. the ability to act on abstract task specifications)?
15. **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)?
16. **Personality:** Can the models support and represent the personality of agents (i.e. the ability to manifest attributes of a "believable" human character)?
17. **Reactivity:** Can the models support and represent reactivity of agents (i.e. the ability to selectively sense and act)?
18. **Temporal continuity:** Can the models support and represent temporal continuity of agents (i.e. persistence of identity and state over long periods of time)?
19. **Deliberative behavior:** Can the models support and represent deliberative behavior of agents (i.e. the ability to decide in a deliberation, or proactiveness)?
20. **Concurrency:** Does the methodology allow for producing models to capture concurrency (e.g. representation of concurrent processes and synchronization of concurrent processes)?
21. **Human Computer Interaction:** Does the methodology allow for producing models to represent user interface and system-user interaction?
22. **Sub-system interaction:** Does the methodology allow for producing models to capture interac-

tion/relationships between subsystems in MAS?

23. **Models Reuse:** Does the methodology provide, or make it possible to use, a library of reusable models?

Supportive Feature Criteria

1. **Software and methodological support:** Is the methodology supported by tools and libraries (e.g. libraries of agents, agent components, organizations, architectures and technical support)?
2. **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)?
3. **Dynamic structure**: Does the methodology provide support for dynamic structure (e.g. the methodology allows for dynamic system reconfiguration when agents are created/destroyed during execution)?
4. **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)?
5. **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)?
6. **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)?
7. **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)?
8. **Support for ontology**: Does the methodology cater for the use/integration of ontology in MAS (i.e. ontology-driven agent systems)?

To evaluate the criteria "*Steps in the development process*" and "*Concepts*", it is helpful to have a list of "standard" process steps and concepts to serve as an assessment checklist. To date, no study has been found that identifies the representative steps and concepts to be supported by a typical MAS-development process and models. We therefore provide a pioneering effort in this area. The following list of standard steps and concepts (Fig. 1) were determined from our investigation of the existing MAS-development methodologies (references [10] to [15][1]). Full details on the specification of these standard steps and concepts will be presented in a separate paper.

Steps	Identify system goals	Specify conflict resolution mechanisms	Specify agent relationships (inheritance, aggregation & association)
	Identify system roles	Define agent architecture	Specify co-existing entities
	Develop system use cases/scenarios	Define agents' mental attitudes (goals, plans, beliefs...)	Specify environment facilities
	Identify system functionality	Define agents' interface (capabilities, services...)	Specify agent-environment interaction
	Identify design requirements	Fulfill agent architecture	Instantiate agent classes
	Identify agent classes	Define system architecture	Specify agent instances location
	Specify agent interaction pathways	Specify organizational structure	
	Define exchanged messages	Specify group behavior	
	Specify interaction protocols		
	Specify contracts/commitments		
	Specify ACL		
Concepts	System goals	Agent's functionality	Agent aggregation
	System roles	Percepts/Events	Agent association
	System functionality	Agent mobility	Co-existing entities
	Task responsibilities/procedures	Interaction pathways	Environment facilities
	Design requirements	Exchanged messages	Organizational structure
	Use case/scenarios	Interaction protocols	Group behavior
	Agent classes	Interaction constraints	Agent-environment interaction
	Agent instances	Conflict resolution mechanisms	Environment characteristics
	Agent's knowledge/beliefs	Contracts/commitments	Agent architecture
	Agent's plans	ACL	System architecture
	Agent's goals	Ontology	Location of agent instances
	Agent's roles	Agent inheritance	Sources of agent instances

Fig. 1. List of standard steps and concepts to be supported by a MAS-development process and models

[1] Due to space constraints, only some of the investigated methodologies are listed here.

3 Conclusion

The completeness and relevance of our evaluation framework for MAS-development methodologies are reflected via its attention to both system engineering dimensions and agent-specific aspects, its focus on all three major components of the methodology (i.e. process, techniques and models), and its representative, case-generic evaluation criteria which center on the capabilities and usefulness of the methodology. We also proposed a list of standard steps and concepts to be supported by a MAS development process and models. Future work includes applying the framework to a comparative analysis of existing MAS-development methodologies, validating the proposed list of standard steps and concepts, and using the framework and the list to develop a new, unified MAS-development methodology.

References

1. 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)
2. Jayaratna, N.: Understanding and Evaluating Methodologies - NIMSAD A Systematic Framework. McGraw-Hill, England (1994)
3. Olle, T.W., Sol, H.G., Tully, C.J. (eds.): Information Systems Design Methodologies - A Feature Analysis. Elsevier Science Publishers, Amsterdam (1983)
4. The Object Agency Inc.: A Comparison of Object-Oriented Development methodologies. http://www.toa.com/smnn?mcr.html (1995)
5. Shehory, O., Sturm, A.: Evaluation of modeling techniques for agent-based systems. Proc. of the 5th Int. Conf. on Autonomous agents (2001) 624-631.
6. 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).
7. Cernuzzi, L., Rossi, G.: On the Evaluation of Agent-Oriented Modelling Methods. Proc. of the OOPSLA Workshop on Agent-Oriented Methodologies (2002)
8. 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.
9. Henderson-Sellers, B., Simons, A., Younessi, H.: The OPEN Toolbox of Techniques. Addison Wesley Longman Ltd., England (1998)
10. Wood, M.: Multiagent Systems Engineering: A Methodology for Analysis and Design of Multiagent Systems. MS Thesis, Air Force Institute of Technology, Ohio (2000)
11. Lind, J.: MASSIVE: Software Engineering for Multiagent Systems. PhD Thesis, University of Saarland, Saarbrucken (1999)
12. Wooldridge, M., Jennings, N.R., Kinny, D.: The Gaia Methodology for Agent-Oriented Analysis and Design. Autonomous Agents and Multi-Agent Systems 3(3) (2000) 285-312
13. Eurescom: MESSAGE - Methodology for Engineering Systems of Software Agents. http://www.eurescom.de/public/projectresults/P900-series/907ti1.asp (2001)
14. Padgham, L., Winikoff, M.: Prometheus: A Methodology for Developing Intelligent Agents. Proc. of the 3rd Int. Workshop on Agent-Oriented Software Engineering (AOSE) (2002)
15. Glaser, N.: Contribution to Knowledge Acquisition and Modelling in a Multi-Agent Framework (the CoMoMAS Approach). PhD Thesis, University of Nancy 1, France (1996)