Evaluation Complexity Problem in Agent based Software Development Methodology

Amin Saremi[1], Mostafa Esmaeili[2], Mahshid Rahnama[1]
[1]Department of Computer Engineering, Sharif University of Technology, Iran
[2]Department of Electrical and Computer Engineering, Shahid Beheshti University, Iran
{saremi, esmaeili, rahnama}@ce.sharif.edu

Abstract — **Several methodologies with their own characteristics have been proposed in the area of agent-oriented software engineering. Consequently, deciding which methodology to select in a specific case is an important issue and it can lead to decrease software development cost and effort. Thus, importance of evaluation of methodologies will be highlighted in choosing the appropriate methodology in the development process of an application. It can also help in developing new methodologies and improving existing ones. In this paper, we are going to provide an evaluation framework of agent oriented methodologies. To demonstrate the usage of the suggested framework, it is applied to evaluate two methodologies (MESSAGE and Prometheus) using a proper example. Results show that, using our method, methodologies can be truly compared and evaluated.**

I. Introduction

Agent oriented methodologies assist developers in all phases of agent based software development process. A methodology provides a set of useful techniques to resolve complexity of developing large scale systems. It also helps to manage development life cycle process easier.

There are several agent oriented methodologies. Each methodology provides some guidelines, activities and modeling concepts. It may suggest specific analysis and design techniques, deliverables and notations. It also has its own supporting tools. Therefore variety of agent oriented methodologies result in some difficulties in deciding, reaching standards and finding sufficient samples and case studies on a methodology. Therefore, choosing an appropriate methodology in a specific case is one of the most important issues in the process of software development. Being aware of strong points and weaknesses of all methodologies and being familiar with existing differences among various methodologies can really help developers in deciding. Hence, evaluation of methodologies in terms of their suggested models and deliverables, development process and capabilities is truly advantageous.

Agent-oriented methodologies are categorized into three major classes based on influenced approaches: Methodologies based on knowledge engineering approaches, such as MAS-CommonKADS [29,30] and CoMoMAS [33] which focus on agent's knowledge level concepts. MaSE [26], MESSAGE [20] and Prometheus [28], [29] are based on object oriented approaches and focus on object-oriented concepts. Other methodologies such as Gaia [6] have neither knowledge engineering approach nor object oriented idea. They are completely based on agent concepts. [7].

Several methods to evaluate agent oriented methodologies have been proposed.

Shehory and Sturm [2] developed a set of criteria for feature based analysis of methodologies consist of software Engineering criteria and agent based characteristics. The similar works have been done in [11] and [13].

Another evaluation of methodologies is provided in [4]. In their research, a case study based on some specific criteria has been selected, and the evaluation of some agent oriented methodologies is provided in the context of the proposed case study. Their evaluation framework consists of a set of questions to be delineated to each methodology.

In the study presented in [5] Cernuzzi and Rossi proposed a quantitative analysis employing quantitative methods. It checks the level to which some specific qualitative features exist in a methodology using attributes tree, it calculated the level to which the evaluated method meets the requirements of its users by assigning a weight to each feature of the method, grading each feature, and calculating a weighted average grade.

By studying evaluation techniques which are proposed until now, it seems that: 1) there is not any appropriate framework for evaluating methodologies. 2) These methods are mostly based on feature based analysis and a small amount of work has been performed to compare agent oriented methodologies in a quantitative approach. 3) There is not any evaluation on complexity of models and techniques offered by different analysis and design phases of methodologies. 4) No study is carried out in order to evaluate consistency and stability of produced models and artifacts.

Therefore, presenting a proper method to perform a drastic evaluation of methodologies based on their processes and proposed models can be really helpful. In this paper, a new quantitative framework for comparison of agent oriented methodologies is introduced which studies models, deliverables, development life cycle and overall process of methodologies. Evaluation of two methodologies, MESSAGE and Prometheus, is also presented by using a simple fruit market as an example. However the evaluation framework can be applied to other methodologies.

The remainder of this paper is organized as follows: The evaluation framework is explained in section 2; section 3 describes our quantitative approach in the context of a case study applied to two different methodologies. We introduce these methodologies in brief and an overview of each one is presented. Finally conclusion is provided in section 4 and Section 5 introduces prospects for future works.

II. Comparison Methodology

In this section, the evaluation framework is presented. In our framework, we evaluate complexity and quality of produced models. High level of abstraction leads to a decrement in complexity. The main question which arises is: How we can measure the abstraction level of methodology? In addition to abstraction level, we need to qualify artifacts but how we can compare designed models? We are going to answer these questions by providing an accurate model of methodology constructs. Each methodology consists of stages and each stage has some deliverables. Each deliverable must be evaluated from different views such as size, diversity of components, topology of components and relations, effectiveness in software development process and etc. Size of models and diversity of components represents complexity of a deliverable. On the other hand, topology of components and relations could be used to qualify deliverable design.

In our modeling methodology, constructs are categorized in three major classes: diagrams, tables and textual templates. Diagrams have been considered as a directed graph. *Magnitude* of a graph depends on number of nodes and edges. Throughout this paper, we denote by *n(G)* and *m(G)* the number of nodes and edges of the graph G, respectively. The number *n(G)* is called the *order of* G and *m(G)* is the *size of* G. Increasing order or size of a graph leads to an increase in its *magnitude*. So we can evaluate the *magnitude* of a graph as follows,

$$Magnitude = n(G) \bullet m(G) \qquad (1)$$

Another aspect of complexity is diversity of components in a model. So, we need a way to measure diversity of components in diagrams and models. To quantify diversity, objects in a model are counted and scores are assigned to objects based on their frequency of occurrence in the model. Scores assigned to each object is the position of object in the ranking of occurrence frequency in descending order. For example, if Oi is an object and has the most frequency of occurrence, its score (Si) will be set to 1. So, diversity of a model is defined as

$$Diversity = \frac{\sum S_i \bullet F_i}{N} \qquad (2)$$

Where Sj stands for Oj's score and Fj shows its frequency of occurrence. N represents number of objects such as nodes. Diversity of components converges to 1 when investigated model consists of only one object type and there is no diversity in components. Diversity could be calculated for both objects and their associations in a proposed model or deliverable. For example, we are going to calculate diversity of the model which is shown in Fig.1.

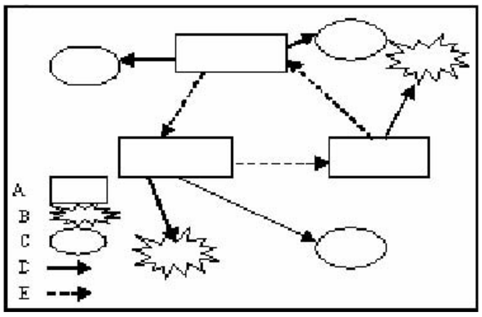

Fig. 1. Sample model

Table 1 shows the frequencies and scores of components in this model. Therefore total diversity is 3.1875.

TABLE 1

Frequency and Score of the Sample Model

Component	Frequency	score
A	3	4
B	2	5
C	3	4
D	5	1
E	3	4

Another type of deliverables is table. Tables in methodologies describe specific types of components such as roles, agents and etc. Tables comprise entries which must be filled by analysts. Tables could be assumed as an adjacency matrix of a bipartite graph. So, count of entries represents size of the graph and sum of rows and columns represents order of assumed graph. Magnitude of a table can be obtained as follows,

$$Magnitude = (C + R) \bullet N \tag{3}$$

Where C and R show number of columns and rows and N represents number of filled entries in table. Complexity of a model is directly affected by the magnitude of that model. Since tables have no components, their diversities are considered 1. Similarly, number of items discussed in a textual template shows its magnitude and its diversity is assumed to be 1, the same as tables.

Using above factors, required effort to prepare a specific deliverable could be estimated. Complexity of a deliverable estimates required time for preparing that construct and understanding it. Increasing model magntude or diversity results in a growth in complexity of models. So, complexity can be calculated as follows,

$$Complexity = Magnitude \bullet Diversity \tag{4}$$

Using complexity approximation, we can assess total process of software development.

To evaluate design of deliverables, we require a general framework to evaluate models. We analyze diagrams in two ways. One is simply an analysis of the density of links within the diagrams. This measure is the ratio of the actual coupling to the maximal possible coupling. For example, if the system has four components, then each component could potentially be linked to a maximum of three other components, giving a total number of 6 possible links. To get the link density, we simply count the links and divide by the number of links in the complete graph. So, we will have link density as follows,

$$LinkDensity = \frac{m(G)}{\frac{n(G)(n(G)-1)}{2}} \tag{5}$$

It is likely that all components in the system are linked in some way, and if this is the case, the minimum number of links will be one less than the number of components. An obvious design that has such a number of links is a star design, where each component is coupled to one central component. Although this has a low level of coupling, it is an undesirable design for two reasons: firstly, change to the central component type is likely to affect many of the other components; and secondly, it is a situation that can potentially lead to a bottleneck at run time when many components are trying to communicate with a single component. If we ignore component cardinality, the bottleneck factor can be assessed simply by examining the number of links that a component has. The largest number of links from a component type can be used as an indication of the worst bottleneck in the system for the purpose of comparing designs. To evaluate design of models, we used *TScore* which can be calculated as bellow,

$$TScore = Variance(D)^{\frac{1}{2}} \tag{6}$$

Where D shows degree of each node in a graph. If a specific node has a high degree and others have low degrees, we would have star topology or something like this. Consequently, *TScore* can measure and qualify topology of a graph.

LinkDensity and *TScore* can also be used to evaluate models and consequently methodologies.

III. Case Study and Evaluation

To evaluate methodologies, a simple FruitMarket is used as an example. To keep it small, three participants have been assumed: OrchardBot, ShopBot and SupplyBot. First participant, OrchardBot is the representative of an orchard

who is attempting to sell its products. Second participant, SupplyBot is a representative of several fruit producer. As the supplier needs to remain well stocked it has a budget to fund purchases. The last participant, ShopBot, is a representative of supermarket chain. It also has a budget to purchase additional stocks.
A broker is also used to register sellers, store sellers' information and provide this information to buyers. This example is a variation of one of Zeus case studies [35].
Two methodologies with object oriented approaches have been chosen as case studies: MESSAGE [20] and Prometheus [28, 29].

A MESSAGE

MESSAGE methodology covers MAS analysis and design and is intended for use in mainstream software engineering departments. The MESSAGE notation extends the UML with agent knowledge-level concepts and diagrams with notations for viewing them. Its analysis and design process is based on the Rational Unified Process (RUP).The methodology distinguishes high-level design from detailed-level design [20].
MESSAGE modeling process consists of three main phases: The first phase is Analysis which follows an iterative and incremental approach, the second phase is High level design and the third is detailed level design. Each phase produces several deliverable construct such as diagrams, textual templates, tables and etc. For example, Fig. 2. shows the Interaction diagram view of the analysis phase in level 1.Three roles Buyer, Seller and Broker are identified and can be seen in the graph.

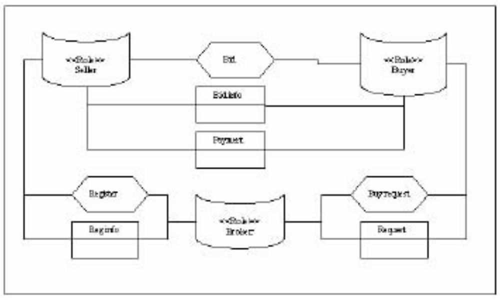

Fig. 2. Interaction view diagram provided by MESSAGE

Table 2 reports the result of applying our quantitative evaluation method on the deliverable constructs produced in the analysis and design phase.

Table 2

Magnitude, Diversity, LinkDensity and TScore of Models in MESSAGE Model

Model	Phase	Magnitude	Diversity	LinkDensity	TScore
Organization diagram view(structural relationships)	Analysis-Level 0	20	1.78	0.4	1.2
Organization diagram view(acquaintance relationships)	Analysis-Level 0	16	1.75	0.67	0.7
Goal view	Analysis-Level 0	342	1.48	0.11	1.7
Delegation Structure	Analysis-Level 1	180	1.1	0.11	2.36
Interaction view	Analysis-Level 1	70	1.53	0.15	1.02
Agent diagram view(Orchard agent)	High level design	42	1.31	0.28	1.8
Agent diagram view(Shop agent)	High level design	30	1.33	0.33	1.48
Agent diagram view(Broker agent)	High level design	72	1.9	0.22	2.2
Agent diagram view(Supply agent)	High level design	30	1.33	0.33	1.48
Task workflow	High level design	42	2.31	0.47	0.9
Task workflow	High level design	80	2.56	0.36	1.06
Task workflow	High level design	70	1.89	0.48	1.05
Interaction protocol diagram	High level design	39	1.2	6.5	2.1
Organization-based architecture	Detailed level design	30	1.45	0.33	1.49
Agent goal decomposition into tasks	Detailed level design	6	1.4	0.6	0.47
Agent goal decomposition into tasks	Detailed level design	30	1.45	0.33	0.94
Class diagram	Detailed level design	72	1.71	0.32	1.14

B Prometheus

Prometheus has been studied as the second methodology. It consists of three main phases: specification, architecture design and detailed design. Prometheus differs from existing methodologies in that it is a detailed and complete (start to end) methodology for developing intelligent agents which has evolved out of industrial and pedagogical experience [28]. Incremental process in this methodology made it very helpful and easy to use. Starting with roles, determining functionalities, data coupling techniques to specifying agents and other useful techniques provided by this methodology guide novice developers well. Using this methodology, we detected three main roles which have been shown in Fig. 3.

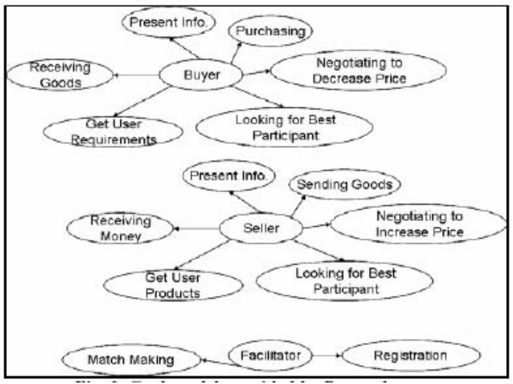

Fig. 3. Goal model provided by Prometheus

As shown in Fig. 3, each role contains some sub goals. Sub goals would be assigned to functionalities. These functionalities must be clustered in groups and groups will be transforms to agents. Various techniques to specify agents and their features are provided in this methodology. We use these techniques and finally complete specification of system would be resulted. Table 3 reports models which are produced during this process.
According to Table 3, total complexity of methodology will be 2429.49.

C Results and Comparison

In this section, we are going to compare the result of applying our framework to proposed methodologies. Based on the result of our framework factors (i.e. *Complexity*, *LinkDensity* and *TScore*), strengths and weaknesses of discussed methodologies will be revealed. Fig. 4 shows that the LinkDensity of MESSAGE is greater than the Prometheus in Detail Design phase. Using complicated relations between components may lead to have more communications between separated components. So, it seems that Prometheus will show better performance.

Table 3

Magnitude, Diversity, LinkDensity and TScore of Models in Prometheus

Model	Phase	Magnitude	Diversity	LinkDensity	Tscore
Goal Model	Specification	238.00	1.45	0.10	2.66
Functionalities Diagram	Specification	374.00	1.82	0.07	1.50
Functionalities Descriptions	Specification	45.00	1.00	0.53	1.06
Scenarios	Specification	12.00	1.00	0.57	0.29
Percepts and Actions	Specification	10.00	1.00	0.14	0.00
Data Coupling Diagram	Architecture Design	72.00	1.76	0.22	0.44
Agent Acquaintance Diagram	Architecture Design	2.00	1.30	1.00	0.00
Interaction Diagrams	Architecture Design	72.00	1.52	0.34	1.13
Protocols	Architecture Design	28.00	1.75	1.33	5.33
System Overview	Architecture Design	121.00	1.90	0.18	3.36
Agent Descriptions	Architecture Design	16.00	1.00	0.22	5.44
Capability Descriptors	Detailed Design	8.00	1.00	0.30	1.50
Agent Overview Diagram	Detailed Design	285.00	1.81	0.09	1.26
Process	Detailed Design	120.00	1.81	0.26	0.68
Capability Diagram	Detailed Design	0.00	0.00	0.00	0.00
Event Descriptors	Detailed Design	20.00	1.00	0.10	1.14
Data Descriptors	Detailed Design	6.00	1.00	0.17	0.50
Plan Descriptors	Detailed Design	36.00	1.00	0.00	0.00

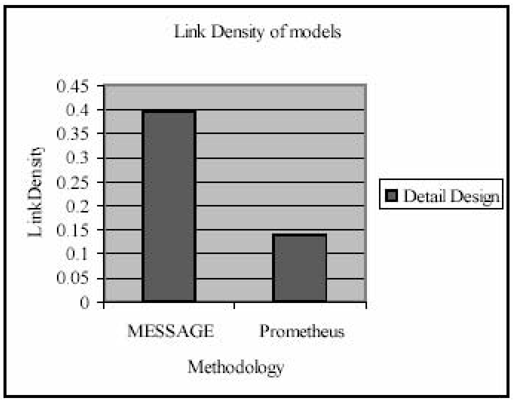

Fig. 4. Comparison of LinkDesnsity of models in Detail Design

Fig. 5 shows that the TScore of MESSAGE is greater than Prometheus in Detail Design phase. This fact leads to prefer Prometheus because of two main reasons: first, modification of one component leads to modify small amount of components. Second, possibility of bottleneck will be decreased.

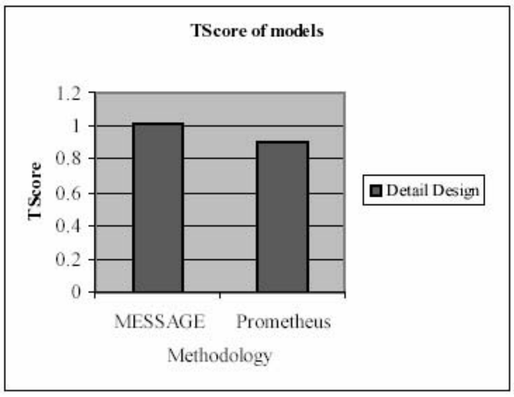

Fig. 5. Comparision of TScore of models in Detail Design

As mentioned above, Prometheus's complexity is larger than MESSAGE. This fact makes Prometheus, hard to use for beginners. On the other hand, Prometheus describes different aspects of agent oriented software in detail and this will help implementers to implement the final system easier. Fig. 6. presents a brief view of complexity increment in different phases of these methodologies. As shown in this figure, complexity in MESSAGE has a trend line as bellow:

$$y = -60.171x + 228.23 \qquad (7)$$

And complexity of Prometheus has a trend line as bellow:

$$y = -58.087x + 257.66 \qquad (8)$$

As shown in these equations, using MESSAGE methodology, complexity in models will decrease more than Prometheus during the project.

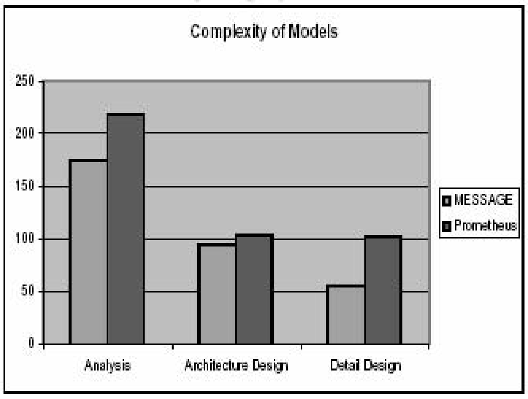

Fig. 6. Comparison of Complexity of models in Different phases

IV. Conclusion

We proposed a framework to evaluate and compare methodologies based on quantitative approach. Using this framework, developers are able to choose a suitable methodology in the development of their agent based systems. Our framework focused on models and artifacts of methodologies by evaluating their complexity and other characteristics. We follow decreases or increases of complexity in different phases in order to assess effectiveness of each stage in overall process. It also helps to resolve shortcomings of existing methodologies and exploiting their strengths.

Further more, we applied our framework to two methodologies with object oriented approaches (MESSAGE and Prometheus) using a FruitMarket as a case study.

We tried to measure abstraction level in terms of model magnitude and diversity of components. It seems that, methodologies with higher level of abstraction are more suitable for large scale systems.

However, our consideration is restricted to a particular case study with its specific requirements. Other case studies may lead to slightly different results.

V. Future Work

As a future work, our introduced framework can be used to evaluate other existing or upcoming methodologies and consequently, a good report about characteristics of each methodology could be made.

Applying the framework to a large scale system with a more complex domain can also be helpful to reach more powerful comparison of these methodologies.

REFERENCES

[1] Graham, I., and Sellers B. H. *The OPEN Process Specification*, Addison-Wesley, 1997.

[2] Sturm, A., and Shehory, O. A Framework for Evaluating Agent-Oriented Methodologies. In *Proceedings of the Int'l Bi-Conference Workshop on Agent-Oriented Information System (AOIS)*, Melbourne, Australia, 2003.

[3] Zambonelli, F., and Omicini, A. Challenges and Research Directions in Agent-Oriented Software Engineering. *Journal of Autonomous Agents and Multiagent Systems 9(3)*, 2004. 253-284.

[4] Yu,E., and Cysneiros, L.M. Agent-Oriented Methodologies-Towards a Challenge Exemplar. In *Proceedings of the 4th Int'l. Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS 2002)*, Toronto May 2002.

[5] Cernuzzi, L., and Rossi, G. On the evaluation of agent oriented modeling methods. In *Proceedings of Agent Oriented Methodology Workshop*, Seattle, November 2002.

[6] Wooldridge, M., Jennings, N. R., and Kinny, D. The Gaia Methodology for Agent-Oriented Analysis and Design. *Int'l Journal of Autonomous Agents and Multi-Agent Systems, Vol 3*, 2000. 285-312.

[7] Luck, M., Ashri, R., and D'Inverno, M. *Agent-Based Software Development.* ARTECH HOUSE Publishers, 2004.

[8] Zeng, Z.M., Meng, B., and Zeng Y.Y. An Intelligent Agent-Based System in Internet Commerce. In Proceedings of 4 Int'l Conference on Machine Learning and Cybernetics, Guangzhou, August 18-21, 2005.

[9] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Lorensen, W., *Object–Oriented Modeling and Design.* Prentice–Hall, 1991.

[10] Cernuzzi, L. and Rossi, G. On the evaluation of agent oriented modeling methods. In *Proceedings of Agent Oriented Methodology Workshop*, Seattle, 2002.

[11] Cuesta, P., Gómez, A., González, J.C., and Rodríguez, F.J. A Framework for Evaluation of Agent Oriented Methodologies. *Actas del Taller de Agentes Inteligentes en el tercer milenio (CAEPIA'2003)* San Sebastian, 11 November 2003.

[12] Shehory, O., and Sturm, A. Evaluation of modeling techniques for agent-based systems. In *Proceedings of the 5th Int'l Conference on Autonomous Agents*, ACM Press, 2001.

[13] Hoa K. Evaluating and Comparing Agent-Oriented Software Engineering Methodologies. MS Thesis, RMIT University, Melbourne, Australia, 2003.

[14] Iglesias, C., Garijo, M., and Gonzales, J. C. A survey of agent-oriented methodologies. In *Proceedings of the ATAL'98*, Intelligent Agents V, volume 1555 of LNAI. Springer, 1999.

[15] Rolland, C., Souveyet, C., and Moreno, M. An approach for defining ways-of-working. In *Information Systems Journal, 20(4)*, 2004.

[16] Nurcan, S., Rolland, C. Meta-modelling for cooperative processe. In *Proceedings of the 7 European-Japanese Conference on Information Modeling and Knowledge Bases*, Toulouse, France, May 27-30, 1997.

[17] Tran, Q. N. N., Low, G., Williams M. A. A Feature Analysis Framework for Evaluating Multi-agent System Development Methodologies. In *Proceedings of Foundations of Intelligent Systems, 14th International Symposium, ISMIS 2003*, Maebashi City, Japan, October 28-31, 2003, 613-617.

[18] Bădică, C., Ganzha, M., Paprzycki, M., and Pîrvănescu, A. Experimenting With a Multi-Agent E-Commerce Environment. In *Proceedings of the 2005 PaCT Conference*, Krasnoyarsk, Russia, 2005.

[19] Luck, M., and d'Inverno M. Unifying Agent Systems. *Int'l Journal of Annals of Mathematics and Artificial Intelligence*, Vol. 37, Number 1-2, January 2003.

[20] Caire, G., Coulier, W., Garijo, F. J., Gomez, J., Pavón, J., Leal, F., Chainho, P., Kearney, P. E., Stark, J., Evans, R., Massonet, P. Agent Oriented Analysis Using MESSAGE/UML. In *Proceeding of Agent-Oriented Software Engineering II, Second International Workshop, AOSE 2001*, Montreal, Canada, May 29, 2001,119-135.

[21] Cossentino, M. Burrafato, P., Lombardo, S. and Sabatucci, L. Introducing Pattern Reuse in the Design of Multi-Agent Systems. In *Proceedings of AITA'02 workshop at NODe02*, Erfurt, Germany, 8-9 October 2002.

[22] Cossentino, M., and Potts, M. A CASE tool supported methodology for the design of multi-agent systems. In *Proceedings of the 2002 International Conference on Software Engineering Research and Practice (SERP'02)*, 2002.

[23] Odell, J., Parunak, H. V. D. and Bauer, B. Extending UML for Agents. In *Proceedings of the Agent-Oriented Information Systems Workshop at the 17th National conference on Artificial Intelligence*, 2000.

[24] Wood, M.F., and DeLoach, S.A. An Overview of the Multiagent Systems Engineering Methodology. In *Proceedings of the First International Workshop on Agent-Oriented Software Engineering*, Agent-Oriented Software Engineering, Limerick, Ireland, June 10, 2000.

[25] Wood, M.F. Multiagent Systems Engineering: A Methodology for Analysis and Design of Multiagent Systems. PhD thesis, Air Force Institute of Technology, 2000.

[26] Deloach, S. A. Analysis and Design using MaSE and agentTool. In *Proceedings of 12 Midwest Artificial Intelligence and Cognitive Science Conference (MAICS 2001)*, Miami University, Oxford, Ohio, March 31 – April 1 2001.

[27] Padgham, L. and Winikoff, M. Prometheus: A Methodology for Developing Intelligent Agents. In *Proceedings of the Third International Workshop on AgentOriented Software Engineering (AAMAS 2002)*, Bologna, Italy, July, 2002.

[28] Padgham, L. and Winikoff, M. Prometheus: A Pragmatic Methodology for Engineering Intelligent Agents. In *Proceedings of the workshop on Agent-oriented methodologies at OOPSLA*, 2002.

[29] C. Iglesias, M. Garijo, J.C. Gonzales, and J.R. Velasco. Analysis and design of multiagent systems using mascommonkads. In Intelligent Agents IV (ATAL97), LNAI 1365, pages 313--326. Springer-Verlag, 1998.

[30] Iglesias, C.A., Garijo, M., Gonzalez, J.C., and Velasco, J.R. A methodological proposal for multiagent systems development extending commonkads. In *Proceedings of 10 KAW*, Banff (CAN), 1996. *th*

[31] Yu,E., and Cysneiros.L.M. Agent-Oriented Methodologies-Towards a Challenge Exemplar. In *Proceedings of the 4th Int'l. Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS 2002)*, Toronto, Canada, May 2002.

[32] Dam, K.H., and Winikoff, M. Comparing Agent-Oriented Methodologies. In *Proceedings of the 5th Int'l Bi-Conference Workshop on AgentOriented Information Systems (AOIS)*, Melbourne, Australia, 2003.

[33] Glaser, N. and Morignot, P. The Reorganization of Societies of Autonomous Agents. In *Proceedings of the*

Eighth European Workshop on Modeling Autonomous Agents in a Multi-Agents World, Multi-Agents Rationality, 1997.

[34] Julian, V. and Botti, V. Developing real-time multi-agent systems. In Proceedings of 4 Iberoamerican Workshop on Multi-Agent Systems, Iberagents, 2002.

[35] Colis, J. FruitMarket an Agent Trading Application. ZEUS Methodology Documentation Part II, Case Study 1, 1999 British Telecommunications plc, Release 1.01, November, 1999.