Evaluating the Feasibility of Method Engineering for the Creation of Agent-Oriented Methodologies

Brian Henderson-Sellers

Faculty of Information Technology,
University of Technology, Sydney,
PO Box 123, Broadway,
NSW 2007, Australia
brian@it.uts.edu.au

Abstract. In the context of agent-oriented methodologies, previous work has created a number of specific method fragments for use with the OPF metamodel and repository. These have been derived from an analysis of a large number of stand-alone agent-oriented methodologies. In order to evaluate the feasibility of this method engineering approach, a different AOSE methodology has been selected so that a scientific experiment could be undertaken. The hypothesis that the agent-enhanced OPF repository could be used without change to engineer any other AOSE methodology was proved false since two tasks for the creation of agents based on role modelling needed to be added to the repository.

1 Introduction

Method engineering requires the provision of a repository of method fragments from which industry strength methodologies can be created. For this to be successful, the repository contents need to be comprehensive in their support of a particular paradigm. Over the last two years, studies have been undertaken of what method fragments are needed to fully support agent-oriented software engineering (AOSE) methodologies. These were added to a repository that was originally not agent-oriented – the OPF repository based on an underpinning metamodel [1].

The fragments currently in the agent-enhanced OPF repository were gleaned from a number of AOSE methods. It is thus reasonable to anticipate that the fragments in the repository are sufficient for the support of these methods. As in a scientific experiment where a theory or numerical model is created by using data from *n* sources and then a set of data from an *n+1*th source is used for validation, here we propose the null hypothesis that the OPF repository is now comprehensive with respect to current AO methodological thinking and test this assumption by an evaluation of the mappings to the methodology proposed in [2], a methodology not in the original data set used to construct the repository enhancements.

2 A Method Engineering Framework – Based on OPF

Methodologies for industry may be a single, comprehensive and inter-related set of work units, work products and actors to perform the embedded software development process. Since it is generally recognized (e.g. [3]) that one size doesn't fit all, many researchers have sought for alternative approaches and frameworks. One of the most promising is that of method engineering (ME) [4,5]. In this approach, a "personalized" method is created for a specific organization, a specific division or a specific project by bottom-up construction from a number of method fragments [6]. These fragments have been pre-constructed and placed in a repository from which they can be selected by the organization's method engineer. Method fragments may describe a particular Task, such as AND/OR decomposition, or a particular Work Product, such as an Agent structure diagram etc., each of which is defined in terms of a clearly specified and standardized interface. The creation of these fragments may follow one of several paths [7,8] and ideally should be based upon an underlying conceptual model as embodied in a methodology metamodel [9].

While ME has been used in traditional and object-oriented (OO) methodologies for some years, it is only recently that the approach has been applied to the fragmentization of agent-oriented methodologies. In a precursor to the FAME (Framework for Agent-oriented Method Engineering) project, we have utilized the OPEN Process Framework (OPF) [1] – in particular, its underpinning metamodel and initially OO-based repository (Figure 1) – as a proof of concept. For later work in FAME (a funded three-year project: 2004-2006/7), we will use the OPF-based work for guidance and inspiration as we identify not only appropriate metamodel for agent-oriented modelling languages (see preliminary work in [10,11]) but also the process standard of AS4651 [12] and, subsequently, its ISO incarnation. Here, we report on the final test of the OPF-based research project.

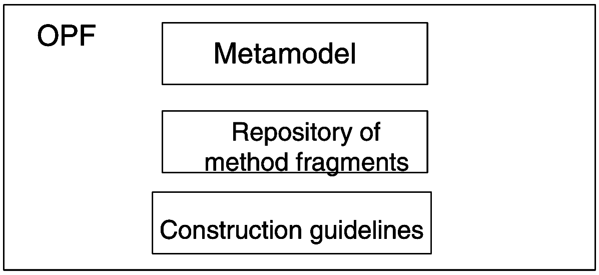

Fig. 1. The OPEN Process Framework consists of a metamodel from which is generated a large number of method fragments stored in repository. The OPF also includes a set of construction guidelines (not discussed here).

In the OPF pilot, we have augmented the OPF repository of method fragments by those derived from a large number of stand-alone agent-oriented methodologies, namely MaSE, Prometheus, Gaia, Cassiopeia, Agent Factory, MAS-Common-KADS, Tropos, PASSI and CAMLE. These span the various kinds of MAS identified in [14] – those based on object technology, either role-based or non-role-based, and those based on Knowledge Engineering (for details see summary in [13] – also Table 1).

Each of these fragments corresponds to one of the classes in the OPF metamodel. There are five major, top-level classes: Work Unit, Work Product, Producer, Stage and Language (Figure 2). There are also three important subtypes of Work Unit. These are Activity, Task and Technique. It was found, when undertaking the

Table 1. Summary of OPF method fragments previously gleaned from a number of AO methodologies. Listed are (a) new Tasks, (b) new Techniques and (c) new Work Products. Source documents referred to are [15-24] (after [25]).

(a) New Tasks and (indented) associated subtasks	**Refs**
Construct agent conversations	[15]
Construct the agent model	[15-18]
Define ontologies	[19]
Design agent internal structure	[16,19,20]
Define actuator module	[19]
Design perceptor module	[19]
Determine agent communication protocol	[21]
Determine agent interaction protocol	[21]
Determine control architecture	[21]
Determine delegation strategy	[21]
Determine reasoning strategies for agents	[21]
Determine security policy for agents	[21]
Determine system operation	[21]
Gather performance knowledge	[21]
Identify emergent behaviour	[21]
Identify system behaviours	[18]
Identify system organization	[21]
Define organizational rules	[17]
Define organizational structures	[17]
Determine agents' organizational behaviours	[18]
Determine agents' organizational roles	[18]
Identify sub-organizations	[17]
Model actors	[22]
Model agent knowledge	[20]
Model agent relationships	[20]
Model agents' roles	[21]
Model responsibilities	[17]
Model permissions	[17]
Model capabilities for actors	[22]
Model dependencies for actors and goals	[22]
Model goals	[22]
Model plans	[22]
Model the agent's environment	[21]
Model environmental resources	[17]
Model events	[16]
Model percepts	[16]
Specify shared data objects	[16]
Undertake agent personalization	[21]
Subtask to Create a System Architecture: Determine MAS infrastructure facilities	[19,20]

(b) New Techniques	**Ref**	**New Techniques**	**Ref**
Activity scheduling	[21]	Environmental evaluation	[23]
Agent delegation strategies	[21]	Environmental resources modelling	[17]
Agent internal design	[15,16]	FIPA KIF compliant language	[23]
AND/OR decomposition	[22]	Learning strategies for agents	[21]
Belief revision of agents	[21]	Market mechanisms	[21]
Capabilities identification & analysis	[22]	Means-end analysis	[22]
		Organizational rules specification	[17]
Commitment management	[21]	Organizational structure specification	[17]
Contract nets	[21]		
Contributions analysis	[22]	Performance evaluation	[21]
Control architecture	[21]	Reactive reasoning: ECA rules	[21]
Deliberative reasoning: Plans	[21]	Task selection by agents	[21]
		3-layer BDI model	[23]

(c) New Work Products	**Ref**	**New Work Products**	**Ref**
Agent acquaintance diagram	[16,17]	Network design model	[20]
Agent class card	[20]	Platform design model	[20]
Agent design model	[20]	Protocol schema	[16,17]
Agent overview diagram	[16]	PSM specification	[20]
Agent structure diagram	[16]	Role diagram	[15]
CAMLE behaviour diagram	[24]	Role schema	[17]
CAMLE scenario diagram	[24]	Service table	[17]
Caste collaboration diagram	[24]	Task hierarchy diagram	[20]
Caste diagram	[24]	Task knowledge specification	[20]
Coupling Graph	[18]	Task textual description	[20]
Domain knowledge ontology	[20]	(Tropos) Actor Diagram	[22]
Functionality descriptor	[16]	(Tropos) Capability Diagram	[22]
Goal hierarchy diagram	[15]	(Tropos) Goal Diagram	[22]
Inference diagram	[20]	(Tropos) Plan Diagram	[22]

augmentation of OPF's repository with AO-focussed fragments, only a small number of Activities were needed and that most of the fragments were instances of the metaclasses Task, Technique and WorkProduct.

Additions to the repository were undertaken incrementally so that for each successive AO methodology analysis, fewer and fewer new fragments were identified. In the penultimate analysis (that of PASSI [26]), only one new task was identified – although as with all other methodologies a significant number of (potentially overlapping) work products were proffered. Now, in this paper, as a check for closure we test the hypothesis that the OPF repository is now replete and capable of modelling and representing the process aspects any other AO methodology (the suites of diagrams used are still too varied between AO methodologies – they require separate, special treatment). To do this, we take the role-focussed methodological work of Kendall and colleagues [1,27-30], identifying the appropriate fragments contained in this approach and then evaluating whether the enhanced OPF repository can adequately model these.

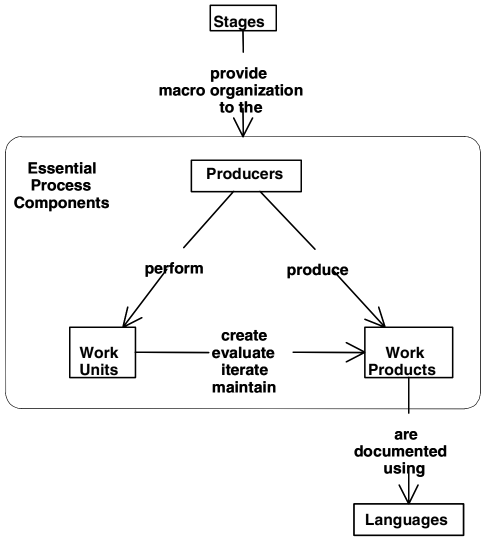
Stages
provide macro organization to the
Essential Process Components
Producers
perform
produce
Work Units
create evaluate iterate maintain
Work Products
are documented using
Languages

Fig. 2. The five major classes in the OPF metamodel

3 Evaluation Test – The Role-Based Methodology of Zhang *et al.*

A standard research methodology in science in the context of model building and model validation is to create a (often numerical) model, exemplifying some hypothesis, by incorporating a number of data sets. The model is then checked and modified until it is in accord with these data sets. However, this only provides an internal validity and consistency check. What then happens is that a "validation" experiment is conducted in which an objective assessment is undertaken against a new data set i.e. one that did NOT contribute to the model building and internal validation.

This is precisely the methodology used here. We have created an agent-oriented method engineering framework by using a large number of data sets. Here, each data set corresponds to our analysis of a single AO methodology as indicated in the references to Table 1. Having accomplished this creation stage, we now seek an external validation by identifying an AOSE methodology that has not yet been used in the creation of the agent-enhanced OPF repository. One such methodology is that of Zhang et al. [2] (referred to henceforth as the ZKJ methodology). This is a role-based methodology that falls clearly within the range of AO methodologies already analysed (as discussed in Section 2)[1].

The ZKJ methodology focusses on the identification of goals and roles. It represents the process by a set of ten "activities", each having an input, an output, a control and a mechanism. Four of these activities are grouped as "object-oriented analysis activities", the rest being focussed on agent goals and roles.

[1] Another possible choice, Adelfe [31], addresses adaptive MASs and is thus deemed out of scope for the present study.

The four object-oriented activities are Identify Actors, Identify Use Cases, Identify Objects and Determine Business Objects. These are said to be derived from [32]. There are six activities focussed on roles and goals. These are named, in this approach, Identify Goals, Develop Goal Cases and Identify Beliefs, Identify Roles, Assign Goals to Responsibilities, Assign and Compose Roles, and Identify Composite Roles.

Use cases, once identified, are used as input to the activity of Identify Goals. A goal in ZKJ is said to be a desired state that identifies what is to be done whereas an activity is viewed as a process that says how things are to be done. Identification of goals from use cases follows an iterative goal decomposition strategy as described in [30]. A variation on the use case is used to describe the interactions of the agents in the context of a specific goal – this is called a "goal case" and seen as central to agent identification describing as they do the set of plans that an agent will finally execute to achieve a particular goal. Here a single activity is used to develop these goal cases and, at the same time, identify the agent's beliefs, said to represent the knowledge that an agent possesses. Although coupled, the granularity of such a linking of two essentially different activities causes a potential clash with the atomic nature of the OPF repository-held method fragments (see Section 4).

Role identification is an important activity in the ZKJ methodology. Roles are seen as able to execute a set of activities in order to fulfil one or more responsibilities. Roles have access to resources and are identified from patterns of interaction and collaboration. The resultant set of roles is documented using a variant of the well-known CRC card [33], here called the Role, Responsibility and Collaboration (RRC) card. Roles are then linked to goals with the activity named Assign Goals to Responsibilities. This uses a goal hierarchy diagram and results in a set of RGC (Responsibility, Goal, Collaborator) cards – another CRC variant. The next activity, Assign and Compose Roles takes the results of the previous two activities to create specifications for agents in terms of their goals, goal cases, collaborators and beliefs. This is expressed with a GCB card (Goals, Goal Cases, Collaborators and Beliefs) used in the subsequent implementation stages. The last ZKJ activity, Identify Composite Roles, is not separately described and would appear to have serious overlaps with the activity of Assign and Compose Roles. With this lack of definitive information in [32], we remove it from the "test data" suite for this evaluation.

4 Test Results

To evaluate whether the agent-enhanced OPF repository provides all the method fragments needed for a method engineer to construct the ZKJ methodology, we need to ensure that all the method fragments described in Section 3 exist in the enhanced OPF repository (Table 1) and to provide a clear mapping from the OPF-based fragments to the new methodology. As noted earlier, we focus on the process-focussed method elements, primarily Activity, Task and Technique. In broad terms, a ZKJ activity maps to a Task in the OPF. However, since, according to [2], a ZKJ activity defines *how* things are to be done, there would appear to be a convolution of both an OPF Task and its corresponding Technique(s) into the ZKJ activity. Nevertheless, the descriptions and examples in [2] tend to focus on the task-like aspects rather than the technique-like aspects (using the OPF style of terminology).

This mapping is provided in Table 2. Here we see that the object-oriented activities of the ZKJ methodology map directly to method fragments (tasks) in the OPE repository. Of more interest here are the agent-focussed fragment mappings. Of the five activities from [2] only two are well supported, one in a many to one case and two are not. Thus the hypothesis is negated, indicating that a further iteration is required in adding method fragments to the OPF repository. Although we have shown previously [15-24] that there are sufficient fragments to recreate those particular methodologies i.e. those used in enhancing the OPF repository, our current study has highlighted areas of deficiency that must be remedied. We thus undertake further analysis in order to recommend yet further enhancements to the OPF repository.

Table 2. Mapping from fragments existing in the enhanced OPF repository to fragments described in the paper of Zhang *et al.* [2]

OPF repository fragment	**Zhang et al. fragment**
Model actors	Identify actors
Construct the object model with Technique: Scenario development	Identify use cases
Identify CIRTs	Identify objects
Develop business object model (BOM)	Identify business objects
Model goals	Identify goals
Model plans Model agent knowledge Technique: Scenario development	Develop goal cases and identify beliefs
Model agent roles	Identify roles
	Assign goals to responsibilities
	Assign and compose roles
Notes: CIRT stands for Class, Instance, Object, Type and is the generic "classifier" advocated in the OPF.	

In particular, we note that the ZKJ methodology's strong emphasis on roles identifies some AOSE-focussed tasks that none of the other methodologies either undertake or stress. The existing support in the agent-enhanced repository in the area of roles follows one standard viewpoint i.e. that agents are identified, followed by identification plus allocation of appropriate roles. This is a very object-oriented viewpoint where the entity (class, object or agent) is the main concept being modelled and roles are typically seen as incidental. In agent-oriented approaches there is a strong conviction by many AOSE methodologists that roles are really the predominant concept that is to be modelled [35-37]. In this viewpoint, roles come first and, having identified roles and perhaps associated goals and responsibilities (as here), then roles are composed into agents. Without entering into a debate about the pros and cons of such a viewpoint, it is certainly necessary to add new method fragments that permit such a viewpoint to be supported. Using the standard OPF interface description style, we thus propose the addition of the following two tasks:

TASK NAME: Assign goals to responsibilities

Typical supportive techniques: Agent internal design, responsibility identification, role assignment, 3-layer BDI model

Explanation: Using a goal hierarchy diagram as input and beginning at the leaves, each goal is assigned to a particular responsibility and collaborations between roles identified. The output is a RGC (Responsibility, Goal, Collaboration) card showing relationships between responsibilities, goals and the role's collaborators.

TASK NAME: Assign and compose roles

Typical supportive techniques: Agent internal design, composition structures, responsibility identification, role assignment, 3-layer BDI model

Explanation: Using the RGC cards, responsibilities and roles can now be composed together to identify agents, along with the agent's goals and collaborator. The steps to be undertaken involve

- Assigning roles for the design of the agent, composing them as needed
- Assigning roles to create agents, bearing in mind coupling and cohesion
- Possibly splitting and/or merging, based on the insight that the goals form the basic expertise of the agents.

In respect of the many-to-one mapping in Table 2, it is worth noting that a goal case is a specific form of a use case attached to a goal describing a set of plans. There is some support for this via the Task: Model plans as well as Technique: Scenario development. The notion of a goal case, per se, is, however, novel.

Finally, although we have stated that the work products are out of scope for this study, it is interesting to note the extensive use made of some variant of the CRC card in the ZKJ methodology. In the original OPF repository, Role Responsibility Collaborator cards were used [34] and goals were added in the Agent Class Card work product introduced in [23].

5 Future Extensions – Replacement of OPF by SMSDM

Since this study has highlighted deficiencies in the comprehensiveness of the method fragments in the enhanced OPF repository, further evaluative studies are needed. Another AOSE methodology, outside of the set so far used, will be identified and the above experiment repeated – iteratively until closure is reached. However, as the FAME project takes over from this prototype project using OPF, we intend to undertake a major replacement of the OPF metamodel and repository contents by a new, standard metamodel (SMSDM), as documented in [12]. Since this has a different conceptual architecture than OPF, some revision of both the conceptual basis and the generated fragments will ensue. One issue for future research is that of the appropriate granularity. For instance, as seen here, using a single activity/task, here the Develop Goal Cases and Identify Beliefs activity in ZKJ which needs three tasks to support it in OPF, indicates some problems with consistent granularity levels. This will also be investigated in an upcoming paper [in preparation].

Acknowledgements

I wish to acknowledge financial support from the University of Technology, Sydney under their Research Excellence Grants Scheme and from the Australian Research Council (grant number DP0451213). Thanks also to Cesar Gonzalez-Perez for his useful comments on an earlier draft. This is Contribution number 05/08 of the Centre for Object Technology Applications and Research.

References

1. Firesmith, D.G. and Henderson-Sellers, B., 2002, *The OPEN Process Framework. AN Introduction*, Addison-Wesley, Harlow, Herts, UK
2. Zhang, T.I., Kendall, E. and Jiang, H., 2002, An agent-oriented software engineering methodology with applications of information gather systems for LLC, *Procs AOIS-2002*, Toronto, May 2002, 32-46
3. Cockburn, A., 2000, Selecting a project's methodology, *IEEE Software*, **17(4)**, 64-71
4. Kumar, K. and Welke, R.J., 1992, Methodology engineering: a proposal for situation-specific methodology construction, in *Challenges and Strategies for Research in Systems Development* (eds. W.W. Cotterman and J.A. Senn), J. Wiley, Chichester, 257-269
5. Brinkkemper, S., 1996, Method engineering: engineering of information systems development methods and tools, *Inf. Software Technol.*, **38(4)**, 275-280.
6. Ralyté, J., Rolland, C. and Deneckère, R., 2004, Towards a meta-tool for change-centric method engineering: a typology of generic operators, *Procs. CAiSE 2004*, LNCS 3084, Springer, 202-218
7. Ralyté, J. and Rolland, C., 2001, An assembly process model for method engineering, *Advanced Information Systems Engineering*), LNCS2068, Springer, 267-283.
8. Ralyté, J., 2004, Towards situational methods for information systems development: engineering reusable method chunks, *Procs. 13th Int. Conf. on Information Systems Development. Advances in Theory, Practice and Education* Vilnius Gediminas Technical University, Vilnius, Lithuania, 271-282
9. Henderson-Sellers, B., 2003, Method engineering for OO system development, *Comm. ACM*, **46(10)**, 73-78
10. Beydoun, G., Gonzalez-Perez, C., Low, G. and Henderson-Sellers, B., 2005, Synthesis of a generic MAS metamodel, *Procs. SELMAS2005* (eds. A. Garcia, R. Choren, C. Lucena, A. Romanovsky, T. Holvoet and P. Giorgini), IEEE Digital Library, IEEE, Los Alamitos, CA, USA, 27-31
11. Beydoun, G., Gonzalez-Perez, C., Low, G. and Henderson-Sellers, B., 2005, Towards method engineering for multi-agent systems: a preliminary validation of a generic MAS metamodel, *Procs. AOSDM'2005 at SEKE'05*, Taipei, 14-16 July 2005
12. Standards Australia, 2004, Australian Standard 4651-2004: Standard metamodel for software development methodologies, 72pp
13. Henderson-Sellers, B., 2005, Creating a comprehensive agent-oriented methodology - using method engineering and the OPEN metamodel, Chapter 13 in *Agent-Oriented Methodologies* (eds. B. Henderson-Sellers and P. Giorgini), Idea Group, Hershey, PA, USA
14. Tran, Q.-N.N., Low, G.C. and Williams, M.-A., 2005, A preliminary comparative feature analysis of multi-agent systems development methodologies, *Agent-Oriented Information Systems II*, LNAI 3508, Springer, 157-168

15. Tran, Q.-N.N., Henderson-Sellers, B. and Debenham, J. 2004, Incorporating the elements of the MASE methodology into Agent OPEN, *Procs. ICEIS2004 - Sixth International Conference on Enterprise Information Systems,* INSTICC Press, **Volume 4**, 380-388
16. Henderson-Sellers, B., Tran, Q.-N.N. and Debenham, J., 2004, Incorporating elements from the Prometheus agent-oriented methodology in the OPEN Process Framework, *Procs. AOIS@CAiSE2004*, Riga Technical University, Latvia, 370-385
17. Henderson-Sellers, B., Debenham, J. and Tran, Q.-N.N., 2004, Adding agent-oriented concepts derived from GAIA to Agent OPEN, *Procs. 16th International Conference, CAiSE 2004,* LNCS 3084, Springer, 98-111
18. Henderson-Sellers, B., Tran, Q.-N.N. and Debenham, J., 2004, Method engineering, the OPEN Process Framework and Cassiopeia, *The Symposium on Professional Practice in AI* (eds. E. Mercier-Laurent and J. Debenham), IFIP, 263-272
19. Henderson-Sellers, B., Tran, Q.-N.N., Debenham, J. and Gonzalez-Perez, C., 2005, Agent-oriented information systems development using OPEN and the Agent Factory, *Procs. ISD 2004,* Vilnius, 9-11 September 2004, Kluwer, 149-160
20. Tran, Q.-N.N., Henderson-Sellers, B., Debenham, J. and Gonzalez-Perez, C., 2004, MAS-CommonKADS and the OPEN method engineering approach, *Procs. ICITA*, Sydney, July 4-7 2005, IEEE Computer Society Press
21. Debenham, J. and Henderson-Sellers, B., 2003, Designing agent-based process systems - extending the OPEN Process Framework, Chapter VIII in *Intelligent Agent Software Engineering* (ed. V. Plekhanova), Idea Group Inc., Hershey, PA, USA, 160-190
22. Henderson-Sellers, B., Giorgini, P. and Bresciani, P., 2004, Enhancing Agent OPEN with concepts used in the Tropos methodology, *Engineering Societies in the Agents World IV. 4th International Workshop, ESAW' 2003,* LNAI 3071, Springer, 328-345
23. Henderson-Sellers, B. and Debenham, J., 2003, Towards OPEN methodological support for agent-oriented systems development, *Procs. First International Conference on Agent-Based Technologies and Systems*, University of Calgary, Canada, 14-24
24. Gonzalez-Perez, C., Henderson-Sellers, B., Debenham, J., Low, G.C. and Tran, Q.-N.N., 2004, Incorporating elements from CAMLE in the OPEN repository, *Intelligent Information Process II*, Springer, 55-64
25. Henderson-Sellers, B., 2005, From object-oriented to agent-oriented software engineering methodologies, *Software Engineering for Multi-Agent Systems - Volume III Research issues and practical applications*, LNCS 3390, Springer, 1-18
26. Henderson-Sellers, B., Debenham, J., Tran, N., Cossentino, M. and Low, G., 2005, Identification of reusable method fragments from the PASSI agent-oriented methodology, *Procs. AOIS@AAMAS2005*, 26 July 2005, 89-96
27. Kendall, E.A., Malkoun, M. and Jiang, C., 1995, A methodology for developing agent based systems for enterprise integration, *EI'95. IFIP TC5 SIG Working Conf. on Modeling and Methodologies for Enterprise Integration*, Heron Island, Queensland, Australia
28. Kendall, E.A., Malkoun, M.T. and Jiang, C., 1997, Multiagent system design based on object-oriented patterns, *J. Obj.-Oriented Prog. (ROAD)*, **10(3),** 41-47
29. Kendall, E.A., 1998, Agent roles and role models: new abstractions for multi-agent system analysis and design, *Int. Workshop on Intelligent Agents in Information and Process Management, German Conference on AI,* Bremen, Germany, September 1998
30. Kendall, E.A., Krishna, M., Pathak, C.V. and Suresh, C.B., 1998, Patterns of intelligent and mobile agents, *Agents '98*, May 1998.
31. Bernon, C., Gleizes, M.-P., Picard, G. And Glize, P., 2002, The ADELFE methodology for an intranet system design, *Procs. AOIS2002*, Univ. Toronto, 27-28 May 2002, 1-15

32. Jacobson, I., Christerson, M., Jonsonn, P. and Overgaard, J., 1992, *Object-Oriented Software Engineering – A Use Case Driven Approach*, Addison Wesley
33. Beck, K. and Cunningham, W., 1989, A laboratory for teaching object-oriented thinking, *Procs. 1989 OOPSLA Conference, ACM SIGPLAN Notices*, **24(10),** 1-6
34. Firesmith, D.G., Henderson-Sellers, B. and Graham, I., 1997, *OPEN Modeling Language (OML) Reference Manual*, SIGS Books, New York, NY, USA, 271pp
35. Odell, J.J., Parunak, H.V.D. and Fleischer, M., 2003, The role of roles in designing effective agent organizations, *Software Engineering for Large-Scale Multi-Agent Systems. Research Issues and Practical Applications*, LNCS 2603, Springer, 27-38
36. Koning, J.-L. and Hernandez, I.R., 2004, Limitations in AUML's roles specification, *Intelligent Information Processing II*, Springer, 79-82
37. Odell, J., Nodine, M. and Levy, R., 2004, A metamodel for agents, roles, and groups, *Agent-Oriented Software Engineering V. 5th International Workshop. AOSE 2004*, LNCS 3382, Springer, 78-92