From Object-Oriented to Agent-Oriented Software Engineering Methodologies

Brian Henderson-Sellers

University of Technology, Sydney, NSW 2007 Australia
brian@it.uts.edu.au

Abstract. Object-oriented methodologies are well-established and have been used as one input for the creation of methodologies suitable to support the development of agent-oriented software systems. While these agent-oriented (AO) methodologies vary in style and, particularly, in heritage and often with a specific focus (either in terms of domain, application style or lifecycle coverage), for industry adoption it is essential that full lifecycle coverage is achieved in a "standardized" way. One way of achieving some degree of standardization yet maintaining full flexibility is through the use of situational method engineering (SME). With this approach, method fragments are created and stored in a repository. For an individual software development, a subset of these is then selected from the repository and a project-specific (or sometimes organization-specific) methodology is constructed. Here, we demonstrate how this might work by using the OPEN approach that already provides a significant coverage of AO method fragments as well as more traditional OO and pre-OO fragments. Those newer fragments supporting AO approaches are detailed, describing, as they do, emerging substantial support for AO methodological creation from the OPEN repository in an SME context.

1 Introduction

Interest in the creation of appropriate software engineering methodologies for supporting the development of agent-oriented (AO) software systems has shown a rapid increase recently. For many AO methodologists, the object paradigm is seen as a useful precursor. Consequently, many AO methodologies exhibits traits inherited from earlier object-oriented (OO) methodologies – either explicitly or implicitly. On the other hand, some AO methodology writers deny any such influence.

In most cases, the meaning of "AO" in the term "agent-oriented methodology" means a methodology to be used for building agent-oriented software systems. However, in one case (Tropos, e.g. Bresciani *et al.,* 2004), it is used to mean that the agent concept is used in the conceptual underpinning of the methodology itself.

It should be noted that although we use the term "methodology", which means a full description of process, people, social structures, project management, modelling language, products etc. (e.g. Henderson-Sellers, 1995; Rolland and Prakash, 1996), some of the methodologies referred to in this paper provide only partial support – perhaps in terms of only addressing analysis and design (as does Gaia e.g. Wooldridge *et al.*, 2000; Zambonelli *et al.*, 2003) or omitting any discussion of the

R. Choren et al. (Eds.): SELMAS 2004, LNCS 3390, pp. 1–18, 2005.

"people element", for instance, MaSE (DeLoach, 1999) or AOR (Wagner, 2004), the latter being primarily a modelling language.

In this paper, we examine the evolution of agent-oriented methodologies and their relationship to earlier AO and OO methodologies leading to suggestions for future AO methodology support that may be of interest to industry. In Section 2, we analyze the various extant AO methodologies in terms of their OO/non-OO lineage. In Section 3 we debate the difference between a "one-size-fits-all" methodological approach versus a more flexible approach, the latter using situational method engineering (SME). The SME approach is then illustrated by a case study (Section 4) using the OPEN metamodel and repository of method fragments (Graham *et al.,* 1997; Henderson-Sellers *et al.,* 1998), recently extended to offer wide support for agents.

2 Methodology Genealogy

The development of AO methodologies has taken many routes. Some methodologists have based their methodological approach on an Artificial Intelligence or Knowledge Representation; others have commenced with basic definitions of objects and then asked what modifications are necessary to support agents; others have commenced with an established OO methodology and asked how agent support can be grafted on.

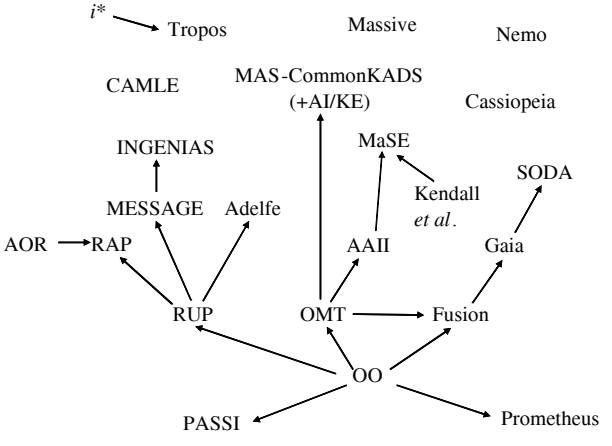

Fig. 1. Genealogy of various AO methodologies and their relationships to OO methodologies

Figure 1 graphically depicts some of these linkages and influences. OO methodologies such as RUP (Kruchten, 1999), OMT (Rumbaugh *et al.*, 1991) and Fusion (Coleman *et al.*, 1994) have all been used by various AO methodology groups as the basis for agent-oriented extensions. RUP has formed the basis for Adelfe (Bernon *et al.*, 2002) and also for MESSAGE (Caire *et al.,* 2001), which, in turn, is the basis for INGENIAS (Pavon *et al.*, 2005) and, more recently, RUP has been a useful input to RAP (Wagner and Taveter, 2005), a direct descendant of AOR (Wagner, 2003). OMT is said to have directly influenced MAS-CommonKADS (Iglesias *et al.*, 1996, 1998), which merges these OO ideas with concepts from AI and Knowledge Engineering, as

well as the AAII approach (Kinny *et al.*, 1996) which, in turn, is said to have been a major influence on MaSE (DeLoach, 1999; Wood and DeLoach, 2000). Fusion has strongly influenced Gaia which, in turn, has influenced SODA (Omicini, 2000). Prometheus (Padgham and Winikoff, 2002a,b) is a fully AO methodology but states that one should use UML-style diagrams when appropriate rather than "reinvent the wheel". All of these AO methodologies are "standalone" – effectively "one size fits all" – approaches.

Other methodologies in Figure 1 do not acknowledge any influence from any OO approach – although clearly some have had an implicit influence. Tropos is said to be based on *i** (Yu, 1995) and has a distinct strength in early requirements modelling. Its use of the *i** modelling language gives it a different look and feel to those that use Agent UML (AUML: Odell *et al.*, 2000) as a notation. It also means that the non-OO mindset permits users of Tropos to take a unique approach to the modelling of agents in the methodological context.

There is no obvious, explicit evidence of an OO influence in the published versions of Nemo (Huget, 2002), MASSIVE (Lind, 2001), Cassiopeia (Collinot *et al.*, 1996; Collinot and Drogoul, 1998), PASSI (Cossentino and Potts, 2002; Burrafato and Cossentino, 2002)[1] and the work of Kendall *et al.* (1996). CAMLE (Shan and Zhu, 2004) does, however, draw some parallels, particularly between a CAMLE caste and an OO class and with respect to UML's composition and aggregation relationships.

Several authors have made direct comparisons of these (and other) AO methodologies. Cernuzzi and Rossi (2002) proposed a framework containing a set of internal attributes (autonomy, reactivity, proactiveness and mental notions), a set of interaction attributes (social ability, interaction with the environment, multiple control, multiple interests and subsystems interaction) and four other requirements (modularity, abstraction, a system view and communication support). They used this framework in a case study to evaluate a BDI focussed methodology (Kinny *et al*., 1996, variously referred to as AAII or BDIM) and MAS-CommonKADS (Iglesias *et al.*, 1998) both qualitatively and, with an appropriate set of metrics, quantitatively. This study and other comparative evaluations of both AO and OO methodologies were used as input to the framework proposals of Dam and Winikoff (2004) who proposed four categories: concepts, modelling language, process and pragmatics. Their contribution is that the evaluation was not only done by the authors but by surveying a set of students who had used the case study methodologies (MaSE, Prometheus and Tropos) on a design problem of a mobile travel planner. The same four categories were used by Sturm and Shehory (2004) and used to evaluate Gaia (as a single example) using a seven point quantitative metric scale. The framework of Tran *et al.* (2003) also has four categories but these are said to be process-related (15 criteria), technique-related (5), model-related (23) and other supportive features (8). The framework was applied by Tran *et al.* (2004b) to five well-referenced AO methodologies – namely MaSE, Gaia, BDIM, Prometheus and MAS-CommonKADS. Different ordinal scales are used for the several criterion sets. A more extensive set of results (the evaluation of 10 AOSE methodologies) is found in Tran and Low (2005).

[1] A more recent manuscript in preparation does, in fact, acknowledge influences from object technology.

3 Specific or General Methodologies?

To support any software development, there would appear to be (at least) three options: (i) create a suite of inflexible methods, each of which is highly tuned to specific operating conditions; (ii) create a single all-inclusive methodology and then permit some removal of unwanted elements (sometimes known as method tailoring); and (iii) create not a methodology but a methodological framework underpinned by the concepts of situational method engineering (see Section 3.2 below) that permits the construction of multiple, specifically configured methodologies – one for each particular operating situation.

Using a suite of methodologies provides perfect alignment with the problem at any given time but, as situations change, provides no route for migration from the current methodology to a second in the suite, however perfect that second one might be for the new problem space. Thus, there is no possibility of encouraging the valuable process of Software Process Improvement or SPI, as advocated by e.g. CMM or SPICE (ISO 15504) because there is no route between these methodological "islands" (Figure 2).

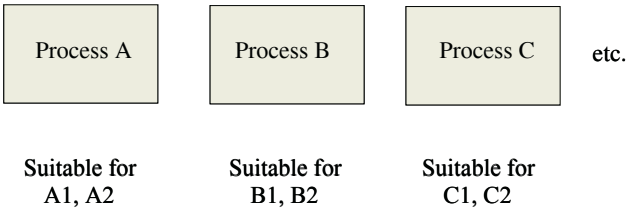

Fig. 2. "Islands" of methodology provide no route to migrate between them and hence there is no potential for SPI

Using a comprehensive methodology typically requires users to understand all elements of the approach before beginning a reduction programme i.e. eliminating the elements of this comprehensive methodology that are not needed for this specific project. This can mean wasted effort and such so-called heavyweight methodologies are often seen as anathema to contemporary problems (Avison and Fitzgerald, 2003) which are often said to require more "agile" approaches to software development.

In many ways, the "best of both these worlds" can be achieved through the third option and that is the one we will explore in this paper in more detail below (Section 3.2) following a brief overview of some of the AO methodology "islands" (Section 3.1) currently available for use.

3.1 Specific AO Methodologies

Many individualistic methodologies have been formulated and published. Here, we review briefly a small selection, focussing on those that have already been analyzed in order to extract method fragments (see Sections 3.2 and 4). Each description below emphasizes the *agent*-oriented aspects of that methodology, needed to go beyond the basic object-oriented concepts that many of them utilize.

Prometheus (Padgham and Winikoff, 2002a,b) is an agent-oriented methodology that reuses as many appropriate elements as possible from object technology including several UML diagram types. In the first phase (of three) of systems specification, the basic functionality of the system is identified, using percepts (inputs), actions (outputs) and any necessary shared data storage. This is followed by the architectural design stage; here, the agents and their interactions are identified. Finally, there is the detailed design phase in which the internal details of each agent are addressed.

MASE (DeLoach, 1999; Wood and DeLoach, 2000) is drawn from the legacy of object-oriented methodologies such as OMT together with influences from the more recent UML as well as pre-existing work in the realm of agents and multiagent systems e.g. Kinny *et al.* (1996) and Kendall and Zhao (1998). It aims to guide the designer through the multiagent-system development process from an initial system specification to a set of formal design documents. It has two phases: analysis and design. The former deals with the specification of system goals, use cases, sequence diagrams, roles and tasks, while the latter uses the analysis phase's outputs to design agent classes, agent interactions and agents' internal components. It is also well supported by a software tool.

Gaia (Wooldridge *et al.,* 2000) views the process of multi-agent system (MAS) development as a process of *organizational design*, where the MAS is modelled as an organized society with agents playing different roles. The methodology allows a developer to move systematically from a statement of requirements to a design that is sufficiently detailed that it can be implemented directly. It supports both macro (societal) and micro (agent) aspects of MAS design, and is also neutral to both application domain and agent architecture. The newest version of Gaia (Zambonelli *et al*., 2003) extends the original version with various organizational abstractions, enabling it to be used for the design of open MAS (which was not achievable previously).

Cassiopeia (Collinot *et al.*, 1996) provides an (arguably incomplete) methodological framework for the development of collective problem-solving MASs. Cassiopeia assumes that, although the agents can have different aims, the goal of the designer is to make them behave cooperatively. It adopts an *organization-oriented approach* to MAS design, as do some other AO approaches, viewing an MAS as an organization of agents that implement/encapsulate *roles*. These roles not only reflect the agents' individual functionality, but also the structure and dynamics of the organization of the MAS.

MAS-CommonKADS (Iglesias *et al.*, 1998) is an agent-oriented methodology that supports the development of MAS from the conceptualization phase through to a detailed design that can be directly implemented. The methodology integrates techniques from a well-known knowledge-engineering methodology, CommonKADS (Schreiber *et al.*, 1994), with those from OO methodologies (e.g. OMT, OOSE and RDD) and protocol engineering. The main modelling concepts in MAS-CommonKADS are agent, knowledge, organization and coordination.

Agent Factory (Collier *et* al, 2003, 2004) is a four-layer framework for designing, implementing and deploying multi-agent systems. It contains (i) an agent-oriented software engineering methodology, (ii) a development environment, (iii) a FIPA-compliant runtime environment and (iv) an agent programming language (AF-APL); with a stated preference for the BDI agent architecture according to the analysis of (Luck *et al.*, 2004). By employing UML and Agent UML, the Agent Factory method-

ology provides a visual, industry-recognized notation for its models - regarded by its authors as a major advantage over other approaches, such as Gaia (Wooldridge *et al.*, 2000) and Tropos (Bresciani *et al.*, 2004), which have non-standard (i.e. non-UML compliant) notations. These models are capable of promoting design reuse (via the central notion of *role*) and being directly implemented by automated code generation (Collier *et al.*, 2004).

CAMLE (Shan and Zhu, 2004) is described as a caste-centric agent-oriented modelling language and environment. It is caste-centric because *castes*, analogous to classes in object-orientation, are argued to provide the major modelling artefact over the lifecycle by providing a type system for agents. A significant difference is claimed between castes and classes: while objects are commonly thought of as statically classified (i.e. an object is created as a member of a class and that is a property for its whole lifetime), agents in CAMLE can join and leave castes as desired, thus allowing dynamic reclassification. CAMLE provides a graphical notation for caste models (similar to class models in OO methodologies), collaboration models and behaviour models. Caste diagrams also include support for the non-OO relationships of congregation, migration and participation. CAMLE relies heavily on the fact that an information system already exists when a new project is started, so that the new system is designed as a modification to the current one. Although this situation is indeed common, the construction of systems from scratch also happens. CAMLE, however, seems to ignore this possibility.

Tropos (Perini *et al.*, 2001; Castro *et al.,* 2002; Bresciani *et al.,* 2004) was designed to support agent-oriented systems development with a particular emphasis on the early requirements engineering phase. The stated aim was to use agent concepts in the description and definition of the methodology rather than using OO concepts in a minor extension to existing OO approaches. Tropos takes the BDI model (Rao and Georgeff, 1995; Kinny *et al.*, 1996), formulated to describe the *internal* view of a single agent, and applies those concepts to the *external* view in terms of problem modelling as part of requirements engineering. It also relies heavily on the *i** framework of Yu (1995) for concepts and notation.

In summary, there is a tendency to reuse significant portions of object-oriented methodological approaches, supplementing them with a new focus on organizations, social interactions, proactivity and roles. There is still discussion about the extent to which UML can be useful. Several AO methodologies use existing UML or, often, AUML diagrams but, at the same time, find deficiencies for which they supply new diagrammatic representations. In particular, there is still argument as to whether an agent concept could be added to the UML metamodel simply as a subtype of the Classifier metaclass or whether a totally different conceptualization is needed (e.g., Silva and Lucena, 2004).

3.2 General Methodologies – The Use of Situational Method Engineering

In contrast to an individual AO methodology, we now explore the third option of creating a methodological framework. In particular, in this section we outline the concepts of situational method engineering or SME (Kumar and Welke, 1992; Brinkkemper, 1996; Ter Hofstede and Verhoef, 1997).

SME suggests that the elements of a methodological can be modularized and encapsulated as "method fragments" (van Slooten and Hodes, 1996). The method fragments can then be connected to form larger fragments and finally the whole methodology. There is thus no initial or default methodology stored in the method repository or methodbase (e.g. Brinkkemper, 1996; Ralyté and Rolland, 2001) and indeed the methodbase may contain conceptual fragments originating from various sources. Ideally, the method fragments should all be instances of a concept captured in a metamodel underpinning the methodbase (Ralyté and Rolland, 2001; Henderson-Sellers, 2003). The metamodel provides essentially a set of rules and prescriptive descriptions of all the kinds of method elements permissible within the methodbase.

The challenge for the method engineer is to select appropriate and compatible fragments and to construct the final methodology (e.g. Wistrand and Karlsson, 2004). This may be from scratch or as an extension to an existing methodology (Ralyté *et al.*, 2003). Thus, construction guidelines (e.g. Klooster *et al.*, 1997; Brinkkemper *et al.*, 1998; Rolland *et al.,* 1999; Ralyté and Rolland, 2001; Ralyté *et al.*, 2004) are critical in the SME approach. Creating a project-specific methodology is currently one of the more difficult and time-consuming jobs of the method engineering approach, since the method engineer has to understand the methodology, the organization, the environment and the software project in order to select the appropriate fragments from the repository to use on the project as well as understanding the rules of construction. Traditionally, this process is carried out using predefined organizational requirements and the experience and knowledge of the method engineer or process engineer (e.g. Fitzgerald *et al.*, 2003), although significant tool support is likely in the near future (Saeki, 2003; Wistrand and Karlsson, 2004).

4 Case Study: Supporting Agent-Oriented Software Engineering Using the OPEN Framework

One example of a method engineering approach that can encompass both object-oriented and agent-oriented methodological thinking is the OPEN Process Framework or OPF (Firesmith and Henderson-Sellers, 2002). OPEN adopts a framework approach based on an underpinning metamodel, and has recently been extended from its original object-oriented base to include methodological support for agents (see, e.g., Henderson-Sellers and Debenham, 2003). As with any method engineering approach, OPEN aims to provide a repository of method fragments that will offer direct as well as extensible support for the construction of individually tailored methodologies for use in both industry and research environments.

OPEN's method fragments are generated directly from its metamodel (Figure 3) and stored in the OPF repository. To create a situated methodology, various method fragments are then chosen from this repository and combined to describe the process, associated people and social issues, deliverables and so on – each of which is defined formally by the corresponding metalevel element in the metamodel (Figure 4). In other words, a full-scale and comprehensive methodology can be constructed from the repository fragments. This could have an object-oriented, an agent-oriented or even a traditional (procedural-focussed) bias.

Using the tenets of SME outlined above, such a methodology can be specifically constructed and tailored towards a specific project or a specific organizational

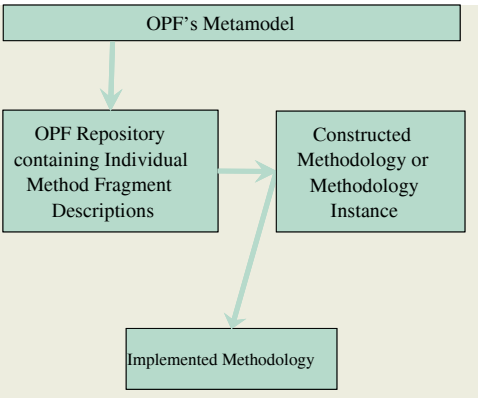

Fig. 3. OPEN defines a framework consisting of a metamodel and a repository of method fragments

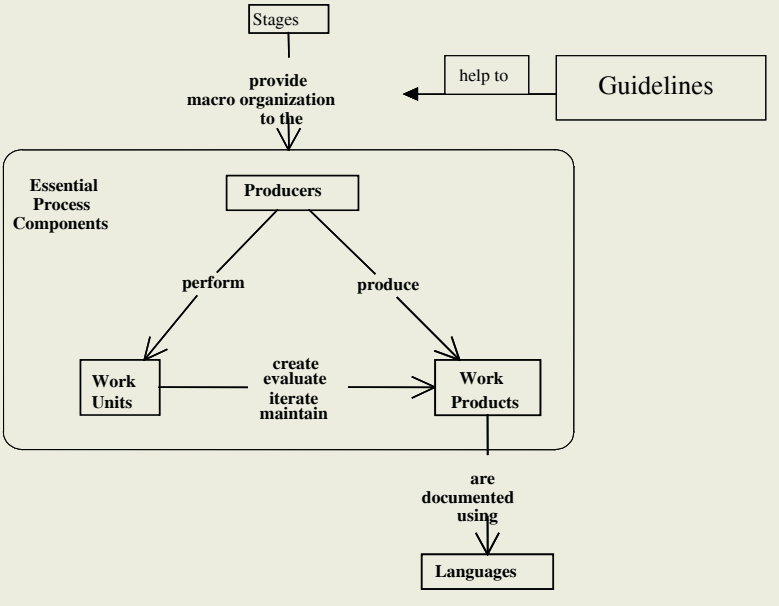

Fig. 4. The five top-level metaclasses of the OPF's metamodel (after Firesmith and Henderson-Sellers, 2002) © Addison-Wesley

"standard" using the supplied construction guidelines (Figure 5) together with a set of deontic matrices (Figure 6). These matrices support the identification of fuzzy relationships between pairs of method fragment types e.g. linkages between tasks and techniques. Deontic values have one of five values ranging from mandatory through optional to forbidden. This gives a high degree of flexibility to the process engineer, perhaps assisted by an automated tool (Nguyen and Henderson-Sellers, 2003), who can allocate appropriate deontic values to any specific pair of process components depending upon the context i.e. the specific project, skills set of the development team etc.

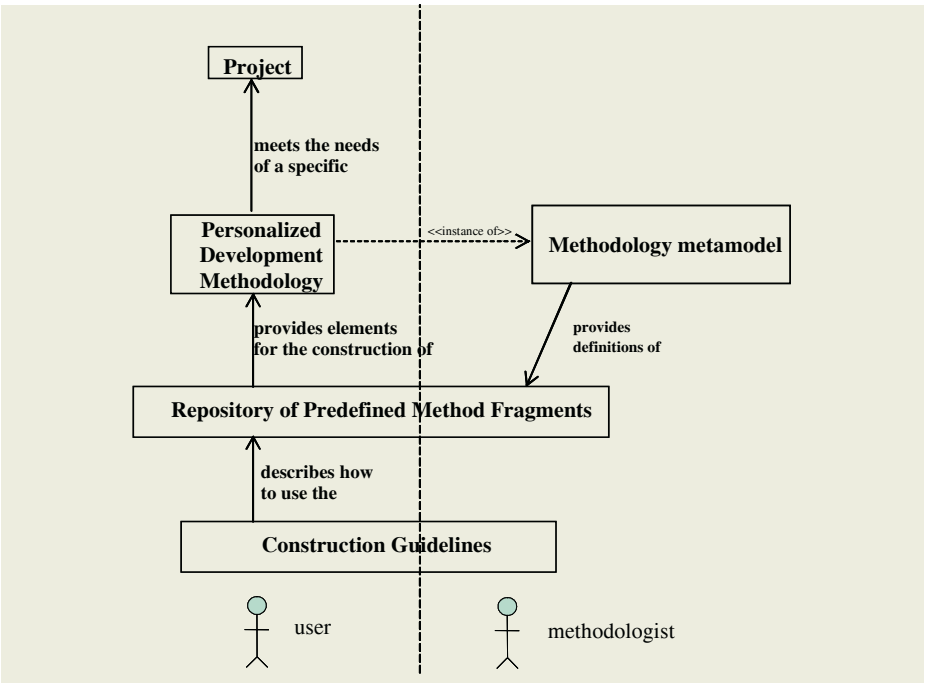

Fig. 5. The methodologist is responsible for the methodology metamodel, creating most of the method fragments in the repository and the guidelines for construction. The user (often the in-house method engineer) uses these guidelines and the contents of the repository (to which they are at liberty to add new fragments) in order to create a "personalized development methodology" attuned to a specific project or context

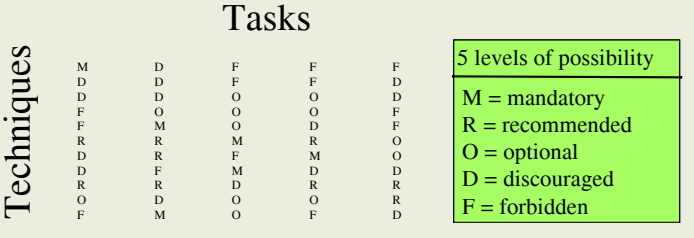

Fig. 6. One of the deontic matrices is used to link Tasks to Techniques. The values in the matrix represent the likelihood of the occurrence of that pair using five levels of possibility (redrawn from Henderson-Sellers *et al.*, 1998) © Addison-Wesley

Initially, the OPF repository contained about 30 predefined instances of Activity, 160 instances of Task and 200 instances of Techniques (the three main kinds of Work Unit) as well as multiple instances of Role, Stage, Language etc. Some of these are orthogonal to all others in their group and some overlap. Consequently, during process construction both association and integration strategies (Ralyté and Rolland, 2001) are needed. For example, there are several Techniques in the repository for finding objects e.g. textual analysis, use case simulations, CRC card techniques.

As noted above, currently one of the hardest tasks in SME construction is the selection of the optimal set of method fragments to suit any specific situation. Syntactic coupling can be verified in terms of the matching of the output from one fragment to the input for a second. This is facilitated by both the generation of the fragments from a metamodel and also by using a standard way of documenting the fragments (as is done in the OPEN book series, for instance). Nevertheless, the current reality is that the semantic aspect of the fragments must be analyzed "by hand", usually by a skilled method engineer (either in-house or as a visiting consultant or mentor). Work towards a more objective approach is under way (e.g. Nguyen and Henderson-Sellers, 2003; Ralyté, 2004).

Although originally created to support object-oriented software development, several additions have been made to the OPF repository since its first publication in 1997 in order to enhance its support for various new technologies, including additions of relevance to agent technology. In a series of papers, summarized in Henderson-Sellers (2005), we have proposed 39 new Tasks and Subtasks, 23 new Techniques and 28 new Work Products as well as a single new Activity. The method fragments, listed by name only in Table 1, thus provide a significant step in creating a fully supportive AO methodology applicable to a wide variety of types of agent-oriented software development approaches.

It should be noted that of these newly added method fragments, there are a number in common to several of the analyzed AO methodologies. For example, the Task "Construct the agent model" is, naturally, common. Prometheus tends to focus on providing extensions to an OO approach. Consequently, some of the diagrams supported in Prometheus (Pagdham and Winikoff, 2002a,b) can be viewed as UML extensions. Tropos (Bresciani *et al.*, 2004), on the other hand, strive to avoid mere OO extensions and use the AO paradigm explicitly in their modelling of the methodology itself. This introduces some novel diagrams and tasks, which focus on capabilities, as well as on goals and plans. Their focus on early requirements also leads to the need to add a new Activity instance, that of Early Requirements Engineering, to the OPEN repository in order that users of OPEN can re-create the Tropos approach to AO systems development. Gaia (Wooldridge *et al.*, 2000; Zambonelli *et al.*, 2003) is more interested in providing supporting for organizational and social interaction aspects of agents – as is Cassiopeia (Collinot *et al*, 1996; Collinot and Drogoul, 1998) and, to a significant extent, Tropos. This leads to the modelling of responsibilities and permissions as well as the specification of organizational rules, roles, structure and behaviour.

Creation of a project-specific or organization-specific agent-oriented methodology then proceeds using the specifically agent-oriented method fragments listed in Table 1 (which tend to focus only on areas *different* from object-oriented approaches) together with a number of non-agent-oriented method fragments that are needed for those elements of software development that are not technology/paradigm-dependent. These include method fragments to describe project management, some metrics, reusability and so on. A fully comprehensive methodology, suitable for direct industry usage, can be constructed in this way; alternatively, one of the existing AO methodologies can be reconstructed by using only those specific AO fragments. For instance, Henderson-Sellers (2005) shows in more detail how a version of the Prometheus methodology enhanced with some Tropos concepts can be put together from the method fragments

in this newly enhanced OPF repository. Figure 7 shows a portion of the Task-Technique matrix enacted (from Figure 6) for this case study. This shows one way of constructing these matrices. Candidate Techniques (in this example) have been

Table 1. Summary of (a) new Tasks, (b) new Techniques and (c) new Work Products so far added to OPEN in the creation of Agent OPEN. Source documents referred to are: 1. Debenham and Henderson-Sellers (2003), 2. Henderson-Sellers and Debenham (2003), 3. Henderson-Sellers *et al.* (2004a), 4. Henderson-Sellers *et al.* (2004c), 5. Tran *et al.* (2004a), 6. Henderson-Sellers *et al.* (2004b), 7. Henderson-Sellers *et al.* (2004d), 8. Tran *et al.* (2004c), 9. Henderson-Sellers *et al.* (2004e) and 10. Gonzalez-Perez *et al.* (2004)

(a) New Tasks and (indented) associated subtasks	**Refs**
Construct agent conversations	5
Construct the agent model	4, 5, 6, 7
Define ontologies	9
Design agent internal structure	4, 8, 9
Define actuator module	9
Design perceptor module	9
Determine agent communication protocol	1
Determine agent interaction protocol	1
Determine control architecture	1
Determine delegation strategy	1
Determine reasoning strategies for agents	1
Determine security policy for agents	1
Determine system operation	1
Gather performance knowledge	1
Identify emergent behaviour	1
Identify system behaviours	7
Identify system organization	1
Define organizational rules	6
Define organizational structures	6
Determine agents' organizational behaviours	7
Determine agents' organizational roles	7
Identify sub-organizations	6
Model actors	3
Model agent knowledge	8
Model agent relationships	8
Model agents' roles	1
Model responsibilities	6
Model permissions	6
Model capabilities for actors	3
Model dependencies for actors and goals	3
Model goals	3
Model plans	3
Model the agent's environment	1
Model environmental resources	6
Model events	4
Model percepts	4
Specify shared data objects	4
Undertake agent personalization	1
Subtask to Create a System Architecture:	
Determine MAS infrastructure facilities	8, 9

Table 1. (Continued)

(b) New Techniques	Ref	New Techniques	Ref
Activity scheduling	1	Environmental evaluation	2
Agent delegation strategies	1	Environmental resources modelling	6
Agent internal design	4, 5	FIPA KIF compliant language	2
AND/OR decomposition	3	Learning strategies for agents	1
Belief revision of agents	1	Market mechanisms	1
Capabilities identification & analysis	3	Means-end analysis	3
		Organizational rules specification	6
Commitment management	1	Organizational structure specification	6
Contract nets	1	Performance evaluation	1
Contributions analysis	3	Reactive reasoning: ECA rules	1
Control architecture	1	Task selection by agents	1
Deliberative reasoning: Plans	1	3-layer BDI model	2

(c) New Work Products	Ref	New Work Products	Ref
Agent acquaintance diagram	4, 6	Network design model	8
Agent class card	8	Platform design model	8
Agent design model	8	Protocol schema	4, 6
Agent overview diagram	4	PSM specification	8
Agent structure diagram	4	Role diagram	5
CAMLE behaviour diagram	10	Role schema	6
CAMLE scenario diagram	10	Service table	6
Caste collaboration diagram Caste	10	Task hierarchy diagram	8
diagram	10	Task knowledge specification	8
Coupling Graph	7	Task textual description	8
Domain knowledge ontology	8	(Tropos) Actor Diagram	3
Functionality descriptor	4	(Tropos) Capability Diagram	3
Goal hierarchy diagram	5	(Tropos) Goal Diagram	3
Inference diagram	8	(Tropos) Plan Diagram	3

	Tasks					
Technique	1	2	3	4	5	6
Abstract class identification						
Agent internal design			Y			
AND/OR decomposition	Y					
Class naming	Y	Y				
Control architecture		Y				
Context modelling	Y			Y		
Delegation analysis	Y	Y				
Event modelling				Y		
Intelligent agent identification		Y				
Means-end analysis	Y					
Role modelling	Y	Y			Y	Y
State modelling		Y				
Textual analysis	Y	Y				
3-layer BDI model		Y	Y			

Key:
1. Model dependencies for actors and goals; 2. Construct the agent model;
3. Design agent internal structure; 4. Model the agent's environment;
5. Model responsibilities; 6. Model permissions

Fig. 7. A small portion of the matrix linking Tasks and Techniques for the extended Prometheus case study described in detail in Henderson-Sellers (2005)

identified for the pre-selected (at a previous stage) Tasks. Linkage decisions (here just binary) are made either subjectively/experientially or by means of an overall assessment of a number of factors relating to the project. These factors include CMM level, specific skills in the workforce, domain of the project etc. Note that, even if a candidate is chosen, there is no danger in over-selection since, for an unnecessary Technique, the completed deontic matrix will simply exhibit a blank line (as for Technique: Abstract class identification in this small example – first line in Figure 7).

A next stage of the project is to critically analyze each of these proposed method fragments to see if they are really unique, to ensure there are no overlaps and to ensure compatibility with non-AO method fragments already in the OPEN repository.

Overall, the strengths of this SME approach are that the finally constructed methodology is highly attuned to local conditions and the people in the organization. The challenges are to construct the several deontic matrices, ensuring that (a) linkages accord to the local situation and (b) that the interfaces of any pair of method fragments to be "plugged together" are compatible. Both of these can be facilitated by the use of software tools, the former with a process construction tool (see, e.g., Nguyen and Henderson-Sellers, 2003), the latter with a database-supported evaluation tool (McBride, 2004), both of which we have prototyped.

5 Summary

To date, the evolution of AO methodologies has been disparate with many groups worldwide creating individual offerings. These vary in style and, particularly, in heritage and have a specific focus, either in terms of domain, application style or lifecycle coverage. For industry adoption, it is essential that full lifecycle coverage is achieved in a "standardized" way. One way of achieving some degree of standardization yet maintaining full flexibility is through the use of situational method engineering (SME). With this approach, method fragments are created and stored in a repository or methodbase. For an individual application, only a subset of these is then selected from the repository and a project-specific (or sometimes organization-specific) methodology is constructed. Here, we have demonstrated how this might work by using the OPEN approach that already provides a significant coverage of AO method fragments as well as more traditional OO and pre-OO fragments. Those newer fragments supporting AO approaches are summarized here, describing as they do emerging substantial support for AO methodological creation from SME and the OPEN repository. Further work is needed to consolidate the AO contributions to this repository, to check for inter-fragment consistency and to create a full suite of construction guidelines specific for the creation of AO methodologies suitable for industrial use.

Acknowledgements

I wish to thank Dr Cesar Gonzalez-Perez for his useful comments on an earlier draft of this manuscript. This is Contribution number 04/28 of the Centre for Object Technology Applications and Research.

References

Avison, D. and Fitzgerald, G., 2003, Where now for development methodologies, *Comm. ACM,* **46(1)**, 79-82

Bernon, C., Gleizes, M.-P., Picard, G. and Glize, P., 2002, The ADELFE methodology for an intranet system design, *Agent-Oriented Information Systems 2002. Procs.AOIS-2002* (eds. P. Giorgini, Y. Lespérance, G. Wagner and E. Yu), 1-15

Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopolous, J. and Perini, A., 2004, Tropos: an agent-oriented software development methodology*, Autonomous Agents and Multi-Agent Systems,* **8(3),** 203-236

Brinkkemper, S., 1996, Method engineering: engineering of information systems development methods and tools, *Inf. Software Technol.*, **38(4)**, 275-280

Brinkkemper, S., Saeki, M. and Harmsen, F., 1998, Assembly techniques for method engineering, *Procs. CAISE 1998*, Springer Verlag, Berlin, Germany, 381-400.

Burrafato, P. and Cossentino, M., 2002, Designing a multi-agent solution for a bookstore with the PASSI methodology, **in** *Procs. Agent-Oriented Information Systems 2002* (eds. P. Giorgini, Y. Lespérance, G. Wagner and E. Yu), 102-118

Caire, G., Coulier, W., Garijo, F., Gomez, J., Pavon, J., Leal, F., Chainho, P., Kearney, P., Stark, J., Evans, R., Massonet, P., 2001, Agent oriented analysis using MESSAGE/UML, *Agent-Oriented Software Engineering II* (eds. M. Wooldridge, G. Wei and P. Ciancarini), LNCS 2222, Springer Verlag, Berlin, Germany, 119-135

Castro J., Kolp M. and Mylopoulos J., 2002, Towards requirements-driven information systems engineering: the Tropos project, *Information Systems*, **27(6),** 365-389

Cernuzzi, L. and Rossi, G., 2002, On the evaluation of agent oriented methodologies, *Procs. OOPSLA 2002 Workshop on Agent-Oriented* Methodologies, Centre for Object Technology Applications and Research, Sydney, 21-30

Coleman, D., Arnold, P., Bodoff, S., Dollin, C. and Gilchrist, H., 1994, Object-Oriented Development. The Fusion Method, Prentice Hall, Englewood Cliffs, NJ, USA, 313pp

Collier, R., et al., 2003, Beyond prototyping in the factory of agents, in: *Multi-Agent Systems and Applications III,* LNCS 2691, V. Marik, J. Muller and M. Pechoucek, eds., Springer-Verlag, New York, pp. 383-393.

Collier, R., O'Hare, G. and Rooney, C., 2004, A UML-based software engineering methodology for Agent Factory, *Procs. SEKE 2004* (in press).

Collinot, A. Drogoul, A. and Benhamou, P. 1996. Agent oriented design of a soccer robot team. *Procs. Second Intl. Conf. on Multi-Agent Systems (ICMAS'96)*

Collinot, A. and Drogoul, A. 1998. Using the Cassiopeia Method to Design a Soccer Robot Team. *Applied Articial Intelligence (AAI) Journal*, 12, 2-3, 127-147.

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

Dam, K.H. and Winikoff, M., 2004, Comparing agent-oriented methodologies, *Agent-Oriented Systems* (eds. P. Giorgini, B. Henderson-Sellers and M. Winikoff), LNAI 3030, Springer-Verlag, Berlin, 78-93

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

DeLoach, S.A. 1999. Multiagent Systems Engineering: A Methodology and Language for Designing Agent Systems, *Procs AOIS '99.*

Firesmith, D.G. and Henderson-Sellers, B., 2002, *The OPEN Process Framework*, Addison Wesley, Harlow, UK.

Fitzgerald, B., Russo, N.L. and O'Kane, T., 2003, Software development method tailoring at Motorola, *Comm. ACM,* **46(4)**, 65-70.

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, *Procs. IIP*, Beijing, 21-23 October 2004

Graham, I., Henderson-Sellers, B. and Younessi, H., 1997, *The OPEN Process Specification*, Addison-Wesley.

Henderson-Sellers, B., 1995, Who needs an OO methodology anyway?, *J. Obj.-Oriented Programming*, **8(6),** 6-8

Henderson-Sellers, B., 2003, Method engineering for OO system development, *Comm. ACM***, 46(10),** 73-78

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 Inc., Hershey, PA, USA

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

Henderson-Sellers, B., Simons, A.J.H. and Younessi, H., 1998, *The OPEN Toolbox of Techniques*, Addison-Wesley, UK, 426pp + CD

Henderson-Sellers, B., Giorgini, P. and Bresciani, P., 2003, Evaluating the potential for integrating the OPEN and Tropos metamodels, *Procs. SERP '03* (eds. B. Al-Ani, H.R. Arabnia and Y. Mun), CSREA Press, Las Vegas, USA, 992-995

Henderson-Sellers, B., Giorgini, P. and Bresciani, P., 2004a, Enhancing Agent OPEN with concepts used in the Tropos methodology, *Engineering Societies in the Agents World IV. 4th International Workshop, ESAW' 2003* (eds. A. Omicini, P. Pettra and J. Pitt), LNAI 3071, Springer-Verlag, Berlin, Germany, 328-345

Henderson-Sellers, B., Debenham, J. and Tran, Q.-N.N., 2004b, Adding agent-oriented concepts derived from GAIA to Agent OPEN, *Advanced Information Systems Engineering. 16th International Conference, CAiSE 2004, Riga, Latvia, June 2004 Proceedings* (eds. A. Persson and J. Stirna), LNCS 3084, Springer-Verlag, Berlin, 98-111

Henderson-Sellers, B., Tran, Q.-N.N. and Debenham, J., 2004c, Incorporating elements from the Prometheus agent-oriented methodology in the OPEN Process Framework, *Procs. AOIS@CAiSE2004*, Faculty of Computer Science and Information, Riga Technical University, Latvia, 370-385

Henderson-Sellers, B., Tran, Q.-N.N. and Debenham, J., 2004d, Method engineering, the OPEN Process Framework and Cassiopeia, *Procs. Symposium on Professional Practice in AI,* Toulouse, France, August 22-27 2004, Kluwer

Henderson-Sellers, B., Tran, Q.-N.N., Debenham, J. and Gonzalez-Perez, C., 2004e, Agent-oriented information systems development using OPEN and the Agent Factory, *Procs. ISD 2004,* Vilnius, 9-11 September 2004, Kluwer

Huget, M.-Ph., 2002, Nemo: an agent-oriented software engineering methodology, **in** *Procs. OOPSLA 2002 Workshop on Agent-Oriented Methodologies*, Centre for Object Technology Applications and Research, Sydney, Australia, 43-53

Iglesias, C.A., Garijo, M., Gonzalez, J.C., Velasco, J.R. 1996. A methodological proposal for multiagent systems development extending commonkads. In *Proc. of 10th KAW*, Banff, Canada

Iglesias, C.A., Garijo, M., Gonzalez, J.C., Velasco, J.R. 1998. Analysis and Design of Multi-Agent Systems using MAS-CommonKADS. In *Intelligent Agents IV: Agent Theories, Architectures, and Languages* (LNAI Volume 1365) (eds. M.P. Singh, A. Rao and M.J. Wooldridge), Springer-Verlag: Berlin, Germany.

Kendall, E.A. and Zhao, L., 1998, Capturing and Structuring Goals, *Workshop on Use Case Patterns, Object Oriented Programming Systems Languages and Architectures*.

Kendall, E.A., Malkoun, M.T. and Jiang, C., 1996, A methodology for developing agent based systems for enterprise integration, **in** *Modelling and Methodologies for Enterprise Integration* (eds. P. Bernus and L. Nemes), Chapman and Hall

Kinny, D., Georgeff, M. and Rao, A., 1996, A methodology and modelling techniques for systems of BDI agents, Technical Note 58, Australian Artificial Intelligence Institute, also published *in Agents Breaking Away: Procs. 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW'96)*, 56-71

Klooster, M., Brinkkemper, S., Harmsen, F. and Wijers, G., 1997, Intranet facilitated knowledge management: A theory and tool for defining situational methods. *Procs. CAISE 1997*, Springer Verlag, Berlin, Germany, 303-317

Kruchten, Ph., 1999, *The Rational Unified Process. An Introduction*, Addison-Wesley, Reading, MA, USA

Kumar, K. and Welke, R.J., 1992, Method engineering: a proposal for situation-specific methodology construction, **in** *Systems Analysis and Design: A Research Agenda,* (eds. W.W. Cotterman and J.A. Senn), John Wiley and Sons, New York, NY, USA, 257-269

Lind, J. 1999. *Iterative Software Engineering for Multiagent Systems. The MASSIVE Method*, LNAI 1994, Springer-Verlag, Berlin

Luck M., Ashri, R. and D'Inverno, M., 2004, *Agent-Based Software Development*, Artech House, Boston, 208pp

McBride, T., 2004, Standards need more rigour, *Information Age*, **Oct/Nov 2004**, 65-66

Nguyen, V.P. and Henderson-Sellers, B., 2003, OPENPC: a tool to automate aspects of method engineering, *Procs. ICSSEA 2003*. Paris, France, **Volume 5,** 7pp

Odell, J., Van Dyke Parunak, H. and Bauer, B., 2000, Extending UML for agents. In G. Wagner, Y. Lesperance and E. Yu (eds.), *Procs. Agent-Oriented Information Systems Workshop,* 17th National Conference on Artificial Intelligence (pp. 3-17). Austin, TX, USA.

Omicini, A., 2000, SODA: Societies and Infrastructures in the analysis and design of agent-based systems, *Procs. First Int. Workshop on Agent-Oriented Software Engineering*

Padgham, L. and Winikoff, M., 2002a, Prometheus: A Methodology for Developing Intelligent Agents. *Procs. Third International Workshop on Agent-Oriented Software Engineering*, at AAMAS'02.

Padgham, L. and Winikoff, M., 2002b, Prometheus: A Pragmatic Methodology for Engineering Intelligent Agents. **In** *Procs.Workshop on Agent-oriented Methodologies at OOPSLA 2002*, November 4, 2002, Seattle.

Pavón, J., Gomez-Sanz, J. and Fuentes, R., 2005, The INGENIAS methodology and tools, Chapter 4, *Agent-Oriented* Methodologies (eds. B. Henderson-Sellers and P. Giorgini), Idea Group Inc., Hershey, PA, USA

Perini A., Bresciani P., Giorgini P., Giunchiglia G. and Mylopoulos J., 2001, A knowledge level software engineering methodology for agent oriented programming, In J.~P. Müller, E. Andre, S. Sen, and C. Frasson, editors, *Proceedings of the Fifth International Conference on Autonomous Agents*, May 2001, Montreal, Canada

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* (eds. O. Vasilecas, A. Caplinskas, W. Wojtkowski, W.G. Wojtkowski, J. Zupancic and S. Wrycza), Vilnius Gediminas Technical University, Vilnius, Lithuania, 271-282

Ralyté, J. and Rolland, C., 2001, An assembly process model for method engineering, *Advanced Information Systems Engineering*), LNCS2068, Springer-Verlag, Berlin, 267-283

Ralyté, J., Deneckère, R and Rolland, C., 2003, Towards a generic model for situational method engineering, *CAiSE2003* (ed. M.M.J. Eder), LNCS 2681, Springer-Verlag, Berlin, 95-110

Ralyté, J., Rolland, C. and Deneckère, R., 2004, Towards a meta-tool for change-centric method engineering: a typology of generic operators, *CAiSE2004* (eds. A. Persson and J. Stirna), LNCS 3084, Springer-Verlag, Berlin, 202-218

Rao, A.S. and Georgeff, M.P., 1995, BDI agents: from theory to practice. In *Procs. First International Conference on Multi-Agent Systems,* San Francisco, CA, USA, 312-319

Rolland, C. and Prakash, N., 1996, A proposal for context-specific method engineering, *Procs. IFIP WG8.1 Conf. on Method Engineering,* Chapman and Hall, 191-208

Rolland, C., Prakash, N. and Benjamen, A., 1999, A multi-model view of process modelling, *Requirements Eng. J.*, **4(4),** 169-187

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. and Lorensen, W., 1991, *Object-Oriented Modeling and Design*, Prentice-Hall, Englewood Cliffs, NJ, USA

Saeki, M., 2003, CAME: the first step to automated software engineering, *Process Engineering for Object-Oriented and Component-Based Development. Procs. OOPSLA 2003 Workshop*, Centre for Object Technology Applications and Research, Sydney, Australia, 7-18

Schreiber, A. Th. Wielinga, B.J., de Hoog, R. Akkermans, J.M and Van de Velde, W., 1994. CommonKADS: A comprehensive methodology for KBS development. *IEEE Expert,* 9(6): 28-37

Shan, L. and H. Zhu, 2004. *CAMLE: A Caste-Centric Agent-Oriented Modeling Language and Environment*. In *Third International Workshop on Software Engineering for Large-Scale Multi-Agent Systems*. Edinburgh, 24-25 May 2004. [in press]. Springer-Verlag.

Silva, V. and Lucena, C., 2004, From a conceptual framework for agents and objects to a multi-agent system modeling language, *Autonomous Agents and Multi-Agent Systems,* **9(1-2),** 145-189

Sturm, A. and Shehory, O., 2004, A framework for evaluating agent-oriented methodologies, *Agent-Oriented Systems* (eds. P. Giorgini, B. Henderson-Sellers and M. Winikoff), LNAI 3030, Springer-Verlag, Berlin, 94-109

Ter Hofstede, A.H.M. and Verhoef, T.F., 1997, On the feasibility of situational method engineering, *Information Systems*, **22**, 401-422

Tran, Q.-N.N. and Low, G.C., 2005, Comparison of methodologies, Chapter 12 **in** *Agent-Oriented Methodologies* (eds. B. Henderson-Sellers and P. Giorgini), Idea Group Inc., Hershey, PA, USA

Tran, Q.N., Low, G. and Williams, M.A., 2003, A feature analysis framework for evaluating multi-agent system development methodologies, **in**. *Foundations of Intelligent Systems – Procs. 14th Int. Symposium on Methodologies for Intelligent Systems ISMIS'03* (eds. N. Zhong, Z.W. Ras, S. Tsumoto and E. Suzuki), 613-617.

Tran, Q.-N.N., Henderson-Sellers, B. and Debenham, J. 2004a, Incorporating the elements of the MASE methodology into Agent OPEN, *Procs. ICEIS2004 - Sixth International Conference on Enterprise Information Systems* (eds. I. Seruca, J. Cordeiro, S. Hammoudi and J. Filipe), INSTICC Press, **Volume 4**, 380-388

Tran, Q.-N.N., Low, G. and Williams, M.-A., 2004b, A preliminary comparative feature analysis of multi-agent systems development methodologies, *Procs. AOIS@CAiSE*04*, Faculty of Computer Science and Information, Riga Technical University, Latvia, 386-398

Tran, Q.-N.N., Henderson-Sellers, B., Debenham, J. and Gonzalez-Perez, C., 2004c, MAS-CommonKADS and the OPEN method engineering approach, submitted for publication

van Slooten, K. and Hodes, B., 1996, Characterizing IS development projects, **in** *Proceedings of the IFIP TC8 Working Conference on Method Engineering: Principles of method construction and tool support* (eds. S. Brinkkemper, K. Lyytinen, R. Welke) Chapman&Hall, Great Britain, 29-44

Wagner, G., 2003, The Agent-Object Relationship metamodel: towards a unified view of state and behaviour, *Inf. Systems*, **28(5)**, 475-504

Wagner, G., 2004, AOR modelling and simulation: towards a general architecture for agent-based discrete event simulation, *Agent-Oriented Information Systems* (eds. P. Giorgini, B. Henderson-Sellers and M. Winikoff), LNAI 3030, Springer-Verlag, Berlin, 174-188

Wagner, G. and Taveter, K., 2005, Towards radical agent-oriented software engineering processes based on AOR modelling, Chapter 10 in *Agent-Oriented Methodologies* (eds. B. Henderson-Sellers and P. Giorgini), Idea Group Inc., Hershey, PA, USA

Wistrand, K. and Karlsson, F., 2004, Method components – rationale revealed, *CAiSE2004* (eds. A. Persson and J. Stirna), LNCS 3084, Springer-Verlag, Berlin, 189-201

Wood, M. and DeLoach, S.A. 2000, An Overview of the Multiagent Systems Engineering Methodology. *Procs. 1st International Workshop on Agent-Oriented Software Engineering (AOSE-2000)*, 207-222

Wooldridge, M., Jennings, N.R. and Kinny, D., 2000, The Gaia methodology for agent-oriented analysis and design, *J. Autonomous Agents and Multi-Agent Systems*, **3,** 285-312.

Yu, E., 1995, *Modelling Strategic Relationships for Process Reengineering*, PhD, University of Toronto, Department of Computer Science

Zambonelli, F., Jennings, N. and Wooldridge, M., 2003, Developing multiagent systems: the Gaia methodology, *ACM Transaction on Software Engineering and Methodology*, **12(3),** 317-370