AI & Soc (2009) 23:379–407
DOI 10.1007/s00146-007-0110-9

ORIGINAL ARTICLE

Methodologies for agent systems development: underlying assumptions and implications for design

Panayiotis Koutsabasis · John Darzentas

Received: 14 April 2006 / Accepted: 23 March 2007 / Published online: 8 May 2007
© Springer-Verlag London Limited 2007

Abstract The area of agent systems design may be safely described as cluttered and disorganized, especially by those that situate themselves outside the ''agent community''. Despite the wealth of bibliography on agent systems design and applications, there are few widely acknowledged design methods that have surfaced from testing and practice, mainly in laboratory settings. The paper contributes to the understanding of the field by presenting a critical review of methodologies that have emerged over the last few years to guide and explain agent systems design and development. The perspective for this review has been mainly formulated by posing important research questions in the field, and by attempting to interpret and discover latent hypotheses and underlying assumptions made by methodologies in reference to relevant research, both in agent systems and cooperative information systems practice and theory. The paper identifies significant challenges for agent systems methodologies that, if pursued, can contribute to a new understanding of the field that shifts the foci of current agent systems research, towards holistic design methods that place human users and information systems stakeholders at the centre of interest and involve them in the design process as much as possible.

Introduction

The bibliography of agent systems design and applications is quite rich, including cross-disciplinary work in terms of numerous areas of concern such as theoretical

P. Koutsabasis (✉) · J. Darzentas
Department of Product and Systems Design Engineering, University of the Aegean, Hermoupolis 84100, Syros, Greece
e-mail: kgp@aegean.gr

J. Darzentas
e-mail: idarz@aegean.gr

background, design methods and architectures, domains of application, problems addressed and technical implementations. However, it is often the case that multiple approaches to agent systems design address similar situations/problems in quite different ways, which in addition may not be safely evaluated and weighted against one another. For many, the concepts of 'autonomous agents' and 'multi-agent systems' have come to be realised as a huge 'melting-pot' of ideas, concerns and specifications regarding cooperative information systems design ranging from 'intelligence' and 'social ability' to 'autonomy' and 'distributed processing'.

In order to make appropriate use of the large amount of work in the field, it is useful to step back and take an external and holistic look at its current status. This will enable a critical look at the directions of the field and allow for gaining an improved understanding of the potential of agent systems approaches to prove responsive to human and social needs. In pursuit of this critical look into agent systems design methods, the paper starts out by posing key questions and discussing important work that has surfaced in scientific literature in terms of, among others, agent definition/identification, architectural design, and coordination/interaction. The main result that can be made out of this discussion comes to support the starting point of this paper, in that almost any attempt to provide an 'answer' may not be evaluated and compared to one-another, which creates apparent obstacles for designers and practitioners. A way to address this polyphony is to focus on selected relevant areas of research, in pursuit of interpretations and explanations about the state of the art, which might not suffice to be generalised further than these areas, but can certainly contribute to a better understanding of the field.

Perhaps the most appropriate area of research in this respect is that of methodologies for agent systems design (agent methodologies). Up to now, there seems to be at least 30 methodologies, not all equally developed or known that attempt to guide agent systems design and development. The appropriateness of selecting this area of research to seek interpretations and explanations for agent systems design may be summarised in that ''in principle, a methodology should include and define all important concepts from the area of concern that are required for providing meaning and structure to the problem situation of interest'' (Checkland and Scholes 1998). In principle, methodologies are not simplistic collections of methods or techniques but are penetrated by an underlying philosophy of thought that needs to be questioned, interpreted, understood and adopted in order to be applied in practice. In particular agent methodologies are not separated from the large body of research on autonomous agents and multi-agent systems but on the contrary they attempt to purposefully synthesize the experiences gained from multifarious work on agent systems design and development to a form that may be easily accessed and used by practitioners.

The paper presents a critical review of methodologies that have emerged over the last few years to guide and explain agent systems design and development. The perspective for this review has been mainly formulated by posing important research questions in the field, such as those related to the suitability of the agent approach to particular problems, the design of the architecture of agent systems and the requirements for agent systems coordination. The paper attempts to interpret and discover the latent hypotheses and underlying assumptions made by methodologies

in reference to relevant research, both in agent systems and cooperative information systems practice and theory. Although there is some work in this respect, for example Jennings (2001) and Yu (2001), this work has not been explicitly related to methodologies for agent systems development. The paper concludes by identifying significant challenges for agent systems methodologies including the realization of the conceptual foundations of methodologies, the identification and inclusion of other methods from related fields and the situation of agent systems design to the wider context of information systems development.

The potential contributions of this work are twofold. On one hand, it enables researchers in the field of agent systems to assess the progress of related methodologies and consider their adoption to their work, which is required for many reasons in the field: much of the pioneering work in the field has an inter-disciplinary background and at large has been developed intuitively, making hard its uptake. On the other hand, it assists information systems designers to consider new ways for the design of cooperative information systems in terms of concepts that have been developed and/or refined in terms of agent systems research and have been recently formulated in terms of methodologies.

Research questions

Every area of research attempts to address some fundamental questions about the nature, the assumptions and the pragmatics of methods, methodologies and practices that are devised and employed to address particular problems. It seems that the large majority of the work in the field incorporates views about important questions, and provides answers to these, either purposefully or unconsciously. These questions cannot be addressed in isolation to one-another; instead they are inseparable concerns for practitioners and need to be addressed by methodologies that guide agent systems design.

Why consider an agent-based approach to cooperative information systems design?

One of the major elements of critique on agent systems may be summarized as only rarely these are based on studies, analogies and metaphors from the physical world. For example, Treur (1999) remarks that ‘most existing multi-agent systems were developed in an ad-hoc manner’; while Weiss (2003a, b) in an investigation of patterns for motivating an agent-based approach notes that ‘the advantages of the agent-based approach are still not widely recognized outside the agent community’. Instead, it is most usually taken for granted that agent systems have to be defined by the designer, and not identified with the participation of other stakeholders, for example in the context of a particular organisation, or human activity system. Furthermore, design decisions about the appropriateness of the agent-based approach are not well-justified by only referring to ambiguous, specification-like attributes of the desired system that are usually drawn on the basis of agent systems definitions. The problem has been identified by related work, such as that referring

to the investigation of patterns, pattern catalogues and pattern languages for justifying/motivating an agent-based approach (Gonzalez-Palacios and Luck 2005; Weiss 2003a, b; Lind 2002), and studies on the identification of agents in specific application domains (e.g. Galland et al. 2003; Bussmann et al. 2001). However, these approaches have not been clearly connected to user requirements but to general guidelines / heuristics, they are largely empirical and they are often drawn from particular domain requirements, with the consequence that they may not be safely generalized to other types of problems.

What should be the architecture of an agent-based system?

Numerous architectures have been proposed and implemented for agent-based systems. This work has provided an essential understanding of the basic concepts and ideas. However, for a long time, it has suffered from the absence of a general theory that can be applied to this work. For example, as Sloman (1996) observes, the task of ''putting agent architectures to a classification scheme lacks definition partly because we have no general theory of types of architectures, and partly because we lack clear and unambiguous specifications of what is to be explained or modeled''. Indeed, the understanding of what agent architecture should be is quite different depending on the background of research in the field. For example, 'symbolic AI' researchers offer conceptually rich, with built-in intelligence, agent architectures at one extreme, while the proponents of 'connectionist AI' design simple architectures by proposing that autonomy and intelligence will arise from the interaction of simple elements, at the other extreme. These have obvious consequences for practitioners who may not easily identify the underlying assumptions of this work and usually develop their own architecture on unclear ground, which further adds confusion and disorder in the field.

What are the requirements of agent systems coordination and interaction?

Agent systems coordination and interaction is considered a research area of its own with work ranging from coordination theory to agent communication languages and technological platforms for multi-agent systems communication. Agent systems coordination refers to the informational content and organisational structure of a multi-agent system and the rules and actions by which this content and structure may 'change' or evolve during the lifetime of the system. Currently there is much debate in the field and approaches range from the discussion of its theoretical foundations, such as those of Gasser (1991) referring to principles underlying action and knowledge for DAI systems; Jennings (1993) referring to commitments and conventions as the foundation of coordination in AI; Malone and Crowston (1994) referring to the interdisciplinarity of coordination; and Castelfranchi (1998) referring to modelling social action for AI agents; to coordination protocols, such as the long dated but still in use contract net protocol (Smith and Davis 1981); partial global planning (Durfee and Lesser 1991); and the synthesis of social laws for artificial agent societies (Shoham and Tennenholz 1992). Most of these approaches still fall short of the development practice since they discuss concepts

and techniques that are not met in mainstream communication technologies. As a result, agent systems are usually designed poorly with respect to coordination and interaction requirements, limiting the potential to demonstrate autonomous and intelligent behaviour.

The work on agent methodologies provides advances on these questions in various ways ranging from the synthesis of related work to the definition of a methodology, to quite new proposals previously unseen in related literature. However, their documentation does not relate the proposed approaches to these questions and therefore one has to carefully study and interpret them with respect to these wide-ranging concerns. During this interpretation task, wider methodological issues emerge and these need to be discussed further. In this respect we further discuss the scope of agent systems methodologies in terms of the types of problems addressed and the phases of the lifecycle, and their conceptual foundations and assumptions in terms of related work in information systems theory and practice.

Why methodologies?

Over the last few years, it has been widely acknowledged that agent systems design and development has been largely ad-hoc. Luck et al. (2003) notes that ''while there are currently object-oriented development methodologies, not such routes exist for agent-oriented systems, which much either use unsuitable or ad hoc methods'' Work on methodologies for agent systems design and development has been quite extensive and is developing in a number of ways. There are many methodologies that directly address agent systems design, while new methodologies continuously appear in literature. Table 1 lists selected methodologies for agent systems design and development, outlining their background and main phases, with main criterion of the soundness of their documentation. Further to this list of methodologies, there are also other works that do not seem to have continued their development up to now in terms of either theoretical work or application, for example, such as those of Kendall et al. (1995), Burmeister (1996), and Iglesias et al. (1996); and methodologies that focus on particular application domains (e.g. Galland et al. 2003). Even from this minimal presentation of methodologies, it is evident that these works are not equally well-developed or known, they differ in a number of aspects such as background, objectives, scope, steps/phases, models, methods (many methodologies do not seem to have specific methods), tools (conceptual, diagrammatic, technical), application to real problems, and others.

There are a few papers that review to some extent this work. Wooldridge and Ciancarini (2001) and Iglesias et al. (1999) summarize the main characteristics of methodologies and mainly classify them as extending object-oriented and knowledge-based approaches. Gómez-Sanz and Pavón (2004) present a review of some methodologies for agent systems development by examining how they support specific agent-related concepts. Furthermore, a few papers perform attribute-based comparisons of methodologies and show that there is still work to be done with regard to identifying their pros and cons in this respect (Sturm and Shehory 2003;

Dam et al. 2003). There are also works that try to assemble a methodology from features of other methodologies, in an attempt to identify complementary and similar methods and techniques (Juan et al. 2002).

Table 1 Outline of background/main objectives and main phases of agent systems design methodologies

	Authors	Background/main objectives	Main phases
Tropos	Bresciani et al. (2004)	Requirements engineering; based on organizational concepts (rather than programming)	Early requirements, late requirements, architectural design, detailed design, implementation
ODAC (Open distributed applications construction)	Gervais (2003)	To provide a set of methods and tools based on ISO ODP (Open distributed processing) that allow control of the complexity of constructing of agent-based systems	System construction, deployment, use and withdrawal
REF: Agent based requirements engineering framework	Bresciani and Donzelli (2003)	Requirements engineering; to allow non-technical stakeholders to elicitate requirements by reducing complexity; to provide a comprehensive requirements analysis methodology	Organisation modelling; hard-goal modelling; soft-goal modelling
ADELFE (Toolkit for designing software with emergent functionalities)	Bernon et al. (2002)	To cover all phases of a classical software design from requirements to deployment	Late requirements, analysis, design
AOSE methodology for information gathering	Zhang et al. (2002)	To develop an agent-based system in a systematic way based on role models	Agent analysis; agent design
Extended OPEN Process framework	Debenham and Henderson-Sellers (2002)	To extend the OPEN Process Framework (OPF, a componentized O–O development methodology underpinned by a full meta-model) so that it supports agent-oriented information systems	N/A (tasks and techniques were defined and added to the OPF)
MESSAGE/UML	Caire et al. (2002)	To cover MAS analysis and design and for use in mainstream software engineering departments	Level 0 analysis; Level 1 analysis
Prometheus	Padgham and Winikoff (2002)	To provide a detailed and complete process with associated deliverables which can be taught to practitioners that do not have a background in agents	System specification; architectural design; detailed design

Table 1 continued

	Authors	Background/main objectives	Main phases
Tool-supported process analysis and design of MAS	Knublauch and Rose (2002)	Extreme programming; to provide a tool-supported methodology that allows the capturing of agentified processes in a format that is simple to be understood and maintained by domain experts	A process model is iteratively evolved into a multi-agent system design
Multiagent systems engineering (MaSE)	DeLoach et al. (2001)	To provide a further abstraction of the O–O paradigm where agents are a specialization of objects	Analysis; design
A semiotic approach based on roles and norms	Chong and Liu (2000)	Semiotics; To establish a social model of the organisation, including agents, roles, patterns of behaviour and ontological dependencies	Semantic analysis; norm analysis
Gaia	Wooldridge et al. (2000)	For analysis, to develop an understanding of the system and its structure. For design, to transform the analysis models into a sufficiently low level of abstraction that traditional design techniques may be applied to implement agents	Analysis, design
High-level and intermediate models	Elammari and Lalonde (1999)	To provide a systematic approach for generating from high-level designs implementable system definitions	Agent discovery; agent definition
Agent oriented and role based workflow modelling	Yu and Schmid (1999)	To model processes in a dynamic social organisation settings, where the business process are made up of social actors who have goals and interests, which they pursue through a network of relationships with other actors	Role-based analysis; agent-oriented design; agent-oriented implementation
An agent-oriented design methodology	Kinny and Georgeff (1996, 1997)	To develop techniques for modelling agents and multi-agent systems which adapt and extend existing O–O techniques, and a methodology which guides system design and specification	Identification of roles; role responsibilities and services; service interactions, performatives, and information content; refinement of the agent hierarchy

This focus of this review is quite different from the aforementioned work since it does not attempt to compare or classify the methodologies in terms of attributes or features. Instead, it accommodates the best practices in terms of agent systems research in order to reveal an understanding about the 'views' of methodologies on critical questions for the field. In this respect, it contributes to the identification of the conceptual foundations of methodologies in terms of both agent systems and cooperative information systems theory and practice. This focus can enable deeper

comprehension and wider uptake of this work both inside and outside the agent community by contributing to the evaluation of the current situation in the field and identifying future areas of research and development. This evaluation can contribute to answering the 'why' question in agent systems design, i.e. why conceptualise/design/deploy/etc. systems constituted by agents. Despite this question might be easy to answer from agent community practitioners, it is true that many people outside this community still do not seem to be convinced for the need of agents in the development of information systems.

A review on methodology status and development inevitably deals with expected obstacles. Analysis of features may be hampered by terminology with the consequence that similar phenomena may be described in different terms (or vice versa). In addition, there are often difficulties to identify the relevance and usefulness of incorporating related work in the structured approach of a methodology. Finally, key methodological issues, such as the conceptual foundations and scope of the methodology are not clearly set out which may result in misinterpretations of the basics of the methodologies.

On the other hand, the interpretation of a methodology is a critical step for its engagement and use, and this is inevitably done by practitioners, consciously or not. Therefore, it is important that agent systems methodologies are reviewed in order to identify fundamental conceptual and practical issues that are still not adequately addressed.

Identification/definition of agent systems

Agent systems identification or definition is discussed in different depths and from different perspectives within the documentation of methodologies. We identify this work as based on domain models, role models and as deriving from functional specifications. Table 2 presents the situation of methodologies in terms of this schema, which is discussed throughout this section. In addition to this work there are also a few other views, which promote the iterative and participatory identification/definition of agent systems, that is implied by the tool supported process methodology (Knublauch and Rose 2002) that exploits extreme programming; as well as approaches that pre-suppose that certain types of agents exist in information systems, such as the agent oriented and role based workflow modelling methodology (Yu and Schmid 1999). The former approach implicitly suggests that the identification or definition of agents will arise from the iterative design and development process, while the latter attempts to ''get away'' from the agent identification/definition problem by providing some general 'types' of autonomous agents, leaving again the designer to decide whether these will be appropriate for any type of development.

Domain models

Some methodologies provide methods for identifying and representing the basic notions or concepts of a problem situation that take into account the perspectives of

Table 2 Approaches for definition/identification of agent systems

	Domain models	Role models	Functional specifications
REF: Agent based requirements engineering framework	√		
ADELFE (Toolkit for designing software with emergent functionalities)			√
AOSE methodology for information gathering		√	
Prometheus			√
Tropos	√		
Multiagent systems engineering (MaSE)		√	
A semiotic approach based on roles and norms	√		
Gaia		√	
High-level and intermediate models			√

involved stakeholders or actors of the domain. We refer to these methods and representations as models of the domain or domain models. In principle these models should be unique for each information system, since involved stakeholders are different people and they have different perspectives and requirements. Therefore it is considered that knowledge about these models needs to be elicited, rather than determined upon experience. The methodologies that most notably employ domain models in order to understand the problem situation and identify agents include Tropos (Bresciani et al. 2004; Castro et al. 2002; Giunchiglia et al. 2002), the REF methodology (Bresciani et al. 2003) and the semiotics approach (Chong and Liu 2000). Table 3 lists the definitions of the basic notions for domain models of each methodology.

The starting point of these methodologies is to understand the current situation, rather than modelling it technically on an empirical basis. These methods provide significant assistance to designers of agent systems with respect to the question of agent definition/identification. The domain models may be safely situated in the phase of analysis and user requirements, and are the starting point for design on the basis of the notions defined (however other notions may be added later). In terms of identifying agents as part of domain models, this identification may happen in collaboration of stakeholders of the domain and it is not simply a designer decision. Furthermore, the fact that the identification task involves interaction and knowledge sharing among project stakeholders, promotes system acceptance and reduces the possibilities for taking up inappropriate design decisions.

Role models

A common approach for agent definition/identification in many methodologies is to associate agents to roles that are identified or ''exist'' in a given information system. The rationale for this approach may be explored in role theory and conceptual modelling work that relates roles to other knowledge notions that constitute agent design, such as beliefs, goals, desires, plans, tasks, intentions, commitments, and others (e.g. Johnson and Johnson 1991). Furthermore, in areas where agent

Table 3 Basic concepts of domain models in agent methodologies

	Agent methodologies		
Agent concepts	Tropos	REF	Semiotics approach
Actors	Has strategic goals and intentionality and can include a physical agent, a software agent, a role or a position	N/A (Not applicable)	Human actors of a domain
Agents	The actors of the domain are identified as agents	The organisational context is modelled as a network of interacting agents (any kind of active entity)	Identified by common patterns of behaviours that characterise actors and roles.
Behaviour patterns	N/A	N/A	Not defined –they are identified by assigning meaning to behaviours of actors and roles
Beliefs	Representations of actors' knowledge of the world	N/A	N/A
Capability	The ability for an actor to define, choose and execute a goal	N/A	N/A
Constraints	N/A	They are associated with other notions of the organisation model to specify quality attributes.	N/A (conceptually similar to norms)
Dependency	(Between actors) Indicates that one actor depends on another to attain some goal, execute some plan or deliver a resource	Defined between agents and other notions e.g. an agent and a goal, when this agent wants this goal to be achieved.	N/A
Goals	Represents the strategic interest of actors. Goals are distinguished between hard and soft, i.e. the latter having no clear-cut definition and/or criteria for accomplishment	Goals model agent relationships and are distinguished between hard and soft in a similar way as in Tropos.	N/A
Norms	N/A	N/A (conceptually similar to constraints)	Norms establish what patterns of behaviour are acceptable for agents.
Plans	A way of satisfying a goal	N/A	N/A
Resource	A physical or informational entity that one actor wants and another can deliver	N/A	N/A
Roles	N/A	N/A	
Tasks	Not defined, seems to imply operational goals (i.e. not strategic)	Not defined, seems to imply operational goals (i.e. not strategic)	N/A (seems similar to behaviours)

identification is not straightforward it seems convenient for many practitioners to identify roles of human actors and organisational structures as agents.

The methodologies that most notably define role models are Gaia, which 'views a multi-agent system as a computational organisation consisting of various interacting roles' (Wooldridge et al. 2000); the Agent Oriented Software Engineering methodology for information gathering (Zhang et al. 2002) and MaSE (Deloach et al. 2001). Wooldridge et al. (2000) note ''it is natural to think of a computer system as being defined by a set of roles, if we adopt an organisational view of the world''. Furthermore, it is relatively easy, in some cases, to build scenarios of methodology application by assuming roles, for example in terms of the administrative structure of an organization, which is helpful for demonstrating the applicability of methodologies. The basic characteristics of the role models provided by methodologies are illustrated in Table 4.

These methodologies consider that roles exist in a given problem situation and that it is relatively straightforward to identify them, e.g. from the organisational chart of a company. However, the latter assumption for dealing with the agent definition/identification question is not contextually sensitive and requires substantial experience and intuition from the design team. For example the responsibilities of a 'secretary' role are quite different in different organizations. This type of approach to the identification of basic roles of the system of concern could be complemented with methods that emphasise the identification of user and domain requirements to achieve the identification of required knowledge in a context-specific manner.

Functional specifications

The identification/definition of agent systems is usually associated with functional specifications in the sense that these provide the basis for identifying agents, usually by applying a set of criteria such as software engineering principles. Functional specifications are usually modelled in UML (Unified Modelling Language, Booch et al. 2001) which is a widely deployed standard that provides multiple notations that can be used for functional specifications and detailed O–O design. The following methodologies emphasise the definition of agent systems on the basis of functional specifications: Prometheus (Padgham and Winikoff 2002), ADELFE

Table 4 Overview of the characteristics of selected methodologies regarding the identification of agent systems on the basis of the 'role' concept

	Gaia	AOSE for IG	MaSE
Definition of 'role'	Descriptive	Descriptive	Descriptive
Relation 'role' to other conceptual attributes of agent systems	Responsibilities, permissions, activities, protocols	Activities, responsibilities, resources	Tasks, protocols
Relation to other related work	O–O software engineering	O–O software engineering	O–O software engineering

(Bernon et al. 2002) and the high levels and intermediate models methodology (Elammari and Lalonde 1999).

Diagrammatic notations such as UML are useful for documenting functional specifications. However it is important that these methodologies are complemented with methods that promote requirements, capture, identify and discuss the notions that they define later in the software engineering process. These methodologies currently rely too much on design experience and intuition, which does not promote the take up of agent-based analysis and design by practitioners outside the agent community. Further to the work of these methodologies in terms of functional specifications there is also work that extends UML notations for specifying functional requirements including notions related to agents (Odell et al. 2000; Bauer 2001). This work contributes to the field by providing conceptual tools by which functional specification can include related concepts to agent-based analysis.

Architectural design

Architectural design is a fundamental and quite complex phase of agent systems design and development. In terms of methodology development, it is fundamental since that the application of the methodology should result in specifications for agents encompassing the issues that distinguish agents from other types of programs. It is also quite complex, since there is little agreement on many conceptual issues that are related to agent architecture development. Some of these issues include the single agent and the multi-agent aspects of architecture (also referred to the 'micro' and 'macro' aspects), the degree that the desired agent (behavioural and mental) attributes are built into the architecture or arise from interactions with the environment, and the effects on closely related issues such as other agent systems architectures, agent ontology and possibly agent technology that may support the design. In terms of agent systems methodologies work, agent architecture development is also addressed in quite different ways and level of details. The paper identifies the following basic strands of work:

- Methodologies that promote the process of architecture development, and typically result in the design of agent system architecture on the basis of the information system requirements:
- Methodologies that focus on the ontological aspects of agent architectures, which typically define the essential attributes that the agent architecture should embed:
- Methodologies that associate their application to specific agent architectures, and typically guide the modelling of the notions of these architectures to a typical/formal level.

Although most agent systems methodologies fall under these strands of work, naturally these are not strictly distinguishable in agent systems methodology development. Furthermore, the degree and depth upon which methodologies explain their approach is quite different. Table 5 presents the situation of methodologies in terms of this schema, which is discussed throughout this section.

Promoting the process of agent systems architecture development

One of the main lessons learnt from the large amount of work on agent architectures seems to be that there is not a unique architecture that can explain coherently the organisation and behaviour of autonomous agents and MAS. Methodologies that take this consideration into account do not focus on the specification (or adoption of an existing agent) architecture but attempt to set the ground for this task. These methodologies mainly attempt to link architecture development to other methods for analysis and design, such as O–O analysis and design, workflows and extreme programming, in a software engineering focus.

The most typical examples of this approach include: MaSE (2000, 2001), which associates the development of (mainly the aspects of single) agent architecture to O–O principles; Gaia (2000) that explicitly states that it is neutral with regard to the agent architecture; the conceptual framework for agent oriented and role based workflow modelling methodology of Yu and Schmid (1999); and the tool-supported process analysis and design methodology (Knublauch and Rose 2002) which follows an extreme programming approach that gradually leads to the identification of the multi-agent architecture for a particular problem domain by the iterative design and refinement of process models.

A particular advantage of these methodologies is that they associate the development of agent architectures to mainstream processes for analysis and design, which enables their uptake by software developers. Furthermore, the fact that they leave the issue of agent architecture open may be of help to some agent practitioners and software developers that have the experience to identify a suitable architecture for a particular problem.

Table 5 Architectural design approaches in agent systems methodologies

	Promoting the process of architecture development	Emphasis on the ontological aspects of agent architectures	Associating design to specific agent architectures
MESSAGE/UML		√	
Tool-supported process analysis and design of MAS	√	√	
Tropos		√	√
Multiagent systems engineering (MaSE)	√		
A semiotic approach based on roles and norms			√
Gaia	√		
Agent oriented and role based workflow modelling	√		
An agent-oriented design methodology			√

On the other hand, the fact that many methodologies do not commit to an agent architecture may also be considered as a disadvantage, especially from the perspective of practitioners outside the agent community. This is mainly due to the fact that this type of approach to the issue of agent architecture implies a weak notion of agency, which may seem poor in explaining why agents are different from other types of computational entities, especially if we take into account that the methods that are provided are based on other areas of work. Methodologies that promote the process of architecture development need to provide advanced methods for architectural design whose outputs can be evaluated in terms of agent systems theory.

Metamodels for ontological specification of agent systems

Another approach that is followed by methodologies with regard to agent architecture development is to define agent metamodels. Metamodels are also employed in terms of software engineering to describe the relationships of the concepts of O–O Analysis and Design to O–O Programming (Hillegersberg and Kuldeep 1999). However, agent systems methodologies define the relationships of conceptual agent attributes in a way that eases the development of ontologies. An ontology defines the basic terms and relations comprising the vocabulary of a topic area as well as the rules for combining terms and relations between terms (Sugumaran and Storey 2002) and may be generic or specific to a problem domain.

In principle, this approach to analysis and design of agent-based systems allows autonomous agents to be based on different architectures, provided they conform to a language that communicates the semantics of a shared ontology. The issue of agent architecture is usually left open as in process-based methods described in the section ?. A metamodel defines the attributes relevant to agent theory that is proposed by the methodology and elaborates on their relationships. The methodologies that define agent metamodels include: the tool-supported process analysis and design methodology (Knublauch and Rose 2002); the Tropos methodology (Sannicolo et al. 2001); and MESSAGE/UML (Caire et al. 2002), which defines a formal metamodel that extends UML with concepts required for agent-oriented modelling; Also, of particular relevance to this work is also the Agent-Object Relationship (AOR) metamodel[1] (Wagner 2003), which is a formal metamodel based on ER and UML that defines the relationships of agent attributes to O–O programming. Table 6 overviews the key attributes of these metamodels. Naturally, these attributes are associated in a different way per metamodel and designers need to further identify their relationships and scope of definition from the documentation of each methodology.

[1] The AOR metamodel is not referred to as a methodology but it is directly relevant to work that can be included in terms of the purposes of this section.

Table 6 Mental agent attributes in agent systems methodologies

	Tool supported process methodology	Tropos	MESSAGE/UML	AOR metamodel
Agent	√		√	√
Action			√	√
Event			√	√
Goal		√	√	
Resource		√	√	
Activity	√			
Actor		√		
Claim				√
Commitment				√
Dependency		√		
Object				√
Plan		√		
Process	√			
Property	√			
Role	√		√	
Service			√	
Task			√	
Notation	ER and class diagrams	UML metamodel	UML metamodel	ER and UML class model

Associating methodology development to a particular architecture

The most straightforward way to incorporate related work on agent architecture into agent systems methodology development is to associate this with particular agent architectures. When the starting point of this work is the information system requirements, then this association enables a top-down approach to the design of agent architectures that is linked with a specific situation. When the starting point is the architecture itself, then this work results in computational models of agent architecture attributes. The methodologies that associate their development with agent architectures include: the agent-oriented design methodology (Kinny et al. 1996; Kinny and Georgeff 1997), which provides modelling techniques for agents based upon the BDI architecture that extend the Object-Oriented (O–O) models of Booch (1994) and Rumbaugh et al. (1991); Tropos, which defines key notions of the actor and dependency model in a way that can be mapped (Castro et al. 2002)to the BDI (Beliefs, Desires, Intentions) agent architecture (Bratman 1987; Kinny and Georgeff 1991)[2]; and the semiotics approach (Chong and Liu 2000), which defines a simple behaviour-based agent architecture that is consistent with their approach based on roles, states and five types of norms that are identified by their analysis.

[2] BDI architectures consider that a rational agent having certain mental attitudes of Belief, Desire and Intention, representing, respectively, the information, motivational, and deliberative states of the agent (Georgeff et al. 1999).

The association of particular agent architectures to the methodology process is very useful since it provides methods that link architectural modelling to other phases of systems development. In particular, the association of agent architectures to the phases of domain requirements and software engineering that is achieved respectively by the first two methodologies shows that these methodologies might be used complementarily to address a wider range of issues of agent systems development than each one alone. When methodologies link their work to agent architectures, they elevate previous related work in the field and enable its application in specific problem situations.

On the other hand, placing a particular focus to the architecture of agent systems is a restricting practice for specific domains, since their requirements may not be readily represented to a particular architecture that is associated with a methodology. However, even in this case the need to associate methodologies to architectures still remains, since methodologies need to provide comprehensive guidance with regard to architectural design, since it is commonplace that enabling (at least the) the autonomy of a computer-based application requires a different architecture of autonomous agents in comparison to other technologies.

Coordination and interaction of agent systems

Agent systems coordination refers to the informational content and organisational structure of a multi-agent system and the rules and actions by which this content and structure may 'change' or evolve during the lifetime of the system. Agent systems methodologies provide guidance to some degree for the identification of agent communicative acts and messages mainly in terms of concepts that are borrowed from O–O programming. For example:

- Gaia (Wooldridge et al. 2000) defines a protocol model in an attempt to reflect dependencies and relationships among agents on the basis of purpose, inputs, outputs and processing and an acquaintance model that illustrates communication pathways to trace communication bottlenecks at runtime.
- In MaSE (DeLoach et al. 2001), the notion of conversations is developed. The basic idea is that when an agent receives a message it compares it to active conversations and if the message is a part of an existing conversation the agent changes its state otherwise the agent compares it to all possible conversations that it can participate in (depending on the agent's role) and if a match is found, it starts a new conversation.
- In Tropos, the interaction requirements are identified in terms of goal and resource dependencies among agents and messages for addressing those dependencies are identified (Giunchiglia et al. 2002).
- Prometheus (Padgham and Winikoff 2002) borrows interaction diagrams from O–O programming (which are based on use cases) and uses interaction protocol diagrams that generalize from interaction diagrams to basic calls.

Many methodologies emphasise the issue of representation of communications among agents (e.g. Prometheus, MESSAGE/UML and MaSE). They provide

notations for representing agent communications or adapt existing notations in order to provide examples of application according to mainstream (usually object oriented) software engineering practices. Despite the importance of representation, it may not provide only by itself a sufficient solution to the problem of coordination and interaction design for agent systems, since these need to be complemented with methods for this task. There is a need to see beyond representation, for identification and relationships of interactions with knowledge level or behavioural notions.

Some methodologies indeed associate the specification of communication protocols to the knowledge level and behavioural attributes of agent systems (especially the semiotics methodology, Tropos and the high level and intermediate models). This is required since the result of a processing cycle of an agent's operation may be a communicative action. However, there is still a need for methodologies to address coordination and control beyond dependencies and norms aiming at the dynamics of interaction. The approaches to coordination and interaction that are provided by methodologies specify the ''fixed'' dimensions of communication and not the dynamics of their interaction, which does not address the heart of the design issues. According to Castelfranchi (1998) ''the notion of social action (which is foundational for the notion of an Agent) cannot be reduced to communication or modelled on the basis of communication ... agent cannot be called 'social' because they communicate, but the other way round: they communicate because they are social.''

Scope of agent systems methodologies

During the detailed analysis and evaluation of methodologies in terms of the above questions, it was made clear that wider methodological issues emerge and need to be discussed further. Thus there is a need to further identify the scope of agent methodologies referring to at least the types of problems that they address and the lifecycle of systems development that they cover.

With respect to the former, it seems that most agent methodologies are general in purpose and they specify few limitations only in terms of technical development with the most comprehensive description in this respect provided by Gaia (Wooldridge et al. 2000). There is also other work that explicitly addresses specific application domains. This includes the MaMAS methodology (Galland et al. 2003) that addresses agent-based simulation of distributed industrial design, the AOSE methodology for information gathering (Zhang et al. 2002) that addresses agent-based systems for information gathering and DACS (Design of Agent-based Control Systems, Bussmann et al. 2001, 2002). Furthermore, there are some methodologies that seem to better fit organisational setting such as Gaia (Wooldridge et al. 2000) and the semiotics approach (Chong and Liu 2000), without however referring explicitly to their scope in this respect.

In order to discuss the aspects of the scope of methodologies that refer to the phases of the lifecycle, we adhere to the definitions of Hirschheim et al. (1995) and

Avison and Fitzgerald (1998). Certainly issues such as: the sequence by which these activities occur, the depth to which each methodology strives in for each of the above phases and the occurrences of each activity during the life-cycle, and others, are quite different per methodology.

- Problem formulation refers to 'the identification of problematic situations and object systems for change, assisting sense-making at the front-end of the life-cycle' (Hirschheim et al. 1995). Problem formulation also includes the social aspects of feasibility that are related to the understanding of the need and the goals for design.
- Feasibility refers to 'principles of accounting and economics to assess the effectiveness and efficiency of design proposals' (Hirschheim et al. 1995). However, Avison and Fitzgerald (1998) imply that a definition referring to economic terms is a narrow definition for feasibility, as ''feasibility may include social, user and technical issues'' (which may also be related to problem formulation, depending on the methodology).
- Analysis is performed on the factual situation identified and presents the current system investigating alternative methods and processes. Perhaps more intensely than any other previous phase, analysis includes the capture of user requirements.
- Design refers to the description of the new system, incorporating the requirements of the analysis, attempting to avoid problems occurred with the old system and without including new ones.
- Implementation refers to the actual realisation of the new system including prototyping and testing.
- Evaluation refers to ensuring that the system follows the requirements identified in the previous phases of the lifecycle.
- Maintenance happens when the new system is operational and aims at ensuring that the operation of the system is efficient and effective.

The investigation of methodologies in terms of the systems lifecycle is useful since it provides a widely accepted set of phases upon which methodologies may be mapped. Inevitably, some methodologies are more comprehensive than others with respect to their orientation to the lifecycle. Although the discussion regarding the coverage of the systems is made with a loose correspondence to the above phases, we note that whether a methodology may be considered as addressing in a comprehensive way a particular phase of the lifecycle is not determined solely on the claims of methodologies but also includes the study of their building blocks or methods provided. Thus in terms of Table 7, the strongly shaded area indicates that a methodology covers the phase in some detail and provides a comprehensive set of methods, models, techniques, etc. of support, while the lightly shaded area means that the methodology addresses the area in less detail.

- Problem formulation: Some methodologies incorporate aspects that might be related to problem formulation (Semiotics, REF and Tropos), since they define models that attempt to elicit information about the problem domain. However they still do not specify clearly important aspects of problem formulation such

Table 7 Indication of the scope of agent systems methodologies in terms of the systems lifecycle

Title	Authors	Problem Formulation	Feasibility	Analysis	Design	Implementation	Evaluation	Maintenance
ODAC (Open distributed Applications Construction)	Gervais, M.P. (2003)							
REF: Agent based Requirements Engineering Framework	Bresciani and Donzelli (2003)							
ADELFE (Toolkit for designing software with emergent functionalities)	Bernon et al (2002)							
AOSE methodology for information gathering	Zhang et al (2002)							
Extended OPEN Process framework	Debenham and Henderson-Sellers (2002)							
MESSAGE/UML	Caire et al (2002)							
Prometheus	Padgham and Winikoff (2002)							
Tropos	Bresciani et al (2004)							
Multiagent Systems Engineering (MaSE)	DeLoach et al (2001)							
A Semiotic Approach based on Roles and Norms	Chong and Liu (2000)							
Gaia	Wooldridge et al (2000)							
High-Level and Intermediate Models	Elammari and Lalonde (1999)							
Agent oriented and role based workflow modelling	Yu and Schmid (1999)							
An Agent-Oriented Design Methodology	Kinny and Georgeff (1996, 1997)							

as the mechanisms for decision making and reaching consensus and the eligibility of methods for data collection.

- Feasibility: Some methodologies provide assistance to practitioners with regard to the feasibility of the proposed approach with respect to technical issues (BDI methodology, MESSAGE/UML, Prometheus, MaSE). Notably, there is as yet no methodology that addresses economic issues of agent systems application[3].
- Analysis: Most methodologies provide methods and techniques for enabling the analysis of the factual situation. However, few connect this analysis to user requirements (Tropos and REF); most of them focus on the representation of functional specifications, mainly in terms of illustrating the use of used cases.
- Design: The majority of methodologies address the technical design of agent systems in terms of models that specify the characteristics of these types of systems. The degree by which methodologies reach to detailed design, which includes the formal specification of agent related concepts (e.g. in terms of O–O techniques and agent metamodels) which determines whether they address this phase comprehensively or not.
- Implementation: This phase is addressed by providing formal specifications of agents systems in notations that can straightforwardly lead to software development. When there are also CASE tools that support the methodology or guidelines for implementation with existing agent software development environments then the methodology is considered as addressing this phase in a comprehensive way.
- Evaluation and maintenance: These phases do not seem to be addressed by any methodology in the field.

Challenges for future work with respect to the methodologies' coverage of the systems lifecycle are related to the degree by which they address the requirements of each phase and their extension of scope targeting at other phases. Some of them include the enrichment of the problem formulation phase to include methods for decision making and data collection, the incorporation of eliciting user requirements to the analysis phase, the provision of formal models for design, addressing the issue of feasibility of agent systems from an economic perspective, and targeting at evaluation and maintenance of agent systems. There are also opportunities for practitioners to make combined use of methodologies with building blocks that address complementary phases of the lifecycle. For example, this may be the case for methodologies that address comprehensively the analysis phase with those that address comprehensively detailed design and implementation.

[3] Some information systems methodologies include guidelines and methods for feasibility studies concerned with economics like SSADM (Structured Systems Analysis and Design Methodology) (Weaver et al. 1998) and DSDM (Dynamic Systems Development Method) (The DSDM consortium, http://www.dsdm.org).

Conceptual foundations of agent systems methodologies: a holistic view

Besides other aspects of a methodology that are usually explicit, a methodology is also guided and formulated by assumptions that determine the capability of a methodology to affect the current organisation of the system to which it is applied. As Lane (1999) remarks 'these assumptions have implications for the type of modelling work that is possible and the nature of any changes that may be made in consequence'. These assumptions form the conceptual foundations of a methodology and largely determine whether practitioners can adhere to it or not. It may need to be stressed that the investigation of a methodology's conceptual foundations is not an idle academic exercise. In particular, unless agent methodology work firmly discusses the ways that they can affect and change the organisational aspects of human activities that they come to support, they will inevitably constrain their applicability solely to technical development.

There is substantial, long-dating work in the field of systems thinking and information systems theory and practice about the nature of methodologies and the paradigms and assumptions by which these are equipped to address decision making and design. It is out of the scope of this paper to provide a detailed account of this work but as a working schema the paper refers to two widely-accepted paradigms for methodology development for the purposes of this work as 'narrowly technical' and 'contextually sensitive'.

The paper refers to the 'narrowly technical' stance in a similar sense to what is referred to as functionalism by Burrell and Morgan (1979), hard systems thinking by Checkland (1981) and rationalistic tradition by Winograd and Flores (1986). The narrowly technical approach consists of assumptions about the nature of methods that emphasise optimisation, effectiveness and efficiency and assumptions about the nature of the world (i.e. a human activity system or a domain) that consider it as ''objectively existing'' with the consequence that methods do not question its organisation and structure. The narrowly technical approach promotes the role of the designer as an 'expert' of mastering design tools and most often considers that it is legitimate for decision making to happen by the most ''appropriate'' people involved in the design process.

In contrast, the paper refers to the 'contextually sensitive' stance in a similar sense to what is referred to as interpretivism by Burrell and Morgan (1979), soft systems thinking by Checkland (1981) and social relativism by Hirchheim et al. (1995). The contextually sensitive approach consists of assumptions about the nature of methods that emphasise learning, knowledge sharing and identify the meaning of interactions, and assumptions about the nature of the world that consider it as subjectively existing. Thus each stakeholder may have a different perspective about the world, with the consequence that the methods provided need to explore its organisation and structure, rather than consider these as objectively existing. The contextually sensitive approach promotes changes to the social environment by means of consensus.

It should be noted that the two kinds of approaches are actually using agent concepts for different purposes and making different claims about agent concepts. The first uses agent concepts for addressing complexity in software engineering

(see e.g. Jennings 2001), while the second uses agent concepts for modelling and analyzing the problem context (see e.g. Yu 2001). A unifying aim of agent systems methodologies is to model the dynamic aspects of information systems and a common general perspective that guides this work is that they view information systems as a set of interacting agents that may have common and conflicting goals. Besides this focus, the study of methodologies' documentation does not elaborate on their conceptual foundations and assumptions about the nature of their approach with respect to the methods they employ and the social environment in which they are applied. The background and objectives of methodologies are the main vehicle for identifying these foundations since these are usually provided by the authors of methodologies explicitly (Table 1).

The 'narrowly-technical' stance

Narrowly technical approaches are largely concerned with addressing the software implementation problems of agent systems. To achieve this, methodologies usually attempt to marry agent research to O–O programming. This is a big step for agent systems development that contributes to the mainstream acceptance of agent-based computing as a new paradigm for development of complex information systems employed with some degree of autonomy and/or intelligence. These approaches provide useful guidance for technical development of agent systems in various ways including, among others, basic steps for software engineering tasks and O–O models for agent concepts. Methodologies that profoundly distinguish this type of focus include MESSAGE/UML (Caire et al. 2002); Prometheus (Padgham and Winikoff 2002); MaSE (DeLoach et al. 2001), and the Agent-Oriented Design Methodology (Kinny and Georgeff 1996, 1997).

An important shortcoming of the narrowly technical approaches is that they do not deal extensively with identifying design requirements in a context specific way. Although some methodologies refer to aspects of the problem situation during the argumentation of their problematique, they do not deal with this as a part of their methodology, i.e. they do not include building blocks for eliciting knowledge from, interacting with and understanding this environment. For example, the Gaia methodology clearly situates its scope outside user requirements (Wooldridge et al. 2000). However, for practical adoption, a methodology should provide guidance and support from the beginning of the lifecycle. As a consequence these approaches need to be complemented with methods that investigate user requirements.

The focus of narrowly technical approaches is on issues such as improving productivity, efficiency, extensibility, robustness, etc. and specifying problematic situations in engineering terms without explaining adequately why specific aspects of the environment are modelled following that methodology. This focus elevates the role of the designer to that of an expert that should be in the position to manage methodology building blocks in order to fit the agent-based system with the technology.

Questions that may arise from this focus are related to the essence of an agent in the physical world/information system and to the corresponding realisation/

observability and evaluation of agents to the new form of the information system as this is formulated by the use of a methodology. Other questions are related to the essential ''characteristics'' of the problematic situations for which software agents may be more appropriate instead of other conceptualisations. These methodologies are largely still in laboratory research stages and stronger impact can be expected as industrial investment and participation increase.

The 'contextually sensitive' stance

Contextually sensitive approaches are largely concerned with eliciting design objectives and applying a methodology's building blocks via disseminating ideas and promoting social interaction with stakeholders of the problem domain. Thus the scientific methods used attempt to acquire knowledge and aim at interpreting the current situation. These methodologies address the specification of user requirements and provide methods that may identify basic aspects of an information system, possibly with the participation of other stakeholders.

There seem to be at least three agent systems methodologies that are influenced to some degree by the assumptions made by the contextually sensitive approach. The semiotics approach (Chong and Liu 2000) describes a semantic model that 'provides a mechanism for identifying a list of terms that should be included into the ontology' and also 'reveals the types and roles of agents that are needed in the system to solve the problems in the business domain'. The Tropos methodology (Bresciani et al. 2004) defines an actor model that represents the identified stakeholders of a domain with their respective objectives (goals) as part of the early requirements phase. REF (Brecianni and Donzelli 2003; Donzelli 2003) specifies an organisation model that represents agents, soft goals, hard goals tasks and dependencies that are identified in a specific problem situation.

The methodologies that follow a contextually sensitive approach attempt to interpret specific situations rather than define them 'objectively', in contrast to the narrowly technical approaches. Thus the role of the designer is best understood as that of a catalyst for the case of eliciting knowledge about the organisation and then that of introducing changes to the organisation in terms of agent-based systems. In this type of approaches it is important that the perspectives of involved stakeholders are included in models of the current situation that can be understood not only by the designer but by all people involved, and that a consensus process exists for reaching to a commonly agreed set of objectives for the new organisation and system that will arise from the introduction of agent systems.

Challenges for agent systems methodology development and implications for design

This review of the landscape of agent systems methodology development suggests a number of open challenges for future work. While these challenges are not mutually independent and may not be applicable for each and every one of the methodologies,

it seems that they are central for the further development of the field. In particular, we identify the following challenges for future research and development:

To further refine, comprehend and shape the underlying conceptual foundations of agent systems methodologies

Currently methodologies for agent systems design do not specify explicitly their underlying conceptual foundations and assumptions. This review has shown that these foundations determine the nature of modelling work that may be conducted and the range of possibilities for taking action to change the situations faced by the methodology. Currently these aspects are largely unclear for agent systems methodologies. These foundations and underlying assumptions need to be clarified in terms of both agent systems and information systems theory and practice.

To identify other methodologies, methods and techniques, etc. that can contribute to the extension of the scope of work

After studying the scope of methodologies, we have seen that most methodologies do not address in depth some aspects of the systems lifecycle. This is particularly evident for the phases of problem formulation, feasibility and analysis, where there is also substantial work outside the field of agent systems. There is a need to identify related work in this respect and incorporate this to the body of agent methodologies. This is also the case with the phases of evaluation and maintenance that are not at all addressed. When methodologies incorporate the above issues, they will be in the position to explain fundamental questions about agent systems design such as the economics of adopting an agent-based approach and the particular requirements of evaluating agent systems.

To situate the agent systems design process to the wider context of information systems development

Currently a large body of methodologies address this challenge in a technical way of associating agent systems design with mainstream software engineering techniques. What is urgently needed is to emphasise the contextual aspects of systems design, for example the active participation of human actors, the role of decision making during problem formulation, the eligibility of data collection methods, etc. Few methodologies provide some assistance on the above but clearly there is much more to be done.

This requirement is particularly important for enabling a wider understanding of the basics of agent concepts and technology and the elicitation of related requirements that can straightforwardly lead to the identification of agents for particular problem situations. Therefore, the characteristics of problematic situations that require an agent-based approach have to be further investigated rather than deciding upon that ''intuitively''—a complex problem that requires distributed problem solving competencies may not always 'require' a multi-agent solution. Currently the identification of agents is mainly considered as a design decision and

the justification of the agent-based approach is made on general requirements, which are in the wish list of all designers, designing agent systems or not.

To further investigate the relationships of other agent systems research to current agent systems methodologies

Currently, many methodologies do not refer to prior work in agent systems research, which is sometimes surprising, for example, if we think about the large body of work in agent architectures. This aspect of methodology development is particularly important since there are works that may be easily incorporated to the body of methodologies in other areas as well, such as for example associating agent systems development with advanced task analysis techniques (e.g. Koutsabasis et al. 1999) and evaluation (e.g. Dikaiakos et al. 2001). This association elevates previous work on agent systems and provides justifications regarding the agent-oriented aspects of this work, which are sometimes very few due to the emphasis on software engineering.

To extend the scope of agent systems coordination/interaction

Currently, methodologies provide limited assistance with regard to the design of coordination/interaction of agent systems. In this respect, agent systems methodologies do not borrow much from related work in the field. There is some work in some agent methodologies, especially the semiotics methodology, Tropos and the high level and intermediate models, that may fit well with theories of coordination (such as the theory of coordination of Malone and Crowston 1984). Besides this work, most methodologies mainly deal with providing notations for representing communication protocols and need to include the design of agent systems dynamics.

To apply and evaluate agent systems methodologies by employing them to specific problem situations

Most methodologies are currently illustrated in the form of scenarios or 'case studies' (though the term is not used correctly, since in most cases these do not refer to real application). A few methodologies provide CASE tools and/or demonstrate the implementation with existing agent development environments. There is certainly a need to link agent systems methodology work with existing environments, multi-agent infrastructures etc. in order to provide practical assistance to agent developers.

Summary and conclusions

The paper presented a critical review of agent systems methodologies that envisages formulate the current knowledge in the field in a form that promotes understanding and uptake by practitioners. We have adopted a broad perspective encompassing fundamental issues from the areas of IS methodology development and agent

systems research and development. Over 30 agent systems methodologies documented in more than 70 journal and conference papers have been examined. We have tried to accommodate the best practices of methodologies rather than insist on the drawbacks of methodologies, in terms of clarity, comprehensiveness, and others.

The work presented in the paper inevitably makes interpretations of the work in agent systems methodologies in order to enable a critical review. When it comes to agent research issues, interpretations mainly take into account other work in agent systems, which has not explicitly been named as 'methodology' by the authors but should be related to it, including work in agent metamodels and architecture. When it comes to more general methodological issues, interpretations mainly take into account related work in the field of information systems theory and practice, which has a long historical background over principled problem solving. We envisage that the new understanding of fundamental issues in agent systems methodology development may be of help to academics and practitioners in the areas of agent systems and information systems research.

The paper identifies significant challenges for agent systems methodologies that, if pursued, can contribute to a new understanding of the field towards holistic design methods that place human users and information systems stakeholders at the centre of interest and involve them to the design process as much as possible. Dealing with the identified challenges can enable deeper comprehension and wider uptake of this work both inside and outside the agent community and contribute to the evaluation of the current situation in the field. The clarifications that this review has made over agent systems methodologies can assist information systems designers to develop agent systems that are more responsive to human and social needs.

References

Avison DE, Fitzgerald G (1998) Information systems development: methodologies, techniques and tools, 2nd edn. The Alden Press, Oxford

Bauer B (2001) UML Class diagrams: revisited in the context of agent-based systems. In: Proceedings of agent-oriented software engineering (AOSE) 2001, Agents 2001, Montreal, pp 1–8

Bernon J, Gleizes MP, Picard G, Glize P (2002) The Adelfe methodology for an intranet system design. In: Proceedings of the 4th international bi-conference workshop on agent-oriented information systems (AOIS-2002 at CAiSE*02) Toronto, 27–28 May 2002

Booch G (1994) Object-oriented analysis and design, 2nd edn. Addison-Wesley, Reading

Booch G, Jacobson I, Rumbaugh J (2001) OMG—Unified modelling language. OMG, September 2001. http://www.omg.org

Bratman M (1987) Intention, plans, and practical reason. Harvard University, Campridge

Bresciani P, Perini A, Giorgini P, Giunchiglia F, Mylopoulos J (2004) Tropos: an agent-oriented software development methodology Auton Agent Multi Agent Syst. 8(3):203–236

Bresciani P, Donzelli P (2003) REF: a practical agent-based requirement engineering framework. In: 5th international bi-conference workshop on agent oriented information systems (AOIS-2003), Chicago, 13 October 2003. http://www.aamas-conference.org/ at ER'03

Burmeister B (1996) Models and methodology for agent-oriented analysis and design, In: Fisher K (ed) Working notes of the KI'96 workshop on agent-oriented programming and distributed artificial intelligence, DFKI document D-96-06. http://www.dfki.uni-kl.de/dfkidok/publications/D/96/06/abstract.html

Burrell G, Morgan G (1979) Sociological paradigms and organisational analysis. Heineman, London

Bussmann S, Jennings NR, Wooldridge M (2001) On the identification of agents in the design of production control systems In: Ciancarini P, Wooldridge M (eds) Agent-oriented software engineering, January 2001. Lecture notes in AI. vol 1957. Springer, Heidelberg

Caire G, Garijo F, Gomez J, Pavon J, Vargas E (2002) Agent oriented analysis using MESSAGE/UML. In: Proceedings of the 4th international bi-conference workshop on agent-oriented information systems (AOIS-2002 at CAiSE*02), Toronto, 27–28 May 2002

Castelfranchi C (1998) Modelling social action for AI agents Artif Intell 103:157–182

Castro J, Kolp M, Mylopoulos J (2002) Towards requirements-driven information systems engineering: the tropos project Inf Syst 27:365–389

Checkland P (1981) Systems thinking, systems practice. Wiley, New York

Checkland P, Scholes J (1998) Soft systems methodology in action. Wiley, New York

Chong S, Liu K (2000) A semiotic approach for modelling and designing agent-based information systems based on roles and norms. In: Proceedings of 2nd conference on agent-oriented information systems. http://www.aois.org/

Dam KH, Winikoff M (2003) Comparing agent-oriented methodologies. In: 5th international bi-conference workshop on agent oriented information systems (AOIS-2003) Melbourne,14 July 2003. At Autonomous agents and multi-agent systems (AAMAS'03)

Debenham J, Henderson-Sellers B (2002) Full lifecycle methodologies for agent-oriented systems—the extended open process framework. In: Proceedings of the 4th international bi-conference workshop on agent-oriented information systems (aois-2002 at caise*02) Toronto, 27–28 May 2002

DeLoach SA, Wood MF, Sparkman HC (2001) Multiagent systems engineering. Int J Softw Eng Knowl Eng 11(3)

Dikaiakos M, Samaras G (2001) performance evaluation of mobile agents: issues and approaches, in performance engineering: state of the art and current trends. In: Dumke, Rautenstrauch, Schmietendorf, Scholz (eds). Lecture Notes in Computer Science Series, state of the art survey, vol. 2047. Springer, pp 148–166

Durfee EH, Lesser VR (1991) Partial global planning: a coordination framework for distributed hypothesis formation IEEE Trans Syst Man Cybern 21(5):1167–1183

Elammari M, Lalonde W (1999) An agent-oriented methodology: high-level and intermediate models. In: Proceedings of 1st conference on agent-oriented information systems, 14–15 June 1999. http://www.aois.org

Galland S, Grimaud F, Beaune P, Campagne JP (2003) MaMAS: an introduction to a methodological approach for the simulation of distributed industrial design Int J Prod Econ 85:11–31

Gasser L (1991) Social conceptions of knowledge and action: DAI foundations and open systems semantics, Special issue on foundations of AI. Artif Intell J

Gervais MP (2003) ODAC: An agent-oriented methodology based on ODP. Auton Agent Multi Agent Syst, vol 7, 30-07-2003, pp 199–228

Giunchiglia F, Mylopoulos J, Perini A (2002) The tropos software development methodology: processes, models and diagrams. In: The 3rd international workshop on agent-oriented software engineering (AOSE-2002), Bologna, 15 July 2002. Held at Autonomous Agents & Multi-Agent Systems (AAMAS 2002)

Gómez-Sanz J, Pavón J (2004) Methodologies for developing multi-agent systems J Universal Comput Sci (J.UCS) 10(4):359–374

Gonzalez-Palacios J, Luck M (2005) A framework for patterns in Gaia: a case-study with organisations. In: Odell J et al (ed) AOSE 2004. LNCS, vol 3382, pp 174–188

Hillegersberg JV, Kuldeep K (1999) Using metamodeling to integrate object-oriented analysis, design and programming concepts Inf Syst 24(2):113–129

Hirschheim R, Klein HK, Lyytinen K (1995) Information systems development and data modelling, conceptual and philosophical foundations. Cambridge University, Great Britain

Iglesias C, Garijo M, Gonzalez J, Velasco J (1996) A methodological proposal for multiagent systems development extending common KADS. In: Proceedings of the 10th knowledge acquisition for knowledge-based systems workshop, 9–14 November 1996, Banff. http://ksi.cpsc.ucalgary.ca/KAW/KAW96/iglesias/ Iglesias.html

Iglesias CA, Carijo M, Gonzalez J (1999) A survey of agent-oriented methodologies, In: Intelligent agents V—Proceedings of the 5th international workshop on agent theories, Architectures and Languages (ATAL-98), Lecture Notes on Artificial Intelligence. Springer, Heidelberg

Jennings NR (1993) commitments and conventions: the foundation of coordination in multi-agent systems Knowl Eng Rev 8(3):223–250

Jennings NR (2001) An agent-based approach for building complex software systems Commun ACM 44(4):35–41
Johnson H, Johnson P (1991) Task knowledge structures: psychological basis and integration into system design Acta Psychol 78:3–26
Juan T, Sterling L, Winikoff M (2002) Assembling agent oriented software engineering methodologies from features. In: The 3rd international workshop on agent-oriented software engineering (AOSE-2002), Bologna, 15 July 2002. Held at Autonomous Agents & Multi-Agent Systems (AAMAS 2002)
Kendall EA, Malkoun MT, Jiang C (1995) A methodology for developing agent based systems for enterprise integration. In: Luckose D, Zhang C (eds) Proceedings of the first Australian workshop on DAI, Lecture Notes on Artificial Intelligence. Springer, Heidelberg
Kinny D, Georgeff M (1991) Commitment and effectiveness of situated agents. In: Proceedings of international joint conference on artificial intelligence, IJCAI91. 1991, pp 82–88
Kinny D, Georgeff M (1997) Modelling and design of multi-agent systems. In: MullerJP, Wooldridge M, Jennings NR (eds) Intelligent agents III (LNAI vol 1193). Springer, Berlin, pp 1–20
Kinny D, Georgeff M, Rao A (1996) A methodology and modelling technique for systems of BDI agents. In: Proceedings of 3rd international conference on agent theories, architectures and languages, 1996 (ATAL'96)
Knublauch H, Rose T (2002) Tool-supported process analysis and design for the development of multi-agent systems. In: The 3rd international workshop on agent-oriented software engineering (AOSE-2002), Bologna, 15 July 2002. Held at Autonomous Agents & Multi-Agent Systems (AAMAS 2002)
Koutsabasis P, Darzentas JS, Spyrou T, Darzentas J (1999) Facilitating user–system interaction: the gaia interaction agent. In: Proceedings of 32nd Hawaiian international conference on system sciences HICSS-32. IEEE Society Press
Lane DC (1999) Social theory and system dynamics practice Eur J Oper Res 113:501–527
Lind J (2002) Patterns in agent-oriented software engineering, In: The 3rd international workshop on agent-oriented software engineering (AOSE-2002), Bologna, 15 July 2002. Held at Autonomous Agents & Multi-Agent Systems (AAMAS 2002)
Luck M, McBurney P, Preist C (2003) Agent technology: enabling next generation computing—a roadmap for agent research, AgentLink, http://www.agentlink.org
Malone T, Crowston K (1994) The interdisciplinary study of coordination ACM Comput Surv 26:87–119
Odell J, Van Dyke H, Bauer B (2000) Extending UML for agents, In: Proceedings of AOIS 1999 (Agent-oriented information systems), Heidelberg, 14–15 June 1999
Padgham L, Winikoff M (2002) Prometheus: a methodology for developing intelligent agents, In: The 3rd international workshop on agent-oriented software engineering (AOSE-2002), Bologna, 15 July 2002. Held at Autonomous agents & multi-agent systems (AAMAS 2002)
Rumbaugh J, Blaha M, Premerlani W, Eddy F, Lorensen W (1991) Object-oriented modelling and design. Prentice Hall, Englewood Cliifs
Sannicolo F, Perini A, Giunchiglia F (2001) The Tropos modelling language: a user guide, Technical report, ITC-irst, December 2001
Shoham Y, Tennenholtz M (1992) On the synthesis of useful social laws for artificial agent societies, In: Proceedings of the 10th national conference on artificial intelligence (AAAI-92), San Jose, pp 276–281
Sloman A (1996) Towards a general theory of representations. In: Peterson D (ed) Forms of representation, Intellect books
Smith RG, Davis R (1981) Frameworks for cooperation in distributed problem solving IEEE Trans Syst Man Cybern 11(1):61–70
Sturm A, Shehory O (2003) A framework for evaluating agent-oriented methodologies, In: 5th international bi-conference workshop on agent oriented information systems (aois-2003), Melbourne, 14 July 2003. At Autonomous Agents and Multi-Agent Systems (AAMAS'03)
Sugumaran V, Storey VC (2002) Ontologies for conceptual modelling: their creation, use and management Data Knowl Eng 42:251–271
Treur J (1999) Methodologies and software engineering for agent systems. AgentLink Newsletter, 2. http://www.agentlink.org
Wagner G (2003) The agent-object-relationship metamodel: towards a unified view of state and dynamics Inf Syst 28:475–504
Weaver PL, Lambrou N, Walkley M (1998) Practical SSADM version 4: a complete tutorial guide. Pentice Hall, London

Weiss M (2003a) Pattern-driven design of agent systems: approach and case study, Conference on advanced information systems engineering (CAiSE), Springer, 2003

Weiss M (2003b) Patterns for motivating an agent-based approach. In: 5th international bi-conference workshop on agent oriented information systems (AOIS-2003), Chicago, 13 October 2003. http://www.aamas-conference.org/ at ER'03

Winograd T, Flores F (1986) Understanding computers and cognition. Addison-Wesley, MA

Wooldridge M, Ciancarini P (2001) Agent-oriented software engineering: the state of the art. In: Ciancarini P, Wooldridge M (eds) Agent-oriented software engineering. January 2001. Lecture Notes in AI, vol 1957. Springer

Wooldridge M, Jennings NR, Kinny D (2000) The gaia methodology for agent-oriented analysis and design J Auton Agent Multi Agent Syst 3(3):285–312

Yu E (2001) Agent-oriented modelling: software versus the world, Agent-oriented software engineering. In: Workshop proceedings. 2002. LNCS 2222. Springer, Heidelberg, pp 206–225

Yu L, Schmid BF (1999) A conceptual framework for agent oriented and role based workflow modelling. In: Proceedings of 1st conference on agent-oriented information systems. http://www.aois.org

Zhang TI, Kendall E, Jiang H (2002) An agent-oriented software engineering methodology with application of information gathering systems for LCC. In: Proceedings of the 4th international bi-conference workshop on agent-oriented information systems (AOIS-2002 at CAiSE*02) Toronto, 27–28 May 2002