Comput Syst Sci & Eng (2006) 3: 87-98
© 2006 CRL Publishing Ltd

International Journal of
Computer Systems
Science & Engineering

Integrating social modelling and agent interaction through goal-oriented analysis

Iyad Rahwan[*†], Thomas Juan[‡] and Leon Sterling[‡]

[*]*Institute of Informatics, The British University in Dubai P.O.Box 502216, Dubai, United Arab Emirates. Email: iyad.rahwan@buid.ac.ae*
[†]*(Fellow) School of Informatics, University of Edinburgh, Edinburgh, Scotland, UK*
[‡]*Department of Computer Science & Software Engineering, University of Melbourne, Parkville 3010, Australia. Email: leon,tlj@cs.mu.oz.au*

In the Agent-oriented software engineering (AOSE) community, it is now widely recognised that interaction is perhaps the most important single characteristic of complex software systems. Recently, the focus of AOSE research has shifted towards social (or organisational) modelling, which allows interaction to be modelled with higher-level constructs like organisational structures and social policies. However, a gap remains between the social models proposed in AOSE methodologies and the increasing number of sophisticated agent interaction frameworks outside AOSE, such as those based on auctions, commitments, and dialogue games. Hence, social models in AOSE methodologies need to have more "hooks" to enable integration with rich domain-specific agent interaction frameworks. Towards closer integration between social modelling in AOSE and other agent interaction frameworks, we extend the ROADMAP methodology and provide a framework for enhancing the analysis of interaction requirements through goal-oriented analysis and social modelling. More precisely, goal models provide the purpose of interaction, while social models provide the social dependencies and policies that govern interaction. We show how the resulting requirements specification, complemented by protocol descriptions, can inform the design of the system's interaction model. The resulting approach bridges the gap described above by linking high-level social concepts to low-level, domain-specific design decisions about interaction.

Keywords: agent interaction, social modelling, goal-oriented analysis

1. INTRODUCTION

The increasing complexity of distributed computer systems has led researchers to utilise various tools of abstractions in order to improve the software engineering process. However, the requirements of an increasing number of computing scenarios go beyond the capabilities of traditional computer science and software engineering abstractions, such as object-orientation. According to Zambonelli and Parunak (2003), four main characteristics distinguish future software systems from traditional ones:

Situatedness: Software components execute in the context of an environment, which they can influence and be influenced by;

Openness: Software systems are subject to decentralised management and can dynamically change their structure;

Locality in control: Software systems components represent autonomous and proactive loci of control;

Locality in interactions: Despite living in a fully connected world, software components interact accordingly to local (geographical or logical) patterns.

These characteristics have led to the emergence of the *agent* paradigm in computing: one that views computer systems in terms of multiple, interacting autonomous agents in a multi-agent system (Wooldridge, 2002). The agent paradigm promises to offer a powerful set of metaphors, concepts and techniques for conceptualising, designing, implementing and verifying complex distributed systems (Jennings, 2001). To this end, our long-term research agenda is to provide an integrated, comprehensive Agent-Oriented Software Engineering (AOSE) methodology. Our work towards this vision has been reported through our expanding ROADMAP methodology (Juan *et al.*, 2002; Juan *et al.*, 2003; Juan and Sterling, 2004; Kuan *et al.*, 2005).

Agents often need to interact in order to fulfil their objectives or improve their individual or collective performance. It has been argued consistently that the analysis and design of multi-agent interaction can benefit from the use of social metaphors (Dignum, 2004). It is argued that modelling interaction can be improved by using higher-level constructs like organisation structures and social policies. However, a gap remains between the social models proposed in AOSE methodologies and the increasing number of sophisticated agent (run-time) interaction frameworks outside AOSE.

On one hand, within AOSE, work has been done on using organisational abstractions (structures, policies etc.) to support the analysis and design stages (Ferber *et al.*, 2003; Zambonelli *et al.*, 2003; Mao and Yu, 2004). On the other hand, various agent interaction frameworks and protocols are being proposed outside the AOSE community, such as those based on auctions (Wurman *et al.*, 2001), commitments (Maudet and Draa, 2002), or dialogue games (Mcburney and Parsons, 2003), with applications ranging from electronic commerce (Sandholm, 2002), to computer games (Gros *et al.*, 2004), to military simulations (Soon *et al.*, 2004) and logistics planning (Perugini *et al.*, 2004). AOSE methodologies such as Gaia seem far too abstract, while domain-specific interaction frameworks are far too precise. Indeed, some multi-agent interaction mechanisms are very specific and prescriptive, much like searching and sorting methods in algorithmics. The gap between the two threads of research makes it difficult for AOSE methodologies to integrate with and exploit such rich interaction frameworks. We believe this is because social models in AOSE lack enough “hooks” to enable easier integration with rich interaction frameworks.

Our objective in this paper is to give software practitioners the appropriate concepts and constructs to enable them to analyse and express their requirements in a way that exploits rich interaction mechanisms, i.e. by improving requirements analysis such that the design-time choices of specific interaction mechanisms become more obvious. To this end, we propose a change in focus by making social modelling in AOSE methodologies more open, in the sense that it easily accommodates various research results on specific frameworks of agent interaction. More precisely, we extend the ROADMAP methodology and provide a framework for enhancing the analysis of *interaction requirements* through goal-oriented analysis and social modelling. Goal models provide the *purpose* of interaction, while social models provide the *social dependencies* and *policies* that govern interaction. We show how the resulting requirements specification, complemented by protocol descriptions, can inform the design of the system’s interaction model. The resulting approach bridges the gap described above by linking high-level social concepts to low-level domain-specific design decisions about interaction.

In a nutshell, the significance of the contributions of this paper is as follows. Our framework enables the capture and analysis of interaction requirements in a *protocol-friendly* manner. In doing so, our framework provides a more integrated approach to goal and social modelling, where *high-level* analysis constructs (e.g. goals, organisational structures) and low-level design constructs (e.g. protocols) fit together. This is done within a framework that is both simple and flexible. Interaction analysis is enriched and linked to design-time constructs without committing to a particular protocol specification language, such as finite state machines, AUML sequence diagrams, or auction rules.

The rest of the paper is organised as follows. In the next section, we provide a more detailed motivation of the need to integrate social modelling with agent interaction frameworks. In Section 3, we present our approach to making social modelling more open through goal-oriented modelling of interaction. We then conclude the paper in Section 4.

2. MOTIVATION

Recall that this paper is motivated by the observation that a gap exists between social modelling in the AOSE literature and research carried out in other areas of agent research, particularly in applying (run-time) agent interaction frameworks to different application domains. We believe the custom nature of social modelling constructs in AOSE methodologies impedes this integration.[1] Existing AOSE methodologies are quick to move into low-level technical details of interaction without supporting the developer in addressing higher-level interaction requirements. For example, in Gaia, the analysis stage starts directly with low-level constructs like protocols, hence not supporting the developer in addressing higher-level interaction requirements such as collaborative learning.

We argue that social modelling in AOSE can be distilled into a cleaner framework with more “hooks” available to enable integration with agent interaction frameworks. We believe that in order for the integration to be successful, there must be no conflict between the semantics of the specific interaction framework and the semantics of the AOSE methodology, no conflict between the notations and model presentations of the interaction framework and AOSE methodology, and no conflict between the development processes of the interaction framework and the AOSE methodology. To this end, the methodologies must present a simple view of social modelling and generic concepts and models as “hooks” around which interactions may happen.

From the analysis before, we propose some initial guidelines to assist the improvement of social modelling concepts and methods in order to integrate more easily with related research on agent interaction and accommodate related work under a unifying framework.

1. The concepts used to describe social modelling should be simple and general. Complex concepts often fail to model

1. By ‘custom nature’ we mean that the social modelling constructs in various methodologies are hard to map to one another. For example, it is not obvious how one could translate social modelling constructs from Opera to Gaia.

simpler situations. Specialized concepts with specialized semantics often have the effect of "fitting the problem to the solution." The semantics of the concepts must not preclude important issues from the wide variety of agent application domains.
2. The concept should be presented at the right level of abstraction, showing the most relevant aspects of the interaction while hiding unnecessary details. Over-specifying the system at early stages of development may encourage developers to focus on low level details and ignore important issues from the application domains.
3. The methods for creating the models should be consistent and should support easy translation from earlier models to later ones. Integration with external interaction frameworks will not be meaningful unless the benefit propagates to design, implementation and later stages of the development life-cycle.

These guidelines can be understood by looking at how *use cases* achieved great success in OOSE. Use cases only offer few and generic concepts, namely the system boundary, actors (stick figures) and the associated use cases (oval bubbles). The representation is simple and intuitive, which serves as visual summaries for more detailed text scenarios. These generic concepts in use cases serve as "hooks" and allow a wide range of domain specific issues to be integrated closely into use case models.

2.1 A mobile computing scenario

To help illustrate the need for richer social modelling, we present a scenario from mobile and pervasive computing. This scenario will then be used throughout the paper to illustrate our approach. Note that this case study is not intended to be a comprehensive application of the ROADMAP methodology. Instead, we shall focus on features developed in this paper.

The natural target for mobile computing is data management – devices will acquire information in one situation and deliver this information (or a processed version thereof) in another situation where deemed useful. In our scenario (Figure 1), the user's fridge recognises the need to purchase more soft drink as the user opens the last bottle. This information is added to the user's shopping list. It will be *useful* to remind the user of this shopping list when he/she is nearby a supermarket (Figure 1a). Hence, information about the shopping list is transferred to a mobile device held by the user which, based on some location-identification service, recognises the user's proximity to the supermarket and triggers the appropriate action. This action can range from simply retrieving a quote from some nearby food store and presenting it to the user (Figure 1c) to performing complex negotiations with multiple food stores to find the best deal (Figures 1b, d).

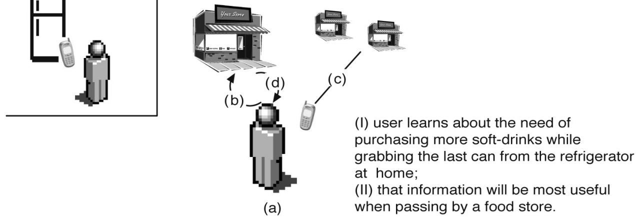

(I)
(II)
Your Store
(b)
(d)
(c)
(a)
(I) user learns about the need of purchasing more soft-drinks while grabbing the last can from the refrigerator at home;
(II) that information will be most useful when passing by a food store.

Figure 1 Mobile-commerce and pervasive computing scenario: shopping assistant

2.2 Shortcomings of existing approaches

In this sub-section, we use the above scenario in order to elaborate the need for integrating social modelling with agent interaction. We explain why existing object-oriented and agent-oriented approaches fall short from achieving this.

Use cases are not suitable for modelling agent applications like the mobile computing scenario above as its features are not tailored to capturing agent requirements. For example, the notion of system boundary is not very relevant in the mobile computing scenario. Traditional software on PCs or workstations has a clear user interface, and various functions are hidden inside the interface, so the distinction between the user outside the system and the functions inside the system is useful. In the mobile computing scenario, it can be argued that the system is following the user around, or the user is immersed inside the system. Indeed, many agent researchers do not make clear distinctions between users and other agents in the system and allow human users and software agents to be swapped dynamically. Therefore, a system boundary is not necessary.

Moreover, use cases focus on concrete *features* of the system exposed through the user interface. Agent systems, on the other hand, are likely to change their features or behaviours according to the context of the use. It is therefore undesirable to hard-wire the system features at such an early stage of development, narrowing prematurely the potential of the target application. For example, the user of the mobile computing example may be reminded when he/she is close to a supermarket. As long as such location/context information is correctly determined, how it is determined is not as important to the user. We consider system features to be at an excessively low-level of abstraction to be used to specify agent requirements. The side-effect for operating at this level is that the developer's effort must be devoted to gathering feature information that is not as important while the more significant information of system objectives may receive insufficient attention. Therefore a more abstract way to elicit requirements at higher-level of abstraction is needed.

A main benefit of agents is their ability to act intelligently, delivering services in a manner appropriate to the context of use. In another words, the non-functional quality requirements of the system play a much more important role in agent systems than in conventional systems. Whether a service is regarded as "intelligent" or "appropriate" depends on meeting the quality expectations of the users. For example, the user of the mobile computing scenario may expect the reminder to be non-intrusive. Use cases offer no assistance in specifying such quality requirements of the system and are inadequate for modelling agent systems.

AOSE methodologies offer different methods and models to elicit social requirements about agent interaction. We examine a number of AOSE analysis methods, namely Prometheus (Padgham and Winikoff, 2004), the *agent use* *cases* described by Heinze *et al.* (2000), Gaia (Wooldridge *et al.,* 2000; Zambonelli *et al.*, 2003) Tropos (Bresciani *et al.*, 2004), and MAS-ML (Dasilva and Lucena, 2004) with the above perspective. Gaia and Tropos are designed to support agent oriented analysis methods natively and are not based on use cases.

Both Prometheus and agent use cases from Heinze *et al.* are based on use cases, and inherit their drawbacks. In particular, Heinze *et al.* use another variant of use cases, allowing agents to be depicted inside the system boundary. The resulting requirements are also likely to be low-level while quality requirements are neglected. The Prometheus methodology employs a similar form of use cases and scenarios for eliciting requirements. Consequently, Prometheus insists on early commitment to a system boundary and does not accommodate quality goals. Finally, by committing to the BDI architecture at the design stage, the resulting requirements are likely to be biased and low-level, with reduced support for other high-level social issues such as learning.

Gaia assumes the availability of requirements and therefore does not support requirements elicitation. This is a significant drawback in terms of integrating with other social modelling frameworks. Once the requirements are gathered, Gaia organises them using organisational structures, roles and protocols. By moving into low-level constructs like roles and protocols early, Gaia also makes integration difficult by committing to specialised constructs early without staying generic.

Tropos utilises a goal-oriented approach based on i* (Yu, 1995) to model requirements. This approach allows high-level requirements to be described in terms of actors (roles, agents and positions) and the relationships between actors. The relationships between actors must involve a goal, a soft-goal, a task or a shared resource. While Tropos uses high-level concepts such as roles, goals and soft-goals, it also includes low-level concepts such as agents, tasks and resources. The number of possible constructs and their relationships can encourage developers to over-specify the system up front with too much detail. The modelling of high-level issues and low-level issues are not clearly separated to better guide the focus of development efforts. Tropos also make commitments to the BDI architecture and therefore places semantic restrictions on integrating to non-BDI interaction frameworks.

MAS-ML (Dasilva and Lucena, 2004) is a multi-agent modelling language based on the *TAO* (Taming Agents and Objects) conceptual framework (Dasilva *et al.,* 2003). However, MAS-ML is mainly concerned with integrating agent-related concepts from TAO with object-oriented UML via their meta-models. When it comes to interaction modelling, MAS-ML extends UML sequence diagrams. Therefore, as it stands, MAS-ML does not offer integration with other agent interaction frameworks.

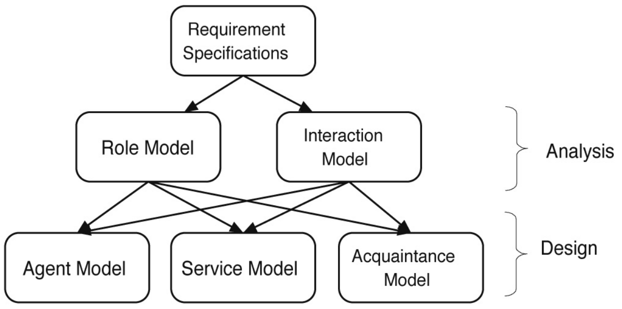


Figure 2 The Gaia models

In summary, existing AOSE methodologies present useful methods to address requirements analysis. However, the concepts and methods of these methodologies tend to be too specialised in terms of semantics, which encourages low-level details to be specified. Consequently, integration to other interaction frameworks of non-AOSE origin is more difficult as high-level and generic “hooks” are often missing or misplaced.

3. OUR APPROACH

In the previous section, we argued for the importance of integrating social modelling with agent interaction. In this section, we describe our approach. In particular, we extend the ROADMAP analysis models in order to provide a structured approach to specifying rich interaction requirements. We then show how these can influence design-time models. But first, we give a brief introduction to ROADMAP as it currently stands.

3.1 The ROADMAP methodology

ROADMAP (Juan *et al.*, 2002) started as an attempt to extend Gaia (Wooldridge *et al.*, 2000), a methodology intended to support the analysis and design of multi-agent systems (see Figure 2). In the analysis stage, a *role model* and *interaction model* are created. A role is described through a *role schema*, which describes the role’s protocols, activities, permissions and responsibilities. Interaction is described by providing simple descriptions of each protocol: the roles responsible for initiating and responding to the protocol, the input and output information, and a brief textual description. In the design stage, Gaia requires the specification of an *agent model*, a *service model* and an *acquaintance model*. These are mappings of the role model and interaction model to concrete constructs (much like the instantiation of classes as objects in object-oriented methodologies). The agent model specifies the agents types, which describe the roles the agent takes, and the number of instances of that type. The service model describes coherent blocks of functionality, as derived from role activities and protocols.Finally, the acquaintance model is a directed graph between agents that denotes communication flow.

From Gaia, ROADMAP inherits the organizational view on multi-agent systems, and the basic definitions of roles, protocols, agents and services. However, over time the semantics of these concepts in ROADMAP has become quite different, making it evolve as a methodology in its own right. ROADMAP provides the following main additional features:[2]

Role Hierarchies: These are organisational structures that specify societies on different levels of abstraction. A role

2. For a more detailed list, refer to Cernuzz *et al.* (2004).

can aggregate sub-roles: roles that are part of the super-role.

Agent Hierarchy: These are actual instantiations of role hierarchies. Loosely, roles serve as system specification, while agents serve as system implementation.

Expanding Role Descriptions: ROADMAP requires specifying knowledge components associated with roles, as well as the specification of pre- and post-conditions of protocols;

Performance Metric: This is a function that evaluates the performance of the system, and can be associated with a role, a particular agent, or a resource.

Recall that in this paper, we are mainly concerned with the requirements analysis stage. Roughly speaking, the analysis stage in ROADMAP involves the following:

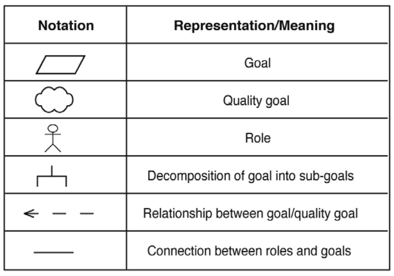
1. Create the *goal model* to analyse the *objectives* and purpose of the system. This model involves a hierarchy of functional and quality goals, roles associated with goals, and quality goals built from the start. The goal model consists of a hierarchy of functional goals (represented via parallelogram symbols) and quality goals (represented via clouds linked to functional goals), roles associated with goals (represented by stick figures). The notation for the ROADMAP goal model is summarised in Figure 3.

2. Create other models to analyse the *requirements* of the system. These models include:
a. The role model involves individual role definitions, and their interrelationship in the form of a role hierarchy.[3]
b. The environment model which is divided into a hierarchy of zones, each with its associated constraints and properties.
c. The knowledge model which specifies the main ontologies in the system organised into a knowledge hierarchy.
d. The social model which specifies social aspects such as norms, interaction policies, team conventions etc.

3. Iteratively improve the models until they are satisfactory

To illustrate how ROADMAP works, we now begin preliminary analysis of the scenario presented in Section 2.1.[4] Figure 4 shows a preliminary goal model based on our scenario. The main goal of the system is to handle mobile shopping and, in this case, this is done by identifying an opportunity for shopping (e.g. based on user proximity to a supermarket), finding a deal, and reporting the result back to the user. The opportunity identification must satisfy the quality goal of being permitted by the user context, however that may be expressed. At the most basic level, finding a deal involves identifying available alternatives and negotiating a specific deal accordingly. Negotiating a deal would naturally involve one or more other parties enacting the *vendor* role. The deal found must be the *best* (however that might be defined) from the point of view of the user; this is indicated as another quality goal.

After identifying the goal model, the analyst begins building the other models. For example, building the role model would involve further defining the roles "user" and "vendor" through role schemas. The environment and knowledge models also need to be defined, but since they are outside the focus of this paper, we now focus on discussing the extended goal and social models.



Notation	Representation/Meaning
▱	Goal
☁	Quality goal
웃	Role
┌┴┐	Decomposition of goal into sub-goals
← – –	Relationship between goal/quality goal
——	Connection between roles and goals

Figure 3 Goal model notation

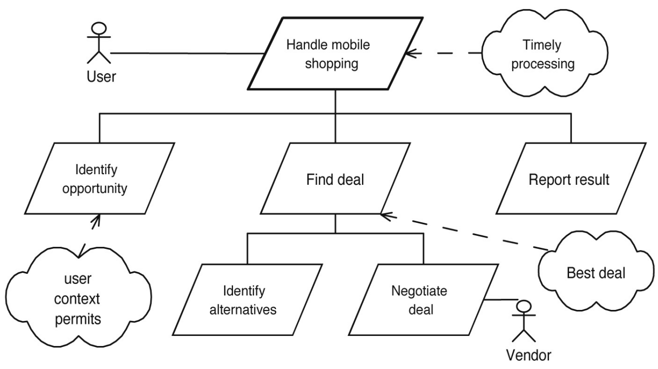


Figure 4 A ROADMAP goal model

3.2 Extending ROADMAP models for interaction

Currently, the ROADMAP goal model does not explicitly distinguish interaction goals at the analysis stage. Interaction is implicitly and directly addressed in the role schemas, where each role has a list of the protocols it uses. At the design stage, this interaction specification is directly instantiated into AUML interaction diagrams (Bauer *et al.*, 2001). However, the above scenario gives room for "teasing out" more domain knowledge about interaction at the analysis stage. For example, what does the interaction between the vendor and the user mobile device aim to achieve? What properties does it need to satisfy? And how could these requirements guide the design-time selection of specific interaction protocols? Our aim is to provide additional features to enable such elicitation and analysis.

In order to improve *interaction requirements* analysis, we extend the goal model and social model. In the design stage, these improved models inform the design of the *protocol model* and *interaction model*. Below is a brief description of our approach:

3. At the detailed design stage, these role hierarchies are instantiated into agent hierarchies which are run-time entities.
4. Note that we only demonstrate ROADMAP to the extent needed to motivate the extensions reported in this paper. Hence, we do not fully demonstrate all ROADMAP's features, such as environment and knowledge models, rich role models etc.

Step 1: Analyse interaction goals through the *extended goal model*. In addition to denoting functional and quality goals, this model now explicitly encodes *interaction goals*.
Step 2: Analyse the organisational structure through the *extended social model*. This model captures the different relationships among agents in order to elicit high-level policies that govern their interaction.
Step 3: Based on the *social policies* resulting from the extended social model, (i) refine the interaction goals in the goal model, and (ii) encode social policies in the form of quality goals over interaction goals.
Step 4: Concretise the social policies by giving them more precise meaning suitable for the design or selection of protocols in the protocol model.
Step 5: Design the protocol model, which is similar to the earlier ROADMAP protocol model. It consists of a set of protocol schemas, each describing various aspects of a particular protocol. Examples of protocols include: auction protocols, team formation protocols, delegation protocols etc.
Step 6: Design the final interaction model. This model is where specific interaction patterns are specified. These interaction patterns use protocols from the protocol model as building blocks, while making sure they fulfil the interaction goals and adhere to the social policies.

The enhanced models, and the flow of information between them, are depicted in Figure 5. The new goal model and social model provide additional tools for eliciting and analysing interaction requirements. Then, the protocols in the protocol model can be identified (from reusable components or designed from scratch) based on the interaction requirements. Together, the goal model, social model and protocol model can then be used to produce the low-level interaction model. This approach clearly extends the original approach taken in ROADMAP, in which protocols are specified in AUML and associated directly with role descriptions.

The ROADMAP models and their interrelationships have undergone various changes (see, for example, Juan *et al.*, (2002, 2003) and Juan and Sterling (2004)). We change these yet again in order to cater for the new above-mentioned models,[5] and the resulting structure of various models within the ROADMAP methodology is shown in Figure 6. The models are grouped into two categories. The environment model and knowledge model contain reusable high-level domain specifications. The protocol model and service model, on the other hand, describe potentially reusable low-level software components. All other models are considered, more or less, *application specific*. The shaded boxes denote those models that form a key part of interaction requirements analysis and interaction design.

Next, we describe our approach to interaction requirements analysis in more detail. In particular, we focus on the elements of the goal model and social model, and then show how these can guide the two design-time models (i.e. protocol model and interaction model).

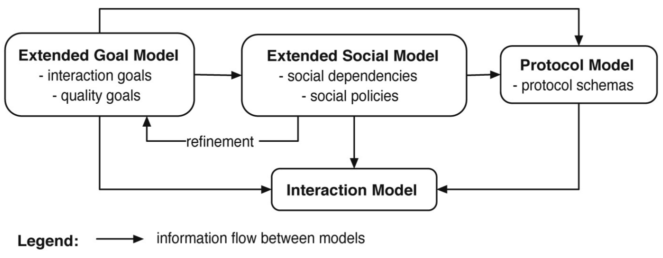


Figure 5 The new design-time interaction-related models

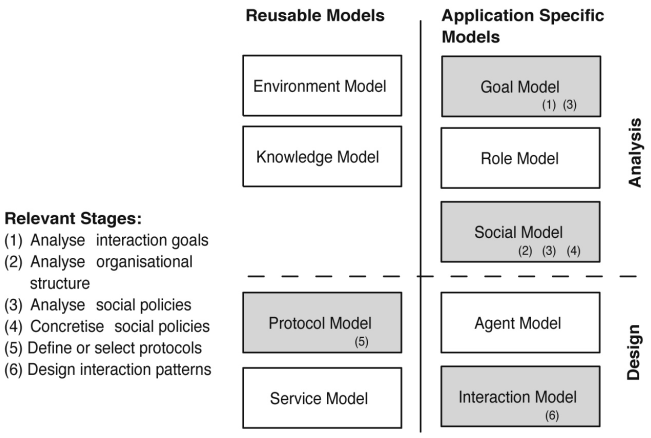

Reusable Models
Application Specific Models
Environment Model
Knowledge Model
Goal Model (1) (3)
Role Model
Social Model (2) (3) (4)
Analysis
Relevant Stages:
(1) Analyse interaction goals
(2) Analyse organisational structure
(3) Analyse social policies
(4) Concretise social policies
(5) Define or select protocols
(6) Design interaction patterns
Protocol Model (5)
Service Model
Agent Model
Interaction Model (6)
Design

Figure 6 The models within the ROADMAP methodology

3.3 Interaction requirements analysis

In this section, we extend the ROADMAP goal model and social model in order to support *interaction requirements analysis*. This process consists of identifying the *interaction goals*, *organisational structure* and *social policies*.

3.3.1 Identify Interaction Goals

The first step in interaction requirements analysis is to identify the objectives of interaction among agents, which we shall refer to as the *interaction goals* (as a specialisation of *functional goals* described earlier). As the name indicates, interaction goals are goals that require, for their achievement, some form of interaction among roles. And any such interaction goal must ultimately contribute towards some of the functional system goals as identified in the goal model (i.e. communication must be purposeful). For example, a functional goal to "place an order" may require communication between a customer role and a vendor role in order to determine the terms of the transaction.

Interaction goals are added to the goal model. For this purpose, we introduce a slightly modified graphical notation (dotted parallelogram) to indicate that a goal is an interaction goal. Figure 7 shows a modified version of Figure 4 with interaction goals. The goal "negotiate deal" directly involves interaction among a vendor and a user. Therefore, it makes more sense to cast it as an interaction goal. The goal "identify alternatives," on the other hand, requires seeking information about products on offer. For this

5. We believe this ability to change is a strength, not a weakness, as it reflects ROADMAP's natural flow of concepts between different stages.

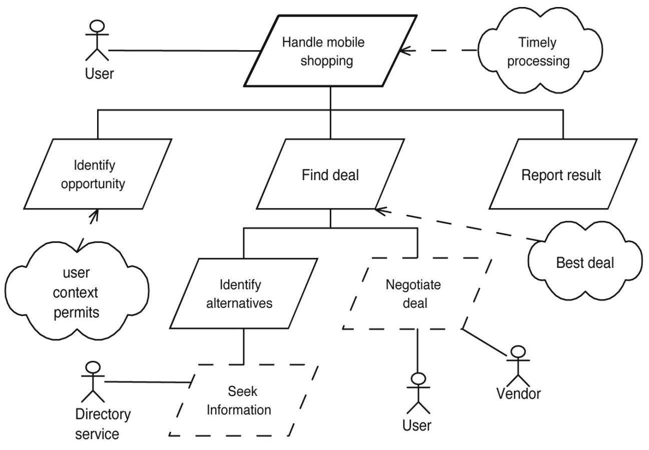


Figure 7 Example goal model with interaction goals

reason, a new interaction goal "seek information" is added to the goal model. Interaction goals themselves might also be decomposed hierarchically. For example, the interaction goal "seek information" may be further decomposed into two goals: "query yellow pages" which requires communication with a role providing a yellow-pages service to find who offers the products needed, and "request products on offer" which requires communication with the vendor role to obtain information about features of specific products. In order to properly exploit interaction analysis, we need to ensure that the new goal model is sufficiently rich to aid subsequent design of interaction protocols. For example, the analyst might be tempted to cast the goal "identify alternatives" as a leaf-level interaction goal. However, this level of specification is not very informative since it does not give clues about the types of interaction protocols that might be needed to achieve this goal. We therefore present the analyst with a set of *primitive interaction goals*, described in Table 1. These interaction goals are sufficiently abstract to provide flexibility while, at the same time, sufficiently concrete to provide useful guidance for the selection of specific interaction protocols at the design stage. These interaction goals are based on a typology of dialogues from argumentation theory (Walton and Krabbe, 1995). Note that this list is neither exhaustive nor disjoint, as some goals may be related or subsumed by one another, which can be done easily through interaction goal decomposition. For example, "negotiation" may involve a form of "persuasion" in which a seller attempts to convince a potential buyer of the quality of a particular product. We believe the list provides a set of potentially useful concepts at the analyst's disposal.

Table 1 Types of primative interaction goals

Interaction Goal	Description	Example Variants
Information exchange	One agent needs information possessed by another	– Information seeking – Synchronisation of knowledge – Verification – Expert consultation
Inquiry	Parties seek an answer to a question that neither of them knows	– Investigation – Examination
Persuasion	One party wishes to change the mental state of another	– Dispute
Deliberate	Parties communicate to reach a joint decision about what to do	– Means-ends discussion – Discussion of goals
Negotiation	Parties with conflicting demands over limited resources attempt to reach agreement over the division of these resources	– Single-attribute bargaining – Multi-party, multi-attribute negotiation

3.3.2 Identify Organisational Structure

The organisational structure describes the relationships and dependencies between roles (Zambonelli *et al.*, 2001) and can therefore provide clues about the types of protocols that might be suitable for their interaction (Mao and Yu, 2004). Hence, specifying the organisational structure is an essential step in the analysis of interaction requirements.

Currently, the ROADMAP social model is simple. Firstly, hierarchies constitute the only form of organisational structure among roles. There is no support (conceptually or notationally) for specifying more complex types of organisational structures. However, the relationship between a vendor and a buyer is clearly non-hierarchical. Therefore, it would be instrumental to have conceptual tools that enable requirements analysis in a way that reflects a variety of social structures.

In our extension, identifying the *organisational structure* takes place by identifying the various relationships based on the existing role model and labelling the resulting structure. Relationships are identified in such a way as to achieve the interaction goals discussed above. Below, we elaborate on how this may be done.

One way is to view relationships in terms of the specific types of *influence* agents can exert on each other and their environments (Ashri *et al.*, 2003a; Panzarasa and Jennings, 2002). Another way to identify relationships is using binary relations (Dignum, 2004). We take the latter approach here since it is more abstract and less committal about the underlying representation of agent mental attitudes and the environment.

Relationships are identified as follows. First, the *roles* that constitute the ROADMAP role model are listed and each is defined through a *role schema* (Juan *et al.*, 2002). Then, a set of *role dependencies* are identified. Essentially, role dependencies correspond to a set of binary relations on the set of roles. For example, the triple <*user*, *directory_service*, *dependency_info*> may be used to denote that the enactor of the *user* role has a dependency on information resources provided by an agent enacting the role of *directory_service*. Role dependencies are not confined to "delegation." They may represent any kind of social relationship. Following we provide a list of typical *types of relationships* that may exist between agents:

- *delegation*: where one role can delegate tasks to another role;
- *control*: where one role has authority over another role, enabling it to change the latter's state;
- *influence*: where one role can influence another role's state (this is weaker than "control," which enables influence over *all* state attributes);
- *dependency*: where one role relies on resources (e.g. actions or information) from another role;
- *collaboration*: where two or more roles rely on each others' resources for the achievement of common goals;
- *authorisation*: where one role requires authorisation from another role in order to execute particular behaviours;

These relationships are extensible. It is possible to define

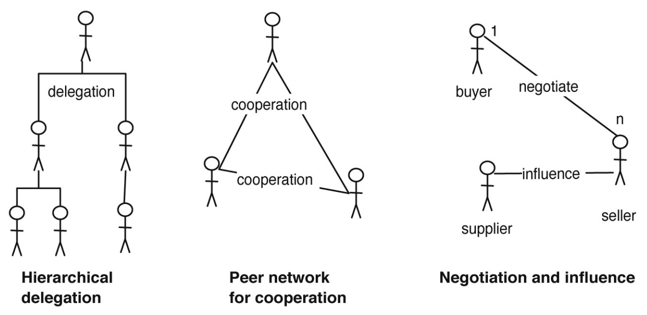


Figure 8 Examples of organisational structures

different types of relationships to the ones mentioned above.

Based on the relationships between roles, the analyst can identify the *organisational structure* of the system. At the most abstract level, there are two main social structures found in the literature, namely:

- *Hierarchy*: where agents are organised into a hierarchy. This structure is suitable for representing relationships such as decompositional dependency, delegation and authority.
- *Peer Network*: where all agents have equal status and interact with one another in a peer-to-peer fashion. This is suitable for representing cooperative teams and competitive markets.

These structures may be specialised into various other types of structures, based on the types of relationships that hold between agents. For example, a market is typically a peer network, since all agents are regarded as equal. A team can be seen either as a peer network (where team members are regarded as equal) or as a hierarchy (where some members have a coordinating function among their subordinates).

In Figure 8 we illustrate the graphical notation for describing different types of structures.[6] The links between roles represents relationships among them, with the annotation describing the relationship type. Note that these relationship types do *not* denote the interaction protocols among these roles. Instead, they denote a more abstract description of their relationship. Later, specific interaction protocols can specialise relationships as appropriate.

While the organisational structure is being identified, the analyst needs to make sure that the structures correspond correctly to the interaction goals, and potentially refining both the interaction goals and organisational structure until they are satisfactory. This iterative process of refinement yields a coherent goal model with clear interaction goals, along with an organisational structure among roles that reflects such a goal model. For example, it would not make sense for an interaction goal that require *negotiation* to be associated with a *control* relationship, since negotiation by definition requires autonomy. In general, what is an appropriate linking between the relationship/organisational structure and the interaction goal model will depend on the application domain.

Note that the organisational structure produced in the analysis stage does not necessarily correspond to that of the design stage. Indeed, the designer may choose more efficient organisational structure. Nevertheless, as we shall see in the next section, the analysis-stage organisational structure provides a means for analysing the social policies (discussed next) that are useful for enriching interaction requirements.

3.3.3. Identify Social Policies

Having defined the organisational structures that *describe* the agent society, the next step is to define the *social policies* that control interaction among various roles. Social policies are constraints on interaction and behaviour. They can represent anything from access rights, to social norms, to obligations, etc. The analyst identifies social policies based on the relationships and dependencies between roles, which are now available from the organisational structure.

In order to integrate these policies with interaction more clearly, they need to be linked to the interaction goals elicited in the goal model. We propose a simple but powerful approach to achieving this. We view social policies as *quality goals over* the interaction goals: they are constraints that *control* or *measure* the achievement of interaction goals. As a result, social policies can be linked to interaction goals in the goal model diagram in a similar way that normal quality goals are added to normal goals.

Figure 9 shows a portion of the model described earlier with three elicited quality goals representing social policies. Below is a textual description of these policies.[7]

- *Pareto efficiency*: This social policy requires that the deal achieved by the "negotiate deal" interaction goal must be Pareto efficient, i.e. there must be no other deal which makes one party better off without making the other party worse off. This policy is a common requirement in frameworks for automated negotiation among self-interested agents (Binmore and Vulkan, 1997).
- *Commitments are binding*: This social policy requires that if agents agree on a deal, they should not be allowed to *decommit* on that deal. In design time, this may translate to a choice of a specific protocol that imposes heavy penalties on decommitment.
- *Timely response*: This quality goal requires that information about the available alternative products or services is

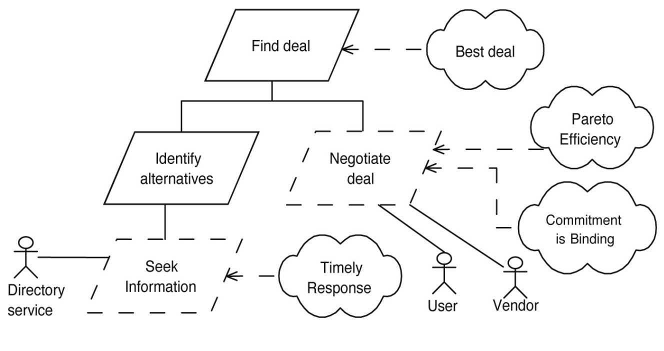


Figure 9 Portion of Figure 7 appended with social policies represented as quality goals over interaction

6. This notation extends the earlier *role model* notation in ROADMAP in which only hierarchies are allowed, and in which links are not labelled.

7. Of course, a more formal description (e.g. using logical axioms) can also be used if needed.

provided in a timely fashion. In the design and implementation stages, this may translate into specific time-out periods by which the response must be provided even though, for example, the list of alternatives is still incomplete.

Other social policies may be specified as quality goals. For example, in an application that requires collaborative interaction and a team structure, one might specify that no team member might leave without notice.

At this stage, developers focus on nominating the social policies and analysing them qualitatively. Therefore, once the social policies have been identified, the developer can put the corresponding quality goals (and other quality goals not related directly to interaction) into a hierarchy. The dependencies and potential conflicts can be modelled using positive or negative contribution relationships (denoted by "+" and "–" signs). The same framework can allow you to represent other influences from the environment that can affect your quality goals. In addition to quality goals, the developer can also add other notions from the environment to the hierarchy to capture their influences. Figure 10 shows an example of such hierarchy. It shows that both goals "timely response" and "commitment is binding" contribute towards achieving a "better deal." However, there is conflict between achieving a better deal and a "timely processing" of the mobile shopping.[8]

Note that the social policies identified at the analysis stage are relatively abstract. The precise manner in which social policies are defined at the design and implementation stages will depend on the specifics of the domain, as we shall see below.

3.4 Moving towards design: the protocol and interaction models

By now, the analysis stage will have produced the complete set of interaction requirements based on the extended goal and social models. These requirements are produced without any commitments about the system design. Yet, these interaction requirements provide very rich models that can aid the design stage. In particular:

- the new goal model shows the main goals that require interaction among agents, described in hierarchies of interaction goals, explicitly stated and linked purposefully to functional goal;
- the organisational structure provides a description of the dependencies between different roles, giving further clues about the kinds of suitable protocols;
- the social policies, described as quality goals over interaction goals, provide a set of criteria that interaction needs to fulfil, and a performance metric that can also be used for validating interaction at run-time;

The bulk of this paper was focused on the analysis stage. For the sake of illustration, however, we now briefly discuss how interaction requirements influence the interaction-related design-time models (namely, the protocol model and interaction model).[9]

3.4.1 Concretise Social Policies

Recall that in the analysis stage, social policies are described abstractly, and not necessarily given precise meaning. For example, a quality goal like "timely response" might have different interpretations when considered in the context of different applications. In the design stage, social policies need to be concretised and given precise meaning. This can be done by nominating measurement functions for quality goals, which can have quantitative meaning.

In order to maintain generality, we do not prescribe a specific method for concretising social policies. They can be expressed, for example, in the form of a quantitative measure of some value, or in the form of logical axioms that must be satisfied by any run of the system. Following is a concretisation of the social policy requiring the Pareto efficiency of negotiation outcomes through a mathematical function:

> Let $\mathcal{O}$ be the set of all possible outcomes and let $U_i(O)$ be the utility that agent i receives if outcome $O \in \mathcal{O}$ was reached. Any negotiation outcome O reached in the system must achieve the following condition: $\nexists\, O' \in \mathcal{O}$: $\exists i \in Agents,\ U_i(O') > U_i(O)$ and $\forall j \in Agents,\ U_j(O') \geq U_j(O)$

Similar definitions might be given for other social policies/quality goals. For example, the performance with respect to the goal "timely processing" might be easily measured using a measurement function based solely on the time logged between the triggered context change (according to the user's location) and the production of offers on the screen. This function gives a range of performances based on, for example, the difference between the time taken and the average expected time.

The relative importance of quality goals can also be quantified. For example, if "timely processing" is three times more important than getting the best deal, then the agent is

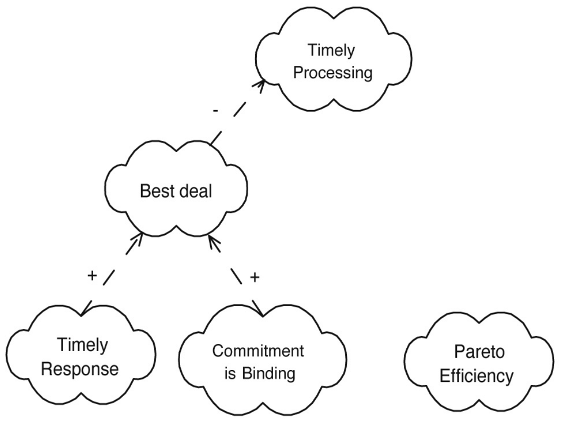


Figure 10 Hierarchy of quality goals (both interaction and non-interaction related)

8. Recall that "timely processing" is a quality goal that relates to the top-level functional goal in Figure 4.

9. Hence, we do not demonstrate all features related to the design stage, such as instantiating agents and defining specific services.

to devote three times the resource on finding *a* deal over obtaining information about all available nearby shops.

Diagrammatically, the designer could annotate the quality goal hierarchy shown in Figure 10 with weights on the arrows (reflecting the strength of the positive of negation contribution between quality goals) and numbers within the clouds (reflecting the degree of importance of that goal).

The concretised policies can serve two purposes. First, they can provide directly implementable specifications of performance measures, which might be handy either for testing or for run-time reasoning about the compliance with social policies. Second, they offer more concrete criteria for selecting and designing interaction protocols.

3.4.2 Identify Protocol Model

ROADMAP has a distinct *protocol model*, which is part of the design stage. The ROADMAP protocol model describes a set of protocols: reusable message sequences that serve specific objectives. Each protocol is given two complementary descriptions:

- An abstract *protocol schema* that describes various aspects of the protocol, including when and for what it can be used;
- A *protocol*, that specifies the precise sequence of messages that constitute the protocol as a message sequence specified in an AUML sequence diagram. This constitutes the runtime realisation of the protocol.

The protocol schema is a table that specifies the following:

- Protocol name;
- Goals achieved (including soft goals);
- Initiating roles: roles that can initiate the protocol;
- Responding roles: role in charge of responding to the protocol;
- Sub-protocols.

The idea of sub-protocols enables capturing more complex protocols in terms of simpler ones. For example, a *purchasing protocol* may consist of an *information-seeking protocol* for finding possible items to purchase, and a *negotiation protocol for* reaching agreement on the terms of transaction for a particular item. Figure 11 shows an instantiated protocol schema for purchasing a service.

Protocol Name: Purchase-Service
Goal: Contract out another agent to perform a service
Sub-Protocols: Request-Service-List; Negotiate-Price;
Initiating Roles : Service-Requester
Responding Roles : Service-Supplier

Figure 11 Protocol schema for purchasing a service

The protocol model can be obtained in two ways: it can be built from scratch;[10] or it can be based on a selected set of protocols from a repertoire of reusable protocols. In both cases, the design or selection of protocols must take into account the interaction goals, social policies and organisational structure.

The first requirement a designed/selected protocol must fulfil is that the protocol's goals must be correspond directly to a (bottom-level) interaction goal. For example, in order to achieve the interaction goal "negotiate deal" shown above, the protocol chosen must be a *negotiation* protocol.

Secondly, the designed/selected protocol must be consistent with the organisational structure. For example, a negotiation protocol does not normally make sense between two roles amongst which there is a strong authority and influence relationship. It is more natural for negotiation to occur among peers since negotiation departs, by definition, from a situation of conflict that cannot be resolved through unilateral authoritative decisions.

Finally, the designed/selected protocol must be consistent with the social policies. This is where the specific properties of the protocol become an important design factor, as the designer relates them to the concretised social policies. In particular, this aspect of design demonstrates the benefit of integrating social concepts more closely with agent interaction. In our example, the negotiation protocol designed/selected must ensure Pareto efficiency while providing a timely response. Not all negotiation protocols ensure this requirement is fulfilled, but some do (Kraus, 2001; Rosenschein and Zlotkin, 1994). Moreover, to ensure the deal reached is the "best deal," the user agent needs to negotiate with all available vendors in the nearby area. This means that a one-to-many negotiation protocol (such as an auction or concurrent bilateral negotiation protocol (Rrahwan *et al.*, 2002) or an auction protocol (Wurman, 1999) is more suitable than a one-to-one protocol (such as the bilateral negotiation protocol of Faratin (2000).

3.4.3 Identify Interaction Model

Having defined or selected a set of suitable protocols that meet the interaction and social requirements, the designer can now instantiate the protocol model in order to identify specific patterns of interaction, resulting in the *interaction model*. We only provide brief clues as to how this may be done since detailed design is not the main focus of this paper.

Since the interaction model is part of the (detailed) design stage, it is built in conjunction with the *agent model* and *service model*. The agent model includes specific agent classes, instantiated based on role schemas from the role model. Similarly, the service model includes specific services and activities agents need to perform. Specific interaction protocols are instantiated in order to facilitate the interaction among agents. In doing so, the interaction model brings social and goal modelling of interaction to design-time constructs.

4. DISCUSSION AND CONCLUSION

In this paper we presented the idea of creating more "hooks" in AOSE methodologies to assist integration to other research effort on agent interaction frameworks. Implementing this idea appropriately allows rich domain-specific interaction frameworks outside AOSE to be reused seamlessly, to serve the needs of specialised agent application domains.

We outlined some initial guidelines to improve AOSE methodologies. The central theme is to avoid unnecessary or

10. For example, using a run-time protocol specification language such as AUML (Bauer *et al*, 2001), auction parametrisation rules (Wurman *et al.*, 2001), dialogue-games (Mcburney and Parsons, 2003), commitment machines (Yolum and Singh, 2001), or Flores and Kremer's framework based on the negotiation of social commitments (Flores and Kremer, 2004).

premature commitments to conceptual, representational and process restrictions in social modelling, or at least delay the commitments if possible. By staying high-level and generic in social modelling practices, external interaction frameworks with their distinctive high-level issues can be incorporated and accommodated in AOSE methodologies.

We illustrated this idea by extending ROADMAP with goal modelling, and linking the goals to social dependencies and policies via interaction goals. The interaction goals are further refined to interaction protocols. The goal models are generic in the sense that no explicit logical relationships are required between sub-goals. We believe the goal modelling techniques are pitched at the right level of abstraction and allow developer to focus on the essence of interaction without going into technical details too early. Quality goals, which also represent social policies, are open to iterative refinement and their semantics are captured in modular measurement functions. The interaction protocols are connected to the interaction goals through smooth transition from analysis to design.

The various entities described in our approach act as generic "hooks" around which interaction frameworks can be integrated and their distinct high-level issues represented and clarified. We consider this approach a successful first step towards integrating AOSE methodologies with agent interaction research outside AOSE.

Our future work includes extending the same approach to integrate broader aspects of social modelling into ROADMAP, such as organisational changes, learning in organisations and organisational knowledge management. Furthermore, the relationships between ROADMAP entities are inherently dynamic and can change at runtime. However, further work is still needed to clarify the details for integrating dynamic relationships in ROADMAP to other dynamic relationship formation frameworks such as the runtime negotiation of *social contracts* described in Dignum (2004). Another area of future work is the integration of our approach with more elaborate requirements analysis frameworks (e.g. Donzelli and Bresciani, 2004) and with other AOSE methodologies such as Tropos.

It is worth noting that while preparing the final version of this paper after acceptance, we learned of a new methodology named *Hermes* (Cheong and Winikoff, 2005), which provides a goal-oriented approach to *designing* interaction protocols and a process for mapping design artefacts to an executable implementation. Indeed, Hermes also uses the notion of *interaction goal hierarchy* as a starting point, and shows how such specification can be turned into run-time protocol implementation. A promising direction of research is the integration between our approach to goal-oriented interaction *requirements analysis,* and the Hermes approach to goal-oriented interaction *design*.

ACKNOWLEDGEMENTS

The authors are grateful to the anonymous reviewers for providing insightful feedback, and particularly to reviewer 2 for bringing to our attention the recent methodology *Hermes* (Cheong and Winikoff, 2005).

During part of this work, Iyad Rahwan was supported by an Australian Research Council's Linkage Grant (ARC LP0348797) and the Special Research Centre on Perceptive and Intelligent Machines in Complex Environments (PIMCE). The first author is grateful to Ronald Ashri for discussions regarding agent relationships, and to Fernando Koch for providing the scenario and Figure 1.

REFERENCES

Ashri, R., Luck, M. and d'Inverno, M. (2003) On identifying and managing relationships in multi-agent systems. In: G. Gottlob and T. Walsh, editors, *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 743–748, San Francisco CA, Morgan Kaufmann.

Bauer, B., Müller, J. and Odell, J. (2001) Agent UML: A formalism for specifying multiagent interaction. In: P. Ciancarini and M. Wooldridge, editors, *Agent-Oriented Software Engineering, volume 1957 of Lecture Notes in Computer Science*, pp. 91–103. Springer Verlag, Berlin.

Binmore, K. and Vulkan, N. (1997) Applying game theory to automated negotiation. *Proceedings of the DIMACS workshop on economics, game theory and the Internet*, New Brunswick NJ, USA, April. Rutgers University.

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

Cernuzzi, L., Juan, T., Sterling, L. and Zambonelli, F. (2004) The Gaia methodology: Basic concepts and extensions. In: F. Bergenti, M. P. Gleizes, and F. Zambonelli, editors, *Methodologies and Software Engineering for Agent Systems*. Kluwer Academic.

Cheong, C. and Winikoff, M. (2005) Hermes: Designing goal-oriented agent interactions. In: J. M¨uller and F. Zambonelli, editors, *Proceedings of the 6th International Workshop on Agent-Oriented Software Engineering (AOSE)*, Utrecht.

da Silva, V. T. and Lucena, C. (2001) 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.

da Silva, V. T., Garcia, A., Brandòao, A., Chavez, C., Lucena, C. and Alencar, P. (2003) Taming agents and objects in software engineering. In: A. Garcia, C. Lucena, F. Zambonelli, A. Omicini, and J. Castro, editors, *Software Engineering for Large-Scale Multi-Agent Systems*, volume 2603 of *Lecture Notes in Computer Science,* pp. 1–26. Springer Verlag, Berlin.

Dignum, V. (2004) A model for organizational interaction: based on agents, founded in logic. *PhD thesis*, Institute of Information and Computing Sciences, Utrecht University, The Netherlands.

Donzelli, P. and Bresciani, P. (2004) Improving requirements engineering by quality modelling: a quality-based requirements engineering framework. *Journal of Research and Practice in Information Technology*, 36(4): 277–294.

Faratin, P. (2000) Automated Service Negotiation Between Autonomous Computational Agents. *PhD thesis,* University of London, Queen Mary and Westfield College, Department of Electronic Engineering.

Ferber, J., Gutknecht, O. and Michel, F. (2003) From agents to organizations: An organizational view of multi-agent systems. In: P. Giorgini, J. P. Müller, and J. Odell, editors, Agent-Oriented Software Engineering IV, volume 2935 of *Lecture Notes in Computer Science*, pp. 214–230. Springer-Verlag, Berlin.

Flores, R. A. and Kremer, R. C. (2004) A principled modular approach to construct flexible conversation protocols. In: A. Y. Taw k and S. D. Goodwin, editors, *Advances in Artificial Intelligence, 17th Conference of the Canadian Society for Computational Studies of Intelligence*, Canadian AI 2004, London, Ontario, Canada, May 17-19, 2004, number 3060 in *Lecture Notes in Computer Science,* pp. 1–15. Springer Verlag, Berlin.

Grosz, S., Kraus, S., Talman, B., Stossel, B. and Havlin, M. (2004) The influence of social dependencies on decision-making: Initial investigations with a new game. *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems – Volume 2 (AAMAS 2004)*, pp. 294–301, New York, NY. IEEE Computer Society.

Heinze, C., Papasimeon, M. and Goss, S. (2000) Specifying agent behaviour with use cases. *Proceedings of the Pacific Rim Workshop on Multi-Agents.*

Jennings, N. R. (2001) An agent-based approach for building complex software systems. *Communications of the ACM*, 44(4): 35–41, 2001.

Juan, T. and Sterling, L. (2004) Achieving dynamic interfaces with agent concepts. In: N. R. Jennings, C. Sierra, L. Sonenberg, and M. Tambe, editors, *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 688–695, Washington DC.

Juan, T., Pearce, A. and Sterling, L. (2002) ROADMAP: Extending the Gaia methodology for complex open systems. In: C. Castelfranchi and W. L. Johnson, editors, *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2002)*, Bologna, Italy, pp. 3–10, New York City, NY.

Juan, T., Sterling, L., Martelli, M. and Mascardi, V. (2003) Customizing aose methodologies by reusing aose features. *Proceedings of the second international joint conference on Autonomous agents and multiagent systems (AAMAS-2003)*, pp. 113–120.

Kraus, S. (2001) *Strategic Negotiation in Multi-Agent Environments.* MIT Press, Cambridge, MA.

Kuan, P. P., Karunasekera, S. and Sterling, L. (2005) Improving goal and role oriented analysis for agent based systems. *Proceedings of the Australian Software Engineering Conference (ASWEC)*, pp. 40–47.

Mao, X. and Yu, E. (2005) Organizational and social concepts in agent oriented software engineering. In: J. Odell, P. Giorgini, and J. P. Müller, editors, *Agent-Oriented Software Engineering V: 5th International Workshop, AOSE 2004*, New York, NY. Volume 3382 of *Lecture Notes in Computer Science*, pp. 1–15. Springer Verlag.

Maudet, N. and Chaib-draa, B. (2003) Commitment-based and dialogue-game based protocols – new trends in agent communication language. *Knowledge Engineering Review*, 17 (2):157–179.

McBurney, P. and Parsons, S. (2003) Dialogue game protocols. In: M.-P. Huget, editor, *Communication in Multiagent Systems*, volume 2650 of *Lecture Notes in Computer Science,* pp. 269–283. Springer Verlag.

Padgham, L. and Winikoff, M. (2004) *Developing Intelligent Agent Systems: A Practical Guid*e. John Wiley.

Panzarasa, P. and Jennings, N. R. (2002) *Social influence, negotiation and cognition. Simulation Modelling Practice and Theory*, 10(5–7): 417–453.

Perugini, D., Lambert, D., Sterling, L. and Pearce, A. (2004) Provisional agreement protocol for global transportation scheduling. *Proceedings of the Workshop on Agents in Traffic and Transportation, held in conjunction with the International Conference on Autonomous Agents and Multi Agent Systems (AAMAS)*, New York, NY.

Rahwan, I., Kowalczyk, R. and Pham, H. H. (2002) Intelligent agents for automated one-to-many e-commerce negotiation. In: M. Oudshoorn, editor, *Proceedings of the 25th Australasian conference on Computer science*, pp. 197–204. Australian Computer Society Press.

Rosenschein, J. and Zlotkin, G. (1994) *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers.* MIT Press, Cambridge, MA.

Sandhol, T. (2002) eMediator: A next generation electronic commerce server. *Computational Intelligence,* Special issue on Agent Technology for Electronic Commerce, 18 (4): 656–676.

Soon, S., Pearce, A. and Noble, M. (2004) Adaptive teamwork coordination using graph matching over hierarchical intentional structures. *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems* – Volume 1 (AAMAS 2004), pp. 294–301, New York, NY.

Walton, D. N. and Krabbe, E. C. W. (1995) *Commitment in Dialogue: Basic Concepts of Interpersonal Reasoning.* SUNY Press, Albany, NY.

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

Wooldridge, M. J. (2002) *An Introduction to MultiAgent Systems.* John Wiley, Chichester.

Wurman, P. R. (1999) Market Structure and Multidimensional Auction Design for Computational Economies. *PhD thesis,* University of Michigan, August.

Wurman, P. R., Wellman, M. P. and Walsh, W. E. (2001) A parametrization of the auction design space. *Games and Economic Behavior*, 35(1–2): 304–338.

Yolum, P. and Singh, M. P. (2001) Commitment machines. In: J.-J. C. Meyer and M. Tambe, editors, *Intelligent Agents VIII, 8th International Workshop, ATAL 2001* Seattle, WA, volume 2333 of *Lecture Notes in Computer Science*, pp. 235–247. Springer Verlag.

Yu, E. (1995) Modelling Strategic Relationships for Process Reengineering. *PhD thesis*, University of Toronto, Toronto, Canada.

Zambonelli, F. and Parunak, H. V. D. (2003) Towards a paradigm change in computer science and software engineering: a synthesis. *The Knowledge Engineering Review*, 18(4): 329–342.

Zambonelli, F., Jennings, N. R. and Wooldridge, M. (2001) Organisational rules as an abstraction for the analysis and design of multi-agent systems. *Software Engineering and Knowledge Engineering*, 11(3): 303–328.

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