Designing Agent-Oriented Systems by Analysing Agent Interactions

Simon Miles Mike Joy Michael Luck

Department of Computer Science, University of Warwick
Coventry, CV4 7AL, United Kingdom
smiles@dcs.warwick.ac.uk

Abstract. We propose a preliminary methodology for agent-oriented software engineering based on the idea of *agent interaction analysis*. This approach uses interactions between undetermined agents as the primary component of analysis and design. Agents as a basis for software engineering are useful because they provide a powerful and intuitive abstraction which can increase the comprehensiblity of a complex design. The paper describes a process by which the designer can derive the interactions that can occur in a system satisfying the given requirements and use them to design the structure of an agent-based system, including the identification of the agents themselves. We suggest that this approach has the flexibility necessary to provide agent-oriented designs for open and complex applications, and has value for future maintenance and extension of these systems.

1 Introduction

The agent paradigm has, over recent years, given rise to a large amount of research on the internal structure of agents as general problem-solvers capable of effective intelligent behaviour in dynamic environments. This concentration of work on the development of *agent architectures* has been just one side of the many aspects illuminated by the agent metaphor. More recently, however, agents have been used as an abstraction for general software engineering. This paper explores the rôle of agent-oriented methods for just such a purpose, and introduces a preliminary methodology that may be used as a basis for designing agent-based systems.

The full potential of agent-based systems in solving problems in complex domains depends upon the systems themselves, and the designs from which they are constructed, being tailored to the conditions that vary across the domain. They also need to be sufficiently adaptable and fault tolerant to cope with changes in the domain that arise due to maintenance, extension and so on [12, 24]. To achieve this, a methodology that produces agent-based designs must be flexible enough to describe these varying requirements and their interconnections. In particular, the significant area of open systems should be addressed [26].

An abstraction of the social aspects of an agent can be given as a system *rôle*, and this concept is used in many of the emerging agent-oriented methodologies [13, 17, 27]. Rôles are useful as they provide a way to describe a multi-agent system as analogous to an organisation without placing heavy restrictions on the behaviour of concrete agents at

runtime. The usefulness of this approach can be extended by identifying organisational patterns using rôle modeling [16] which can then be used repeatedly as the basis for new systems.

However, the choice of organisation or rôle models will effect the performance of the system [7, 23, 29]. The applicability of the organisation to the domain and the effects of changes in connected systems over time must be accounted for. One way of doing this is to analyse and tailor the behaviour of groups of agents to the domain either by providing *coordination media* to societies of agents [23] or *organisational rules* to the system which influence the dynamic form of the organisation [29]. While these approaches provide appropriately constrained flexibility to the organisations, the constraints and infrastructures which are appropriate to the particular domain must be identified.

We explore a different approach, named *agent interaction analysis*, based on using the interactions between agents as a primary component of analysis. As with the MaSE methodology [7, 25], this approach does not constrain the organisation or rôle model until late in the analysis, after domain-specific requirements have been identified. *Agent interaction analysis*, by translating requirements into agent interactions rather than rôles, also allows for the system behaviour to be flexibly distributed between the designed system and connected systems. The product of the steps described in this paper is an organisational structure and justifications for that structure. This allows for other rôle-based methodologies to continue the design process from that organisation to an implementation.

The next section provides detail on some of the terms used in, and sets the scene for, the rest of this paper. Section 3 provides an overview of *agent interaction analysis*. Later sections then look at each step of this process in turn. Section 4 examines the initial analysis of the system requirements and explains what information is necessary for *agent interaction analysis*. Next, Section 5 describes how the system aims can be decomposed into a structure which represents the system functionality in a modularised way. Section 6 describes the concept of an *interaction* between agents and details the part it plays in the design process. The next stage, described in Section 7, derives useful design information from the interactions and system requirements. Finally, Section 8 shows how the design is reduced to a form which is implementable and more comprehensible as a whole. The final two sections describe how the process is useful for maintenance and extension, and then mention the other work done on this project.

2 Agents and Goals

The term *agent* is used in a variety of related ways in the literature surrounding agent-based systems[10]. On the one hand, it is used to describe software artefacts that satisfy certain architectural requirements in order to achieve particular functionality. On the other, as in this paper, agents are used as a software engineering abstraction that enables complex software to be decomposed into a collection of sophisticated interacting components (that may also share the qualities of the previous view). In developing a methodology for agent-oriented design, we aim for it to be sufficiently flexible and general to apply to designs using a wide range of entities, but there are certain proper-

ties that we assume system components will have or can be thought of as having. These assumptions enable the designer to develop systems with those properties implicit, thus making the rest of the design easier to comprehend.

The properties are fairly standard (e.g., see [15]): the agents are considered to be *autonomous*, *decision making*, *social*, *flexible* and *reactive* entities. *Agent interaction analysis* is concerned with the social aspects of agents and, therefore, doesn't directly address the analysis of reaction to a non-autonomous environment in the system design.

These properties describe the broad expectations we have in all the agents in systems resulting from our design approach. We do not require any further particular architectural implementation constraints; indeed it should not impose any further constraints since we cannot know in advance the form agents will take in an open system.

In this paper we describe the direction agents have in their autonomy in terms of declarative *goals* possessed by the agents. To increase the flexibility and range of systems which the methodology can produce, we also allow that goals can have a variety of solutions with different quantative *worths* [6].

3 Overview of Agent Interaction Analysis

The primary building blocks used by our approach are *interactions* between agents. The approach of *agent interaction analysis* is as follows.

- We assume the system contains an arbitrary number of flexible agents with the properties described in the next section (and no other details known as yet).
- We interpret the system requirements as goals and preferences for their achievement.
- We decompose the system goals into independent hierarchies of goals, comparable to hierarchical plans, and actions which achieve the lowest level goals.
- We treat the successful *engagement* of one or more agents to pursue a goal as an interaction.
- From the particular requirements of each interaction and the system preferences we derive the forms of architecture and particular coordination mechanism to make agents taking part in the interaction behave in a way that fits the preferences well.

The overall structure of the agent interaction analysis process is shown in Figure 1, in which the primary entities are written in larger letters. The arrows indicate the transformations between entities which make up the design process.

Requirements The requirements must be analysed to extract the structure of a multi-agent system which will usefully implement them.

Goals As described above, goals describe desired states of the system.

Preferences Preferences denote an encompassing concept that includes different constraints relating to the other significant information in the requirements aside from goals. Particular types of preferences are recognised in *agent interaction analysis*, such as quality of goals, but they can also represent other domain restrictions and desired system properties that require taking account of.

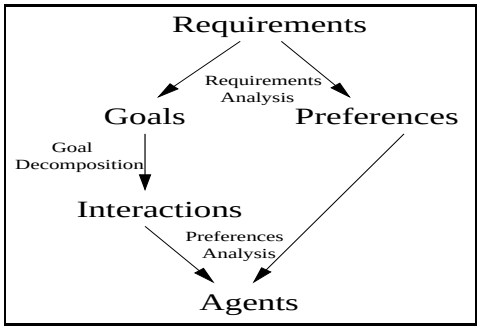

Fig. 1. The transformations involved in agent interaction analysis

Interactions Agreements on coordination between agents (or *place-holders* for where agents will exist) are known as *interactions*. Section 6 expands further on the meaning of interactions.

Agents Agents, or rôles which agents can take, are derived from the other entities and implemented.

While this process is not a fully defined or mature methodology, it does describe a part of such a methodology with beneficial properties for agent-oriented design. The remainder of this paper describes the details of how such a process works and contains arguments for why it is useful.

4 Agent Interaction Analysis Requirements

As with every development methodology, an early stage of the design process is to derive the needed information, in a useful form, from the given requirements. This stage is the first of relevance to *agent interaction analysis* and, in this case, what we want are *goals* and *preferences*. The goals may be continuous over the system lifetime or dependent on context, e.g., at regular intervals or invoked by a user. The preferences may take different forms determining, for instance, the measures of success for goals or restrictions on resources. Examples of preferences are given below.

Now, traditional *requirements analysis* attempts to decompose systems into objects, functions and states so as to understand the problem [5]. *Agent interaction analysis*, however, is concerned with design, i.e., reaching an implementable solution. Therefore, the technique and end product of the artefacts derived from the requirements may be very different. Nevertheless, regardless of the way in which *agent interaction analysis* produces a useful decomposition, the requirements analysis should at least indicate the highest level of system goals, the contexts in which they can occur, and the preferences attached to them. In terms of requirements analysis, this means clarification of definite states which functions should achieve. Importantly, because of the attention to the high-level and the goal-orientation, this stage of the design process can usefully borrow from requirements analysis ideas.

A range of requirements analysis techniques are available (with several being described in [5], and one aimed at agent-based systems described in [1]). It is interesting to note that these techniques derive relations between agents both internal and external to the system being designed and, because we are concerned primarily with interactions, there is no explicit distinction between these different classes of agent.

As an example of the products of the analysis, the following simplistic translations could be made.

- "When a user clicks on a button in the graphical interface the document being edited should be saved" translates to an interaction in which agents cooperate on the goal to save the document. It also states a fact about the interface, i.e., that pressing the button is one way of causing this interaction. This is the *context-based* appearance of goals.
- "The temperature should not go above 100 degrees" places a restriction on the system and describes a system goal to keep the temperature below 100. The goal is considered to be part of an interaction (as they all are) but is likely to be implemented by a single agent repeatedly cooperating with itself. This concept of an agent cooperating with itself reduces to the internal processing of the agent and, while a valuable way to provide an overarching framework for design, is only cooperation in the broadest sense.
  The 100 degrees part describes the measure of quality for the goal, which is high when the temperature is below 100 degrees and low when it is above. A different application might have a wider range of qualities, e.g., "the cooler the better." This is is the *continuous* appearance of goals.
- Aside from context-based and continuous appearance of goals, another possibility is *regular* appearance, e.g., "Back up the system at midnight every night." This sort of interaction is described by the (very simple) cooperation between a clock and another agent.
- "The transferred file should contain the least amount of corruption as possible given other preferences, such as sufficient rapidity" describes a set of preferences concerning the result of a process, which are the quality measures of the goal to transfer the file.

5 Goal Decomposition

Once the system goals are known, we need to examine how the system could be broken up to enable the designer to identify agents and their properties, as well as allowing a division of labour in designing the system [18]. Decomposing goals means finding states that allow the goal state to be achieved more easily or finding independent parts of the goal state such that when the parts are achieved the whole is achieved.

An example of a graphically represented goal decomposition is given in Figure 2. In this diagram, *Goal 1* has been decomposed into three independent goals which, if achieved in some specified order, will cause the achievement of *Goal 1*. Similarly, *Goal 4* has also been decomposed. A real world example which would follow this structure is if the goals were representing the following environment states. *Goal 1* is the state in which a table has been moved to another location, *Goal 2* is the state where one end

has been lifted in the air, *Goal 3* is the state where the other end has been lifted, *Goal 4* is the state where the lifted table has been moved to another location, *Goal 5* is the state where one end has been moved along, and *Goal 6* is the state where the other end has been moved. Some of these decomposed goals must be done in parallel, some sequentially.

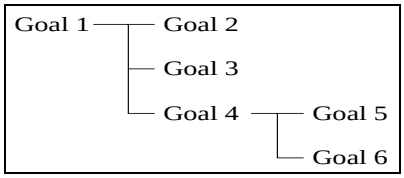

Fig. 2. An example of goal decomposition

This task of decomposition is similar to the production of hierarchical plans. Goals reduce to a series, or some other combination, of subgoals. Unlike a hierarchical plan, this structure is not necessarily possessed by an individual or a group of agents, and it does not require the decomposed system goal to be achieved in this way when the system is running. At this stage, the decomposition only allows us to state that the enactment of that decomposition *could* occur in the system if circumstances were correct. The weakness of the implications of the decompositions is essential to retain the system and agent flexibility as far as possible, with the tightening of the design coming at later stages. Importantly, multiple decompositions can be developed for the same goal. Now, although it may appear that the decomposition produces a full design itself, this may lose the benefits of the agent-oriented approach. In particular, if the fully decomposed subgoals can be translated into actions, then that series of actions achieves the system goal. However, without the agent-based system, or another form of system, to execute them in a flexible manner, we could lose the benefits of such systems such as loss of robustness, decisions on the use of one connecting system or another, reactions to the state of the domain and scope for extension. It could also leave the system as a whole harder to comprehend.

In a full object-oriented design, for instance, we would construct entities and describe message passing between them. What is needed in such a design and not in goal decomposition is the specification of both how to achieve aims at a high level and the mechanisms to allow those functions to happen (and happen robustly in a well designed system). In goal decomposition we have ignored all description of underlying structure and decision-making, leaving that to the agent-based system.

6 Interactions

At this stage of the process we have:

– an assumed multi-agent system with an arbitrary number of minimally defined agents;

– a set of goal hierarchies suggesting ways to decompose the goals in the system requirements so as to hopefully be able to implement them; and
– a set of preferences or restrictions derived from the requirements.

The next step is to combine the first two of the above, i.e., state how a multi-agent system would enact the goal decompositions. In doing this, it is important to retain the whole tree of subgoals from the decompositions. This allows agents to share sections of a problem so that the system remains efficient, robust and conforms to other system preferences. It also allows for high level goals to be delegated to externally connected systems where we may not need to be concerned with how they are further decomposed or otherwise achieved. The notion of goal decomposition in this way means that we wish goals to be passed around the multi-agent system involving cooperation between agents. The agreed cooperation between agents on a goal is called an *interaction*. However, this cooperation is not between particular agents as they have not yet been defined, regardless, we do not necessarily know which agents will cooperate to achieve a goal. Therefore, to define interactions without agents we use *place-holders* for agents. Figure 3 illustrates the structure of an interaction with three rôles for cooperating agents to take in achieving the labeling goal.

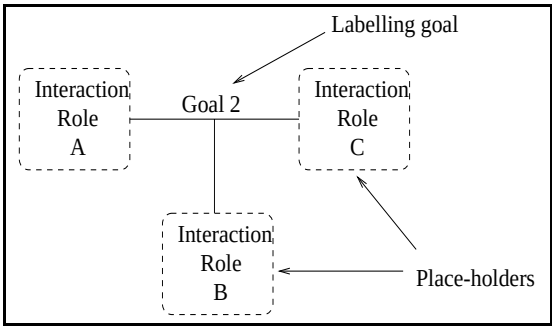

Fig. 3. The structure of an example interaction

Each interaction is labeled by a goal. There is more complexity to interaction than message passing between objects but it has a similar eventual effect in that control is transferred between system components in a limited and meaningful way. Before cooperation can take place for a goal, one of the agents involved must possess that goal, so that interactions have *originators* of the labeling goal which is one of the place-holders involved. The other place-holding agents in an interaction also have rôles within that interaction such as being delegated goals, acting in parallel, monitoring the appropriate execution of the goal, and so on, with the exact roles depending on the goal. Part of the flexibility achieved by not specifying the agents involved in an interaction can mean that the agents filling the interaction rôles may be inside or outside of the designed system or may, in fact, be the same agent. It is important to note that, due to the fact that agents may refuse to cooperate, an interaction only illustrates the successful case where

the originator has eventually found a cooperator. It could also represent a chain of delegations where only the last agent in the chain knows how to continue its decomposition or completion [20].

At this point we can start to refine the design of the system. We have a possible, but not mandatory or inflexible, way of decomposing system goals so as to achieve them. The interactions specify what must be possible and the preferences specify what it means for them to be achieved well. From these we can derive a form of multi-agent system which will behave in a way that is both flexible and consistent with requirements.

The other important aim of a methodology is to make the design comprehensible at the implementation level. As a result of the potential division into several pieces developed by separate people and consisting of complex hierarchies, at this stage the design may be hard to comprehend as a whole or implement efficiently. This will be addressed later: for now the different areas of functionality described by goals can be understood and developed separately.

7 Preference Analysis

Continuing the process of agent interaction analysis, we now develop the design of agents, or system rôles which agents can hold, from the interactions we would like to be possible. There are several aspects of these interactions we can take into account.

1. Which agents should be developed from the analysis of an interaction?
2. How can the agents coordinate so as to best achieve the goal?
3. How can we ensure a system goal *will* always be achieved?
4. How can preferences be accounted for in the design?

Each of these is discussed in the subsections below.

7.1 Classes of Agents

The first aspect above concerns the set of agents to which the results of the other three aspects are to be applied; different sets of agents may be applicable to different rôles in an interaction. The sets can be restricted by a number of factors as follows.

Comprehensibility Restricting the set of agents designed to take part in an interaction could aid the understanding of the resulting system.

Preferences on available resources Restricting or expanding the sets of agents may be desirable to reduce complexity or the number of agents in the system.

Preferences on the monitoring of conflicting goals The best way to resolve conflicts between goals may be to have single agents with multiple goals able to decide actions based on their combination of goals. This requires overlapping sets between interactions.

Preferences on limiting function availability For efficiency or security reasons the designer may wish to limit the number of agents with the ability to achieve a goal.

Restrictions from external systems Only some external agents may be able to complete the goal.

Preferences on state of agents The state of an agent may restrict which classes it can belong to. For example, local mobile agents may be suitable for an interaction while non-local ones aren't.

Likelihood of high load If the goal is going to be required to be achieved repeatedly or requires a lot of resources to complete, the designer may want to expand the set of agents available to distribute the task more widely.

We can also distinguish between those agents designed to achieve success in a particular interaction and those which *may* be able to do so. In some systems it may be preferable for most or all agents to be able to adopt a goal even if they will not achieve it as effectively or efficiently as those specifically designed to. This may be the case when these latter agents already have other goals to achieve so that the system is better balanced by others adopting the goal. Similarly, a further distinction can be made between agents designed for an interaction and those *compelled* to adopt such goals. This is discussed further below in Section 7.3.

7.2 Assurance Analysis

Much work has gone into the development of coordination mechanisms. In the broadest sense we can include commitments [14], trust [21], contracts, system roles [3, 28], social laws [11] and so on. Different mechanisms are suitable for different preferences on coordination; some are more rapid while others are more secure or robust. Different goals in a system may be best served by different mechanisms.

In *agent interaction analysis* we need to find suitable mechanisms for agents without knowing in all cases which agents are taking part. For this reason, we need a set of design tools for examining, from the point of view of a single agent in an interaction, how best to fulfill the goal and system preferences. We call this *assurance analysis* as it examines how an agent can measure or alter the likelihood of success for a particular interaction, e.g., if A delegates to B what *assurance* does A have that B will achieve the goal as A would prefer it to be achieved?

Due to space limitations we will not discuss assurance analysis further here, except to list below some of the things that it should address.

- The *priority* at which the cooperators place a goal, relative to other goals.
- The methods used to complete goals and their side-effects.
- Use of resources, allocated for one goal, in pursuit of other goals.
- Errors in communication.
- Quality of the solution of a goal (as measured by preferences).

7.3 Satisfaction

There is one preference which will generally be implicit in the requirements: that the system goals always *will* be achieved when they should. Of course, the designer can't decide how external systems will behave but it should be possible, in a lot of cases, to restrict the design so that the system will always achieve the goals, even if to minimal standards in certain circumstances.

If a goal is possessed by an agent it will act towards it at some stage because it is pro-active and continuous. However, if an agent cannot complete a goal by itself it will need to cooperate. Unfortunately, a request for cooperation may be rejected by all possible cooperators so that the goal is never achieved.

To address this, the designer may want to make it mandatory for particular agents to take on particular goals if no other agent is doing so. This is an extreme form of assurance where the minimum standards are guaranteed. Such a restriction is comparable to *services* in other methodologies [9, 19, 26] but we note that, while these agents must take on the goals if required, they might not be the agents that actually do so at run-time. It is also one reason why agents should often be able to possess and deal with multiple goals.

7.4 Preferences

While it appears that a vast amount of analysis would be done for the system because of all the interactions within it, many interactions will have a similar form and suggest a certain structure for the agents and overall system [2]. For example, certain things can be assumed about an interaction which we restrict to occurring within the designed system, such as honest communication of information; or a goal which requires rapidity over all else, which may require low communication costs.

8 Design Collation

Agent interaction analysis can provide a justified structure for a flexible multi-agent system. The result of the above design methods is a set of designs leading from the goals and preferences of requirements to the desirable properties of sets of agents involved in particular interactions. The final stage is to reach a design which is coherent and comprehensible enough to implement.

The preference analysis produces descriptions of agents with restrictions on their architecture and the system in which they exist, possibly including physical location in a highly distributed system. This collection of agents needs to be reduced to a set that is useful to implement. We identify some criteria for situations in which agents or systems may be most suitable for being merged, as follows.

- Where the assurance analysis suggests that two agents in an interaction must have such high assurance that it would be best to make them the same agent, the interaction will become an internal processing of the goal (perhaps implicit).
- Where functionality of, and requirements on, agents are similar or the same.
- Where assurance analysis suggests similar coordination mechanisms for agents in different interactions, it may be suitable to merge those place-holders. For instance, a single agent may act as a gateway to less reliable connected systems to provide especially secure coordination.
- Where the low level goals place-holders are dealing with require similar actions to be performed.

After this iterative collation we should be left with a multi-agent system which is implementable and justified in its design. The latter property refers to the fact that the designer can indicate where the system has been tailored to the likely demand the application will place on it to improve its efficiency and has maintained flexibility to increase robustness.

9 Maintenance and Extension

One obvious use of the flexibility of agent-based systems is to ease the incorporation of changes. This helps both in repairing faulty software and in extending the requirements of the system. There are two aspects to this robustness: within the design and during use.

The methodology used affects how easily the design can be altered. Object-oriented designs provide highly cohesive, separated entities which can be altered individually so changes that fall in the bounds of a single object will be relatively simple to make. Agent interaction analysis offers a comparable division in terms of goals. Changes are applied to the design early in its documentation, usually at the goal decomposition stage. At this point the functionality is divided allowing targeting of changes, and there are only very few assumptions or restrictions making it unlikely for features of the system to conflict. After these changes have been made we can follow the documentation from the early to late stages, examining design decisions and replacing them where necessary. Finally the alterations will be merged into the final version of the design with minimal effect on the rest of the system.

At execution time multi-agent systems have an advantage over most other forms of system in adapting to change [12]. By having multiple independently running, decision-making, reactive, communicating and balancing entities, changes can be adapted to by being treated as differences in options available and then taken advantage of. To provide most scope for this adaptation the agents should remain as flexible as the system requirements permit. By emphasizing the interactions between agents and not limiting the interactions too much, agents within the system are free to choose good coordination strategies that the current form of the system allows.

10 Summary and Conclusion

In this paper we have proposed the idea of a form for agent-oriented software engineering methodologies, *agent interaction analysis*, which takes abstract interactions between agents as its primary components of analysis. By decomposing the system requirements into interactions, based around the exchange of goals, we suggest that a more robust design, and one that is well tailored to maintenance and changes in the domain, can be produced.

We have shown how the interactions can be examined to determine efficient forms for the participating agents, and have used the idea of *assurance* to examine the requirements of individual roles in interactions, particularly in terms of extracting helpful information to allow designers to choose effective coordination mechanisms.

Clearly, this work only introduces the general principles of the approach, and there are many issues remaining to be tackled. One is creation of techniques for assurance analysis, so that coordination can be described and judged with respect to the rest of the design. We have done work on a range of specific techniques for assurance analysis in BDI agents. One such technique uses a generalised agent architecture which can be used to express and compare a variety of coordination mechanisms such as trust and basic commitments. Another technique measures the cost, as given by preferences, of using mechanisms between many agents such as brokers, references to agents to static rôles and commitments with or without reference to the committing agents. This work is left for discussion elsewhere due to space limitations.

As this paper only describes a framework, a full methodology can only be produced together with detailed techniques for suitably documenting these designs. The approach presented in this paper is deliberately distinguished from any notation that it may use. As a foundation, some of the concepts may be best illustrated in the recent work on agent-oriented extensions to UML [4, 8, 22].

In future work, we aim to develop a fully usable methodology which uses agent interaction analysis, and test it in the production of suitable applications. We are also working on using assurance analysis to describe and assess coordination mechanisms to help in the development of other such mechanisms tailored to particular domains.

The approach of agent interaction analysis, as suggested in this paper, clears the way for general effective methodologies targeted at the design of multi-agent systems for complex and open applications flexible enough to cope with a wide range of uncertainty and dynamism.

References

1. D. Amyot, L. Logrippo, R. J. A. Buhr, and T. Gray. Use Case Maps for the capture and validation of distributed systems requirements. In *Fourth International Symposium on Requirements Engineering (RE-99)*, 1999.
2. Y. Aridor and D. B. Lange. Agent design patterns: Elements of agent application design. In *Proceedings of the Second International Conference on Autonomous Agents (Agents-98)*, Minneapolis, USA, 1998.
3. M. Barbuceanu. Coordinating agents by role based social constraints and conversion plans. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, pages 16–21, 1997.
4. B. Bauer, J. P. Müller, and J. Odell. Agent UML: A formalism for specifying multiagent software systems. In *this volume*.
5. A. M. Davis. *Software Requirements: Objects, States and Functions*. Prentice Hall, 1993.
6. K. S. Decker and V. R. Lesser. Designing a family of coordination algorithms. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 73–80, 1995.
7. S. A. DeLoach and M. Wood. Developing multiagent systems with agentTool. In *Proceedings of the Seventh International Workshop on Agent Theories, Architectures and Languages (ATAL-00)*, 2000.
8. R. Depke, R. Heckel, and J. M. Küster. Requirement specification and design of agent-based systems with graph transformation, roles and UML. In *this volume*.

9. Elammari, M. and Lalonde, W. An agent-oriented methodology: High-level and intermediate models. In *Proceedings of Agent-Oriented Information Systems 1999 (AOIS-99)*, 1999.
10. S. Franklin and A. Graesser. Is it an agent, or just a program?: A taxonomy for autonomous agents. In J.P. Müller, M.J. Wooldridge, and N.R. Jennings, editors, *Intelligent Agents III: Proceedings of the Third International Workshop on Agent Theories, Architectures and Languages (ATAL-96)*, pages 21–35. Springer-Verlag, 1997.
11. C. V. Goldman and J. S. Rosenschein. Mutual adaptation enhanced by social laws. Technical Report CS98-5, The Hebrew University, Israel, 1998.
12. M. N. Huhns. Interaction-oriented programming. In *this volume*.
13. C. A. Iglesias, M. Garijo, and J. C. Gonzalez. A survey of agent-oriented methodologies. In J. P. Müller, M. P. Singh, and A. S. Rao, editors, *Intelligent Agents V: Proceedings of the Fifth International Workshop on Agent Theories, Architectures and Languages (ATAL-98)*, 1998.
14. N. R. Jennings. Commitments and conventions: The foundation of coordination in multi-agent systems. *Knowledge Engineering Review*, 8(3):223–250, 1993.
15. N. R. Jennings, K. Sycara, and M. Wooldridge. A roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems*, 1(1):275–306, 1998.
16. E. A. Kendall. Agent software engineering with role modelling. In *this volume*.
17. E. A. Kendall. Role modelling for agent system analysis and design. In *Proceedings of the First International Symposium on Agent Systems and Applications (ASA/MA'99)*, November 1999.
18. D. Kinny, M. Georgeff, and A. Rao. A methodology and modelling technique for systems of BDI agents. In Walter Van de Velde and J.W. Perram, editors, *Agents Breaking Away: Proceedings of the Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW-96)*, January 1996.
19. C. Landauer and K. Bellman. Agent-based information infrastructure. In *Proceedings of Agent-Oriented Information Systems 1999 (AOIS-99)*, 1999.
20. M. Luck and M. d'Inverno. Engagement and cooperation in motivated agent modelling. In C. Zhang and D. Lukose, editors, *Distributed Artificial Intelligence Architecture and Modelling: Proceedings of the First Australian DAI Workshop*, pages 70–84. Springer-Verlag, 1996.
21. S. P. Marsh. *Formalising Trust as a Computational Concept*. PhD thesis, Department of Computing Science and Mathematics, University of Stirling, 1994.
22. J. Odell, H. V. D. Parunak, and B. Bauer. Representing agent interaction protocols in UML. In *this volume*.
23. A. Omicini. SODA: Societies and infrastructures in the analysis and design of agent-based systems. In *this volume*.
24. I. Sommerville. *Software Engineering*. Addison-Wesley, fifth edition edition, 1995.
25. M. Wood and S. A. DeLoach. An overview of the multiagent system engineering methodology. In *this volume*.
26. M. Wooldridge, N. R. Jennings, and D. Kinny. A methodology for agent-oriented analysis and design. In *Proceedings of the Third International Conference on Autonomous Agents (Agents-99)*, Seattle, USA, 1999.
27. M. Wooldridge, N. R. Jennings, and D. Kinny. The Gaia methodology for agent-oriented analysis and design. *Journal of Autonomous Agents and Multi-Agent Systems*, 3, 2000.
28. L. Yu and B. F. Schmid. A conceptual framework for agent oriented and role based workflow modeling. In *Proceedings of Agent-Oriented Information Systems 1999 (AOIS-99)*, 1999.
29. F. Zambonelli, N. R. Jennings, and M. Wooldridge. Organisational abstractions for the analysis and design of multi-agent systems. In *this volume*.