A Common Basis for Agent Organisations in BDI Languages*

Anthony Hepple, Louise Dennis, and Michael Fisher

Department of Computer Science, University of Liverpool, Liverpool, U.K.
{A.J.Hepple,L.A.Dennis,M.Fisher}@csc.liv.ac.uk

Abstract. Programming languages based on the BDI style of agent model are now common. Within these there appears to be some, limited, agreement on the core functionality of agents. However, when we come to multi-agent organisations, not only do many BDI languages have no specific organisational structures, but those that do exist are very diverse. In this paper, we aim to provide a unifying framework for the core aspects of agent organisation, covering groups, teams and roles, as well as organisations. Thus, we describe a simple organisational mechanism, and show how several well known approaches can be embedded within it. Although the mechanism we use is derived from the METATEM language, we do not assume any specific BDI programming language. The organisational mechanism is thus intended to be independent of the underlying agent language and so provides a common core for future developments in agent organisation.

1 Introduction

As hardware and software platforms become more sophisticated, and as these are deployed in more unpredictable environments, so the level of *autonomy* built into such systems has increased. This has allowed systems to work effectively without detailed, and constant, human intervention. However, autonomous systems can be hard to understand and even harder to develop reliably. In order to help in this area, the concept of an *agent* was developed to capture the abstraction of an autonomously acting entity. Based on this concept, new techniques were developed for analysing, designing and implementing agents. In particular, several new programming languages were developed explicitly for implementing autonomous agents.

We can simply characterise an agent as an autonomous software component having certain goals and being able to communicate with other agents in order to accomplish these goals [31]. The ability of agents to act independently, to react to unexpected situations and to cooperate with other agents has made them a popular choice for developing software in a number of areas. At one extreme there are agents that are used to search the INTERNET, navigating autonomously in order to retrieve information; these are relatively lightweight agents, with few goals but significant domain-specific knowledge. At the other end of the spectrum, there are agents developed for independent process control in unpredictable environments. This second form of agent is often constructed using complex software architectures, and has been applied in areas such as real-time process control [25, 19]. Perhaps the most impressive use of such agents is as part of the real-time fault monitoring and diagnosis carried out in the NASA Deep Space One mission [22].

The key reason why an agent-based approach is advantageous in the modelling and programming of autonomous systems, is that it permits the clear and concise representation, not just of

* Work partially supported by EPSRC under grant EP/D052548.

what the autonomous components within the system do, but *why* they do it. This allows us to abstract away from the low-level control aspects and to concentrate on the key feature of autonomy, namely the goals the component has and the choices it makes. Thus, in modelling a system in terms of agents, we often describe each agent's *beliefs* and *goals*, which in turn determine the agent's *intentions*. Such agents then make decisions about what action to perform, given their beliefs and goals/intentions. This kind of approach has been popularised through the influential BDI (Belief-Desire-Intention) model of agent-based systems [25]. This representation of behaviour using *mental* notions is initially unusual, yet has several benefits. The first is that, ideally, it abstracts away from low-level issues: we simply present some goal that we wish to be achieved, and we expect it to act as an agent would given such a goal. Secondly, because we are used to understanding and predicting the behaviour of rational agents, the behaviour of autonomous software should be relatively easy for humans to understand and predict too. Not surprisingly, the modelling of complex systems, even space exploration systems, in terms of rational agents has been very successful [27, 26, 16]. Thus, the BDI approach to agent modelling has been successful. Unsurprisingly, this has led to many novel programming languages based (at least in some part) upon this model; these are often termed *BDI Languages*. Although a wide variety of such languages have been developed [3] few have strong and flexible mechanisms for *organising* multiple agents, and those that do provide no agreement on their organisational mechanisms. Thus, while BDI languages have converged to a common core relating to the activity of individual agents [8], no such convergence is apparent in terms of multi-agent structuring.

As part of our overall aim is to provide a common logically based framework for BDI style agent programming, incorporating organisational aspects, and so facilitate agent verification via model checking, a clear goal is to develop a simple, intuitive and semantically consistent organisation mechanism. In this paper we do this, in addition showing how this simple model can, in BDI languages, encompass many previously proposed models of multi-agent organisation and teamwork.

The structure of this paper is as follows. Section 2 surveys some of the leading approaches to agent organisation and illustrates their diverse nature. In Section 3 we describe the structuring mechanism we propose to unify the multi-agent concepts uncovered within our survey. Section 4 demonstrates how our framework can be used to model concepts such as joint-intentions, roles, etc. Finally, in Section 5, we provide concluding remarks and outline future work.

2 Approaches to Agent Organisation

In this section we overview some of the key approaches to the organisation of agents that have been proposed. It is important to note that we are particularly concerned with *rational agents*, predominantly using the BDI model of computation. In addition, while we have not listed *all* the various approaches, the selection we give below covers many of the leading attempts at teamwork, organisational structuring and role-based computation. In addition, while we are primarily interested in developing BDI languages with clear logical semantics and logic-based mechanisms, we also consider organisational approaches beyond this class.

2.1 Cohen and Levesque: *Joint Intentions*

Offering a respected philosophical view on agent co-operation, Cohen and Levesque produced a significant paper 'Teamwork' [7] extending previous work [21, 5, 6]. They persuasively argue that a team of agents should *not* be modelled as an aggregate agent but propose new (logical) concepts of *joint intentions*, *joint commitments* and *joint persistent goals* to ensure that teamwork does not break down due to any divergence of individual team members' beliefs or intentions. The authors' proposals oblige agents working in a team to retain team goals until it is mutually agreed amongst team members that the goal has been achieved, is no longer relevant or is impossible. This level of commitment is stronger than an agent's commitment to its individual goals which are dropped the moment it (individually) believes they are satisfied. Joint intentions can be reduced to individual intentions if supplemented with mutual beliefs.

2.2 Tidhar, Cavedon and Rao: *Team-Oriented Programming*

Tidhar [29] introduced the concept of *team-oriented programming* with social structure. Essentially this is an agent-centred approach that defines joint goals and intentions for teams but stops short of forcing individual team members to adopt those goals and intentions. An attempt to clarify the definition of a 'team' and what team formation entails was made using concepts such as 'mind-set synchronisation' and 'role assignment'. Team behaviour was defined by a temporal ordering of plans which guided (but did not constrain) agent behaviour. A social structure is proposed by the creation of *command* and *control* teams which assign roles, identify sub-teams and permit inter-team relationships.

2.3 Ferber, Gutknecht and Michel: *Roles and Organisations*

Ferber *et al.* [10] present the case for an organisational-centred approach to the design and engineering of complex multi-agent systems. They cite disadvantages of the predominant agent-centred approaches such as: lack of access rights control; inability to accommodate heterogeneous agents; and inappropriate abstraction for describing organisational scenarios. The authors propose a model for designing language independent multi-agent systems in terms of *agents*, *roles* and *groups*. Agents and groups are proposed as distinct first class entities although it is suggested that an agent ought to be able to transform itself into a group. (We will see later that this is close to our approach.)

In [11], Ferber continues to argue for an organisational-centred approach, advocating the complete omission of mental states at the organisational level, defining an organisation of agents in terms of its capabilities, constraints, roles, group tasks and interaction protocols. Clearly articulated here is a manifesto of design principles.

2.4 Pynadath and Tambe: *TEAMCORE*

Pynadath *et al.* [24] the authors describe their interpretation of 'team-oriented programming' that aims to organise groups of heterogeneous agents to achieve team goals. A framework for defining teams is given that provides the following concepts:

Team — an agent without domain abilities;
Team-ready — agents with domain abilities that can interface with a team agent;
Sub-goal — a goal that contributes to the team goal; and
Task — the allocation of a sub-goal to a team-ready agent.

An implementation of their framework, TEAMCORE, provides organisational functionality such as enabling multicast communication between agents, assigning tasks, maintaining group beliefs and maintaining hierarchies of agents (by role). Also, heterogeneous agents are accommodated by wrapper agents that act as proxies for the domain agent.

2.5 Fisher, Hirsch and Ghidini: *Groups as Agents*

Beginning within the context of executable temporal logics [1], Fisher *et al.* produced a series of papers [12–15] that developed the METATEM language into a generalised approach for expressing dynamic distributed computations. As we will see more about this model in Section 3, we just provide a brief outline below.

Organisational structuring within the METATEM [12] language consists of a simple nested grouping structure where groups comprise communicating elements (objects, agents, or other software components). The key aspect of this approach is that groups themselves are also agents, providing a homogeneous, simple, yet expressive, model. In [14], it is argued that systems composed of components as diverse as objects, web services and abstract features can be modelled within this general language.

2.6 Hübner, Sichman and Boissier: *Roles and Permissions*

Hübner and his co-authors believed that the agent organisational frameworks proposed prior to their 2002 paper [18] overlooked the significant relationship between structural and functional properties of an organisation. Thus, in [18], they propose a three component approach to the specification of agent organisations that combines independent structural and functional specifications with a deontic specification, the latter defining, among other things, the roles (structural) having permission to carry out group tasks (functional). The approach provides a proliferation of constructs for specifying multi-agent systems, including the ability to concisely express many previously unmentioned situations, such as:

– the ability to specify *compatibility* of group membership, akin to the members of a government expressing a conflict of interest;
– enabling the *cardinality* of group membership to be defined and thus defining a well formed group as a group who's membership is between its specified minimum and maximum size;
– the ability to express a variance in the agents' permissions over time.

It is argued that such an approach improves the efficiency of multi-agent systems by focusing agents on the organisation's goals. However, we note that of all the proposals discussed in this section this approach applies the most restrictions to agent autonomy.

2.7 Summary

It should be noted that none of the above organisational approaches can comprehensively model all forms of co-operative multi-agent systems. Rather they represent attempts to discover practical and beneficial ways of specifying distributed computational systems, and facilitating the focus of computation on a system's main purpose whilst not compromising the autonomy of the system's components. In achieving this aim it may be convenient to categorise groups of agents in terms of cohesion and co-operation. For instance, a *group* of agents may be individually autonomous, existing as a group solely due to their proximity to one another rather than their co-operation. In contrast, the word *team*, implies a high degree of co-operation and adhesion with an *organisation* fitting somewhere in between. As Cohen stated in [7]

> "teamwork is more than co-ordinated individual behaviour".

Thus, the more expressive proposals reviewed here enable the specification of more cohesive groups but often at significant cost to the agents involved.

3 Structuring Mechanisms

The approach we propose is based on that of METATEM described previously [12]. However, we advocate this grouping approach, independent of the underlying language for agents. The only restrictions we put on any underlying language is that, as in most BDI-based languages, there are logically coherent mechanisms for explicitly describing *beliefs* and *goals*.

The aim of our grouping structure is to provide a simple organisational construct that enables the definition of a wide range of multi-agent systems — from unstructured collections of unco-ordinated agents to complex systems that are often described using the high-level abstractions described in the last section.

3.1 Prerequisites

As in the METATEM framework, the grouping approach involves very few additional constructs within the language. Specifically, we require just two additional elements within each agent's state. We also, as is common, require that first-class elements, such as beliefs, goals, etc, can be communicated between agents. Delivery of messages should be guaranteed, though the delay between send and receipt is not fixed. Finally, we expect asynchronously concurrent execution of agents.

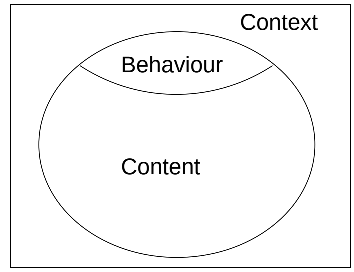
3.2 Extending Agents

Assuming that the underlying agent language can describe the behaviour of an agent, as has been shown for example in [8], we now extend the concept of agent with two sets, `Content` and `Context`. The agent's `Content` describes the set of agents it contains, while the agent's `Context` describes a set of agents it is contained within. Thus, the formal definition of an agent is as follows.

$$\begin{array}{lll} \texttt{Agent} ::= & \texttt{Behaviour:} & \texttt{Specification} \\ & \texttt{Content:} & \mathcal{P}(\texttt{Agent}) \\ & \texttt{Context:} & \mathcal{P}(\texttt{Agent}) \end{array}$$

Here, $\mathcal{P}(\texttt{Agent})$ are sets of agents and `Specification` is the description of the individual agent's behaviour, given as appropriate in the target BDI language.

On the right, we provide a graphical representation of such an agent. The agent (the circle) resides within a context and itself comprises its own behavioural layer and its content. This content can again contain further agents. Note that, for formal development purposes, the Behaviour may well be a logical specification.



The addition of `Content` and `Context` sets to each agent provides significant flexibility for agent organisation. Agent teams, groups or organisations, which might alternatively be seen as separate entities, are now just agents with non-empty `Content`. This allows these organisations to be hierarchical and dynamic, and so, as we will see later, provides possibilities for a multitude of other co-ordinated behaviours. Similarly, agents can have several agents within their `Context`. Not only does this allow agents to be part of several organisational structures simultaneously, but it allows the agent to benefit from `Context` representing diverse attributes/behaviours. So an agent might be in a context related to its physical locality (ie agents in that set are 'close' to each other), yet also might be in a context that provides certain roles or abilities. Intriguingly, agents can be within many, overlapping and diverse, contexts. This gives the ability to produce complex organisations, in a way similar to multiple inheritance in traditional object/concept systems. For example configurations, see Fig. 1.

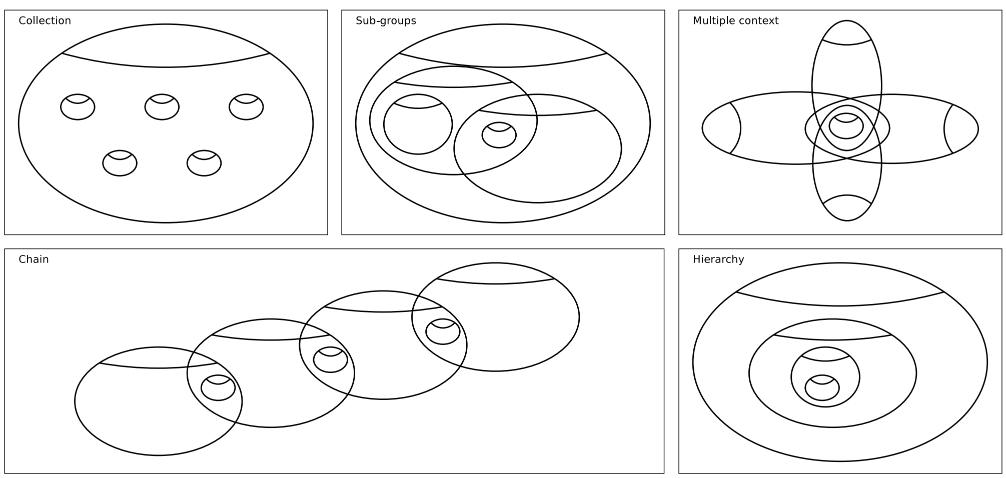


Fig. 1. A selection of possible organisation structures.

An important aspect is that this whole structure is very dynamic. Agents can move in and out of `Content` and `Context` sets, while new agents (and, hence, organisations) can be spawned easily and discarded. This allows for the possibility of a range of organisations, from the *transient* to the *permanent*. From the above it is clear that there is no enforced distinction between an agent and an agent organisation. All are agents, all may be treated similarly. On the other hand it is possible to distinguish between agents (with empty `Content`) and organisations (with non-empty `Content`) to allow an organisation-centred approach, if required.

Finally, it is essential that the agent's internal behaviour, be it a program or a specification, have direct access to both the `Content` and `Context` sets. As we will see below, this allows each agent to become more than just a 'dumb' container. It can control access to, restructure, and share information and behaviours with, its `Content`. Note that, in order to describe fragments of the agent's behaviour during the rest of the paper, we will use simple **IF**...**THEN**...**ELSE** statements. Again, this does not prescribe any particular style of BDI language.

3.3 Communication

The core communication mechanism between agents in our model is broadcast message-passing. The use of broadcast is very appealing, allowing agent-based systems to be developed without being concerned about addresses/names of the agents to be communicated with. The potential inefficiency of broadcast communication is avoided by the use of the agents' `Content` and `Context` structures. By default, when an agent broadcasts a message, it is sent to all members of the agent's `Context` sets with the message being forwarded to agents within the same context. This, effectively, produces *multicast*, rather than full broadcast, message-passing.

This is clearly a simple, flexible and intuitive model, and the system developer is encouraged to think in this way. However, it is useful to note that multicast, or 'broadcast within a set', is actually built on top of point-to-point message passing! We will assume that the BDI language has a communication construct that can be modelled as the action $send(recipient, m)$ which means that the message m has been sent to the agent $recipient$, and a corresponsing $received(sender, m)$ which become true when the $recipient$ agent receives the message. Let us consider an example where an agent wishes to broadcast to all other members of one of its `Context` sets. For simplicity, let us term this context set '$group$'. An agent wishing to 'broadcast' a message, m, to members of the $group$ sends a message, $send(group, broadcast(m))$, to the group agent alone, as illustrated in Fig. 2.

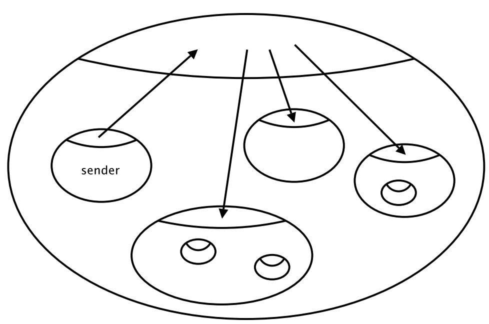


Fig. 2. Broadcast within a Group.

The effect of sending a broadcast message to the $group$ agent is that the $group$ acts as a proxy and forwards the message to its `Content`, modifying the message such that the message appears to have originated from the proxy. In this way agents maintain their anonymity within the group.

$$\begin{array}{l}\textbf{IF}\ received(from, broadcast(m))\\ \textbf{THEN for each}\ x\ \textbf{in}\ \{\texttt{Content} \setminus from\} \quad send(x, m)\end{array}$$

Being an agent-centred approach to multi-agent organisation there does not exist an [accessible] entity that references *all* agents in the agent space, thus a true broadcast is not possible. However a number of recursive group broadcasts can be specified, allowing a message to be propagated to all agents with an organisational link to the sender.

For example, reaching all accessible agents requires the sending agent to send a message to all members of its `Context` and `Content` sets and for each first-time recipient to recursively forward that message to the union of their `Context` and `Content` (excluding the sender). Clearly this is not an efficient method of communication as it is possible for agents to receive multiple copies of the same message, and so it may not be practical in very large societies, but what it lacks in sophistication it makes up for in simplicity and clarity [14].

$$\begin{array}{l}\textbf{IF}\ received(from, broadcastAll(m))\ \textbf{AND not}\ received(\_, m)\\ \textbf{THEN for each}\ x\ \textbf{in}\ \{\texttt{Content} \cup \texttt{Context}\}\ send(x, m)\ \textbf{AND}\ send(x, broadcastAll(m))\end{array}$$

Perhaps more useful than indiscriminate broadcasting would be the case of an agent who wants to reach all other members of the 'greatest' organisation to which it belongs. This requires a message to propagate up through the agent structure until it reaches a group with an empty context, at which point the message is sent downwards until all members and sub-members have been reached.

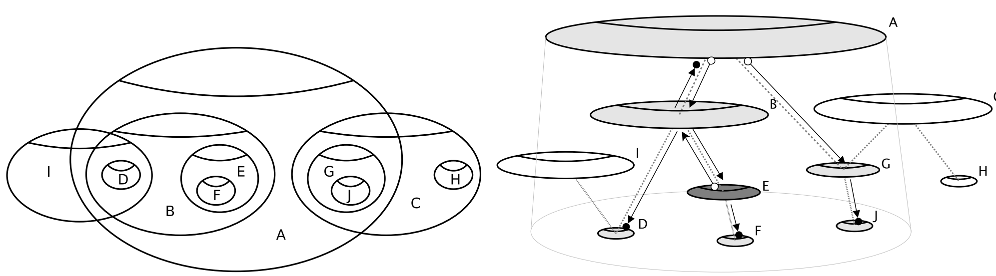


Fig. 3. (a) Nested Organisations **(b)**. Propagation of Messages

To illustrate this, consider the situation of agent `E` in Fig. 3(a), who wants to send a message to its entire organisation — the organisation specified by `A`. A $propagateUp(m)$ message originates from agent `E` who sends it to agent `B`. `B`'s context is non-empty so the message continues upwards to `A`. Since `A` is the widest organisation to which `E` belongs (it has an empty `Context` set), it modifies the message, converting it to $propagateDown(message)$ and broadcasts it along with the message to all members of its content. Upon receipt of this message, agents `B` and `G` send it to their contents and so it continues until the message reaches an agent with an empty content set as illustrated by Fig. 3(b).

This might be specified as follows;

$\textbf{IF } \mathit{received}(\_, \mathit{propagateUp}(m)) \textbf{ AND } \texttt{Context} \neq \emptyset$
$\textbf{THEN for each } x \textbf{ in } \{\texttt{Context}\} \quad \mathit{send}(x, \mathit{propagateUp}(m))$

$\textbf{IF } \mathit{received}(\_, \mathit{propagateUp}(m)) \textbf{ AND } \texttt{Context} = \emptyset$
$\textbf{THEN for each } x \textbf{ in } \{\texttt{Content}\} \quad \mathit{send}(x, m) \textbf{ AND } \mathit{send}(x, \mathit{propagateDown}(m))$

$\textbf{IF } \mathit{received}(\_, \mathit{propagateDown}(m)) \textbf{ AND } \texttt{Content} \neq \emptyset$
$\textbf{THEN for each } x \textbf{ in } \{\texttt{Content}\} \quad \mathit{send}(x, m) \textbf{ AND } \mathit{send}(x, \mathit{propagateDown}(m))$

3.4 Refining and Restricting Communications

Further communication restriction is possible by, for example, restricting the type of communications agents can make. Employing the concept of speech acts [28] we can use the group agent as a communication filter that restricts intra-group messaging to those that conform to permissible protocols or structures.

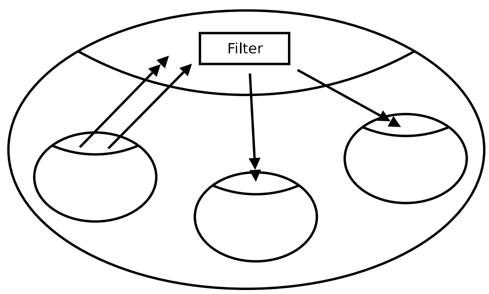


Fig. 4. Filtering communication by group.

If, for example, a fact-finding agent contains a number of agents with access to information resources, it may be necessary to restrict their intra-group communication to **inform** speech acts. In such circumstances it is possible to modify the default behaviour by imposing a message filter.

$$\textbf{IF } \mathit{received}(\mathit{from}, \mathit{broadcast}(m))\textbf{AND } \mathit{informFilter}(m)$$
$$\textbf{THEN for each } x \textbf{ in } \{\texttt{Content} \setminus \mathit{from}\} \quad \mathit{send}(x, m)$$

See Fig. 4 for an example of this. In this way filters can be adapted for many purposes, enabling organisations to maintain:

relevance — ensuring communication remains relevant to to group goal(s), intentions or tasks;
fairness — allowing each member of a group an equal opportunity to speak; and
legality — assigning permissions to group members to restrict communication channels.

3.5 Communication Semantics

The above variations on *broadcast* define varying semantics for a message. A key feature of the grouping approach is that the semantics of communication is flexible and, potentially, in the hands of the programmer. Such semantics can also, potentially, be communicated between agents in the form of plans allowing an agent to adopt different semantics for communication as its `Context` changes.

Adherence to particular common communication protocols/semantics also allows groups to establish the extent to which a member is autonomous (e.g., a group can use a semantics for **achieve** speech acts which forces recipients to adopt the communicated goal).

4 Common Multi-Agent Structures

In this section we will examine some of the key structuring mechanisms that are either explicit or implicit within the approaches surveyed in Section 2, and show how each can be represented appropriately, and simply, using the `Content`/`Context` approach outlined above.

Table 1 lists the mechanisms identified by our surveyed authors as being useful in the specification of agent co-operation. We believe that our approach is flexible enough to model all of these but for brevity we will demonstrate a sample of them only.

	Joint intentions	Roles	Joint beliefs	Joint commitments	Global goals	Communication	Organisation/Team abstraction	Mutual beliefs	Social structure	inter-[team|group] relationships	Sub-teams	Social commitment	Groups	Task structure	Organisational centred	Agent centred
Cohen & Levesque	✓			✓		✓		✓								✓
Ferber		✓				✓	✓		✓				✓	✓	✓	
Pynadath & Tambe		✓	✓			✓	✓							✓		
Hubner		✓			✓	✓			✓	✓	✓	✓	✓	✓	✓	
Cavedon	✓	✓		✓	✓				✓			✓				✓
Vazquez, Dignum & Dignum		✓			✓	✓	✓		✓	✓		✓			✓	✓
Fisher et al.						✓							✓			✓
Tidhar	✓	✓			✓		✓		✓	✓	✓	✓			✓	

Table 1. Multi-agent organisation concepts.

4.1 Sharing Information

Shared beliefs Being a member of all but the least cohesive groups requires that some shared beliefs exist between its members. Making the contentious assumption that all agents are honest and that joining the group is both individual rational and group rational, let agent i hold a belief set BS_i. When an agent joins a group j it receives beliefs BS_j from the group and adds them to its own belief base (invoking its own belief revision mechanism is case of conficting beliefs).

The agent in receipt of the new beliefs may or may not disseminate them to the agents in its content, depending on the nature and purpose of the group. Once held, beliefs are retained until contradicted.

Joint beliefs Joint beliefs are stronger than shared beliefs. To maintain the levels of cohesion found in teams each member must not only believe a joint belief but must also believe that its team members believe the joint belief. Let us assume the agent is capable of internal actions such as $addBelief(Belief, RelevantTo)$ adding $Belief$ to its belief base, and recording the context that $Belief$ is relvant to. Upon joining a group, an agent is supplied the beliefs relevant to that context, which it stores in its belief base along with the context in which they hold.

$$\begin{array}{l}\textbf{IF}\ received(from, membershipConfirm(beliefSet)) \\ \textbf{THEN for each}\ b\ \textbf{in}\ \{beliefSet\} \quad addBelief(b, from)\end{array}$$

The presence of such `Context` meta-information can be used to apply boundaries on agent deliberation, thus mitigating the complexity caused by introducing another variable. When leaving a `Context` an agent might choose to drop the beliefs relevant to that context or retain them.

4.2 Sharing Capabilities

Let agent Ag_i have a goal G, for which plan P exists but that Ag_i does not have and therefore must find an agent that does. Two options available to Ag_i are to find an agent Ag_j, who has P, and either: request that Ag_j carries out the plan; or request that Ag_j sends P to Ag_i so that Ag_i can carry out the plan itself. The first possibility suggests a closer degree of co-operation between agents i and j, perhaps even the sub-ordination of agent j by agent i. Whereas, in the second possibility, agent i benefits from information supplied by j.

In the first scenario we might envisage a group in which a member (or the group agent itself) asks another member to execute the plan. In the second case, we can envisage agents i and j *sharing* a plan. This second scenario is typical if groups are to capture certain capabilities — agents who join the `Content` of such a group are sent (or at least can request) plans shared amongst the group.

4.3 Joint Intentions

An agent acting in an independent self-interested way need not inform any other entity of its beliefs, or changes to them. On the other hand, an agent who is working, as part of a team, towards a goal shared by itself and all other members of the team has both an obligation and a rational interest in sharing relevant beliefs with the other team members [7]. Providing an agent with a persistent goal with respect to a team, such that the agent must intend the goal whilst it is the team's mutual belief that the goal is valid (not yet achieved, achievable and relevant). The implications of this impact on agent's individual behaviour when it learns, independently, that the goal is no longer valid - in such a situation the team working agent maintains its commitment to the invalid goal but informs its team members of the antecedent(s) that lead it to believe the goal is invalid. Only when the agent receives confirmation that the entire team share its belief does it drop its commitment.

The intuitive implementation of this joint intention is not via a team construct but with an extension of an agent's attributes. However, increases in expressiveness of this sort do not come without penalty — increased undecidability usually accompanies them. The organisational or team construct may overcome this problem but we believe that our simple group approach is sufficient to implement joint intentions, mutual beliefs and common goals. Consider the scenario given in Fig. 5.

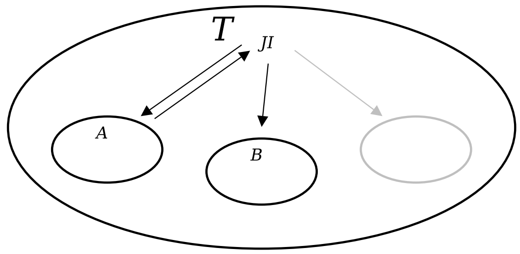


Fig. 5. Communicating Joint Intentions.

Agent A. On joining the group T, agent A accepts goal JI and confirms its adoption of the goal. Whilst T remains a member of A's `Context`, A informs T of all beliefs that are relevant to JI. Finally, all communications from agent T must be acknowledged, with an indication of the agent's acceptance (or non-acceptance) of the message.

A simple specification of this might be:

$$\begin{array}{l}\textbf{IF } received(from, jointIntention(JI)) \\ \textbf{THEN } achieve(JI) \textbf{ AND } send(from, ack(JI))\end{array}$$

$$\begin{array}{l}\textbf{IF } belief(\varphi) \textbf{ AND there is } x \textbf{ in } \{\texttt{Context}\}\ relevantTo(\varphi, x) \\ \textbf{THEN } send(x, \mathsf{inform}(\varphi))\end{array}$$

$$\begin{array}{l}\textbf{IF } goal(\gamma) \textbf{ AND there is } x \textbf{ in } \{\texttt{Context}\}\ relevantTo(\gamma, x) \\ \textbf{THEN } \mathsf{achieve}(\gamma)\end{array}$$

Thus, an agent is obliged to inform its group of beliefs relevant to jointly held intentions and will maintain a goal whilst it remains relevant to its `Context`.

Agent T. Evaluates group beliefs and communicates the taking on, and dropping, of intentions when mutual agreement is established. Since T has details of the agents in its `Content` and can send messages to interrogate them, it can maintain knowledge of *common* information and behaviours, and reason with it.

4.4 Roles

The concept of a role is a common abstraction used by many authors for a variety of purposes [18, 11, 30], including:

- to define the collective abilities necessary to achieve a global goal;
- to provide an agent with abilities suitable for team activity;
- to constrain or modify agent behaviour for conformance with team norms; and
- to describe a hierarchy of authority in an organisation of agents and hence create a permissions structure.

Below we examine a variety of such roles and consider how each could fit into our model.

Ability roles Let plan P be a complex plan that requires abilities x,y and z if it is to be fulfilled. An agent A is created (without any domain abilities of its own) to gather together agents that do have the necessary abilities. Agent A might generate an agent in its content for each of the abilities required to fulfil plan P.

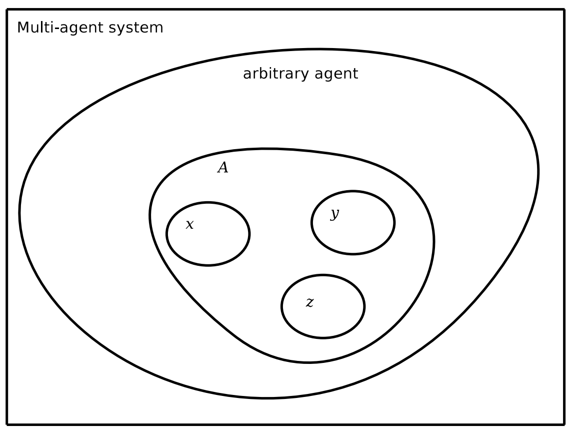


Fig. 6. Roles according to abilities.

When agent A encounters an agent with ability x, y or z it adds the agent to the `Content` of the appropriate group (agent), analogous to assigning roles.

A talented agent might become a member of several ability sets. The ability set, itself an agent, may be a simple container or exhibit complex behaviour of its own. One basic behaviour might be to periodically request (of the agents in its `Content`) the execution of its designated ability. Note that in the case of an ability that is hard to carry out, it may be provident to include many agents with that ability. Similarly, the desired ability might be a complex ability that must be subjected to further planning, resulting in a number of nested abilities.

Roles in society Joining a society, organisation or team of agents commonly involves the adoption of the norms of that society, organisation or team. Whether these norms are expressed as beliefs, goals, preferences or communication protocols, our approach allows them to be transmitted between group members, particularly at the time of joining.

For example, if team membership requires that members acknowledge receipt of messages then each new member of a group might be given the new rule (behaviour)

$$\textbf{IF } received(ag, \theta) \textbf{ THEN } send(ag, ack(\theta))$$

A stronger constraint might require an agent to believe all messages received from its `Context`:

$$\textbf{IF } received(ag, \theta) \textbf{ AND } ag \in \texttt{Context} \textbf{ THEN } addBelief(\theta, ag) \textbf{ AND } send(ag, ack(\theta))$$

Of course, agents can not be certain that another agent will keep with given constraints or comply with norms of the society, the most it can do is demand formal acknowledgement of its request and a commitment to do so. Group membership can be denied if an agent fails to satisfy the *entry criteria*.

Authority roles None of the structures discussed so far result in a structure that usefully reflects a hierarchy of authority.

As each of the above structures allow almost arbitrary group membership, with transitive and cyclic structures possible they are not suitable for expressing a hierarchy of authority, which by its nature must be acyclic with exactly one root.

A common use for such a hierarchy is for creating channels of communication. Our approach to grouping enables communication restrictions for free such that agents may only communicate with their immediate superiors (context), or their subordinates (content). Multicast message passing requires sending a single $broadcast$ message to an agent in its context. The receiving agent will, if it deems it appropriate, forward the message to all other agents in the [superior] agent's content.

5 Concluding Remarks

In this paper, we have proposed a simple but clear model for multi-agent structuring in a wide range of agent languages based on varieties of the logical BDI approach. Although derived from work on METATEM, we propose this as a general approach for many languages. To support this, we first show how simple and intuitive the approach is and how the underlying structures of any appropriate language can be modified. (Note that more detailed operational semantics for our grouping approach in logic-based BDI languages is given in [9].) We then showed, in a necessarily brief way, how many of the common teamwork and organisation aspects can be modelled using our approach.

In order to evaluate the approach, we have also implemented it in AgentSpeak (actually, Jason [4]) and have developed several simple examples of dynamic organisations. This simple additional layer has so far proved to be convenient and powerful. Obviously, the `Content`/`Context` approach has also been extensively used in previous work on METATEM [13–15, 17]. In addition, it has been incorporated in the semantics of AIL [8], a common semantics basis for a number of languages, including AgentSpeak and 3APL; see [9].

5.1 Future Work

Our immediate aim with this work is to apply the model to larger applications, particularly in the areas of ubiquitous computing and social organisations. This will give a more severe test for the approach and will highlight any areas of difficulty.

As mentioned above, the approach is being integrated into the AIL semantics [8], which provides a common semantics basis for a number of BDI languages. Since translations from AgentSpeak, 3APL, etc are being produced, we also aim to translate the organisational aspects used into the above model.

Finally, the aim of the work on AIL is to provide generic verification techniques for BDI languages (that can be translated to AIL). In extending the AIL semantics, we also aim to provide verification techniques for teams, roles and organisations developed within BDI languages.

References

1. H. Barringer, M. Fisher, D. Gabbay, R. Owens, and M. Reynolds, editors. *The Imperative Future: Principles of Executable Temporal Logic*. Research Studies Press, May 1996.
2. R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, editors. *Multi-Agent Programming: Languages, Platforms and Applications*. Number 15 in Multiagent Systems, Artificial Societies, and Simulated Organizations. Springer-Verlag, 2005.
3. R. H. Bordini, M. Dastani, J. Dix, and A. E. Fallah-Seghrouchni, editors. *Multi-Agent Programming: Languages, Platforms and Applications*, volume 15 of *Multiagent Systems, Artificial Societies, and Simulated Organizations*. Springer, 2005.
4. R. H. Bordini, J. F. Hübner, and R. Vieira. ***Jason*** and the golden fleece of agent-oriented programming. In Bordini et al. [2], chapter 1, pages 3–37.
5. P. R. Cohen and H. J. Levesque. Intention is Choice with Commitment. *Artificial Intelligence*, 42(2-3):213–261, 1990.
6. P. R. Cohen and H. J. Levesque. Confirmations and Joint Action. In *Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, pages 951–959, 1991.
7. P. R. Cohen and H. J. Levesque. Teamwork. Technical Report 504, SRI International, Menlo Park, CA, 1991.
8. L. A. Dennis, B. Farwer, R. H. Bordini, M. Fisher, and M. Wooldridge. A Common Semantic Basis for BDI Languages. In *Proc. Seventh International Workshop on Programming Multiagent Systems (ProMAS)*, Lecture Notes in Artificial Intelligence. Springer Verlag, 2007 (to appear).
9. L. A. Dennis, M. Fisher, and A. Hepple. Foundations of Flexible Multi-Agent Programming. Submitted, 2007.
10. J. Ferber and O. Gutknecht. A Meta-model for the Analysis and Design of Organizations in Multi-agent Systems. In *Proc. Third International Conference on Multi-Agent Systems (ICMAS)*, pages 128–135, 1998.
11. J. Ferber, O. Gutknecht, and F. Michel. From Agents to Organizations: An Organizational View of Multi-agent Systems. In *Proc. Workshop on Agent-Oriented Software Engineering (AOSE)*, volume 2935 of *LNAI*, pages 214–230. Springer-Verlag, 2003.
12. M. Fisher. METATEM: The Story so Far. In *Proc. Third International Workshop on Programming Multiagent Systems (ProMAS)*, volume 3862 of *Lecture Notes in Artificial Intelligence*, pages 3–22. Springer Verlag, 2006.
13. M. Fisher, C. Ghidini, and B. Hirsch. Organising Logic-Based Agents. In *Formal Approaches to Agent-Based Systems*, volume 2699 of *LNAI*, pages 15–27. Springer-Verlag, October 2003.

14. M. Fisher, C. Ghidini, and B. Hirsch. Programming groups of rational agents. In *Proc. International Workshop on Computational Logic in Multi-Agent Sytems (CLIMA)*, volume 3259 of *LNAI*. Springer-Verlag, November 2004.
15. M. Fisher and T. Kakoudakis. Flexible Agent Grouping in Executable Temporal Logic. In Gergatsoulis and Rondogiannis, editors, *Intensional Programming II*. World Scientific Publishing Co., 2000.
16. M. Fisher, E. Pearce, M. Wooldridge, M. Sierhuis, W. Visser, and R. Bordini. Towards the Verification of Human-Robot Teams. In *IEEE Workshop on Leveraging Applications of Formal Methods, Verification, and Validation (ISoLA)*, Washington D.C., USA., 2005.
17. B. Hirsch. *Programming Rational Agents*. PhD thesis, Department of Computer Science, University of Liverpool, June 2005.
18. J. F. Hübner, J. S. Sichman, and O. Boissier. A Model for the Structural, Functional, and Deontic Specification of Organizations in Multiagent Systems. In *Proc. Sixteenth Brazilian Symposium on Artificial Intelligence (SBIA)*, pages 118–128, London, UK, 2002. Springer-Verlag.
19. N. R. Jennings and M. Wooldridge. Applications of agent technology. In *Agent Technology: Foundations, Applications, and Markets*. Springer-Verlag, Heidelberg, 1998.
20. V. Lesser. Reflections on the Nature of Multi-Agent Coordination and its Implications for an Agent Architecture. *Autonomous Agents and Multi-Agent Systems*, 1:89–111, March 1998.
21. H. J. Levesque, P. R. Cohen, and J. H. T. Nunes. On Acting Together. In *Proc. Eighth National Conference on Artificial Intelligence (AAAI)*, pages 94–99, 1990.
22. N. Muscettola, P. P. Nayak, B. Pell, and B. Williams. Remote Agent: To Boldly Go Where No AI System Has Gone Before. *Artificial Intelligence*, 103(1-2):5–48, 1998.
23. H. E. Pattison, D. D. Corkill, and V. R. Lesser. Instantiating Descriptions of Organizational Structures. In M. N. Huhns, editor, *Distributed Artificial Intelligence, Research Notes in Artificial Intelligence*, volume I, pages 59–96. Pitman Publishers, 1987.
24. D. V. Pynadath, M. Tambe, N. Chauvat, and L. Cavedon. Towards Team-Oriented Programming. In *Intelligent Agents VI — Proc. Sixth International Workshop on Agent Theories, Architectures, and Languages (ATAL)*, volume 1757 of *LNAI*, pages 233–247. Springer-Verlag, 1999.
25. A. S. Rao and M. Georgeff. BDI Agents: from theory to practice. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 312–319, San Francisco, CA, June 1995.
26. M. Sierhuis. Multiagent Modeling and Simulation in Human-Robot Mission Operations. (See `http://ic.arc.nasa.gov/ic/publications`), 2006.
27. M. Sierhuis, J. M. Bradshaw, A. Acquisti, R. V. Hoof, R. Jeffers, and A. Uszok. Human-Agent Teamwork and Adjustable Autonomy in Practice. In *Proceedings of the 7th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*, Nara, Japan, 2003.
28. I. A. Smith and P. R. Cohen. Toward a Semantics for an Agent Communications Language Based on Speech-Acts. In *Proc. AAAI (Vol. 1)*, pages 24–31, 1996.
29. G. Tidhar. Team-Oriented Programming: Preliminary Report. Technical Report 1993-41, Australian Artificial Intelligence Institute, April 1993.
30. J. Vazquez-Salceda, V. Dignum, and F. Dignum. Organizing Multiagent Systems. Technical Report 2004-015, Institute of Information & Computing Sciences, Utrecht University, March 2004.
31. M. Wooldridge and N. R. Jennings. Intelligent Agents: Theory and Practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.