SPECIFICATION AND IMPLEMENTATION OF A BELIEF-DESIRE-JOINT-INTENTION ARCHITECTURE FOR COLLABORATIVE PROBLEM SOLVING

NICK R. JENNINGS

Department of Electronic Engineering, Queen Mary and Westfield College, University of London, Mile End Road, London E1 4NS, UK.

N.R.Jennings@qmw.ac.uk

ABSTRACT

Systems composed of multiple interacting problem solvers are becoming increasingly pervasive and have been championed in some quarters as the basis of the next generation of intelligent information systems. If this technology is to fulfill its true potential then it is important that the systems which are developed have a sound theoretical grounding. One aspect of this foundation, namely the model of collaborative problem solving, is examined in this paper. A synergistic review of existing models of cooperation is presented, their weaknesses are highlighted and a new model (called *joint responsibility*) is introduced. Joint responsibility is then used to specify a novel high-level agent architecture for cooperative problem solving in which the mentalistic notions of belief, desire, intention and joint intention play a central role in guiding an individual's and the group's problem solving behaviour. An implementation of this high-level architecture is then discussed and its utility is illustrated for the real-world domain of electricity transportation management.

Keywords: Distributed Artificial Intelligence, Multi-Agent Systems, BDI Architecture, Joint Intentions, Intentionality, Rational Problem Solvers.

1. Introduction

Systems composed of multiple problem solving entities, interacting with one another to enhance their performance, are becoming an increasingly popular means of conceptualising a diverse range of applications. This trend towards decentralised and cooperative problem solving occurs as a consequence of the desire to: increase the level of information integration across organisations [1, 2]; overcome the inherent limitations on intelligence present in any finite artificial system [3, 4]; develop increasingly complex software systems [5, 6] and provide a more natural representation of the problem being tackled [7, 8, 9, 10].

In Distributed Artificial Intelligence (DAI) systems, agents are grouped together to form communities which cooperate to achieve the goals of the individuals and of the system as a whole. Agents are usually capable of a range of useful problem solving activities, have their own aims and objectives and can communicate with others. The ability to solve some problems alone (*coarse granularity*) distinguishes components of DAI systems from those of connectionist or neural systems in which individual nodes have very simple states and only by combining many thousands of them can problem solving expertise be recognised.

Agents in a community usually have problem solving expertise which is related, but distinct, and which frequently has to be combined to solve problems. Such joint work is needed because of the dependencies between agents' actions, the necessity of meeting global constraints and because of the fact that often no one individual has sufficient competence to solve the entire problem alone. There are two main causes of such interdependence (adapted from [11]). Firstly, problem partitioning may yield components which cannot be solved in isolation. In speech recognition, for example, it is possible to segment an utterance and work on each component in isolation, but the amount of progress which can be made on each segment is limited. Allowing the sharing of hypotheses is a far more effective approach [12]. Secondly, even if subproblems are solvable in isolation, it may be impossible to synthesize their results because the solutions are incompatible or because they violate global constraints. For instance when constructing a house, many subproblems are highly interdependent (eg determining the size and location of rooms, wiring, plumbing, etc.). Each is solvable independently, but conflicts which arise when the solutions are collected are likely to be so severe that no amount of work can make them compatible. It is also unlikely that global constraints (eg total cost less than £250,000) would be satisfied. In such cases, compatible solutions can only be developed by having interaction and agreements between the agents during problem solving.

An important feature of any DAI system is the model of collaboration which is used to guide the society's behaviour. This model can be used at many different levels: to offer a behaviouristic explanation of problem solving communities, to prescribe the mental state of the collaborating agents, to define agents' actions, to help structure the knowledge elicitation process and so on. This work concentrates predominantly on the mental state and associated action perspectives, although the impact of these choices are also explored as a means of structuring the elicitation process.

A comprehensive model of collaboration should specify: (i) under what circumstances a social action should be initiated; (ii) what conditions need to be established before cooperation can proceed; (iii) how the individual agents should behave when carrying out their local activities related to the joint action; (iv) how (when) agents should interact with their fellow team members; and (v) when the joint action should terminate. As agents are often situated in evolving and unpredictable environments, and also because they have to take decisions using partial and imprecise information, it is important that the model specifies how to behave in exceptional as well as nominal situations [13]. In particular the model should specify how joint actions may become unstuck, how any problems in the collaboration can be repaired and how agents should respond to these problems both in their local activities and with respect to their acquaintances. In addition, the model should also have a clear path to implementation level systems.

At present there is a large chasm between the theoretical work which examines and specifies various models of collaboration and the implemented multi-agent systems which often use *ad hoc* models for social interaction. This paper seeks to bridge this gap; synthesizing and extending the salient characteristics of several theoretical models into a unifying structure called joint responsibility (section two), proposing a high level agent architecture based upon the responsibility model (section three) and discussing one realisation of this architecture in a general purpose cooperation framework called GRATE* (section four). To highlight the utility and applicability of the model, the description of GRATE* is couched in terms of the real-world application of electricity transportation management [14]. Finally relationships to existing research are explored (section five) and the seemingly clear demarcation between intentional and reactive systems is questioned (section six).

2. Intentions and Joint Intentions

2.1 Role of Individual Intentions

Intentions are one of the most popular means of describing the behaviour of rational problem solvers [15, 16, 17, 18, 19, 20]. They account for three important facets of an agent's practical reasoning process [21]. Firstly as agents are resource bounded, they cannot continually weigh their competing desires and their associated beliefs in deciding what to do next. At some point, the agent must just settle on one state of affairs for which to aim, thus creating a *commitment* to that objective. Secondly intentions are needed to plan future actions. Once a future action has been decided upon (intended), the agent must make subsequent decisions within the context that it will perform the said action at the specified time. Finally, intentions pose problems for means-end analysis as agents often commit themselves to activities whose exact means of achievement has still to be decided upon.

From this description of the role of intentions, two important properties can be derived. Firstly, intentions should be both internally consistent and consistent with the agent's beliefs. The former means that an agent's intentions should not conflict with one another; the latter that if an agent's intended actions are executed in a world in which its beliefs are true, the desired state of affairs should ensue [16]. Secondly intentions should have a degree of stability. If intentions were constantly reconsidered and abandoned they would be of little use either in coordinating activity or in helping to deal with resource limitations. However intentions should not be irrevocable because circumstances may change and it is not always possible to correctly predict the future. These desiderata mean that general policies (*conventions* [22]) for governing the reconsideration of intentions are needed and also that agents need to monitor the achievement of their intentions. Such tracking is necessary because agents care whether their attempts succeed; failure using one approach means they may replan and try another.

2.2 Limitations of Individual Intention Approaches

Intentions meet some of the desiderata for a model of collaborative problem solving which were specified in section one. They define an individual's local behaviour when everything is progressing satisfactorily (i.e. honour commitments) and, through the definition of conventions, offer some insight into the types of difficulties which may arise during problem solving (eg that the objective is already satisfied, that the objective will never be satisfied or that the motivation is no longer valid [16]).

However intentions only define individual behaviour and, as such, are an insufficiently rich base on which to build a principled representation of collaboration. Social acts need representations which address higher order units of analysis (i.e. groups and teams). There are two main limitations with the individualistic approach. Firstly, joint action is more than just the sum of individual actions, even if the actions happen to be coordinated. For example it would be somewhat unrealistic to claim that there is any real teamwork involved in ordinary automobile traffic, even though the drivers act simultaneously and are coordinated by traffic signs and rules of the road [23]. To represent the "something else" part, formalisms and structures specifically related to collaborative activity are needed.

Secondly there is a fundamental difference between individuals and groups. This can be most strikingly illustrated by considering the notion of commitment. Group commitment cannot simply be a version of individual commitment where a team is taken to be the agent because members of teams may diverge in their beliefs whereas an individual cannot [24]. If an individual comes to believe that a particular goal is impossible, then it is rational for it to give it up. Similarly when the team's overall objective is impossible, the team must stop. In the former case the individual can drop the goal because it knows enough to do so. However in the latter case the whole team may not know enough to do so. For example, consider a situation in which a group of agents are attempting to lift a table and one of them observes that it is nailed to the floor - meaning the overall objective cannot be attained. However this observant agent cannot reasonably assume that all the other team members have been able to make this observation and the corresponding deduction. Hence although not every member believes that the goal is achievable, the whole team does not yet believe that the goal is unachievable.

2.3 Joint Intentions

To overcome the limitations of individual intentions, researchers started investigating joint intentions [23, 24, 25, 26, 27, 28, 29, 30] - indeed it has been argued that the basic premises of commitments and conventions which are central to the joint intention work are the basis of all coordination mechanisms [22]. Joint intentions can be intuitively defined as a joint commitment to perform a collective action while in a certain *shared mental state* [24]. This means collaboration is not an intrinsic property of the actions themselves, rather it is dependent on the mental state of the participants. Thus two groups of agents could be performing exactly the same actions - one could be acting collaboratively, if they share the necessary mental state, the other not. Previous work has addressed various aspects of this shared mental state; although no individual formulation is anything like complete. A comprehensive review of these extant models highlighted the list of important features given below [22]; agents must:

- agree on a common goal [24, 25, 26, 27, 28, 30]

- agree they wish to collaborate to achieve their shared aim [24, 25, 28, 30]

- agree a common means (plan) of reaching their objective [25, 26]
- acknowledge that actions performed by different agents are related [25, 26, 29]
- have criteria for tracking the rationality of their commitments [24, 25]
- have behavioural rules which define how to behave locally and towards others both when joint action is progressing as planned and when it runs into difficulty [24, 25]

Almost all of the formulations state that joint action requires an objective the group wishes to achieve and a recognition that they wish to achieve it in a collaborative manner. This recognition may be through necessity (no individual is capable of solving the problem alone) or through belief that the best means of tackling the problem is to collaborate with others. The shared objective provides the glue to bind individuals' actions into a cohesive whole and the fact that it should be pursued collaboratively allows it to be distinguished from the situation in which a collection of agents independently have a goal which just happens to be the same. This distinction is important because the two relationships imply different consequences in social interaction; the latter results in cooperation and coordination whereas the former gives rise to competition if resources are scarce. Individuals may be recruited into the group simply by making a request if agents are benevolent towards one another or may involve persuasion and negotiation if they are more autonomous in nature. Once a common goal has been agreed, the group becomes committed to achieving it in a collaborative manner.

Many accounts fail to acknowledge that existence of a common aim is not sufficient to guarantee that cooperative problem solving will ensue. Consider, for example, a country in which both the government and its national bank have the objective of lowering inflation, that they wish to achieve this by operating in a coordinated fashion and that they are both mutually aware of this fact (i.e. they have a joint goal). Unless both parties are also capable of agreeing upon a common means for achieving their objective, there will never be joint action despite the fact that there is a joint goal. The common solution provides a context for the performance of actions in much the same way as the shared aim guides the objectives of the individuals. At every stage there must be a commonly agreed (partial) solution which the agents are working under[a]. Explicitly including such a notion in the definition places an additional functional role on joint intentions; namely that they will drive the participants' actions and behaviour towards agreement of a common solution.

This stipulation does not imply that the common solution must be developed before joint action can commence nor that it cannot evolve over time. Rather it reflects the fact that team members must believe that eventually they will be able to agree upon, and work under, a common solution with respect to their shared objective. Agents must be convinced that the group is capable of reaching an agreement and that if all goes well this solution will be adhered to and the desired objective will ensue as a consequence of the actions taken. At any instant in time the common solution is likely to be partial. It may be *temporally partial* in that the exact ordering between some plan elements may not be

[a] As shown later, if an agent becomes uncommitted to the solution then this may no longer be the case; however this situation will be ignored for the present.

specified because it does not affect the plan [18]. It can also be *structurally partial* reflecting the fact that agents frequently decide upon objectives they wish to achieve, leaving open for later deliberation questions about the means.

Developing or refining the common solution is a complex activity, possibly involving persuasion and the resolution of conflicts. Participating agents need to augment their individual intentions to comply with those of others because actions performed by different agents are intertwined and therefore need to be synchronised. There are several paradigms which may be used to generate the common plan [7, 31, 32, 33] - it may be undertaken before any action has been started or interleaved with execution, it may be carried out by one agent or in a collaborative fashion.

Assessment criteria and the concomitant causal link to behaviour are especially important when agents are situated in changing and complex environments. In such circumstances it is difficult to ensure that a group's behaviour remains coordinated because initial assumptions and subsequent deductions may be incorrect or inappropriate. Therefore a comprehensive model of collaboration must provide a grounded basis from which robust problem solving communities can be constructed. Robustness is also a prime consideration when agents decide they will work together because they believe a team approach is the best means of solving the problem (contrast this with situations in which agents *have* to work together to solve a problem). In this case it is especially important that the benefits of collaboration outweigh the overhead associated with coordination activities. One way in which this can be achieved, and hence one of the advantages of team problem solving, is to ensure that the group as a whole is more robust against failures and unanticipated events than any of its individual constituents.

Cohen and Levesque's work on joint intentions [23, 24] makes an initial stab at defining criteria for tracking the rationality of commitments and of prescribing how to behave both locally and towards others. They state that each team member should remain committed to the common aim until it believes that one of the following is true: the group's objective is satisfied, the objective will never be satisfied or that the objective is not relevant because the original motivation is no longer present. Until one of these conditions prevail, agents have a *normal achievement goal* to bring about the group's objective. This ensures they will strive to ensure the common aim is fulfilled.

Joint responsibility [25, 34] extends these ideas to include plan states. Responsibility specifies that each individual should remain committed to achieving the common objective by the commonly agreed solution (*individual solution commitment*) until one of the following becomes true: the desired outcome of a plan step is already available, following the agreed action sequence does not achieve the desired consequence, one of the specified actions cannot be carried out or one of the agreed actions has not been carried out. Whilst in this state the agent will honour its commitments to its agreed actions.

An agent cannot simply abandon a joint action once it is no longer committed to the shared objective or to the agreed means of attaining it because its accomplices may not have been able to detect that there is a problem (recall the collaborative table lift example of section 2.2). As they are still committed, they will continue to honour their commitments and carry out the processing associated with the joint action. If one of the team members believes that this processing is inappropriate, because it knows that it is no longer rational to be committed to the common objective or the means of attaining it, then it must disseminate this information as widely as possible. For this reason the responsibility model stipulates that when a team member is no longer jointly committed to the joint action it must endeavour to ensure that all of its acquaintances mutually believe[b] that it is no longer committed and the reason for this change of state. This enables the team to reassess the viability of the joint action and in particular the actions involving the agent which is no longer committed. So if the joint action needs to be abandoned or refined, the amount of wasted resource in the community is kept to a minimum because futile activities are stopped at the earliest possible opportunity.

An important reason for distinguishing between goal and plan states becomes evident by examining what happens after the two different types of commitment failure. In the former case, the group's action with respect to the particular goal is over. If the goal is achieved, the group has satisfactorily completed its objective; if the motivation is no longer present or the objective will never be attained, there is no point in continuing. However if the group becomes uncommitted to the common solution there may still be useful processing to be carried out. For instance if the plan is deemed invalid, the agents may try a different sequence of actions which produce the same result. Similarly if the plan is violated, agents could reschedule their activities and carry on with the same plan. Thus dropping commitment to the common solution plays a different functional role to that of dropping a goal. By highlighting these two distinct roles, a clearer guidance is given to the system designer in terms of the functionality to be supported and when it is likely to be invoked.

3. A Belief-Desire-Joint-Intention Architecture for Collaborative Problem Solving

Joint responsibility defines the role that joint and individual intentions play in the collaborative problem solving process. They are used both to coordinate actions (future-directed intentions [15]) and to control the execution of current ones (present-directed intentions [15]). Examination of the formal definition of responsibility [34] also reveals that agents need to possess other mentalistic notions - particularly prominent are beliefs (information the agent believes to be true of itself, its acquaintances and its external environment) and desires (what the agent wants to achieve). Architectures in which such mental states play a central role are grouped under the generic term of Belief-Desire-Intention (BDI) architectures [36]. Previous BDI proposals are reviewed in section five.

From the mental state description of joint responsibility it is possible to identify the associated computational processing components of a high-level agent architecture. Section 3.1 concentrates on the role of future directed intentions in coordinating group problem solving activity, whilst section 3.2 deals with the way in which intentions can be

[b] Mutual belief is the infinite conjunction of beliefs about other agents' beliefs about other agents' beliefs and so on to any depth about some proposition [35]

used to monitor and control task execution. These descriptions are at a suitably high level of abstraction to ensure there is significant leeway in how the concepts will be realised. This is in the spirit of other knowledge level descriptions in which knowledge is characterised entirely functionally, in terms of what it does, not structurally in terms of physical objects with particular properties and relations [37]. Therefore the architecture does not specify how intentions are represented, how commitment is described, what mechanisms are used to obtain agreements nor how to develop the common solution.

3.1 Coordinating Actions

As figure 1 illustrates, an agent has to be aware of two sources of events; those which occur locally (eg as a result of local problem solving or of changes in the environment which the agent knowingly causes) and those which occur elsewhere in the community. The latter may be detected directly, through receipt of a message, or indirectly through perception or deduction. With respect to future directed intentions, the monitor-event process has to identify when potential new objectives are raised - the same process also plays a part in the present directed role of intentions (see section 3.2). For instance an agent may realise there is a fault in the network and that it ought to start a diagnosis process or it may deduce that the cause of a fault has been located and restoration activities should commence.

In order to describe the operation of this architecture, assume an event which signifies the potential need for fresh activity has indeed been detected by the monitor event process. This new objective serves as input to the means-end reasoning process and the agent must determine whether it should be met and, if it should, how best to realise it. This reasoning process involves more than simply starting fresh activity for each new objective which is deemed sufficiently desirable. For instance the agent may already have an active intention which is capable of satisfying the new objective, in which case the means-end analysis should conclude that this activity should be utilised.

When deciding whether to adopt a new objective, the agent must consider its library of recipes[c] and its current intentions. The recipe library constrains the courses of action which are available and also the type of activity required to carry them out. It indicates whether the objective can be satisfied locally (i.e. there is a recipe whose actions can all be performed locally), whether it necessitates social activity (i.e. there is no local recipe which achieves the desired objective) or whether a choice between the two alternatives is required (i.e. there is both a local recipe and one involving other agents). Existing intentions must be taken into consideration because they reflect activities the agent has already indicated that it will perform and therefore to which it must devote resources. Thus if an agent is already committed to several important and time consuming tasks, it is irrational for it to take on fresh activity which is relatively unimportant or which could be performed elsewhere in the community. The output of the means-end reasoning is either a decision not to pursue the objective at all, to pursue it locally or to pursue it in a collaborative fashion.

[c] Recipes are sequences of actions known by an agent for fulfilling a particular objective [18].

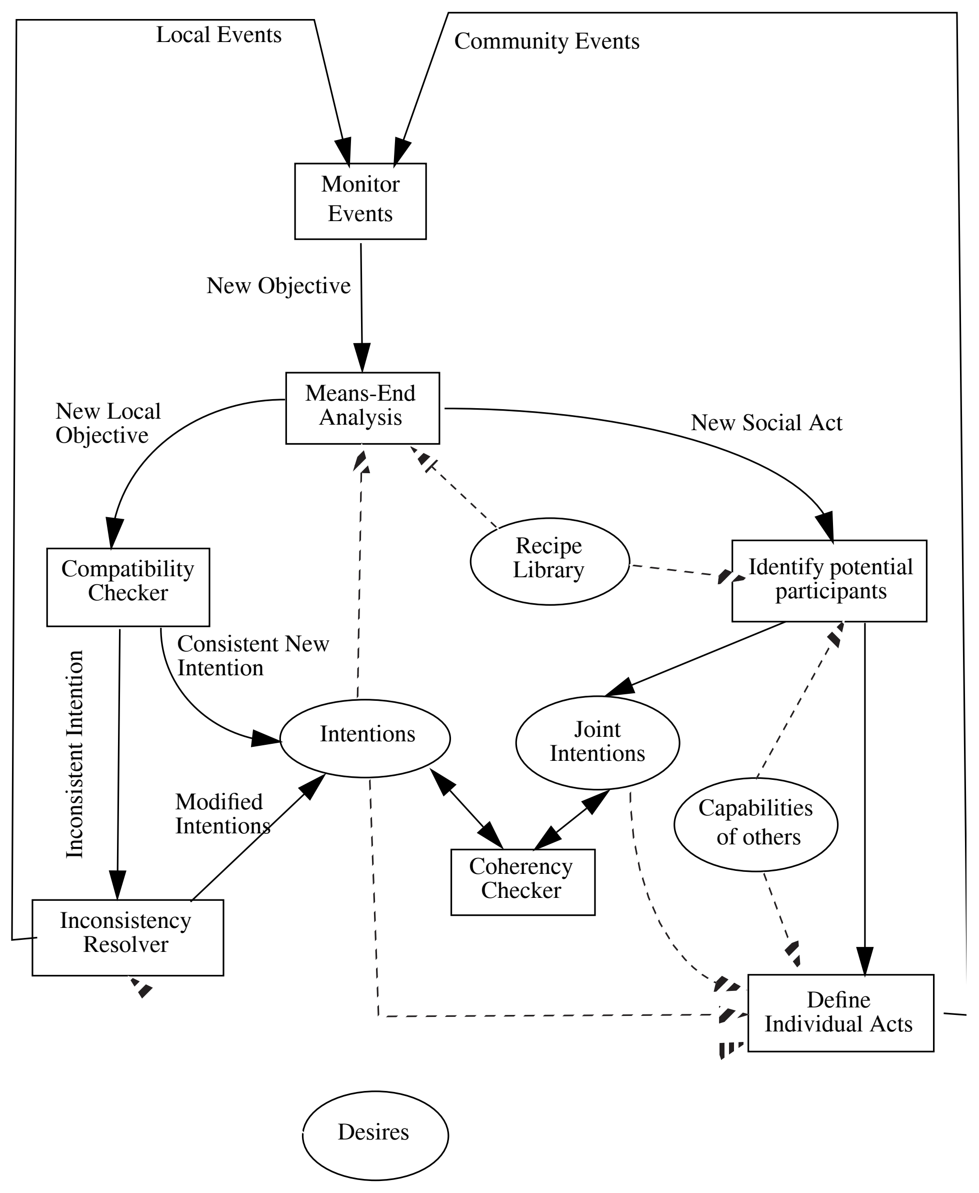
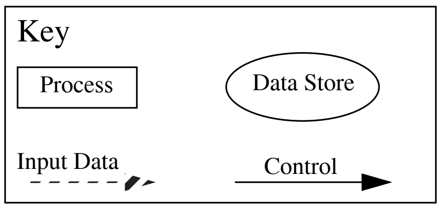


Fig. 1. Functional Agent Architecture - Future directed intentions



Note: The belief data store is not represented explicitly, but it provides input to each of the processes and is modified by the event monitoring task

If the agent decides to pursue the objective locally, it must ensure this new intention is compatible with its existing ones. Compatibility means that it does not conflict with anything that the agent has already committed itself to. For example for an agent capable of working on one task at a time, the decision to perform task t_1 from time 6 to time 11 is inconsistent with an intention to perform task t_2 from time 9 to time 14. In addition to time, consistency can be defined with respect to any scarce or non-shareable resource such as communication bandwidth, environmental state or physical space. If there are no inconsistencies, the new objective is added to the list of individual intentions and the agent commits itself to performing it.

If the new objective conflicts with existing intentions, the inconsistency must be resolved; either by modifying the existing commitments or by altering the new objective so that it is no longer in conflict. An important consideration in such situations is the agent's desires. If the new task is less important (desirable) than existing ones, then it is the one which should be modified; conversely if it is more desirable then it is the existing one which should be adapted. Alteration may vary from delaying an intention until the competing one is finished, to deciding that it cannot be accommodated in its present form. In many cases, the outcome of the conflict resolution process will require a further round of means-end analysis to be undertaken to resolve any remaining discrepancies. Any modifications made by the resolution mechanisms to existing intentions have to be reflected in the intention representation.

The other outcome of the initial means-end reasoning process may be that the agent decides the new objective is best met collaboratively. In this case a social action must be established. The first phase is to identify those agents in the community who are potentially interested in being involved. The means by which this is achieved is not specified at this level. So one agent may dictate to the others that they should participate (i.e. a hierarchically controlled system) or there may be a negotiation phase if agents are more autonomous. The process of establishing the social action will be guided by several factors; including defining the intended organisation of the group (will it be flat or hierarchical? will the solution be decided by one agent or a group of them?), the strategy for group formation (eg large or small groups?) and the known capabilities of other community members. Once the potential members have been identified a skeletal joint intention exists in which potential participants have registered their interest but no firm selection of either the final team or the common solution has been made.

Joint intentions cannot be executed directly, their role is to serve as an overarching description and problem solving context which binds the actions of multiple agents together. It is the group members who have the ability to act and hence only individual intentions are directly related to problem solving actions. However there is a causal link in that each team member would be expected to adopt at least one individual intention as a consequence of its participation in a joint action. Also there must be a coherency between the two representations. So, for example, if an individual has an intention to perform task t_1 from times 6 to 9, this must be consistent with any of the related actions in the joint intention. Consistency between individual and joint intentions is ensured by the coherency checker which is invoked every time a new individual or joint action proposal is made. Its aim is to ensure that there are no conflicts between an agent's local and social perspectives and that the agent is in a position to honour all its commitments.

The outline proposal for team members and common recipes must then be finalised. Decisions about the mapping between the actions to be carried out and the individuals within the group who can undertake them must be made. This process is guided by factors such as strategy, desired organisation and known capabilities as discussed previously. The specification of individual acts may be done in an incremental fashion or all in one go. Again these options are left open, allowing the designer to tailor them to best fit the particular circumstances. The output of this process is passed onto each member of the group and ultimately requires them to perform local means-end reasoning to fit the primitive actions in with their existing commitments whilst satisfying the relationships with their associated actions. Any inconsistencies which arise during this process must be detected by the consistency checker and may require the mapping from joint to individual intentions to be re-examined.

3.2 Monitoring and Executing Actions

As well as providing a means for coordinating actions, intentions also act as a guide for task execution and monitoring. In this present directed role, the responsibility model specifies that a group of agents $\alpha_1 ... \alpha_n$ collaborating to achieve a joint goal σ using a common solution Σ should behave in the following manner:

```
JOINT-RESPONSIBILITY({$\alpha_1 ... \alpha_n$}, $\sigma$, $\Sigma$) $\equiv$
FORALL $\alpha_i$ ε { $\alpha_1 ... \alpha_n$}
   WHILE Normal-Achievement-Goal($\alpha_i$, $\sigma$) AND Individual-Solution-Commitment($\alpha_i$, $\sigma$, $\Sigma$) DO
         PARALLEL
            Honour commitments
            Monitor rationality of commitment to $\sigma$
            Monitor rationality of commitment to $\Sigma$
         END-PARALLEL
   Suspend processing of local activities related to $\Sigma$
   CASE Reason for non commitment OF
      NOT Normal-Achievement-Goal($\alpha_i$, $\sigma$): Abandon commitments to $\sigma$ and subsequent actions in $\Sigma$
      NOT Individual-Solution-Commitment($\alpha_i$, $\sigma$, $\Sigma$): IF remedial action available
                                                  THEN select possible remedies
                                                  ELSE seek assistance
   END-CASE
   Inform other team members of lack of commitment, reason for commitment failure and remedial
            action if it exists
```

Fig. 2. Joint Action Monitoring and Execution According to the Responsibility Model

This behavioural description can be expanded to give a more detailed functional architecture for the present directed role of joint intentions (figure 3). This architecture embodies the fact that intentions are used to control the execution of actions and also the fact that commitments must be continually monitored to ensure they are still rational.

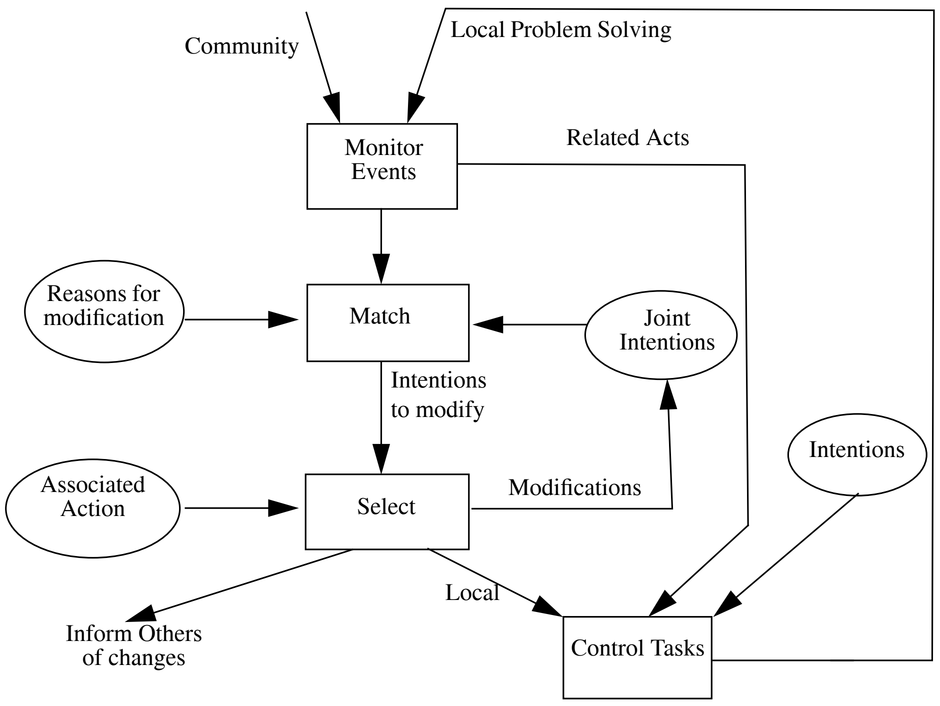


Fig. 3. Functional Agent Architecture - Present Directed Intentions

With respect to executing actions, it is the responsibility of the control-tasks process to ensure commitments are honoured. So if an agent commits itself to start task t_1 at time 10, when time 10 arrives, t_1 must actually be started. This process may also suspend or abort tasks to ensure commitments are kept. Control-tasks must be kept informed when related actions have been performed, either locally or by other community members, so that its actions can be synchronised and any relationships upheld.

With respect to monitoring commitments, the responsibility model provides a clear separation between identification of the reasons why a joint action ought to be modified or terminated and the choice of actions which should be taken in such circumstances. The “reasons for modification” component provides a domain-independent characterisation of collaborative problem solving; defining conditions under which joint actions should be terminated or re-examined based on the responsibility model (eg objective has been satisfied, goal motivation no longer present, plan invalid, etc.). These criteria can be explicitly modelled and enable the agent’s tracking and subsequent recovery to be based on a firm theoretical grounding.

Once the match process has identified joint intentions which need to be re-examined, the agent must decide which remedial actions, if any, should be taken. Such actions may include rescheduling the plan, entering a replanning phase or abandoning the joint action completely. The "associated action" model describes appropriate activities for whole classes of events. For example when an agent drops its commitment to the commonly agreed solution, it must stop processing any of its associated local actions, inform others that it is no longer committed and then enter a replanning phase. There may be many different types of event which cause commitment to the common solution to be dropped, but using the responsibility model they can be grouped together since they require the agent to respond in a similar manner. If commitment to the overall goal is dropped, the agent must again stop all its associated current and planned actions and ensure that the other team members are informed immediately.

4. GRATE*: Realising the Belief-Desire-Joint-Intention Architecture

The functional architectures of the previous section define the computational processes necessary to support the responsibility model. As with all high-level architecture proposals there are a number of potential realisations of which GRATE* is but one. This section describes GRATE*'s implementation architecture (§ 4.1) and deals with the mappings with the schemes of figures one (§ 4.2) and three (§ 4.3). Then the implementation of the consistency checking component is dealt with (§ 4.4).

4.1 Implementation Architecture

GRATE* agents have two clearly identifiable components - a *cooperation and control layer* and a *domain level* system (see figure 4). The latter solves problems for the organisation; be it in the domain of industrial control, finance or transportation. Problems are expressed as tasks - atomic units of processing when viewed from the cooperation layer. The cooperation layer is a meta-level controller which operates on the domain level system. Its objective is to ensure that the agent's domain level activities are coordinated with those of others within the community. It decides which tasks should be performed locally, determines when social activity is appropriate, receives requests for cooperation from other community members and so on.

The cooperation layer has three main problem solving modules. Each module is implemented as a separate forward-chaining production system with its own inference engine and local working memory. Communication between the modules is via message passing. The rules built into each problem solving module are generic and have been used to control cooperative activity in a broad class of industrial applications [14, 38, 39, 40]. The *control module* is the interface to the domain level system and is responsible for managing all interactions with it. The *situation assessment module* makes decisions which affect both of the other two modules. It decides which activities should be performed locally and which should be delegated, which requests for cooperation should be honoured, how requests should be realised, what actions should be taken as a result of freshly arriving information and so on. It issues instructions to, and receives feedback from, the other modules. Typical requests to the cooperation module include "get information X" and "get

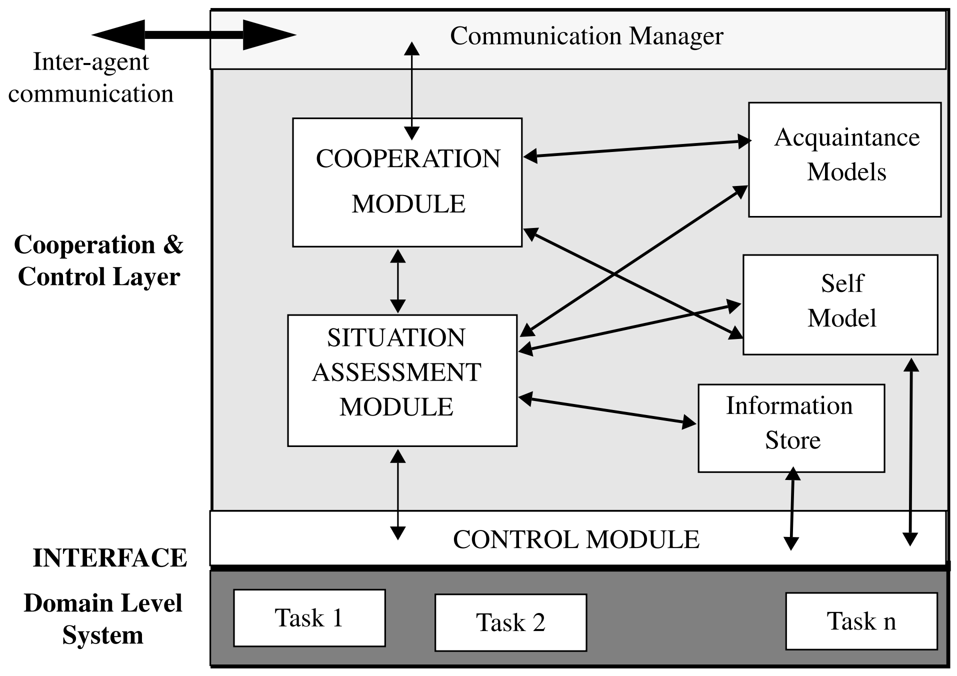


Fig. 4. GRATE* implementation architecture

task Y executed by an acquaintance". Requests to the control module are of the form "stop task T1" and "start task T2". The *cooperation module* is responsible for managing the agent's social activities, the need being detected by the situation assessment module. These activities can be divided into three distinct categories. Firstly, new social interactions have to be established (e.g. find an agent capable of supplying a desired piece of information). Secondly, ongoing cooperative activity must be tracked (see section 3.2). Finally cooperative initiations from other agents must be responded to (e.g. if a request is received for information i, then the agent must determine whether it is both capable and willing to carry out the necessary processing to produce it).

The remaining components provide support functions. The *information store* is a repository for all the data which the underlying domain level system has generated or which has been received as a result of interaction with others. *Acquaintance and self models* are representations of other agents and the local domain level system respectively. They represent information about the state, capabilities, intentions, evaluations, etc. of the agent being modelled [14]. The communication manager sends messages to and receives messages from others in the community. The following key primitives were used to support the GRATE* implementation of the responsibility model; all messages are automatically stamped with a sender, a unique identifier and a time.

- `(PROPOSE-GROUP <name> <motive> <potential-participants>`
  `<importance> <expected-outcome>)`

  Sent by the agent which detects the need for collaborative problem solving to acquaintances it believes will be interested in being involved. Indicates the name of the joint action, why it is proposed to carry out the action, other agents who are being contacted, a measure of the action's importance and the results which can be expected. Recipients respond by registering their interest or stating that they do not wish to be involved.

- `(CONTRIBUTION-PROPOSAL <name> <action> <time> <context>)`

  This message is sent once the individual actions and the agents who will execute them have been decided upon. It informs each agent of the actions it has been chosen to perform, the time at which they should be executed and their execution context (relationships which must be satisfied before the action can be started). Recipients can either accept the proposal or suggest a different (later) time which fits in better with their existing commitments.

- `(FINAL-AGREEMENT <name> <participants> <solution>)`

  Once all the actions have been assigned times and actors, the final agreed solution for the joint action is sent to all the relevant participants.

- `(INFORM <information>)`

  Used to convey information deemed relevant by the sender to the recipient. Examples include the fact that a particular information has been produced or that a particular task has finished executing.

- `(COMMITMENT-FAILURE <name> <reason> <remedy>)`

  Indicates that something has gone wrong with the joint action and that the sender is no longer committed, specifies the reason for this change of state (according to the responsibility model definition) and gives the proposed remedy if it exists.

The following functional architecture components of figure 1 are mapped into GRATE*'s situation assessment module: monitor events, means-end analysis, compatibility checker, inconsistency resolver, consistency checker and defining individual acts. The cooperation module is responsible for identifying the potential participants. An agent's recipe library, intentions, joint intentions and desires are stored in the self model and the capabilities of others are stored in the acquaintance models. In terms of figure 3, the match and select processes are present in the situation assessment module, the control of local activities in the control module and the informing of others in the cooperation module.

4.2 Establishing Joint Action

To establish joint activity, one or more agents must firstly recognise the need - the agent which does this is deemed the *organiser* or leader. Each social action has one organiser and at least one team member (an acquaintance who has (tentatively) agreed to participate). The leader's role involves contacting all the acquaintances which it would like to be involved in the joint action, deciding upon the common solution recipe, assigning the actions to team members and determining when the actions will be performed. Any community member has the potential to be an organiser and there is no restriction on the number of joint actions which can be active at any one time.

The need to start fresh activities is detected by the organiser's situation assessment module. As figure 1 shows, the organiser's first concern is to determine whether it will be able to complete the act alone or whether it requires assistance from others. This decision is taken by computing the number of tasks from the chosen recipe which the agent is unable to complete by itself. There are three potential outcomes of this analysis; joint action is definitely needed, the action can be solved entirely locally or that some assistance is required (e.g. sharing of information and processing capabilities) but not sufficient to warrant a full-scale joint action. The rationale for the last option is that it may not be worth incurring the overhead associated with a distributed planning approach [13] in order to satisfy a small number of interdependencies. In this case assistance will be asked for as and when it is required.

Once the need for joint action has been ascertained, the responsibility model requires the following conditions to be fulfilled before a social can commence; (i) other community members who are willing to participate must be identified; (ii) the fact that a common solution is required needs to be acknowledged; (iii) adherence to the responsibility convention must be agreed; and (iv) the common solution by which the group's objective will be attained must be developed. Rather than having a protracted protocol in which each precondition is agreed upon separately, a more concise version in which several conditions are settled in a single message interchange is more appropriate for industrial control domains. In other applications, however, it may be perfectly legitimate and desirable to go through each phase separately. GRATE* uses a two phase protocol to satisfy these preconditions. The first phase establishes the common recipe, the agents who will participate and the ground rules which they must adhere to; the second phase specifies the detailed actions and their timings.

To provide a firm basis for the discussions of this section, the concepts will be illustrated using the real world problem of electricity transportation management [14]. The exemplar scenario involves three agents; the alarm analysis agent (AAA) which acts as the organiser, the blackout area identifier (BAI) and the control system interface (CSI). This group of agents collaborate to detect and locate faults - a more detailed description of their interactions is presented in [41]. The CSI continually receives status information about the electricity network which is analysed to determine whether a disturbance has occurred. If so, this information is passed onto the AAA and BAI which attempt to locate the fault using different perspectives of the data and to differing levels of accuracy. The AAA performs a two phase diagnosis, a fast approximate one which turns up a large number of potential hypotheses and then a detailed case by case examination. This continues until the element at fault is located. The BAI can only locate the approximate region out of fault, however if this information is supplied to the AAA in a timely fashion then it can be used to reduce the number of hypotheses which need to be considered in the detailed diagnosis phase. Meanwhile the CSI continues to monitor the evolving state of the network and pass results on to the other two agents.

After establishing the desirability of cooperation, the organiser instantiates a representation of the joint intention in its self model (figure 5). The motivation slot indicates the reason for carrying out the joint intention - in this example only one motive is shown, the fact that there is a fault in the network, but in practice there may be several such reasons. The recipe is a series of actions which need to be performed, together with some partial ordering constraints, to produce the desired outcome. A more detailed specification of the primitives of the recipe language is contained in [42]. The recipe indicates what is to be done, not whom is to do it nor the time at which it should be done. The detailed timings and duration of the action are left until the second phase of the protocol.

```
Name: (DIAGNOSE-FAULT)
Motivation: ((FAULT-IN-NETWORK))
Recipe:
     (((START-OFF (IDENTIFY-INITIAL-BOA ?INITIAL-ELTS-OUT-OF-SERVICE)))
     ((START-OFF (GENERATE-TENTATIVE-DIAGNOSIS
                         ?DISTURBANCE-DETECTION-MESSAGE
                         ?BLOCK-ALARM-MESSAGES
                         ?INITIAL-FAULT-HYPOTHESES)))
     ((START-OFF (MONITOR-DISTURBANCE)))
     ((START (PERFORM-FINAL-HYPOTHESIS-DIAGNOSIS
                         ?INITIAL-ELTS-OUT-OF-SERVICE
                         ?INITIAL-FAULT-HYPOTHESES))))
Start Time: -                    Maximum End Time: -
Duration: -                      Priority: 20
Status: ESTABLISHING-GROUP       Outcome: (VALIDATED-FAULT-HYPOTHESES)
Participants: ((SELF PROPOSER AGREED-OBJECTIVE)
               (CSI TEAM-MEMBER AGREEING-OBJECTIVE)
               (BAI TEAM-MEMBER AGREEING-OBJECTIVE))
Bindings: NIL
Proposed Contribution:
       ( (SELF   ((GENERATE-TENTATIVE-DIAGNOSIS) (YES))
                 ((PERFORM-FINAL-HYPOTHESIS-DIAGNOSIS) (YES)))
         (BAI ((IDENTIFY-INITIAL-BOA) ?))
         (CSI ((MONITOR-DISTURBANCE) ?)))
```

Fig. 5. Representation of joint intention in AAA's self model

The priority slot indicates the local agent's assessment of the importance of the intention and is used as the basis for computing its desirability. Values must be positive integers and the higher the value the more desirable the action. Priorities have two components; an intrinsic (static) value associated with each recipe and a dynamic component which reflects the current problem solving context. The intrinsic priority of the diagnose fault recipe is 10. The second component is used to provide flexibility and takes into account the agent's current problem solving state (e.g. whether it is working on a local action, a joint action or a social request). Each distinct type of action has an associated desirability rating, set by the system developer (e.g. local action = 15, joint action = 10, social request = 2), which is combined with the intrinsic value to give the intention's desirability. Thus as the diagnose fault intention is invoked as a joint action it has priority 20 (10 for the intrinsic value and 10 for being used in the context of a joint action). If an intention has more than one motivation (e.g. part of a joint goal and part of a social action request), then both context values are added to the intrinsic one. There is a straightforward mapping from the priority of a joint action to that of its constituent components. If the joint intention has priority X and it is composed of N tasks, then each will be assigned priority X / N. Thus identify black out area, generate tentative diagnosis and so on will each have a priority of 5 (20 / 4).

Status refers to the current phase of the distributed planning protocol and can take on the value "establishing-group", "developing-solution" or "executing-joint-action". The outcome slot enumerates the expected results of performing the intention. The participants slot indicates the organisational structure of the group and the current stage of each agent's involvement. In the diagnose fault joint action, there is one team organiser (AAA) and two other potential team members (BAI and CSI). The AAA has acknowledged that it is interested in fulfilling the objective, since it was the one who suggested it in the first place, and is in the process of establishing whether the other two wish to joint it. The bindings slot is used exclusively in the second phase of the protocol.

Proposed contribution records those agents who are adjudged, by the organiser, to be capable of contributing to the joint act and whether or not they have agreed to make that contribution. It can be seen that the AAA has consented to contribute by performing the "generate tentative diagnosis" and "final diagnosis" recipes. The BAI and CSI have not yet agreed, or even been asked, to contribute anything, though the organiser has already defined privately what roles it would like them to play. In this example there is only one agent indicated per task, although this need not always be the case. The organiser may establish whether several agents are interested in contributing to a particular activity and then decide between them at a later stage when more information is available.

Having identified the need for joint action, the process of establishing it can commence. The first phase of the protocol is to ascertain which acquaintances are ready and able to participate (figure 6). Firstly the organiser's situation assessment module informs its cooperation module that a social act is required (action 1). The cooperation module identifies those acquaintances who it believes can contribute to actions which are part of the recipe, based on capability knowledge represented in its acquaintance models. The strategy guiding this process is to identify the maximum number of agents who are capable of participating so that the organiser is able to take better decisions in the second phase of the protocol. After the potential contributors have been identified, they are each sent a `propose-group` message requesting their participation in the joint action (act 2).

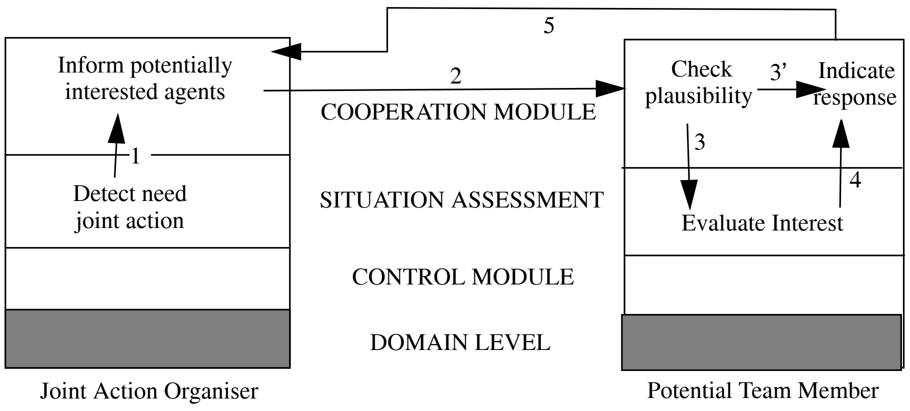


Fig. 6. Establishing a joint action

Each participant receives the proposal in its cooperation module and checks that it understands the request. This involves ensuring the agent is willing to accept new requests and that it has the potential to carry out the requested activity. If the request is rejected, a negative response is prepared (action 3') and passed back to the group organiser. Assuming the request is accepted at the cooperation module level, it is then passed to the situation assessment module (action 3) to see whether the agent has sufficient resources to tackle the problem. This involves analyzing existing commitments to ensure the proposal is consistent and can be accommodated successfully - this process is elaborated upon in section 4.4. The result of this evaluation, either yes or no, is returned to the cooperation module (action 4) which passes the answer back to the organiser (action 5). If the agent does wish to participate, it sets up a joint intention description in its self model.

By answering affirmative to the group initiation response, agents acknowledge that they are interested in participating in the collaborative fault diagnosis, that a common solution is needed to solve the problem and that they will obey the responsibility convention for that action. These criteria are not communicated explicitly because all agents have a common understanding of the semantics of performing a joint action.

This phase of the planning protocol identifies those agents who, in the opinion of the team organiser, are potentially able to contribute something useful. This is achieved by contacting a chosen subset of the community about specific activities, an approach which works well when the organiser has a good model of other agents in the community. In industrial control applications the community is fairly static and hence capability knowledge, such as which agents are able to perform which actions, can be assumed to be accurate and comprehensive. However in more open environments in which agents enter and leave the community frequently this approach may be less appropriate.

Once all the community members who were identified as potential participants have replied, the second phase of the planning protocol is entered into. In this phase the fine details of the common solution, including the exact action timings, are agreed upon. The organiser updates the joint action's status slot to "developing-solution" and modifies the participants slot to reflect the fact that the group is now in the solution planning phase.

From the agents who are willing to participate, the organiser selects those it wishes to be in the team and the contributions it would like them to make. The criteria upon which selection is based, in this instance, is to minimise the number of group members. This strategy was chosen because the fewer agents there are in the team the lower the likelihood of one of them defaulting, since its individual loss would be greater. This strategy also reduces the coordination overhead because messages need to be sent to fewer agents. In other instances it may be desirable to have the maximum number of agents in the team to balance the load throughout the community or to select the most competent agent to perform each task. The team leader updates its proposed contribution slot to indicate those agents which were selected and those which were not.

```
Proposed Contribution:
    ((SELF   ((GENERATE-TENTATIVE-DIAGNOSIS) (YES SELECTED))
             ((PERFORM-FINAL-HYPOTHESIS-DIAGNOSIS) (YES SELECTED)))
    (BAI ((IDENTIFY-INITIAL-BOA) (YES SELECTED)))
    (CSI ((MONITOR-DISTURBANCE) (YES SELECTED))))
```

The organiser then makes an initial proposal for the timings of the individual actions which is represented in the bindings slot. Start, end and duration times are also filled in.

```
Bindings: (   (BAI (IDENTIFY-INITIAL-BOA) 19)
              (SELF (GENERATE-TENTATIVE-DIAGNOSIS) 20)
              (CSI (MONITOR-DISTURBANCE) 21)
              (SELF (PERFORM-FINAL-HYPOTHESIS-DIAGNOSIS) 36))
```

The proposal takes into account the fact that some time is required to agree the solution. Work cannot commence immediately because for each action which needs to be performed by an acquaintance at least two messages must be sent to establish its start time. Associated with each of these messages is a communication delay while they are sent and a delay before they are processed by the recipient. A final delay is caused by the fact that a `final-agreement` message must be sent to all community members when the final timings are agreed upon.

Due to the complexities of the application and the agents, the team leader cannot have a complete picture of all activities within the community - it has bounded rationality [43]. In particular the team leader does not know the existing commitments and desires of all its potential team members, so exact timings cannot be *dictated*. To avoid chaotic oscillations and many iterations, the organiser takes each action in the recipe in a temporally sequential order and agrees with the agent concerned the appropriate time at which it should be performed. Thus the BAI is contacted first, about the identify initial blackout area task, and a time is agreed which fits in with its existing obligations. Then the AAA is contacted about the generate tentative diagnosis task and a time agreed, and so on for each of the actions.

Upon receipt of a `contribution-proposal` message, the team member evaluates it to determine whether it is acceptable. If there is no conflict, the agent adopts the intention and sets up an individual intention description[d] in its self model (figure 7). It then returns a message indicating its acceptance to the organiser.

Name: (ACHIEVE (IDENTIFY-INITIAL-BOA))
Motivation: (SATISFY-JOINT-ACTION (DIAGNOSE-FAULT)))
Recipe: (IDENTIFY-INITIAL-BOA)

Start Time: 19	**Maximum End Time**: 34
Duration: 15	**Priority**:5
Status: PENDING	**Outcome**: (Black-Out-Area)

Fig. 7. Individual intention representation for BAI agent

If the suggested time is unacceptable the recipient proposes a (later) time at which the action can be fitted in with its existing commitments, makes a tentative commitment for this time and returns the suggestion to the team leader. If the modified time is acceptable, the organiser will make appropriate adjustments to the solution bindings and will proceed with the next action. If the modified time proposal is unacceptable, the organiser will look for another agent to perform the action from its list of proposed contributors.

The process of agreeing a time for each action continues until all of them have been successfully dealt with. At this point the common solution is agreed upon and the organiser sends out a `final-agreement` message to all team members and the joint action becomes operational. The organiser changes the status of the social action to "executing-joint-action" and the participants slot is updated to indicate that all team members are now in the process of executing the joint action. All the necessary preliminaries for joint action have been satisfied and the group can begin its problem solving in earnest.

4.3 Monitoring Joint Action

Once a joint action has been established, the tracking aspect of the responsibility model comes to the fore. The match process of figure 3 takes place in the situation assessment module and identifies reasons for terminating a joint action. Such circumstances may arise as a result of local problem solving:

[d] There are two slight differences between this representation and that of joint intentions. Firstly, the recipe slot contains the name of a local recipe rather than a list of actions and their relationships. Secondly, the status is simply "executing" or "pending" since there is no distributed planning phase.

```
if   a local task has finished execution and
     it has produced the outcome of the overall joint action
then the objective of the joint action has been met

if   a plan is unattainable and
     the plan is part of a joint action
then the joint action is unattainable
```

or as a consequence of events occurring elsewhere in the community:

```
if   social action component has been delayed and
     delayed component is related to a local act
then joint action plan violation has occurred
```

These rules take events which have occurred locally or elsewhere in the community and determine whether they mean that the collaborative action has become unsustainable in its present form (according to the model of joint responsibility) - respectively the result of a task satisfies the overall objective, the unattainable plan is part of a joint action and the delayed component is related to a local act. If fired, they identify a joint action which needs modification.

Once intentions which require attention have been identified, the agent must decide upon the appropriate course of action. This decision is achieved, in the agent's situation assessment module, by matching the reasons for failure against the possible courses of action and picking the most appropriate remedy. Examples of corresponding repair actions for the rules cited above are as follows:

```
if objective of joint action has been satisfied
then   inform cooperation module joint act successfully finished
       abandon all associated local activities

if a joint action is unattainable
then   suspend local activities associated with joint act
       inform cooperation module joint act unattainable

if   a joint action component has been violated and
     it has been successfully rescheduled
then   suspend current activities associated with joint act
       reset the descriptions and timings of the joint action
       inform cooperation module of violation & give new times
```

As well as taking actions locally, such as suspending or rescheduling tasks, the responsibility model stipulates that the other team members must be informed when an individual's commitment is compromised. This aspect of the model is realised by the cooperation module, based on information provided by the situation assessment module. In the above cases, the assessment module indicates that the joint action has been obtained, that it is unattainable or that the solution has been violated. It is then up to the cooperation module to ensure that the other team members are informed by sending out a `commitment- failure` message.

```
if joint action has been successful
then   inform all other members of successful completion
       see if results should be disseminated outside the team

if receive local indication that joint action is unattainable
then inform other members of need to abandon it

if local joint action violation has occurred
then inform other team members of violation & new timings
```

4.4 Honouring Commitments and Ensuring Consistency

An important aspect of the responsibility model's functional architecture is the notion of consistent intentions and ensuring that any new proposals fit in with existing commitments. In GRATE*, time is the sole criteria on which consistency is judged. Two intentions are deemed inconsistent if the times for which they are scheduled overlap, they are consistent if they are distinct. So if the BAI agent has two intentions, say check restoration premises and update network topology, they will be represented in its self model as shown in figure 8. They are consistent because the times at which they are to be carried out, 19-29 and 38-44, do not intersect.

Name: (CHECK-RESTORATION-PREMISES)
Motivation: ((SATISFY-LOCAL-GOAL (CHECK-RESTORATION-PREMISES)))
Chosen Recipe: CHECK-RESTORATION-PREMISES

Start Time: 19	**Maximum End Time**: 29
Duration: 10	**Priority**: 6
Status: PENDING	**Outcome**: (RESTORATION-PREMISES)

Name: (UPDATE-NETWORK-TOPOLOGY)
Motivation: ((SATISFY-LOCAL-GOAL (UPDATE-TOPOLOGY-USING-SNAPSHOTS)))
Chosen Recipe: UPDATE-NETWORK-TOPOLOGY-USING-SNAPSHOTS

Start Time: 38	**Maximum End Time**: 44
Duration: 6	**Priority**: 3
Status: PENDING	**Outcome**: (UPDATED-NETWORK)

Fig. 8. BAI's existing intentions

Assume a new intention to identify the initial black out area is proposed by the BAI's situation assessment module (figure 9). The situation assessment module's compatibility checker has to determine whether the new proposal is consistent with the agent's existing intentions. As a result of its analysis, the compatibility checker will indicate that the new intention is inconsistent, since the BAI cannot identify the initial black out area and check the restoration premises at the same time.

Name: (ACHIEVE (IDENTIFY-INITIAL-BOA))
Motivation: ((SATISFY-JOINT-ACTION (DIAGNOSE-FAULT)))
Chosen Recipe: (IDENTIFY-INITIAL-BOA)

Start Time: 20	**Maximum End Time**: 35
Duration: 15	**Priority**: 7
Status: PENDING	**Outcome**: (INITIAL-ELTS-OUT-OF-SERVICE)

Fig. 9. New intention proposal

The consistency resolver is invoked in order to make the two intentions compatible. It makes use of the agent's desires, represented by priority values, for each of the intentions in order to reschedule its activities. In this case, identification of the black out area is marginally more important than checking the restoration premises - ratings of 7 and 6 respectively. Therefore the BAI adopts the intention of identifying the initial black out area from time 20 to 35; meaning that the intention to check the restoration premises has been violated and must be modified. The inconsistency resolver process will propose, based on the model of associated repair actions depicted in figure 3, that the best means of remedying this violation is to reschedule the actions. Therefore it attempts to move the check restoration premises recipe to after the black out area has been produced (i.e. from time 36 to time 46). However this conflicts with the agent's intention to update its network topology model. Again the consistency resolver computes its preferences for these two actions and decides that checking the restoration premises is more desirable. Thus it commits itself, all other things being equal, to check the restoration premises from time 36 to time 46. The updating network topology recipe is violated by this rescheduling and so has to be made consistent. It is moved to after the restoration premises have been checked; starting at time 47. Now there are no conflicts and all the BAI's intentions are again consistent, at least until its beliefs change!

Name: (ACHIEVE (IDENTIFY-INITIAL-BOA))
Motivation: ((SATISFY-JOINT-ACTION (DIAGNOSE-FAULT)))
Chosen Recipe: (IDENTIFY-INITIAL-BOA)

Start Time: 20	**Maximum End Time**: 35
Duration: 15	**Priority**: 7
Status: PENDING	**Outcome**: (INITIAL-ELTS-OUT-OF-SERVICE)

Name: (CHECK-RESTORATION-PREMISES)
Motivation: ((SATISFY-LOCAL-GOAL (CHECK-RESTORATION-PREMISES)))
Chosen Recipe: CHECK-RESTORATION-PREMISES

Start Time: 36	**Maximum End Time**: 46
Duration: 10	**Priority**: 6
Status: PENDING	**Outcome**: (RESTORATION-PREMISES)

Name: (UPDATE-NETWORK-TOPOLOGY)

Motivation: ((SATISFY-LOCAL-GOAL (UPDATE-TOPOLOGY-USING-SNAPSHOTS)))

Chosen Recipe: UPDATE-NETWORK-TOPOLOGY-USING-SNAPSHOTS

Start Time: 47	**Maximum End Time**: 53
Duration: 6	**Priority**: 3
Status: PENDING	**Outcome**: (UPDATED-NETWORK)

Fig. 10. Final consistent intentions

5. Relationship with Other Work

Many DAI systems do not embody a principled model of collaboration upon which agents can base their decisions and subsequent actions about cooperation. This shortcoming is one of the reasons why there are so few-real size applications employing DAI technology [13, 34]. Also the theoretical work on developing models of collaboration is, in general, a considerable distance from implementation level systems because of the unrealistic assumptions which are employed. However there is some other work which is trying to bridge this great divide against which GRATE* can be compared.

Bratman *et al.* have proposed a high-level agent architecture in which the mentalistic notions of beliefs, desires and intentions play a pivotal role [36]. Their architecture is based on Bratman's discursive accounts of intentions [15, 21] and the role they play in coping with an agent's inherent resource boundedness. Their proposal is similar to that part of the responsibility agent architecture which deals with individual intentions. However their intention override mechanism is not as well defined since it does not enumerate conditions under which a joint action may fail - it merely recognises that in some instances an agent may wish to reconsider the rationality of its commitments. Although acknowledging that intentions must be tracked, their proposal offers no high level design for this process. Also their proposal concentrates on defining the behaviour of an individual in an asocial situation. There are also no provisions for collaborative activity and no joint intentions.

Burmeister and Sundermeyer have defined and implemented a generic agent model based upon intentions and desires [44]. Their agents are characterised by their perceptive capabilities, the actions they can undertake, their intentions and the resources needed for executing the different types of behaviour. They identify two types of intention - long term ones which correspond to high level objectives and desires and tactical intentions which are intimately related to actions. However their implementation is not based on a well founded theory of intentionality. Although using notions of commitment and long term intentions the semantics of these terms are not rigorously defined (either formally or discursively). Their agents are explicitly designed to operate in environments containing other entities, however they do not represent joint intentions. Interaction between agents is mediated by individual intentions and hence the level of sophistication in collaboration which can be attained by their architecture is limited.

Shoham's work on agent-oriented programming offers a different perspective on the problem [45]. Agents are specialised objects which consist of components called beliefs, choices, capabilities and commitments. This mental state is captured formally in an extension to standard epistemic logic in which the knowledge and belief operators are temporalised. These well-defined and theoretical notions are then used as the basis of a declarative programming regime in which agents and their interactions can be specified. This approach has the advantage that any theoretical model of collaboration expressed in this language can automatically be translated into executable code. This contrasts with the responsibility model definition which first had to be mapped from dynamic and temporal logic into production rules before it could be executed. As it represents a programming language there is no explicit model of either individual or joint intentions. Such a model would have to be constructed from the basic primitives, which, given their fine level of granularity, would require considerable effort.

6. Conclusions

This paper has specified a high-level agent architecture for collaborative problem solving in which intentions and joint intentions play a central role. The inherent limitations of individual intentions for describing collaborative activity have been exposed and a review of existing formalisms for joint intentions has been presented. This review synthesizes and extends a number of these existing formalisms into a unifying framework which clearly identifies and specifies the role of individual and joint intentions in cooperative problem solving. The responsibility model identifies preconditions which must be fulfilled before joint action can commence and prescribes how cooperating agents should behave, in terms of their own activity and their interactions with others, both when the joint action is progressing satisfactorily and when it runs into difficulty. From the mental state description of responsibility, the computational processes required to implement the model are identified and presented in a generic high-level agent architecture. The GRATE* realisation of this architecture is then described in detail with illustrative examples taken from the real-world domain of electricity transportation management.

A series of experiments have been undertaken as a means of assessing the benefits of the approach advocated in this paper. A GRATE* community was compared with one in which interacting agents did not form explicit groups. In the latter, agents exchanged information, shared processing and had individual intentions but there were no joint intentions. These experiments showed that: (i) the responsibility community behaves in a more coherent manner, especially when agents are situated in complex and dynamic environments; (ii) implementing responsibility does not require large amounts of resource to be made available for coordination [13].

The view that intentionality is the key component of cooperating agents is contradicted by the reactive planning school of thought [46, 47]. In reactive systems, agents merely respond to situations and do not reason about the world (i.e. they do not explicitly represent mental states such as belief, desires and intentions). Usually both the agents and the actions are relatively simple and global properties are seen as emerging from the interaction of behaviours. The advantage of this approach is that because of their lack of explicit reasoning, agents are fast and can respond to changing environmental conditions so long as they have a predefined stimulus-response pairing.

As a consequence of this seemingly contradictory view, there has been a vigourous debate between proponents of the intention based (reflective) approach and those of the reactive system approach [48, 49]. However, here it is hypothesised that reactive agents merely exhibit a special type of intentionality. Each individual has a number of fixed, simple intentions which are specified by the system designer and are implicitly available to the agent. Thus when the designer defines an agent's behaviour he is in fact installing its intentions, the means-end reasoning undertaken at runtime in reflective systems is set through the stimulus conditions. These fixed intentions (pre-compiled behaviours) can then be invoked automatically whenever their conditions for activation are satisfied. Compliance with this viewpoint means that reactive and reflective systems are just opposite ends of a spectrum rather than fundamentally different technologies. Similar reasoning is carried out by both types of system, although at different stages of the development process - run time for reflective systems and design time for reactive ones. Following on from this hypothesis, there should be problems which can be solved by reflective systems but not by reactive ones, since the former are a superset of the latter. There is some evidence to support this view: reactive systems have difficulty solving recursive problems and can fail in environments which contain dead ends [50]. Also the observation that coordination among interacting plans must be done by the programmers of reactive systems when they are designing appropriate actions [51] (rather than by the run-time reasoning mechanisms) is further supporting evidence.

Acknowledgments

The work described in this paper has been partially supported by the ESPRIT II project ARCHON (P2256) whose partners are: Atlas Elektronik, Framentec-Cognitech, Labein, QMW, IRIDIA, Iberdrola, EA Technology, Amber, Univ. of Athens, Univ. of Amsterdam, Volmac, CERN and Univ. of Porto.

References

[1] M. P. Papazoglou, S. C. Laufman and T. K. Sellis, An Organisational Framework for Cooperating Intelligent Information Systems, *Int. Journal of Intelligent and Cooperative Information Systems* **1** (1992) 169-202.

[2] J. Y. C. Pan and J. M. Tenenbaum, An Intelligent Agent Framework for Enterprise Integration, *IEEE Trans. on Systems Man and Cybernetics* **21** (1991) 1409-1419.

[3] E. H. Durfee, The Distributed Artificial Intelligence Melting Pot, *IEEE Trans. on Systems Man and Cybernetics* **21** (1991) 1301-1306.

[4] M. Minsky, *The Society of Mind* (Simon & Schuster, 1985).

[5] N. R. Jennings and T. Wittig, ARCHON: Theory and Practice, in *Distributed Artificial Intelligence: Theory and Praxis*, eds. N. M. Avouris and L. Gasser (Kluwer Academic Press,

1992) 179-195.

[6] R. Neches, R. Fikes, T. Finin, T. Gruber, R. Patil, T. Senator and T. Swartout, Enabling Technology for Knowledge Sharing, *AI Magazine* (1991) 36-56.

[7] S. Cammarata, D. McArthur and R. Steeb, Strategies of Cooperation in Distributed Problem Solving, *Proc. Int. Joint Conference on Artificial Intelligence*, Karlsruhe, Germany (1983) 767-770.

[8] F. Hayes-Roth, Towards a Framework for Distributed AI, *SIGART Newsletter* (1980) 51-52.

[9] M. N. Huhns, U. Mukhopadhyay, L. Stephens and R. Bonnell, DAI for Document Retrieval in *Distributed Artificial Intelligence*, ed. M. N. Huhns (Pitman Publishing 1988) 249-284.

[10] V. R. Lesser and D. D. Corkill, The Distributed Vehicle Monitoring Testbed: A Tool for Investigating Distributed Problem Solving Networks, *AI Magazine* (1983) 15-33.

[11] R. G. Smith and R. Davis, Frameworks for Cooperation in Distributed Problem Solving, *IEEE Trans. on Systems Man and Cybernetics* **11** (1981) 61-70.

[12] V. R. Lesser and L. D. Erman, An Experiment in Distributed Interpretation, *IEEE Trans. on Computers* **29** (1980) 1144-1163.

[13] N. R. Jennings and E. H. Mamdani, Using Joint Responsibility to Coordinate Collaborative Problem Solving in Dynamic Environments, *Proc. of Tenth National Conference on Artificial Intelligence*, San Jose, USA (1992) 269-275

[14] N. R. Jennings, E. H. Mamdani, I. Laresgoiti, J. Perez and J. Corera, GRATE: A General Framework for Cooperative Problem Solving, *IEE-BCS Journal of Intelligent Systems Engineering* **1** (1992), 102-114.

[15] M. E. Bratman, Two Faces of Intention, *Philosophical Review* **93** (1984) 375-405.

[16] P. R. Cohen and H. J. Levesque, Intention is Choice with Commitment, *Artificial Intelligence* **42** (1990) 213-261.

[17] D. C. Dennett, *The Intentional Stance* (MIT Press, 1987).

[18] M. E. Pollack, Plans as Complex Mental Attitudes, in *Intentions in Communication*, eds. P. R. Cohen, J. Morgan and M. E. Pollack (MIT Press, 1990) 77-105.

[19] J. Searle, *Intentionality: An Essay in the Philosophy of Mind* (Cambridge University, 1983).

[20] E. Werner, Cooperating Agents: A Unified Theory of Communication and Social Structure, in *Distributed Artificial Intelligence Vol II*, eds. L. Gasser and M. N. Huhns (Pitman Publishing, 1989) 3 -36.

[21] M. E. Bratman, What is Intention?, in *Intentions in Communication*, eds. P. R. Cohen, J.

Morgan and M. E. Pollack (MIT Press, 1990) 15-33.

[22] N. R. Jennings, Commitments and Conventions: The Foundation of Coordination in Multi-Agent Systems, *The Knowledge Engineering Review* **8** (1993).

[23] H. J. Levesque, P. R. Cohen and J. H. Nunes, On Acting Together, *Proc. of Eighth National Conference on AI*, Boston, USA (1990) 94-99.

[24] P. R. Cohen and H. J. Levesque, Teamwork, *Nous* **25** (1991) 487-512.

[25] N. R. Jennings, On Being Responsible, in *Decentralised AI 3*, eds. E. Werner and Y. Demazeau (North Holland Publishers, 1992) 93-102.

[26] K. E. Lochbaum, B. J. Grosz and C. L. Sidner, Models of Plans to Support Communication, *Proc. of Eighth National Conference on AI*, Boston, USA (1990) 485-490.

[27] A. S. Rao, M. P. Georgeff and E. Sonenberg, Social Plans: A Preliminary Report, in *Decentralised AI 3*, eds. E. Werner and Y. Demazeau (North Holland Publishers, 1992) 57-76.

[28] J. Searle, Collective Intentions and Actions, in *Intentions in Communication*, eds. P. R. Cohen, J. Morgan and M. E. Pollack (MIT Press, 1990) 401-416.

[29] M. P. Singh, Group Ability and Structure, *Proc. Modelling An Autonomous Agent in a Multi-Agent World*, Saint-Quentin en Yveslines, France (1990) 87-100.

[30] R. Tuomela and K. Miller, We-Intentions, *Philosophical Studies* **53** (1988) 367-389.

[31] D. D. Corkill, Hierarchical Planning in a Distributed Environment, *Proc. Sixth Int. Joint Conference on AI*, Tblisi, Georgia (1979) 168-175.

[32] E. H. Durfee, *Coordination of Distributed Problem Solvers* (Kluwer Academic Press, 1988).

[33] M. P. Georgeff, A Theory of Action for Multi-Agent Planning, *Proc. of Second National Conference on Artificial Intelligence*, Austin, USA, (1984) 125-129.

[34] N. R. Jennings, Towards a Cooperation Knowledge Level for Collaborative Problem Solving, *Proc. Tenth European Conference on AI*, Vienna, Austria (1992) 224-228.

[35] J. Y. Halpern and Y. O. Moses, Knowledge and Common Knowledge in a Distributed Environment, in *Theoretical Aspects of Reasoning about Knowledge*, ed. J. Y. Halpern (Morgan Kaufmann, 1984) 50-61.

[36] M. E. Bratman, D. J. Israel and M. E. Pollack, Plans and Resource Bounded Practical Reasoning, *Computational Intelligence* **4** (1988) 349-355.

[37] A. Newell, The Knowledge Level, *Artificial Intelligence* **18** (1982) 87-127.

[38] N. R. Jennings, L. Z. Varga, R. P. Aarnts, J. Fuchs and P. Skarek, Transforming Standalone

Expert Systems into a Community of Cooperating Agents, *Int. Journal of Engineering Applications of Artificial Intelligence* **6** (1993).

[39] L. Z. Varga, N. R. Jennings and D. Cockburn, Integrating Intelligent Systems into a Cooperating Community for Electricity Distribution Management, *Expert Systems with Applications* **7** (1994).

[40] G. Stassinopoulos and E. Lembessis, Application of a Multi-Agent Cooperative Architecture to Process Control in the Cement Factory, ARCHON Technical Report 43/3-93, 1993.

[41] R. P. Aarnts, J. Corera, J. Perez, D. Gureghian and N. R. Jennings, Examples of Cooperative Situations and their Implementation, *Vleermuis Journal of Software Research* **3** (1991) 74- 81.

[42] N. R. Jennings, *Cooperation in Industrial Multi-Agent Systems* (World Scientific Press, 1994).

[43] H. A. Simon, *Models of Man* (Wiley, 1957)

[44] B. Burmeister and K. Sundermeyer, Cooperative Problem Solving Guided by Intentions and Perception, in *Decentralised AI 3*, eds. E. Werner and Y. Demazeau (North Holland Publishers, 1992) 77-92.

[45] Y. Shoham, Agent-Oriented Programming, *Artificial Intelligence* **60** (1993) 51-92.

[46] R. A. Brooks, Intelligence without Representation, *Artificial Intelligence* **47** (1991) 139-159.

[47] P. Maes, ed., *Designing Autonomous Agents* (MIT Press, 1991).

[48] Y. Demazeau and J. P. Muller, eds., *Decentralized AI* (Elesevier Science Publishers, 1990).

[49] Y. Demazeau and J. P. Muller, eds., *Decentralized AI 2* (Elesevier Science Publishers, 1991).

[50] J. Feber and A. Drogoul, Using Reactive Multi-Agent Systems in Simulation and Problem Solving, in *Distributed Artificial Intelligence: Theory and Praxis*, eds. N. M. Avouris and L. Gasser, (Kluwer Academic Press, 1992) 53-80.

[51] A. L. Lansky, A Perspective on Multi-Agent Planning, SRI International, Technical Note 474, Center for Study of Language and Information, Stanford University, 1989.