Joint Intentions as a Model of Multi-Agent Cooperation in Complex Dynamic Environments

by

Nicholas Robert Jennings

Thesis submitted for the degree of
Doctor of Philosophy
in the Faculty of Engineering

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

AUGUST 1992

ABSTRACT

Computer-based systems are being used to tackle increasingly complex problems in ever more demanding domains. The size and amount of knowledge needed by such systems means they are becoming unwieldy and difficult to engineer into reliable, consistent products. One paradigm for overcoming this barrier is to decompose the problem into smaller more manageable components which can communicate and cooperate at the level of sharing processing responsibilities and information.

Until recently, research in multi-agent systems has been based on *ad hoc* models of action and interaction; however, the notion of intentions is beginning to emerge as a prime candidate upon which a sound theory could be based. This research develops a new model of joint intentions as a means of describing the activities of groups of agents working collaboratively. The model stresses the role of intentions in controlling agents' current and future actions; defining preconditions which must be satisfied before joint problem solving can commence and prescribing how individual agents should behave once it has been established. Such a model becomes especially important in dynamic environments in which agents may possess neither complete nor correct beliefs about their world or other agents, have changeable goals and fallible actions and be subject to interruption from external events.

The theory has been implemented in a general purpose cooperation framework, called GRATE*, and applied to the real-world problem of electricity transportation management. In this application, individual problem solvers have to take decisions using partial, imprecise information and respond to an ever changing external world. This fertile environment enabled the quantitative benefits of the theory to be assessed and comparisons with other models of collaborative problem solving to be undertaken. These experiments highlighted the high degree of coherence attained by GRATE* problem solving groups, even in the most dynamic and unpredictable application contexts.

ACKNOWLEDGMENTS

I would like to express my gratitude to the following people for their help in this thesis. Firstly to my supervisor Professor Abe Mamdani for his help, suggestions and encouragement throughout this research. Other members of the Knowledge Engineering Applications Group have also assisted in many different ways - especially John Bigham, Claudia Roda, Jeff Pople, Laszlo Varga and Simon Parsons. Thanks are also due to Andy Martin who has provided excellent computer support.

This research has benefitted tremendously from interactions with all the partners on the ARCHON project. In particular, Inaki Laresgoiti, Juan Perez and Jose Corera helped me to understand the electricity transport domain; Rob Aarnts, Joachim Fuchs and Paul Skarek provided valuable feedback by applying GRATE to the problem of cooperative fault detection for particle accelerator beams; and Thies Wittig gave general guidance on system design issues. Erick Gaussens deserves a special mention for the interest and enthusiasm he has shown in my research - reading papers and chapters of this thesis, offering helpful comments and insights, and criticising when I have been vague. I would also like to acknowledge the contribution of Chris Whitney and others at BT Research Labs (Martlesham Heath) who investigated the feasibility of applying GRATE to the domain of telecommunications network management.

Finally, but in no way least, I would like to thank my parents for their valued support and interest in my academic activities and Jo who has given me constant encouragement, support and friendship throughout every stage of my Ph.D.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

This chapter provides the background, motivation and context for this research. It describes the need for a new approach to building large software systems and why a Distributed Artificial Intelligence (DAI) approach has been investigated. The three major subfields of DAI - distributed problem solving, multi-agent systems and parallel artificial intelligence - are introduced in order to place this work in context. Several fundamental modes of cooperative social interaction are described and the rationale behind some basic assumptions of system design are noted. The notion of intention is introduced because of the pivotal role it plays in the definition of the new theory of coordination developed in this research.

The domain of electricity transport management is then introduced. Transport management is concerned with monitoring the transmission of electrical energy from where it is produced (eg power stations) to where it is consumed (eg homes and businesses). It involves ensuring the electrical network is operating correctly, detecting when faults occur and performing remedial actions to restore the network in such cases. The scenario described is based on a real application running in the north of Spain. This real world testbed enables the applicability of the concepts developed in this research to be judged. The problem is couched in terms of a typical industrial control application, the solution adopted by a Spanish distribution company is explained and the benefits of employing a DAI approach are enumerated.

1.1 Motivation For Research

As computer hardware and software becomes increasingly powerful, so applications which used to be considered beyond the scope of automation come into reach. To cope with these increased demands, software systems are becoming correspondingly larger and more complex (Jones, 1990). However the problems encountered in building such systems are not simply scaled up versions of those faced when writing small computer programs (Sommerville, 1992). As the “software crisis” which was first noted in the late 1960’s illustrates, large systems require radically different techniques and methods.

Artificial Intelligence (AI) techniques, and expert systems technology in particular, have often been used to tackle some of the most difficult automation problems. After more than a decade of exploitation there are now thousands of expert systems being used in hundreds of companies all over the world to solve complex problems in numerous domains (Feigenbaum *et al.*, 1988). However as this technology has proliferated and individual systems have increased in size and complexity, new problems and limitations have been noted (Partridge, 1987; Steels, 1985).

- *Scaleability*: the complexity of an expert system may rise faster than the complexity of the domain.
- *Versatility*: a complex application may require the combination of multiple

problem solving paradigms.

- *Reusability*: several applications may have requirements for similar expertise which has to be coded afresh in each new situation.
- *Brittleness*: expert systems operate on a high plateau of knowledge and competence until they reach the extremity of this knowledge when they fall off sharply to the level of ultimate incompetence.
- *Inconsistency*: As knowledge bases increase in size, it becomes correspondingly more difficult to ensure that the knowledge they embody remains consistent and valid.

Various approaches for circumventing these problems have been advocated. The first proposal involves building an extremely large base of common-sense knowledge (Guha and Lenat, 1990). This work is predicated on two important assumptions; firstly, performing a complex task well requires a great deal of knowledge about the world (eg water flows downhill, living things get diseases, doing work requires energy, etc.)[1]. Secondly, to behave intelligently in unexpected situations requires the ability to fall back on increasingly general knowledge and analogising to specific but superficially far-flung knowledge (Lenat and Feigenbaum, 1991).

A second approach is to allow the sharing and reuse of knowledge (Neches *et al.*, 1991). In this vision, rather than constructing knowledge based systems afresh, as is the case today, it could be done by assembling reusable components. System developers would then only need to worry about creating the specialised knowledge and reasoners for the specific task at hand, thus facilitating the building of bigger and better systems more cheaply. This work is based on the observation that application systems contain many different kinds and levels of knowledge. The top-level ontologies[2] embody representational choices ranging from topic independent (eg models of time) to topic specific but still application independent knowledge (eg domain knowledge about different kinds of testing operations represented in a manufacturing system). This top level of knowledge is elaborated by more application specific models (eg knowledge about chip-testing operations in a specific manufacturing application). Together they define how a particular application describes the world. At the bottom level, assertions using the vocabulary of these models captures the current state of the system's knowledge. Knowledge at the higher levels, being less specialised, is easier to share and re-use. Knowledge at the lower levels can only be shared if the other system accepts the models in the levels above.

Finally, the approach pursued in this research, is to build systems of smaller more manageable components which can communicate and cooperate (Bond and Gasser, 1988; Gasser and Huhns, 1989; Huhns, 1988). This approach has several major advantages. Firstly divide and conquer has long been championed as a means of constructing large systems because it limits the scope of each processor. The reduced

1. Guha and Lenat's system currently embodies over a million assertions
2. Shared sets of explicitly defined terminology

size of the input domain means the complexity of the computation is lower, thus enabling the components to be simpler and more reliable. Decomposition also aids problem conceptualisation; many tasks appear difficult because of their sheer size, they are simply too big to contemplate all at once. For example, the working of a national economy or the operation of a large multi-national corporation would fall into this category (Smith and Davis, 1981).

A second major advantage is that a distributed approach often provides a more natural fit to the problem. Examples include distributed sensor networks (Lesser and Corkill, 1983), air traffic control (Cammarata *et al.*, 1983) and distributed information retrieval (Huhns *et al.*, 1988). Indeed, Hayes-Roth (1980) even goes so far as to state that “all real systems are distributed”.

Other potential advantages include: reusability of problem solving components, greater robustness in the case of component failure, enhanced problem solving due to the combination of multiple problem solving paradigms and sources of information, problem solving speed up due to parallel execution and increased system modularity (Bond and Gasser, 1988).

The initial research described in this thesis addresses one of the major open issues in DAI: that of simplifying the task of building multi-agent systems (Bond and Gasser, 1988). The solution adopted here is to provide the application designer with a new type of development environment, one which comes with *built-in generic knowledge* not just specialised programming features. This knowledge is related to three main areas; control of local problem solving activity, assessing the local and global situation and controlling social activity. The system constructed is called *GRATE* (Generic Rules and Agent model Testbed Environment) (Jennings *et al.*, 1992). It has been applied to three real-world exemplar domains: electricity transportation management, fault diagnosis of a particle accelerator controller and detection of focused overloads in a telecommunications network[3]. This thesis concentrates on electricity transportation management as the other two were built at laboratories which were not readily accessible and use expert systems which are not easily portable. Further details of these experiments can be found in (Fuchs et al., 1992; Jennings et al., sub) and (Whitney, 1992) respectively.

One of the problems which arose when GRATE was used in transportation management was that when anything unexpected happened (eg new information became available which invalidated existing goals, timings between actions were disrupted or agents had misinterpreted the situation) the community of cooperating agents often became uncoordinated. For instance, agents would continue work on a goal despite the fact that one community member knew their processing was obsolete; agents would simply stop processing requests if a more important task arose, without informing the agent who originated the request; agents would wait for the results of a

[3]. The first two applications are taken from the ESPRIT II Project ARCHON and their development has benefitted from interaction with other partners within the project. In particular the first one has been developed with the invaluable assistance of Iberdrola and Labein and the second with Volmac and CERN. The third application was undertaken with BT Research Labs (Martlesham Heath).

task which had been abandoned and so on.

As a consequence, the problem of coordinating behaviour in environments in which agents possess neither complete nor correct beliefs, have changeable goals and fallible actions and are subject to interruption from external events needed to be investigated further. Uncoordinated behaviour was attributed to the fact that GRATE agents did not embody sufficient knowledge of the collaborative problem solving process to operate in dynamic, complex and unpredictable domains.

To rectify this problem, it was decided that agents should have a well-grounded theory of interaction on which their behaviour could be based; that is, a deep theory of coordination was required. This model could then be embodied in a declarative style so that community members could reason about it and use it to guide their actions. A key desiderata of the theory is that it should prescribe how to behave when team problem solving is progressing satisfactorily and also when things go wrong (as is often the case in industrial control applications!). Previous theoretical work had not satisfactorily addressed this issue; therefore a new model for describing agent behaviour in such circumstances was developed as part of this thesis. This model, called *joint responsibility*, is based on the notion of joint intention and has been implemented in an extended version of GRATE called *GRATE**. An empirical investigation of a community of cooperating agents, in electricity transport management, using the responsibility model to guide their interactions was undertaken. These experiments aimed to assess the hypothesis that adherence to the model ensures robust collaborative behaviour even in the most hostile environments.

1.2 Introduction to Distributed AI

This section provides an introduction to the field of DAI so that the contribution of this research can be placed in perspective. The major subfields of DAI are defined and reasons why agents need to communicate and cooperate during problem solving are put forward. Various forms of social interaction supported by GRATE and GRATE* are defined and important assumptions underlying their design are described. Finally, intentions and the central role they play in the problem solving process are discussed.

1.2.1 Defining Distributed AI

As is the case with “conventional” AI, there exists a plethora of seemingly inconsistent definitions for Distributed AI. Because of its broad and multidisciplinary nature, no universally acceptable definition has yet been produced; Sridharan (1987) presents a historical perspective citing definitions from 1980 onwards, but fails to summarise them into a consistent whole. Therefore only a loose working will be used in this thesis:

Distributing and coordinating knowledge and actions in multiple agent environments (Bond and Gasser, 1988)

Individual problem solving entities are called *agents* and are grouped together to form *communities*. It is assumed that each agent is capable of a range of useful problem solving activities in its own right, has its own goals and objectives and can communicate with others. This ability to solve some problems alone (coarse granularity (Sridharan, 1987)) contrasts with other types of community which have fine grained agents. For instance in neural networks (McClelland and Rumelhart, 1986) individual nodes have very simple states (either on or off) and only by combining many thousands of them can problem solving expertise be recognised[4].

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 to meet global constraints and the fact that often no single agent has sufficient competence to solve the entire problem. There are two main causes of such interdependence: adapted from (Davis and Smith, 1983):

- When problem partitioning yields components which cannot be solved in isolation.

  In speech recognition, for example, it is possible to segment an utterance and work on each segment 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 (Lesser and Erman, 1980).

- 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 any global constraint (eg total cost must be less than £70,000) would be satisfied. In such cases, compatible solutions can be developed only by having interaction and agreements between the agents during problem solving.

This analysis provides an indication of the type of social interactions which should take place. In the former case, agents should transmit information which most strongly reduces ambiguity. In the latter case, they should exchange partial solutions (speech recognition), underlying assumptions (building renovation) or similar information which will ensure consistency of the eventual subproblem solutions and satisfaction of any global constraints.

---

[4]. Parunak (1989) proposes that the boundary between coarse and fine grained systems is not as large as it may appear. This manifests itself when agents are capable of carrying out a wide range of intelligent behaviour, but constrain their interactions in order to provide a greater measure of coherence in the system as a whole. He terms this the "marching band" syndrome, however the value of this observation has still to be assessed.

Even when multiple agents can work independently (eg inferencing in standard logic or execution of pure functional languages), information discovered by one agent can be of sufficient use to another that the two agents can solve the problem more than twice as fast. This “accelerator effect” or combinatorial implosion (Hewitt and Kornfield, 1980) can be particularly useful in domains with large search spaces. It has led to the postulation that cooperative search, when sufficiently large, can display universal characteristics which are independent of the detailed nature of either the individual processes or the particular domain being tackled (Clearwater *et al.*, 1991). This theory has been partially tested by having a number of agents solve a set of cryptarithmetic problems and measuring their individual and global performance.

Depending on the number of problems being solved, their nature and the way in which they are distributed, two types of DAI system can be identified: distributed problem solvers and multi-agent systems. A third sub-area, called parallel AI, deals with more traditional issues related to concurrent languages and hardware. Each of these is expanded upon below.

1.2.1.1 Distributed Problem Solving (DPS)

DPS deals with the interactions of groups of intelligent agents which act together to solve a single task (Decker, 1987). It can be characterised by the nomenclature given in figure 1.1 (Smith and Davis, 1981). In the first phase, the problem is continuously decomposed until solvable sub-tasks are generated. Then the individual subproblems are solved by agents which communicate and cooperate as necessary. Finally, subproblem results are integrated to produce the overall solution. Depending on the underlying application, the three phases may vary in complexity and importance. For example in a distributed traffic light control system, problem decomposition involves no computational effort because controllers are simply positioned at each intersection. There is also no answer synthesis, the overall solution is a smooth flow of traffic through the intersection.

Early DPS research was often confused with distributed processing. Distributed processing often aims to synthesize a network of machines capable of carrying out a number of independent tasks (Coulouris and Dollimore, 1988). The major motivation being to find a way of reconciling any conflicts and disadvantages arising from the desire to carry out disparate tasks, in order to gain the benefits of using multiple machines. Conversely, in DPS, a single task is envisaged for the system and the available resources have no predefined, allocated roles. The concerns of researchers are thus with developing frameworks for cooperative behaviour between willing entities, rather than enforcing cooperation as a form of compromise between incompatible entities. The final distinction results from the lack of substantial cooperation in most distributed processing systems; typically most of the processing is done at a central site and remote processors are limited to basic data collection. Distributed is usually taken to mean spatial distribution of data, distribution of function or control is not usually considered.

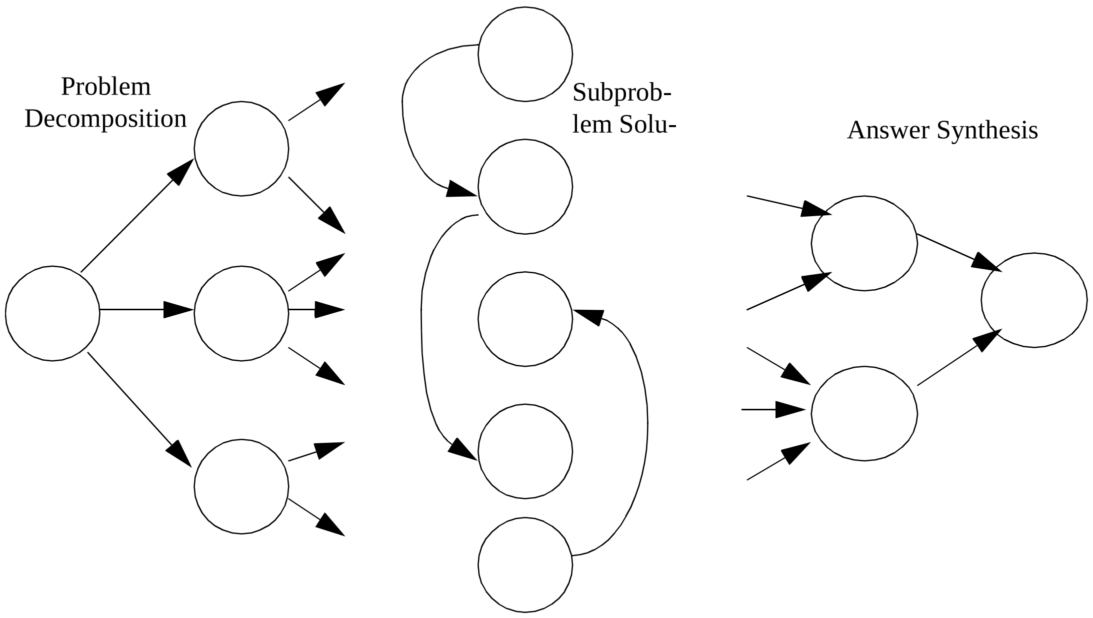

Figure 1.1: Phases of Distributed Problem Solving

Although GRATE enables agents to share information and request others to carry out tasks for them, *it is not targeted towards distributed problem solving*. Firstly, problem decomposition has been carried out *a priori* by the system designer. Secondly, the matching of agents to tasks is usually straightforward because most tasks can only be performed by one or a few agents. In DPS systems, agents are generally homogeneous and hence most agents will be able to undertake any given task. Finally in industrial control applications, there are typically several problems active at the same time. For instance in electricity transport management some agents may be attempting to diagnose a fault whilst others are trying to repair an earlier one.

1.2.1.2 Multi-Agent Systems

Multi-agent systems are concerned with coordinating the knowledge, goals, skills and plans of autonomous intelligent agents so that they can jointly take actions or solve problems (Bond and Gasser, 1988). Agents may be working towards a single global goal or separate individual goals that interact in some way. In common with DPS systems, agents must share knowledge; however they must additionally reason about the processes of coordination to ensure: (Lesser and Corkill, 1987).

- *Coverage*: all necessary portions of the overall problem must be included in the activities of at least one agent.
- *Connectivity*: agents must interact in a manner which permits the covering activities to be developed and integrated into an overall solution.
- *Capability*: coverage and connectivity must be achievable within the network’s communication and computation resource limitations.

- *Coherence*: team members are acting in a purposeful and consistent manner.

It was the problem of maintaining coordinated behaviour in dynamic and unpredictable environments which led to the notion of joint responsibility and ultimately to GRATE*. In general, two major paradigms are used to coordinate activity:

- Planning for multiple agents. A single intelligent agent constructs a plan to be carried out by a group of agents and then hands out the pieces to the relevant individuals (Rosenschein, 1982).

- A group of intelligent agents together construct and possibly execute the plan (eg Partial Global Plans (Durfee and Lesser, 1987)).

GRATE* uses a combination of the two approaches for organising collaborative activities. One agent, the organiser, is responsible for deciding which actions should be carried out and by whom. The organiser contacts those agents it wants to be involved, they evaluate whether they are interested in participating and return their responses. Once all potential participants have replied, the detailed timings of each plan action are then made. This is the collaborative part. For each action in the plan, the organiser proposes the earliest time at which it can be started, then the chosen agent evaluates its existing activities to see if it can be performed at this time. If so, this is the agreed time; if not the chosen agent proposes the earliest time at which the task can be performed and this becomes the agreed time.

The autonomy of the agents makes the social interactions more complex. Firstly agents must decide whether they wish to participate in a collaborative act, hence the organiser must send out requests for participation rather than tacitly assuming other agents will always be willing to contribute. In such systems the primary motivation for cooperation is self-interest (Axelrod, 1984; Durfee *et al.*, 1987): the belief that participation in the social activity will be more beneficial than abstaining. This hypothesis is examined extensively by Axelrod (1984), through the prisoners dilemma[5], in the domains of political science, trench warfare and biological systems. Novak and Sigmund (1992) have also investigated this approach for biological situations in which errors of recognition were also taken into consideration and have found it applicable. Finally genes can also be viewed as self interested agents which cooperate to improve their chances of reproducing (Dawkins, 1976).

Secondly it cannot be assumed that agents are benevolent: ready to change their goals to suit the needs of others and always be in agreement with each other

[5.] Two suspects are taken into custody and separated. The police are certain that they are guilty of a specific crime, but do not have enough evidence to convict them. Each prisoner is offered two alternatives: to confess or not. If they both do not confess, they will both be sentenced on some minor charge and will receive minor punishment. If they both confess they will be prosecuted but will receive less than the most severe sentence. But if one confesses and the other does not, then the confessor will receive lenient treatment for helping the police whereas the latter will receive the full punishment (Luce and Raiffa, 1957).

(Rosenschein and Genesereth, 1985; Genesereth *et al.*, 1988). In GRATE*, this is illustrated by the fact that the organiser and the participants must jointly agree upon the action timings. Participants may already have activities planned for the time proposed by the organiser and may therefore be unwilling to drop them simply because they have been asked to do so. Agents only drop activities if they prefer the new proposal more than their current one (Galliers, 1988).

In truly autonomous systems the agent initiating the social interaction must exert sufficient influence over the others to persuade them to contribute (Parunak 1989), rather than assuming they will adopt goals upon recognition (Castelfranchi, 1990). In GRATE* goal adoption is not explicitly reasoned about, however the recipient may give the request such a low priority that it never gets executed.

1.2.1.3 Parallel AI

Parallel AI is concerned with developing parallel computer architectures, languages and algorithms for AI (Bond and Gasser, 1988). Examples of systems which fall into this category include: AGORA (Bisiani *et al.*, 1988) which provides a parallel virtual machine that is language independent, allows a number of different programming models and can be mapped into a number of different computer architectures; AF (Green, 1988) which supports the implementation of real-time AI programs on multiple interconnected computers; and ABE (Hayes-Roth *et al.*, 1988) which addresses the problem of combining conventional computing functions with knowledge processing capabilities.

The primary motivation for this research is to improve the performance of current AI systems. It differs from the other two sub-fields of DAI because it does not aim to advance the conceptual understanding of the nature of reasoning and intelligent behaviour among multiple agents. However developments in concurrent languages and architectures may have significant impacts on DAI architectures, reliability, knowledge representation and so on. The research described here does not make any advances in this area.

1.2.2 Forms of Social Interaction

The simplest form of social interaction is communication. There are two major models for interagent communication:

• Shared Memory

The most widely used architecture is the blackboard system (Engelmore and Morgan, 1988; Hayes-Roth, 1983; Nii, 1986a/b). The blackboard is a global database containing entries generated by the agents (knowledge sources (KSs)). Entries are intermediate results generated during problem solving and include both elements of the problem solution and information deemed important in generating solution elements. KSs monitor portions of the blackboard waiting for particular patterns of data. When such patterns occur, the KS takes the data and processes it, typically forming new combinations of data which it places on another portion of the blackboard.

The blackboard forms the basis of most extant DAI systems. HEARSAY II (Erman and Lesser, 1975), a system devised to understand connected speech in real-time, was the first example of this communication model. Other important systems include the Pilot's Associate which acts as an expert pilot aid (Smith and Broadwell, 1988) and the Distributed Vehicle Monitoring Testbed which monitors and interprets data from a set of sensors at spatially distributed locations (Durfee *et al.*, 1988a; Lesser and Erman, 1980).

• Message Passing.

Message passing ideas have been drawn from conventional object oriented programming and in particular from object based concurrent programming - eg ACTORS (Agha and Hewitt, 1985; Agha, 1986). It does not require shared memory data structures between the communicating agents.

Both GRATE and GRATE* use a message passing paradigm for communicating between agents. This approach was chosen because:

- It has well understood semantics and offers a more abstract means of communication (Hewitt and Kornfield, 1980). No hidden interactions can occur; so there is greater comprehensibility, reliability and control over access rights.

- It makes fewer assumptions about the system architecture.

- Shared memory systems do not scale up. If only a single blackboard exists then it becomes a severe bottleneck and if several exist the semantics revert to message passing (Hewitt and Lieberman, 1984).

Communication is the basis for more sophisticated social activities such as cooperation and coordination[6]. However establishing a precise distinction is the subject of an intense philosophical debate within DAI, see (Power, 1984) and (Axelrod, 1984) for a sample of these arguments. Rather than reproducing such discussions here, a pragmatic approach of defining necessary and sufficient conditions will be adopted; cooperation on a particular goal G occurs if and only if: (Galliers, 1988/89)

- The agents share G as a common goal.

[6]. The one exception to this work is outlined in (Genesereth *et al.*, 1988). In this work game theoretic notions and agent rationality are used to investigate cooperation in situations where agents cannot communicate. Although some cooperation is possible in such circumstances, it is fairly limited and relies on somewhat unrealistic assumptions (eg agents have common knowledge of the payoff matrix).

- Agents are aware of the others' goals; ensuring cooperation is more than mere accidental coordination[7], as it incorporates an element of helpfulness.

- As well as recognising another agent's goal the agent must have a personal preference for this goal state to be achieved. As a consequence of these conditions there is commitment to achieving the common goal.

- If attainment of the common goal requires attainment of sub-goals, then these are adopted on the same basis as above.

The inclusion of the context, G in this case, is needed to relativise the discussion. It is not usual to say that two agents cooperate *per se*; rather they may cooperate on goal G_1, be independent of each other on goals G_2 and G_3 and be in conflict about goal G_4.

Two of the most fundamental forms of social interaction are task and result sharing (Smith and Davis, 1981), both of which are supported by GRATE. Task sharing is depicted in figure 1.2, the numbers on the data flows represent the temporal ordering of events. In this form of interaction, one agent asks another to perform some problem solving activity for it (eg to calculate a list of elements at fault in diagnosis or identify the type of an object in sensor interpretation). If the recipient accepts the request, it completes the task and informs the originating agent of the outcome. Task sharing has similar semantics to a remote procedure call in conventional distributed systems (Coulouris and Dollimore, 1988), although in some cases the requestor may be able to continue with subsequent activity.

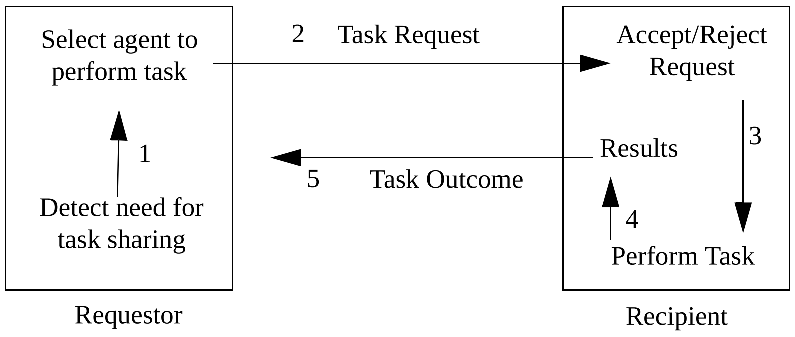

Figure 1.2: Task Sharing

A task sharing form of interaction may be invoked because the originating agent cannot perform the task itself (used in GRATE), because it deems another agent more capable or in order to balance the system's computational load.

A key concern for this form of interaction is the mechanism by which agents

[7]. Two art enthusiasts arrive independently at an art gallery with the goal of destroying a particularly shocking painting. One is intercepted by a guard, but the other succeeds in tearing the picture. One agent achieved the goal state of both of them and yet was unaware of the other's goal, hence this act is deemed "accidental coordination", not cooperation (Power, 1984).

decide who will perform which task. GRATE uses a simple client-server model to direct a single request to the agent deemed most appropriate; this is sufficient for most industrial applications because tasks can generally be undertaken by only a few community members. In situations where many agents can perform each activity and the most appropriate choice cannot be identified *a priori*, a more sophisticated protocol may be appropriate. For example in the contract net protocol (Smith, 1980), the requesting agent sends out proposals to all community members, who then evaluate it and return a bid indicating why they should be chosen. Once all the bids have been received the originator selects one agent and a contract is established.

The second fundamental cooperation paradigm is that of result sharing, see figure 1.3, again the numbers indicate sequential events. Agents assist one another by spontaneously sharing partial results which are based on differing perspectives of the global problem. Different perspectives may arise because agents use different knowledge (eg syntax versus acoustics in the case of a speech understanding system (Lesser and Erman, 1980)) or different data (eg data that is sensed at different locations in the case of a distributed sensing system (Durfee *et al.*, 1987)). The agent which generates the information, evaluates whether it could be of use to any of the others. If it finds such an agent, the information is sent. When the recipient receives the information, it is evaluated to determine whether it could be usefully incorporated into the current problem solving context. If it can, then it is included; if it cannot then it is simply discarded.

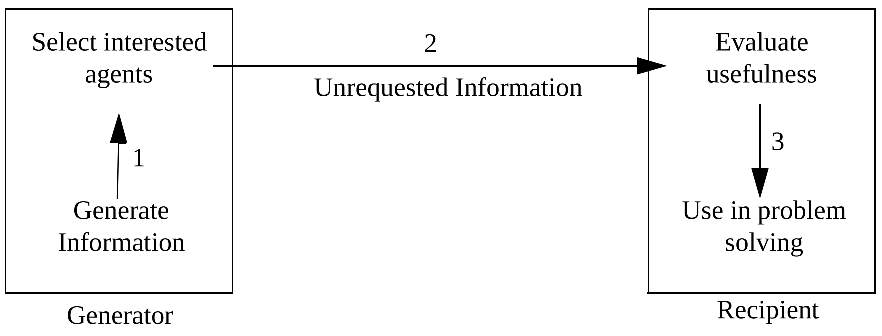

Figure 1.3: Result Sharing

Information sharing is most beneficial when subproblems cannot be solved by working independently, when results achieved by one agent influence those produced by another and when sharing causes convergence not oscillation. The data may be used: to confirm or deny competing results, to aggregate partial results, to prune an agent’s search space or to indicate potentially profitable lines of problem solving. It assumes that problem partitioning has been carried out *a priori*, and that individual experts work on subproblems which have some degree of commonality

Task and result sharing occur in both distributed problem solving and multi-agent systems. However they have their limitations. Task sharing assumes the requests can be solved by working independently. Result sharing does not incorporate a problem decomposition mechanism nor does it offer a feedback mechanism for coordinating the actions of groups of agents.

A further type of social activity (called team or group or joint problem solving), neglected by Smith and Davis, is required to describe scenarios in which the actions of several agents are intertwined. That is, the goals of one agent interact with the goals of another (McArthur *et al.*, 1982). Relevant examples include several agents lifting a heavy object, musicians in an orchestra, driving in a convoy and playing cricket. A collaborative lift, see figure 1.4 for example, cannot be decomposed into independent actions. The lifting times and positions of the participants must be coordinated since they are related. Team members must share partial solutions and future plans if the objective is to be satisfied - for instance they may exchange information about what time the lift should occur, the relative lifting positions, is it going according to plan? and so on.

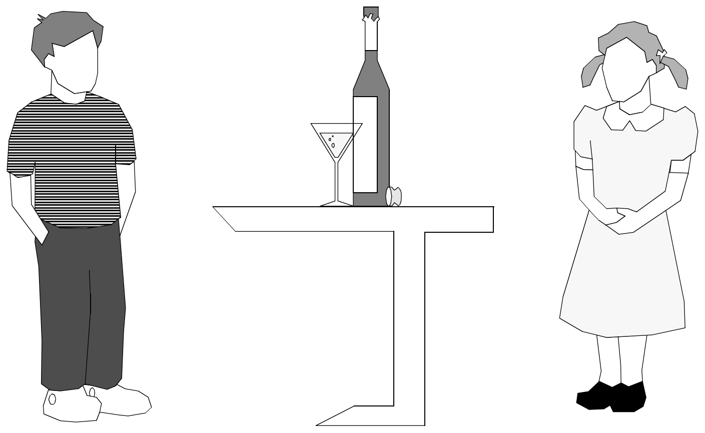
Figure 1.4: Illustration of joint action scenario

There are two primary motivations for groups of agents forming teams and acting collaboratively.

• through necessity, because no individual is capable of achieving the overall objective

eg one agent cannot lift a heavy object alone, playing tennis requires at least two people and performing an orchestral piece requires several people

• through belief that a team approach is best

eg when searching for a lost object in a large area it is often better, though certainly not essential, to do so as a team

Joint action differs from both task and result sharing because it is a reciprocal process in which participating agents augment their actions to comply with those of others. Such adaptation is necessary because of the task interdependencies; Smith and Davis assumed actions could be performed as if the agents were operating in a vacuum. In result sharing, the aims of the information generator are unaltered; it is of no direct concern whether the recipient acts upon the information or simply chooses to ignore it. In task sharing the originator's overall aims are unaffected and the recipient typically adopts a new intention rather than modifying an existing one.

Collaborative problem solving is not an intrinsic property of the actions performed. Examples can be constructed in which identical actions can be construed as individual behaviour in one case and joint action in another. Imagine a group of people sitting in a park (Searle, 1990). As a result of a sudden downpour all of them run to a tree in the middle of the park because it is the only available source of shelter. This is individual behaviour because each person has the intention of stopping themselves from becoming wet and even if they are aware of what others are doing and their goals, it does not affect their actions. This contrasts with the situation in which the group are ballet dancers and the choreography calls for them to converge on a common point (the tree). This is joint action because individuals have the aim of meeting at the central point as a result of the overall intention of executing the ballet.

Joint action also involves more than just the union of simultaneous individual actions, even when these actions are coordinated. For example, there is no real teamwork involved in driving along a busy road, even though the drivers act simultaneously and are coordinated by the traffic signs and rules of the road (Levesque *et al.*, 1990). Only if the drivers decide to do something together, such as driving somewhere in a convoy, is there any collaborative action. This distinction is also reflected in Werner's (1989) classification of positive and negative cooperation. Positive cooperation in which an agent only wants to perform certain actions if others are performing complementary ones can be equated to joint action and negative cooperation in which agents try to avoid doing certain sub-processes simultaneously to coordination.

Joint action does not imply either cooperative or coordinated action. For example, agents may agree that the best means of solving a common goal is to pursue purely individualistic actions. Suppose one member of a cricket team looses his wallet at the game, team members gather together and decide their chances of finding it are best if they each act separately and search for the wallet in their own way. They then set out upon the joint action of finding the wallet with a common solution which is completely lacking in coordination and cooperation (Searle, 1990).

As these examples illustrate, it is difficult to base a definition of joint action on external behavioural characteristics. Therefore this thesis assumes that the crucial component is the mental state of the participants. A joint activity is one which is performed by individuals sharing specific mental properties, such as beliefs, desires and goals.

1.2.3 System Design Issues

If all agents had complete knowledge of the goals, actions and interactions of their fellow community members, it would be possible to know exactly what each agent was doing at present and what it is intending to do in the future. In such instances, it would be possible to avoid conflicting and redundant efforts - systems could be perfectly coordinated and the cost of achieving this state would not be prohibitively expensive (Malone, 1987). However such complete knowledge is often infeasible, in any community of reasonable complexity, because:

- Bandwidth limitations make it impossible for agents to be constantly informed of all developments in the system (Davis and Smith, 1983).
- A local perspective simplifies conceptualisation and implementation. Problems become more complex if agents have to monitor the activities of all the others - the theory of bounded rationality (Simon, 1957).

As a consequence of these points and the inherent complexity of the industrial control domain it was decided to build systems in which agents have only local, incomplete perspectives of their world. Because of this partiality, it is possible that actions may be mutually interfering, rather than supportive. For example misleading and distracting hypotheses may be spread, multiple agents may compete for unshareable resources simultaneously, agents may unwittingly undo the results of each other and the same actions may be carried out redundantly.

If all community members cannot have an overall perspective, the next easiest way of ensuring coherent behaviour is to have one agent with a complete picture. This global controller could then direct the activities of the others, assign agents to tasks and focus problem solving to ensure coherent behaviour. However in systems of reasonable complexity, such an approach is also infeasible because:

- Even just keeping one agent informed of all activities in the community would swamp the available bandwidth.
- The controller would become a severe communication and computational bottleneck and would cause the whole system to collapse if it failed (Lesser and Corkill, 1987).

To produce a system without bottlenecks and which exhibits graceful degradation of performance, this research concentrated on developing agents in which both control and data were distributed. Distributed control means individuals have a degree of autonomy in generating new activities and in deciding which tasks to do next. The disadvantage of this approach is that it becomes correspondingly more difficult to attain coherent global behaviour because of the increased uncertainty of each agent’s actions. Similar problems have also been noted in organisational science:

> “the greater the task uncertainty, the greater the amount of information

that must be processed among decision makers during task execution in order to achieve a given level of performance" (Galbraith, 1973)

Organisational science proposes that one way of tackling this problem is to introduce rules and procedures which define how to behave in certain circumstances and "if everybody adopts the appropriate behaviour the resultant aggregate response is a coordinated pattern of behaviour" (Galbraith, 1973). GRATE agents did not possess the necessary procedures to enable them to operate in uncertain environments and so the level of performance in such situations was significantly degraded. The notion of joint responsibility, on the other hand, defines such a procedure for team problem solving. It specifies how individuals should behave when problem solving is progressing as expected and also when unanticipated events occur. Hence in GRATE*, which uses the model of responsibility to guide the actions of group members, the community's performance should not deteriorate as badly. An empirical investigation of this hypothesis is undertaken in chapter six.

To produce an efficient system it is important that agents spend the bulk of their time engaged on computational activities, rather than in communication which is significantly slower. Agents should be loosely coupled (Sridharan, 1987). In such cases, communication is only necessary to request assistance with sub-goals, to exchange results and to coordinate activities - interactions which form a relatively small proportion of the total time available[8]. To construct such agents, the overall problem needs to be decomposed into the most modular units possible whilst ensuring they are still of sufficient granularity to warrant the overhead inherent in goal distribution (Davis and Smith, 1983). Empirical evidence supporting the importance of the latter point has shown that distributing small tasks can prove substantially more expensive than performing them in one place (Wesson *et al.*, 1981; Durfee *et al.*, 1988a).

1.2.4 Intentions

Since the days of Aristotle, philosophers have argued over the constituent components of rational behaviour (Dennett, 1987). Most have agreed that the mental states of *belief* (eg I believe you are enjoying reading this thesis, Tom believes it will rain today) and *desires* (eg John wants an ice cream) are important. Until recently the reductive model of rational behaviour has predominated, in which action is viewed as being reducible to desires and beliefs (Bratman, 1984). However Bratman (1984) and others have argued that a separate mental state of *intentions* is needed. Intentions are related to the other two but are not reducible to them.

Bratman makes two justifications for his claim. Firstly noting that agents are resource bounded (i.e. they do not have infinite time or information), he suggests that

[8]. In tightly coupled systems, such as neural networks, ACTOR systems (Agha, 1986) and BEINGS (Lenat, 1975), the reverse is true. Agents are unable to operate without extensive assistance and communication from their fellow agents. This is because each component has little or no knowledge of the problem solving task as a whole or of general techniques for communication and cooperation. Advanced behaviour stems from the predefined interactions between that particular group.

no agent can continually weigh its 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 obtain that objective. Secondly intentions are needed to coordinate future actions. Once a future action has been decided upon (intended), the agent must make subsequent decisions with it as given. For example if I intend to play cricket for my local side on Sunday afternoon then it would be irrational to plan to go to Paris for the weekend. Thus the ability to plan to do some action A in the future, and what to do subsequently from A, requires that a rational agent does not simultaneously believe he will not do A. If this were to be the case, the agent would not be able to plan past A since it believes A will not be done. Without a notion of commitment, deciding what else to do would be impossible.

Two types of intention have been identified; future-directed (or prior) and present directed (or intentions in action) (Bratman, 1984; Searle, 1983). The former guide agents' planning activities, their coordination with others and the adoption of subsequent activities; the latter function causally in producing behaviour. For example a future-directed intention may include playing football on Saturday and a present directed one may be to press the keyboard to type a full-stop now. In the context of coordinating multi-agent behaviour, the notion of future directed intentions is more useful and so will be the primary focus of this thesis. Hereafter the term intention will be used in this sense only, unless explicitly indicated to the contrary.

Although intentions provide a degree of stability for planning future activity, they are not irrevocable because circumstances may change and it is not always possible to correctly predict the future. For example consider a robot whose master has instructed it to go and fetch him a beer (Cohen and Levesque, 1990a). The robot forms the intention of going to get the beer from the fridge and goes into the kitchen where it believes the fridge is located. Once in the kitchen the robot notices that a chip pan is on fire, which it knows to be dangerous, but because it is committed to getting the beer for his master it simply opens the fridge, picks up the beer and leaves the kitchen. His master is just taking his first sip of the beer when there is an explosion in the kitchen and he has to flee before the whole house is engulfed in flames. Although the robot honoured its commitments it did not behave in an intelligent manner; when it noticed the fire it should have dropped (or suspended) the intention of fetching the beer and attempted to put it out. Therefore it is important that as well as providing a commitment to carry out actions, intentions are not carried out fanatically and can be reneged upon in appropriate circumstances.

The notions of individual intentions have been used to analyse a variety of phenomena in DAI, philosophy, social science and management science and are discussed further in chapter three. However these formulations concentrate on defining how *individuals* should behave. When describing social activity it is necessary to have representations of action which address higher-order units of analysis (Gasser and Huhns, 1989). This leads to the idea of joint or group intentions. For example when describing a collaborative lift, it is important to specify how the team as a whole should behave, as well as the behaviour of the individual participants.

Any comprehensive formulation of joint intention must clearly define their functional role in group problem solving, address the issue of the relationship with individual actions and provide a clear mapping into an implementation architecture.

It is a central tenet of this thesis that notions such as joint intention will provide the basis for deeper theories of coordination, which is presently one of the major open issues in contemporary DAI. Joint intentions are discussed in greater depth in chapter three, form the basis of the joint responsibility model of coordination developed in chapter four and provide the foundation of the implementation of GRATE* discussed in chapter five.

1.3 Industrial Control Applications

This section describes the details of the domain of *electricity transport management* which was used as the main demonstrator for the concepts developed in this research. It has been written with the assistance of Inaki Laresgoiti, Juan Perez (both Labein) and Jose Corera (Iberdrola). Iberdrola is the largest electricity utility in Spain in terms of market share and the second in terms of power generation. Labein is a technology institute with a strong involvement in applied AI and power engineering.

To be available at the required consumption sites, electrical energy has to pass through a series of complex processes: generation, transportation and distribution. Generation transforms “raw” energy (eg hydraulic, thermal, nuclear and solar) into a more manageable and useful form such as electrical energy. Ideally consumption sites would be near generation ones, however due to various economic, social and political factors this is often impossible. Hence energy needs to be transported from its generation site to the customer. To minimise losses during transportation, the electrical voltage is made high (132kV or above) before it is placed on a *transport network* and sent over many hundreds of kilometres. The transmission network controlled by the Northern Power Dispatch Centre of Iberdrola extends over an area of 100,000 km^2 and consists of 120 substations, 300 lines and 300 transformers in the range 132kV to 380kV. The final operation necessary to make energy available to consumers is the lowering of the voltage and its distribution. This is achieved using a *distribution network* which involves many kilometres of network (all below 132kV).

The work described in this thesis focuses on the construction of a multi-agent system for monitoring electricity transportation networks in which the primary objective is to maintain a balance between safety and economical factors. Aiming to provide a reliable service to the customer whilst producing profits for the electricity utility providing the service.

This application was chosen because contacts made within the ARCHON project afforded the opportunity to work on a real-world problem, with the constraints that this imposes. From a DAI point of view the application is of appropriate complexity, neither too difficult nor too complex (Wittig, 1992). A number of subsystems already exist, each with a specific task to perform, but sometimes with overlapping capabilities. Finally the application is typical of a large class of industrial control systems, meaning that results and insights gained should be applicable to a broad class of problems.

Like many industrial applications, the objective of the system is to maintain some property at a specified value over time. In electricity transport management the property is the network status and the value is operational. The operation of an industrial control system can be expressed as a function (F) relating system inputs (I_s), outputs (O_s) and time (t) (Leveson, 1990). At any moment in time, a unique set of relationships exists between inputs and outputs; the output value is related to past and present values of the inputs and time. In this case the relationships involve fundamental electrical and mechanical laws as embodied in the nature and construction of the system. The system is built from components whose interaction implements F, including a control component whose function is to ensure F is achieved correctly. Besides the basic objective implemented by the process, these systems often have constraints on their operating conditions. In this example there are safety constraints (ensuring manœuvres in the network do not become dangerous), physical limitations and equipment capacities (so equipment overload must be avoided) and economic considerations (operating the network must not become too expensive).

As figure 1.5 shows, function F is often modelled as a closed-loop with four basic components: the process, the control component, actuators and sensors. The behaviour of process P is controlled through manipulated variables V_m and the actual behaviour is monitored through control variables V_c. The process's state can alter because of changes in internal conditions and manipulated variables and also because of disturbances (D) not subject to adjustment and control. Actuators manipulate the behaviour of the process by changing the manipulated variables. Sensors monitor the process's actual behaviour by measuring controlled variables.

Deriving a mathematical model of the process is usually very difficult because in general it is highly non-linear. Therefore engineers often derive a control function using a description of part of the control process and by making certain simplifying assumptions. The purpose of the control component C is to adjust the manipulated variables to achieve the system goals despite disturbances. The controlled variables provide feedback, which the control component uses to monitor the process's behaviour. This feedback information is then used as a foundation for future control decisions as well as an indicator of whether the changes initiated previously have been achieved. Feedback is needed because the control component may have only partial control over the process.

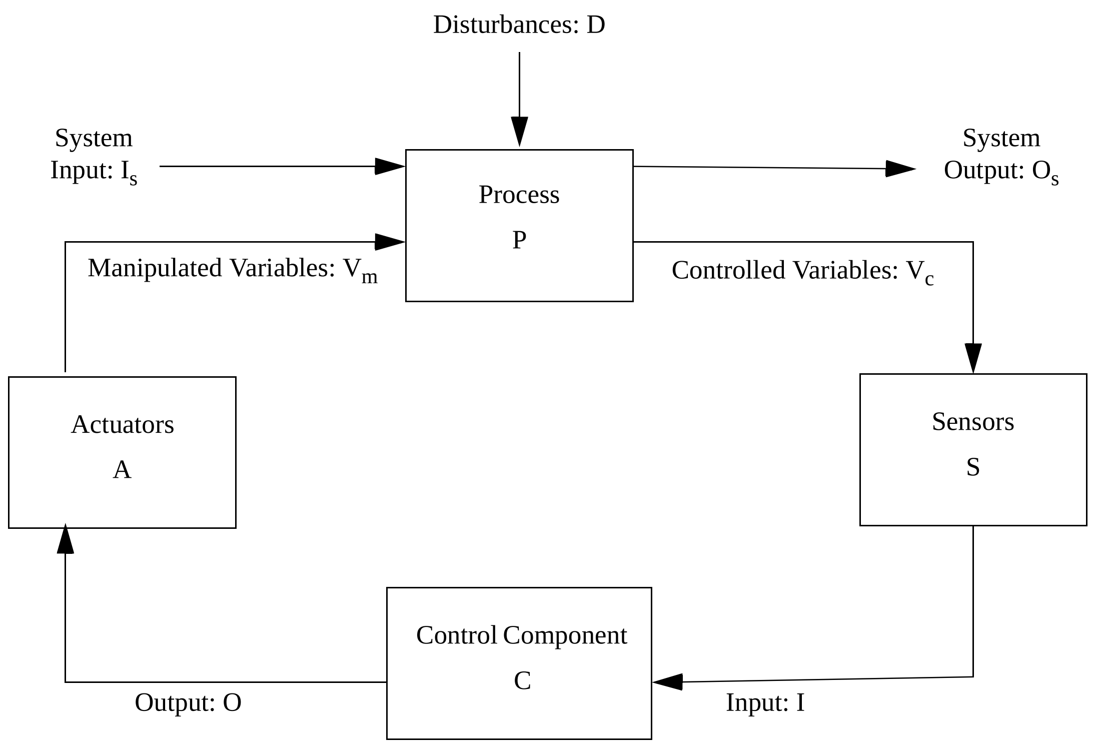

Figure 1.5: Model of a Typical Industrial Control Application (Leveson, 1990)

1.3.1 Monitoring Electricity Transportation Networks

To ensure the transportation network remains within the desired safety and economical constraints, it is equipped with a sophisticated data acquisition system. This system acquires enormous amounts of data. It provides information on the status of 20,000 elements (breakers, switches and protective relays) with a maximum polling cycle of ten seconds; and the value of 2000 analogue measures (power flows, voltages and frequency) every twenty seconds. All this information is sent to a central Dispatching Control Room (DCR) for analysis by control engineers (CEs) - see figure 1.6. The CEs analyse the information to determine when disturbances occur and how they can be repaired.

Before automation, analysis and planning the subsequent restoration were carried out manually. However as the amount of data grew and customers became ever more reliant on electricity for their businesses, the importance of making correct and timely decisions increased. A wrong choice or any substantial delay in reaching a decision could damage the network, necessitating the dispatch of a maintenance team and a network down time of several hours. This causes inconvenience to private consumers and may cost industrial concerns hundreds of thousands of pounds in terms of lost revenue or damage to equipment and goods. Even if no damage is caused immediately, the network may be left in an unstable state, which could cause all (or part of) it to be lost at a later stage.

To combat these problems, expert systems were exploited to provide an aid to the CE as and when the appropriate technology became available. Therefore a typical DCR contains a variety of automated functions, implemented at different times and by different people using different problem solving paradigms. As automation proliferated, the need for an integrated framework, which allows information and problem solving expertise to be shared, became apparent. Rather than scrap the existing systems and re-implement their functionality; it was decided to utilise the pre-existing systems and integrate them through a cooperative problem solving environment.

Figure 1.6: Partial view of a Dispatch Control Room (DCR)

Whenever there is an unplanned or sudden disconnection of the network, an emergency situation is signalled to the CE by displaying alarm messages related to changes in network elements on his screen. Elements which are monitored include breakers (devices which allow the (dis)connection of lines, transformers and busbars between them and which may be opened or closed from the DCR), fault recorders (mechanisms that record the voltage and intensity during the disturbance) and protective relays (mechanisms which trip breakers after sensing a fault). Figure 1.7 provides an example of breakers and intensity transformers at the hydroelectric power station in Aldeadavila (North Spain). A breaker has three phases, each consisting of two “V” shaped devices in series, as shown in the left part of the figure. The intensity transformers are on the righthand side, one for each phase of the breaker.

Figure 1.7 Breakers and Intensity Transformers

If the disturbance is sufficiently large and involves many elements, the job of ensuring the network remains in a safe state is too complex for a single CE and so several (typically 3 or more) cooperatively undertake the task. Each CE is assigned a specific task which involves interacting with a particular software system and with other CEs. One CE will be in charge of the system which analyses the alarm messages received and diagnoses their origin; he will also identify any missoperation of the network equipment. Another CE will be in charge of analyzing the opening and closing of network elements to determine those which are out of service, looking at the importance of the elements lost and of monitoring the evolution of the restoration process. Such analysis allows CEs to judge the criticality of the situation and how restoration is progressing.

While the CEs are engaged in their analysis, the auto-reclosing mechanisms[9] will be operating, provoking a reaction in the network. This reaction has to be analyzed by the CE attending to the diagnosis and by the one attending to the breakers in order to confirm the previous hypotheses, refine them or even change them.

After a while, the network reaches a steady state[10] and the CEs can start planning the actions required to restore the network. In this phase, one CE is responsible for planning the restoration. Once he arrives at a plan which complies with all the known constraints, its execution and monitoring starts. Monitoring is necessary because the plan may have been constructed incorrectly or be based upon incorrect assumptions about the status of the network.

The CE who produces the restoration plan then starts executing it; issuing commands from his console. While this is happening, a second CE analyzes the reaction of the network from the point of view of successful or unsuccessful restoration operations; a third CE checks the evolution of the area out of service to ensure that the plan’s critical elements are still in service. Any evolution of the disturbance which deviates from that which is expected is communicated by the second CE to the person in charge of the restoration, along with some suggestions imposed by the new constraints. The CE in charge of producing the restoration plan considers this information and decides whether to modify the current plan or to build a completely new one. Once an action is taken, the above process is repeated until the network reaches a state in which both safety and economical considerations are satisfied.

1.3.2 Detailed Functionality for Electricity Transportation Management

Diagnosis of disturbances and network restoration in the electricity transportation domain, involves several distinct tasks:

- Distinguish between messages which relate to disturbances and those which represent pre-planned maintenance operations. The latter are not relevant to the operator from the point of view of diagnosis and restoration, the former are.

[9]. Breakers have a mechanism for automatic reclosing. This is triggered when sufficient voltage returns or if sufficient time has elapsed for a non-permanent fault to disappear.
[10]. A state (configuration, voltages and energy flows) which remains until some action is performed by the CEs. This contrasts with the transient state, generated initially by a fault, which is constantly and automatically changing. Steady states are signified by the fact that during a certain period of time (eg 1 minute) there are no alarm messages.

- Once a disturbance has been identified, find out its type (transient or permanent), origin and extent.

- Safely restore the network to the best possible configuration, minimizing the amount of time that customers are without electricity.

Based on these functional requirements, the following software systems can be found in the DCR at Iberdrola.

CSI (*Control System Interface*): Surveys the network and signals whenever a disturbance occurs. Receives information from the field (from substation equipment monitoring breaker and relay status and electrical measurements in lines and transformers) and organises it into an understandable form.

BAI (*Blackout Area Identifier*): Calculates the group of elements out of service and monitors the evolution of the network (detecting whether the situation is progressing or if it is getting worse) in order to advise whether a new restoration plan is needed.

AAA (*Alarm Analysis Agent*) and **BRS** (*Breakers and Relays Supervisor*): Both systems locate the individual element at fault, analyze the permanence of the fault and indicate any damaged breakers. Two systems were built because there are different sources and types of data which provide information with different levels of certainty, different arrival times and different amounts of completeness. This functional duplication can enhance the speed of response, increase reliability through cross-checking and offer graceful degradation of performance.

SRA (*Service Restoration Agent*): Provides the user with a sequence of steps which restore the network. This job is currently performed by one operator, while other operators monitor the network and indicate if the situation is deteriorating.

A full exposition of the cooperative scenarios between this group of agents is given by Aarnts *et al.* (1991). The research described here concentrates on three of the agents - the AAA, BAI and CSI - and on the task of detecting and diagnosing faults. The restoration phase is not dealt with.

As the real systems are rather large and run on dedicated platforms, it was not possible to use them directly. Simulations were therefore employed instead. These “dummy” systems were written by Juan Perez and kindly made available for this work.

1.3.3 Benefits of Distributed AI in Electricity Transport Management

Many of the typical advantages of distributing both data and control are also applicable to the domain of electricity transportation management. However the following are especially pertinent in this application area (Jennings, 1991a):

- Enhanced problem solving

  by sharing knowledge and data, systems can make use of information which would not normally have been available to them (due to resource limitations, for example). Thus problem solving can be faster, of a higher quality, more accurate and so on.

- Ease of Extensibility

  new agents can be added as and when they become available without the need for significant system restructuring.

- Ease the burden on the operator

  in emergency situations operators are typically faced with a barrage of information from multiple sources which they have to collate, interpret and act upon. By allowing the subsystems to communicate directly the operator's cognitive load is reduced considerably, leaving him to concentrate on the strategic decisions to which he is better suited.

- Increased reliability

  because problem solving capability may be available at multiple sites within a community, failure at one node does not lead to total system collapse. The system exhibits graceful degradation of performance.

1.4 Objective and Contribution of Research

The main objectives and achievements of this research are to:

- Implement a novel multi-agent shell which contains significant amounts of inbuilt knowledge related to cooperation and control.

- Use the shell to illustrate a new paradigm for building multi-agent systems in which the designer starts by assembling pre-existing components and then augments them with any necessary domain specific knowledge.

- Develop a new model for coordinating the behaviour of groups of agents engaged in collaborative problem solving based on the notion of joint intentions.

- Use the model of collaborative problem solving to specify a high level architecture which clarifies and highlights the role of joint intentions in social environments.

- Implement the high level architecture and assess its benefits for guiding collaborative problem solving in complex, dynamic and unpredictable environments.

In pursuit of these aims, two multi-agent system shells have been built. The original system, GRATE, defines the type of generic knowledge about cooperation which needs to be included in a “component based” multi-agent system shell. As a consequence of using GRATE for electricity transport management, the need for a deeper theory of coordination became apparent. Such a theory was required to cope with situations in which things did not go according to plan or unpredictable events occurred. This work resulted in the development of GRATE*, which has a deep representation of multi-agent collaboration, meaning it should be flexible enough to perform well even in the most dynamic and unpredictable environments. Electricity transportation management was chosen to test this hypothesis because it is typical of a broad class of industrial applications. Individual agents have to take decisions using partial, imprecise views of the system and the process is itself inherently unpredictable and dynamic.

1.5 Organisation of Thesis

Chapter two provides a review of existing multi-agent system shells in order to place this work in perspective. It also describes the GRATE system in detail. Particular attention is given to the notion of agent models upon which much of GRATE is based. GRATE’s novel approach to building multi-agent applications is highlighted and the shortcomings which led to the joint responsibility work are outlined. The application of GRATE to cooperative fault detection in electricity transport management is explained in depth.

Chapter three reviews existing models of individual and joint intentions. Prominent models are described and evaluated in order to place the theoretical contribution of this research into perspective. The limitations of using individual intentions to describe collaborative problem solving are discussed. Finally the failings of existing models of joint intentions in complex and dynamic environments are enumerated.

Chapter four represents the main theoretical contribution of this research. The model of joint responsibility is defined using modal, temporal logics. A description of how it circumvents the shortcomings of the other models of joint intentions is given. The usefulness of the theory is then illustrated by applying the concepts to the electricity transportation domain.

Chapter five outlines the arguments for a new computer level specifically for social problem solving systems. It describes a high-level architecture for collaborative problem solving based on joint intentions. Implementation of the responsibility model is then discussed. Again examples are taken from the electricity transport domain.

Chapter six provides an empirical evaluation of the performance characteristics of joint responsibility. In particular, the hypothesis that the model of joint responsibility allows a high-level of coherence to be sustained even in the most dynamic and unpredictable environments is evaluated. The model's communication and resource overheads are also examined. To facilitate evaluation, comparative analysis with groups of agents which do not form explicit joint intentions and with teams in which the agents behave as selfish problem solvers is undertaken.

Finally chapter seven draws together the strands of research presented in this thesis and highlights some areas for further investigation. A brief comment on the methodology used to guide this research is also given.

2. DEVELOPING MULTI-AGENT SYSTEMS

This chapter describes the types of development environment which are available to the designer of multi-agent systems. Two broad approaches can be identified; systems which are purpose built to solve particular problems and systems which are general and can be applied to several types of problem or domain. This research is concerned with the latter type. Several exemplar frameworks are reviewed in order to highlight their salient characteristics and to enable the contribution of GRATE to be assessed.

GRATE differs from other development environments in that it contains generic, *in-built* knowledge related to cooperation and control. This knowledge can be utilised when constructing an application; meaning the designer does not have to start from scratch. A significant proportion of the knowledge which would have to have been encoded is already available, thus the amount of effort is reduced. An important feature of GRATE agents is their use of models to represent information about themselves and other community members. GRATE's models are placed in the context of existing work in this area and their role in attaining generality is explained. A novel paradigm for building systems based on generic knowledge is discussed and possible enhancements to the basic GRATE approach are described.

An illustration and assessment of the use of GRATE for cooperative fault diagnosis in electricity transportation management is given. The shortcomings of this work which necessitated a deeper model of collaborative problem solving are also highlighted.

2.1 Existing Multi-Agent System Environments

Within the field of DAI, two main types of software tool have been developed: *integrative systems* and *experimental testbeds* (Bond and Gasser, 1988). Integrative systems provide a framework to combine a variety of problem specific tools and methods into a useful whole, whereas experimental testbeds have their main emphasis on controlled experiments in which measurement and monitoring of problem solving activity are a prime consideration. Within the former category, two broad classes of system can be identified:

- systems which test a specific type of problem solver, a specific coordination technique or a particular domain

  eg DVMT which performs distributed sensing and interpretation (Lesser and Corkill, 1983), ATC for air traffic control (Cammarata *et al.*, 1983), HEARSAY II for the domain of speech understanding (Erman and Lesser, 1975) and YAMS a control system for a discrete manufacturing environment (Parunak, 1988).

- systems which are applicable to a broad class of problems or are general purpose eg MACE (Gasser *et al.*, 1988), ABE (Hayes-Roth *et al.*, 1988), MCS (Doran *et al.*, 1990), RATMAN (Bürckert and Müller, 1990), MAGES (Bouron *et al.*, 1990), CooperA (Avouris *et al.*, 1989) and DASEDIS (Burmeister and Sundermeyer, 1991).

The general testbeds are not targeted towards any particular application domain, they are appropriate for a broad class of problems. They each offer a range of facilities such as debugging and tracing tools, knowledge representation, reasoning capabilities, inter-agent communication and planning which the designer can utilise to construct a multi-agent application.

General purpose programming languages, and concurrent object-oriented ones (eg ACTORS (Agha, 1985) and ABCL/1 (Yonezawa and Tokoro, 1987)) in particular, are often used to build multi-agent systems. However they are not considered because of the limited facilities they provide to the application builder. They are general in the sense that the designer has complete flexibility over the system to be built and thus they can be used for a broad class of problems. However a significant amount of effort needs to be expended to impose the necessary structure. Language primitives are at a low level of abstraction and so many features essential for multi-agent systems must be coded explicitly. In the testbed approach, some structure and tools appropriate for social activity are provided; the designer then builds upon these facilities to construct a working system.

To provide a context for the GRATE work, it is necessary to provide a brief overview of some existing testbeds. One of the earliest, and perhaps the most quoted, system is MACE (Gasser *et al.*, 1988). MACE (Multi-Agent Computing Environment) is an instrumented testbed for building a wide range of experimental DAI systems at different levels of granularity. It is primarily a prototyping tool and has many similarities with a distributed object oriented system. It has been implemented in two versions; a parallel version on a hypercube and a simulated parallel system for carrying out repeatable experiments and testing.

MACE consists of five main components - a collection of problem solving agents which are specified using the MACE agent description language; system agents which interpret MACE commands, provide a standard user interface, contain agent builders and offer error handling, tracing and execution monitoring; facilities which all agents can share such as a pattern matcher, standard agent engines and standard error messages; a description database which contains information about all the agents and from which the executable agent images are constructed and finally kernels which handle communication and message routing, perform input and output and map agents onto processors.

CooperA (COOPERating Agents) is a software environment which supports the cooperation of heterogeneous, distributed, semi-autonomous knowledge based systems which are presented to the user via a customised interface (Avouris *et al.*, 1989). The CooperA environment has four layers. Firstly, the kernel which represents the cooperation shell and consists of a collection of initialisation procedures and system facilities that each agent can use. Secondly, a message passing mechanism which handles communication between agents. Thirdly, a collection of CooperA system agents which are independent of the application and perform operations of common interest; one example is the user interface agent which handles all input and output operations. The final component is the community of application specific agents; each having the capability to fulfill one or more tasks and embodying knowledge about the external world such as other agents, their location and capabilities.

MAGES (Multi AGEnt System) is a testbed which facilitates experimentation with different types of interactions between agents using heterogeneous architectures and behaviours (Bouron *et al.*, 1990). It allows a variety of agent architectures to be described through a hierarchy of agent types, sophisticated environment models, communication between different grain size agents and user friendly interface and debugging tools. MACE and CooperA provide tools for homogeneous agents (i.e. those whose internal architecture, control strategy and basic behaviour bear the same structure), whereas MAGES is built with the idea that heterogeneous agents can communicate without restraining themselves to a standard communication protocol.

MCS (Multiple agent Core System) differs from other testbeds in that it has a non-linear hierarchical planner as an inference engine which integrates plan generation, execution and monitoring (Doran *et al.*, 1990). One of the main motivations for its development was to experiment with the relationship between predictive planning and situated action. Each agent has independent access to the planner and has a set of plans and goals currently being worked on. Agents also have a database of beliefs and an associated set of demons which fire whenever their conditions match against beliefs in their database. There is also a user interface which allows the initial set up of agents and their contents, monitoring events and interactive updating of agents at run-time.

DASEDIS (Development And Simulation Environment for Distributed Intelligent Systems) is a testbed which has the notion of intentions as its main driving force. As indicated in the introduction, there are two types of intention; strategic ones which represent long term objectives and tactical ones which represent the short to mid term. In DASEDIS, the latter type correspond to sub-goals and plan steps and are directly bound to actions. The agent architecture is modular and contains a cognition component which represents the cognitive resources of the agent. This component is divided into two parts, one which performs local actions and one which performs actions which are inherently social in nature. However there is no explicit representation of commitment and no notion of joint intentions. DASEDIS agents also have sensors for receiving inputs from the environment, actuators for carrying out actions and a communication component which sends and receives messages. It also provides instruments for implementing, inspecting and observing the interaction between the problem solving modules and for building the interface with the user.

RATMAN (Rational Agents Testbed for Multi-Agent Networks) is a workbench for defining and testing the behaviour of rational agents in a multi-agent environment (Bürckert and Müller, 1990). It consists of four components - a specification kit which serves as the user interface for defining agents, their relations in the world and their status in the society; an agent toolbox which provides a hierarchically structured knowledge base together with reasoning facilities to model all features of the agents; a current world scenario and finally the status sequence and statistics box which visualises the sequence of changing worlds, the activity potential of the agents, the internal clock and the analysis of synchronisation processes. The toolbox provides classes of pre-existing knowledge, but this knowledge is related to problem solving within an agent rather than social problem solving; examples include knowledge related to time and space or learning.

GRATE is not a general testbed in the sense of the systems described above; rather it is an experimental vehicle which allows the notion of generic rules for cooperation and control to be explored. Therefore it is missing many desirable features essential for an easy to use development environment. GRATE has few debugging facilities over and above those of the LISP environment, the interface does not express social interactions in a natural manner, there are no facilities for obtaining statistics, for carrying out plan generation nor is there a language with which the agents can be described which is independent of their execution image. Also GRATE is targeted at agents which have the same level of sophistication and which are fairly homogeneous in nature.

However GRATE differs from all the aforementioned systems in that it contains inbuilt generic knowledge related to cooperation and control. This knowledge offers a new way of building multi-agent systems which places less of a burden on the application developer.

2.2 A New Paradigm for Building Multi-Agent Systems

One of the major open issues in DAI is that of simplifying the task of building multi-agent systems (Bond and Gasser, 1988). The approach adopted in this research is to provide the application developer with a new, more powerful type of environment - one which comes with *built-in generic knowledge*, not just specialised programming features. This knowledge can be broadly divided into three categories; controlling local activities, controlling social activities and assessing the current problem solving state with respect to both local and global considerations. Examples of generic knowledge associated with cooperative and situation assessment activities, respectively, are:

```
if  an agent has generated a piece of information and
    it believes that it could be of use to a fellow agent
then send that agent the information

if  an agent has a task to perform and
    it is not able to perform it locally
then seek assistance from another agent
```

Knowledge like this forms the basis of many social activities in a wide range of problems. Therefore it will have been encoded, either implicitly or explicitly, in a large number of multi-agent applications. The GRATE approach is to identify and utilise such generally applicable knowledge and build it into the development environment. Thus when constructing a system, the designer can utilise this pre-coded knowledge rather than have to represent it himself every time (see figure 2.1a). Any additional knowledge which is specific to the particular problem, and therefore not pre-existing in the environment, can be used to augment the available corpus.

This offers a novel paradigm for building multi-agent systems, in which the designer augments pre-existing knowledge; rather than building the whole system from scratch in each and every case. As figure 2.1b illustrates, there will be duplication between the knowledge required to build a multi-agent system for application P_1 and that for P_2. GRATE seeks to exploit this commonality by extracting it and building it into a general purpose shell. Note that the combination of the knowledge specific to P_1 and the general knowledge will be a super set of the knowledge required to build P_1. This approach enhances productivity and decreases development time, because some of the necessary knowledge acquisition and coding has already been carried out; the designer need only add domain specific reasoning.

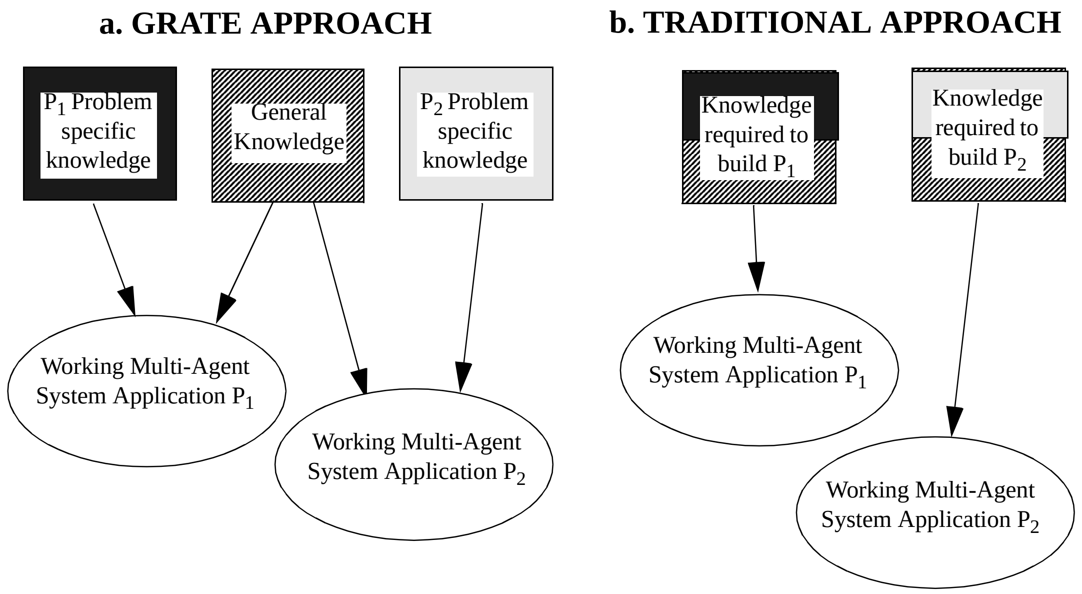
a. GRATE APPROACH
b. TRADITIONAL APPROACH
P1 Problem specific knowledge
General Knowledge
P2 Problem specific knowledge
Working Multi-Agent System Application P1
Working Multi-Agent System Application P2
Knowledge required to build P1
Knowledge required to build P2
Working Multi-Agent System Application P1
Working Multi-Agent System Application P2

Figure 2.1: A New Paradigm for Constructing Multi-Agent Systems

It is possible to construct a general corpus of knowledge about cooperative behaviour because all the domain-dependent information necessary for individuality is grouped together in specific data structures. These structures are called *agent models* and they provide an explicit representation of other agents in the world (Gasser *et al.*, 1988). The information which may be represented (i.e. the model's structure) is consistent across all applications, however some aspects may be left unfilled in particular instances. For example the high-level goals of a database system may not be represented, whereas for an expert system they may be an essential component. Agent model instantiations are highly domain dependent and must be carried out by the application builder.

Agent models provide a structured representation of the information necessary for controlling an agent in a social environment. Thus the generic rules can operate on this homogeneous structure rather than having to cope with the idiosyncracies of a particular problem which are maintained as the model's contents. This approach can be viewed as a further illustration of the notion that generic structure can be exploited when building specialised systems (Chandrasekeran, 1983a/b; Steels, 1990).

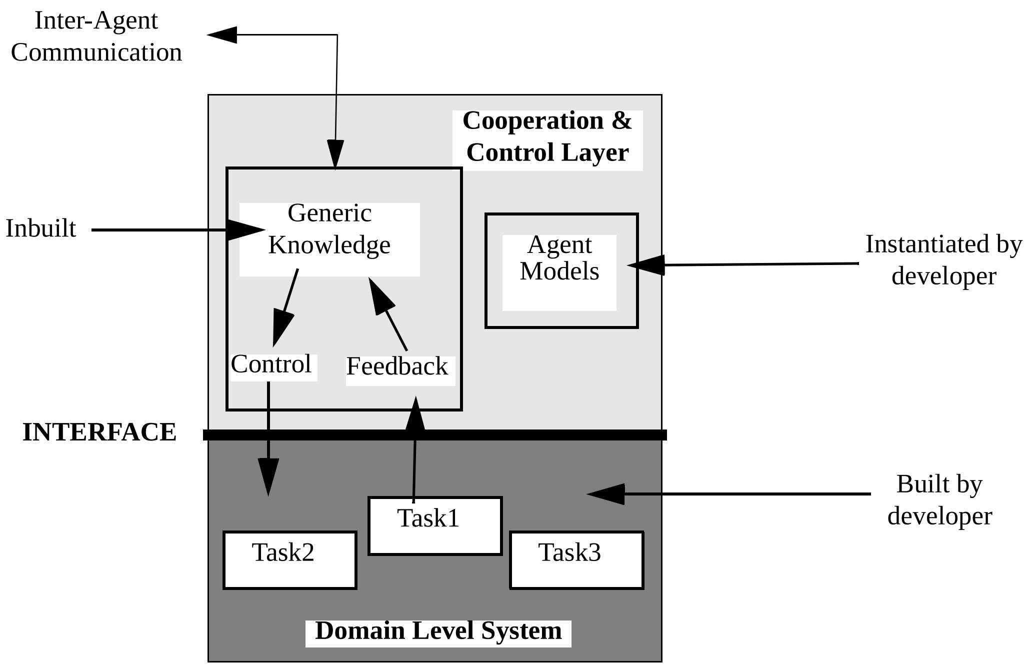

Figure 2.2: Overview of GRATE Agent

GRATE agents have two clearly identifiable components - a *cooperation and control layer* and a *domain level* system (see figure 2.2). This delineation of domain related knowledge and knowledge related to cooperation and control has several advantages. Firstly, it increases software reusability in that the cooperation layer can be used for several applications without having to disentangle the knowledge used to guide social activity from that used to solve domain level problems. Secondly, the two systems can be developed independently provided that they respect the interface definition, meaning greater productivity through concurrency and the ability to incorporate pre-existing systems.

The domain level system solves problems such as detecting when faults occur, locating the problem area or producing a restoration plan. These problems are expressed as tasks - atomic units of processing when viewed from the cooperation and control layer. For example a domain level system which locates faults may be composed of two tasks; one which produces a quick but approximate solution and another which uses this information to guide its accurate but lengthy evaluation. The cooperation layer controls the domain level system through a fixed set of commands such as “start” and “stop” and receives feedback such as “results produced” and “task finished”. This language is specified more extensively in section 2.4.2

To summarise, in order to construct a multi-agent system using GRATE the application builder has to:

1) Conceptualize and construct the domain level system in terms of tasks

2) Implement functions which enable the prescribed interaction with the cooperation and control layer to take place.

> So for example when the cooperation layer issues a command to start a task the actions which take this message and actually initiate the appropriate activities have to be written by the application builder. This is because the way of starting a task will vary from system to system; in some it may involve invoking a function and in others placing a request on an agenda. The same argument can also be applied to the other functions at this interface

3) Determine whether the generic knowledge embodied in GRATE is sufficient for the problem

> If this is not the case then it needs to be augmented with the necessary domain specific knowledge. This must be expressed in the GRATE production rule format and inserted in the appropriate place in the cooperation and control layer

4) Instantiate the agent models.

2.3 Agent Models

Most disciplines which study communities of social entities have recognised that sophisticated, coordinated behaviour requires participants to form and reason about representations of themselves and others. Agents need not model all community members, only those with which they need to interact; that is, those which have similar interests or capabilities, are able to provide useful services or are consumers of local services. This subset is called an agent’s *acquaintances* (Agha and Hewitt, 1985). Each agent maintains models of between zero and all other community members. It is usually mandatory for an agent to have a representation of itself. Such *self models* specify the names of tasks which can be performed locally, the information they need to execute and the results produced.

In the formative years of DAI research, the need for cooperating agents to represent information about others was recognised as a key issue. Wesson *et al.* (1981) stated that: “one of the most important principles we devised involved the use of models to simulate and predict other nodes’ activities”. Nowadays many DAI systems embody explicit models of others to facilitate the process of coordination and collaboration; a sample of which is given later in this section.

Human Computer Interaction researchers also recognise that interaction, between humans and computers in this case, can be improved if explicit models are maintained. Models are used to tailor interactions to individual users needs; taking into consideration their level of expertise, their experience, their goals and so on (Kass and Finin, 1989; Rissland, 1987; Wahlster and Kobsa, 1986).

Sociologists and social psychologists in the symbolic interactionist tradition also postulate that the primary mechanism for creating an organised society is the ability of individuals to generate and use internal models of other agents and of themselves, to reflect upon actions and their effects (Abraham, 1982; Bond 1989).

As well as providing a structure to represent domain dependent information, GRATE’s agent models are used in the coordination process itself. They play an important role in:

- Focusing Activity and Reducing Communication

  To make requests and spontaneously supply *relevant* information, agents must have representations of the capabilities and intentions of acquaintances. If such knowledge is not available requests and information must be sent blindly, meaning that agents who could have benefited might not be aware and those who have no interest may be unnecessarily disturbed. A directed approach also reduces communication (Genesereth *et al.*, 1988; Wesson *et al.*, 1981; Yang *et al.*, 1985).

- Coordination and Coherence

  To effectively coordinate joint actions, agents must have some expectation about the current and future activities of their acquaintances. For example in the RAND ATC work each agent (aeroplane) needed to know the plans of others as well as their speeds, headings, fuel levels, destinations and emergency statuses (Cammarata *et al.*, 1983).

- Self Reasoning

  To reason about local problem solving, agents must have a representation of the underlying domain level system. This information allows them to assess which tasks can be performed locally and which require social interaction, what information must be available to initiate a task, what results will be produced and when a particular activity will be finished.

In the ACTORS work, agent models were confined to representing the name and location of acquaintances. Recently systems have enriched these basic models to incorporate descriptions of the behaviour, capabilities, intentions and current actions of others (eg MACE and CooperA). However more is not always better. There is a tradeoff between computation and communication costs. The more complete the models, the more local computation required to maintain them, but the lower the communication overheads since much of the wastage is removed through greater focusing.

The exact nature of the information to be maintained in still an open issue (Huhns, 1988). Models should only include those facets of direct relevance for the modelling agent, therefore there may be several different representations of the same agent within the community. For instance an agent may possess many skills, however only pertinent ones will be included in any particular representation. Models have both static and dynamic components - examples of the former include the tasks an agent can carry out and the information it can produce; the latter includes active tasks and current objectives.

Most of the models are specified when the system is created. However in open and dynamic environments it is often necessary to be able to create new models after this initiation phase has been completed. Both MACE and CooperA support this, GRATE which is targeted at less dynamic domains does not. In GRATE it is assumed that a model’s static components are correct. So if a model indicates an agent can perform a particular task or is interested in a particular piece of information this represents the actual state of affairs. With more dynamic information, such as an intentions or problem solving status, some leeway for inaccuracy is allowed.

Self models are predominantly used for making decisions about local problem solving activities whereas acquaintance models are mainly used in social interactions. The two types of model are similar in structure but vary in the level of detail; in most cases the acquaintance model is not just a copy of the agent’s self model (cf CooperA). For example, a self model will represent all its currently active tasks, whereas an acquaintance model may represent a higher level description such as active plans (i.e. clusters of tasks).

A further distinction between self and acquaintance models can be found by examining the way in which the two models are built-up. An agent’s self model requires a tight-coupling with the underlying domain level system, therefore in most cases it is specified by the application developer. For this not to be the case, substantial learning and reasoning capabilities would be required. However it is reasonable to expect agents to build up models of acquaintances in a dynamic and flexible manner. Several approaches are possible and they are described in order of increasing system complexity (or decreasing application builder effort).

1) Application builder completely specifies (hardwires) static models.

2) Agents transfer their entire self description to all interested parties before any cooperative activity is started.

3) Agents go through an initiation phase in order to obtain the information they require. They must determine which agents can complete tasks which they need and who can supply essential information.

In CooperA each self model has a field stating the goals the agent cannot satisfy itself (Avouris *et al.*, 1989). When a community is constructed, agents broadcast their unsatisfiable goals to all others. Acquaintances which are able to provide the desired service reply and the originator updates its model to indicate this fact.

4) Agents assume others have the same structure and capabilities as themselves.

This notion of “stereotyping” is often used as a way of installing user models in interface design based on small amounts of information (Benyon *et al.*, 1987; Chin, 1986; Rich, 1979). In most instances attribute stereotyping in which values are assigned to specific model slots would be most appropriate, although in certain cases structural stereotyping may also be useful.

5) Models of other agents are built up dynamically during the course of problem solving and no explicit learning phase is entered into.

Trying to implicitly deduce information about others has been tried in the field of user modelling but success was only achieved in very limited circumstances (Kass and Finin, 1987; Wahlser and Kobsa, 1986).

In any domain which exhibits complex behaviour, such as industrial control, the first and last options in their pure form are impractical because of excessive effort on behalf of the designer and the necessity of sophisticated non-monotonic reasoning capabilities, respectively. Also the assumption of similarity (point 4) is not applicable because agents will typically be heterogeneous in nature. The merits of options two and three depend upon the relative costs of communication and computation and also the percentage of overlap between the agents. The greater the overlap and the lower the cost of communication, the greater the advantage of approach two over approach three (Khoui and Jennings, 1990).

In GRATE a compromise solution was adopted - the application designer specified most of the acquaintance models and then some general rules for dynamic updates were also included to provide some flexibility. Dynamic updating is achieved by relatively simple inference mechanisms (eg If an agent asks for a piece of information then assume it is interested in it). The type of adaption in which agents infer models of their acquaintances based on their behaviour, as proposed in (Sueyoshi and Tokoro, 1991), is unsuitable for real-size problems because of the agent complexity and the sophistication of their interrelationships.

2.3.1 Information Contained in Agent Models

At this stage of DAI research there has been no attempt to define the information which should be contained in agent models in a rigorous manner. Only classifications based on experiences or intuitions have been produced (Bond and Gasser, 1988; Brandau and Weihmayer, 1989; Werner, 1989). This problem is exacerbated by the fact that the knowledge needed is dependent on the type of social interaction and the distribution of tasks within the community (Roda and Jennings, 1991). For instance in order to carry out result sharing, agents only need to know what their acquaintances are interested in. To carry out task sharing in situations where only one agent can carry out each task, an agent need only know who can perform it. When there is a choice meta-information such as reliability and speed must be brought to bear. To provide grounds for comparison, existing systems employing agent models are examined before the GRATE models are detailed.

2.3.2 Agent Models in MACE

Every MACE agent has a personal, partial model of the world in which it resides. This model encompasses knowledge of itself and other agents. It creates an image of it’s environment which it can interact with and dynamically alter as its knowledge grows (Gasser *et al.*, 1988). Models have the following structure:

Name: Name of the agent being modelled.

Class: Name of the modelled agent’s class.

Address: Location of the modelled agent (used for communication purposes).

Role: Relationships the modelled agent bears to this agent. Several predefined roles exist and there are facilities for the user to introduce application specific ones. Examples of predefined roles are “my-org-manager” which indicates that the model is of the top-level manager of the organisation of which this agent is part and “co-worker” which represents the situation in which the model is of a member of the organisation of which this agent is also a part.

Skills: What this agent knows to be the capabilities of the modelled agent. Represented as pairs consisting of patterns which match goals the skill will achieve and a set of alternative procedures for achieving them. There are two types of method - explicit procedures written by the system developer which can be executed directly and agent references which indicate agents who can perform the skill (see 2.3.5 for an example).

Goals: What the agent understands the modelled agent wants to achieve. Represented by patterns which match programmer-specified descriptions of agent goals.

Plans: A view of the way the modelled agent will achieve its goals. Represented by a partially ordered collection of goals or skills.

2.3.3 Agent Models in CooperA

CooperA agents contain a self-description, similar to a self-model, which specifies static information (eg address, skills and goals which are unattainable locally) as well as dynamic information (eg current status and the goals it is currently pursuing) (Avouris *et al.*, 1989).

Agents also maintain models of their acquaintances; these models being based upon the following information structures: Modules, Yellow-Pages and Interested-in-Tables. The Modules table contains a reference to the other agents which are known in the community. The Yellow-Pages is a table of associations which relate each of the modelling agent’s unfulfilled goals to a list of acquaintances which have the potential to satisfy them. The Interested-in-Table associates information the local agent is capable of producing together and agents which could be interested in receiving it.

The CooperA Kernel also provides a mechanism for creation of acquaintance models based upon the self-models of the agents participating in a particular multi-agent community. This operation is performed at configuration time before the community starts actively solving problems. This facility is not available in GRATE; as described earlier, the application builder has to manually construct the acquaintance models.

2.3.4 Formal Agent Models

Formal models, based on game theory payoff matrices (Luce and Raiffa, 1957), have also been examined as a means of modelling agents in multi-agent systems (Genesereth *et al.*, 1988; Ginsberg, 1988; Rosenschein and Genesereth, 1985). The models express the value (utility) agents assign to various options and can be combined with notions of rationality to predict what choices an agent will make. For example given the following payoff matrix for an interaction between agents 1 and 2:

AGENT 2

		C	D
AGENT 1	A	1/4	0/5
	B	3/2	2/7

agent 1 can reason that agent 2 will select option D in all cases since 5 is greater than 4 and 7 is greater than 2. Given this fact, the rational action which agent 1 should take is B since this gives a payoff of 2 as opposed to 0 if it chooses A.

However, the assumptions on which this work is based are at present too limiting to be applicable to the type of environments for which GRATE is designed. Most of this work assumes a single encounter, no consideration is given to any past or future interaction and no attempt is made to influence the other agents' intentions. It is assumed that agents have common knowledge of the payoff matrix, a possibly unrealistic assumption considering the loosely coupled nature of the agents. A final drawback is that, even if the matrix were known, it may quickly become intractable for large games involving many agents and outcomes.

2.3.5 GRATE Agent Models

The information contained in GRATE's models can be classified into the categories listed below. Similar classifications have been attempted by Werner (1989) who includes intentional, evaluative and state knowledge but not domain or capability knowledge and by Brandau and Weihmayer (1989) who include state and capability knowledge but not domain, evaluative or intentional knowledge. Examples of each category are outlined here and specified in greater detail later in the section.

- State Knowledge (eg solution progress)

  Indicates the activities which are currently being carried out, how far they have progressed and when they are likely to be completed.

- Capability Knowledge (eg task descriptions, recipes)

  Knowledge about the tasks which can be performed, how they are combined to achieve particular results, the information they require and the results which can be expected.

- Intentional Knowledge (eg intentions)

  A high level description of the targets an agent is working towards and to which it has committed itself. Complex agents are likely to have several intentions active at any one time and an agent's intentions may conflict with those of others within the community.

- Evaluative Knowledge (eg time to complete task, intention end time)

  When faced with several means for achieving the same end, evaluative knowledge provides a means of distinguishing between them.

- Domain Knowledge (eg recipe priority, recipe triggers)

  Facts and relationships which are true in the environment in which the agent is operating and which are relevant for defining its behaviour.

GRATE’s models contain more information than those of either MACE or CooperA. This manifests itself in terms of the range of information which is maintained and also its depth. For instance neither MACE nor CooperA maintain a representation of evaluative knowledge or of active intentions. Even where there are overlaps, GRATE’s models are more detailed and declarative. Consider the following portion from a self model in MACE for a manager in a contract net scenario (Gasser *et al.*, 1988):

```
[skills (ANNOUNCE-TASK $task-spec
            (ref (acquaintance with
                    [roles (CNODE-MANAGER POTENTIAL-NQ-CONTRACTOR)]
                    [goals (TASK-ANNOUNCEMENT $)])
            (by-sending ‘(TASK-ANNOUNCEMENT $task-spec)))]
```

This indicates that in order to be a manager of a contract net the agent must have the ability to announce tasks and be able to take on the role of a CNODE-MANAGER. It also specifies how this task is to be achieved, by invoking the function TASK-ANNOUNCEMENT. GRATE adopts a more declarative approach, describing the task in terms of its inputs and expected results, keeping execution considerations separate.

This more comprehensive representation is needed because of the specific functional role which agent models play in GRATE. In both MACE and CooperA, the reasoning mechanisms which exploit the models are neither explicit nor general; therefore there is not such a need for clear and detailed knowledge representation.

An actual instantiation for the electricity transportation domain is shown below. It shows a portion of the AAA’s self model, there are other tasks and recipes but these are not shown because of space limitations. The key *types* of knowledge are capitalised and made bold. The start of each individual instance is written in bold.

AQUAINTANCE IDENTIFIER: AAA

RECIPES[1]:

Name: (UPDATE-TOPOLOGY-USING-ALARMS)

```
Trigger: (   (INFO-AVAILABLE BLOCK-ALARM-MESSAGES)
             (NETWORK-UPDATE-NOT-ESSENTIAL)          )
Actions: (((START (UPDATE-TOPOLOGY-WITH-ALARMS
                                         ?BLOCK-ALARM-MESSAGES))))
```

[1] Recipes are a sequence of steps which are known by an agent (Pollack, 1990) - see section 3.3.2 for more details. For example the AAA knows a way of diagnosing faults which involves going through an initial hypothesis generation phase and then performing a detailed validation. The full recipe construction language is given in appendix A along with several examples of its use.

```
Time to Execute: 5            Priority: 3        Outcome: NIL
Instantiations:               Identifier: NIL
Component of: (SATISFY-LOCAL-GOAL (UPDATE-TOPOLOGY-USING-ALARMS))
Current Action: NIL           Local Bindings: NIL
```

Name: (DIAGNOSE-FAULT)

```
Trigger: (   (INFO-AVAILABLE DISTURBANCE-DETECTION-MESSAGE)
             (FAULT-DETECTED))
Actions: (   ((START (HYPOTHESIS-GENERATION (> FAULT-HYPOTHESES)))
              (GET-INFO INITIAL-ELTS-OUT-OF-SERVICE))
           ((WHILE-WHEN[2] (BLACK-OUT-AREA-NOT-AVAILABLE) DO
               (((START (HYPOTHESIS-VALIDATION
                           (< FAULT-HYPOTHESES) (> VALIDATED-HYPOTHESES)))))
              (WHEN ((SUSPEND (HYPOTHESIS-VALIDATION)))
                    ((START (HYPOTHESIS-REFINEMENT
                              (< FAULT-HYPOTHESES) (> REFINED-HYPOTHESES))))
                    ((REACTIVATE (HYPOTHESIS-VALIDATION
                                             (> REFINED-HYPOTHESES))))))))
Time to Execute: 64           Priority: 10       Outcome: (VALIDATED-HYPOTHESES)
Instantiations:               Identifier: G1907
Component of: (SATISFY-LOCAL-GOAL (DIAGNOSE-FAULT))
```

Current Action: (((G1919 ((START (HYPOTHESIS-GENERATION ?DISTURBANCE-DETECTION-MESSAGE ?BLOCK-ALARM-MESSAGES ?FAULT-HYPOTHESES)))) (TOP-LEVEL (G1919 ((START (HYPOTHESIS-GENERATION ?DISTURBANCE-DETECTION-MESSAGE ?BLOCK-ALARM-MESSAGES ?FAULT-HYPOTHESES)))) NIL)))

Local Bindings:((FAULT-HYPOTHESES ?) (BLOCK-ALARM-MESSAGES (BR HERNANI 3 (CO) LOCAL BR HERNANI 4 (CO) LOCAL BR HERNANI 6 (CO) LOCAL)) (DISTURBANCE-DETECTION-MESSAGE (DISTURBANCE)))

SOLUTION PROGRESS:

Task Name: (HYPOTHESIS-GENERATION)

Information Passed: ((BLOCK-ALARM-MESSAGES (BR HERNANI 3 (CO) LOCAL BR HERNANI 4 (CO) LOCAL BR HERNANI 6 (CO) LOCAL)) (DISTURBANCE-DETECTION-MESSAGE (DISTURBANCE)))

Status: EXECUTING Priority: 10

Rationale: ((PLAN-ACTION (DIAGNOSE-FAULT) G1919 G1907))

[2] WHILE-WHEN p DO q WHEN r means while p is true do q; when (if) p becomes false do r.

INTENTIONS:

Name: (ACHIEVE (DIAGNOSE-FAULT))

Motivation: ((SATISFY-LOCAL-GOAL (DIAGNOSE-FAULT)))

Chosen Recipe: (DIAGNOSE-FAULT)   Start Time: 8

Maximum End Time: 72   Duration: 64   Priority: 10

Status: ACTIVATED   Outcome: (FAULT-HYPOTHESES)

Name: (ACHIEVE (UPDATE-TOPOLOGY-USING-ALARMS))

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

Chosen Recipe: (UPDATE-TOPOLOGY-USING-ALARMS)   Start Time: 73

Maximum End Time: 78   Duration: 5   Priority: 3

Status: PENDING   Outcome: NIL

TASKS:

Task Identifier: (HYPOTHESIS-GENERATION)

Necessary Requirements: (DISTURBANCE-DETECTION-MESSAGE
BLOCK-ALARM-MESSAGES)

Results Produced: (FAULT-HYPOTHESES)   Time to execute: 15

Task Identifier: (UPDATE-TOPOLOGY-WITH-ALARMS)

Necessary Requirements: (BLOCK-ALARM-MESSAGES)

Results Produced: NIL   Time to execute: 3

This description indicates that the AAA has two current intentions. The one which is currently active is that of diagnosing faults and a second which will start immediately afterwards is that of updating the network topology model using alarm messages. These two intentions are both being executed for local reasons and are satisfied by carrying out a local recipe. The diagnose fault recipe is currently on the first specified action of hypothesis generation. This task has received two inputs, the block of alarm messages and the fact that a disturbance has taken place and is currently calculating the initial value for the fault hypotheses.

2.3.6 Location of Agent Models

Having determined that agents need to model one another and identified the type of knowledge to be represented, the final decision concerns the location of the models. Three alternatives were considered:

- Agents broadcast their capabilities on demand (i.e. no explicit acquaintance models, only self models).

- Acquaintance models sited in supervisory agent.

• Acquaintance models maintained inside each agent.

In a pure broadcast system, when an agent wants to obtain information or services, it sends requests to all the community members of which it is aware of. For example in the contract net protocol when an agent (the originator) encounters a goal which it is unable to complete locally, it sends out a request to all community members (the recipients) (Smith, 1980). The recipients then evaluate whether they are able to undertake the task and return a bid; when all the bids are received the originator decides with which agent the task will be placed.

The main advantage of this approach is it’s openness (Hewitt, 1985) and flexibility. No assumptions are made about the capabilities of others and changes are easily accommodated. The major disadvantage is the high communication overhead caused by the significant proportion of wastage. To rectify this problem, the basic paradigm has been adapted by various researchers to incorporate some primitive agent modelling notions. These include focused addressing in which agents tag general information about their surplus capacity onto bid proposals (Rammamritham and Stankovic, 1984) and audience restriction in which agents maintain a record of acquaintances who have previously responded to similar bids (Parunak, 1988). The success of broadcast based approaches is also highly reliant on the delay and error rate of the communication channels. For the reasons outlined, it was deemed inappropriate as the primary form of acquaintance model organisation.

If models are stored inside a supervisor then a multi-level hierarchy is imposed on the system. Supervisors maintain models of their group members, of themselves and of others at the same level in the hierarchy. Group members only maintain models of themselves. When subordinate agents encounter goals which they are unable to complete locally, their supervisor is informed. If the supervisor knows of a suitable subordinate then the goal is designated immediately. If not, the supervisor may enter into dialogue with other supervisors, either at the same level or a higher level, to determine whether they are aware of an appropriate agent.

The main advantage of this approach is that a supervisor has a wider perspective than any single agent can reasonably aim to achieve, meaning that coordination is easier. However as indicated in chapter one, hierarchical control has many drawbacks in complex applications even if it is feasible.

The final option is for each agent to maintain its own model of its acquaintances. This is the approach adopted by MACE, CooperA and PAN (Wesson *et al.*, 1981) and the user models of GRUNDY (Rich, 1979) and KBET (Trenouth and Ford, 1989). It was chosen for GRATE because of the limitations in communications bandwidth and the need to support loosely coupled systems - respectively ruling out the broadcast and hierarchical approaches. This decision is also borne out by a series of comparative experiments which concluded that the performance of systems in which control was distributed consistently surpassed that of hierarchical organisations (Wesson *et al.*, 1981). They also concluded that, the distributed structure:

- communicated more effectively, using fewer messages to achieve comparable performance.
- prompted faster communication, with agents responding to requests for information far more quickly.

A major drawback of this approach is that several models of the same agent exist in the community. Thus if new members join the community, each interested agent must form its own model of the new arrival. Similarly, if an agent's capabilities change (eg through adaption) then all the models must be updated accordingly.

2.4 A GRATE System

GRATE is implemented in Allegro Common LISP and CLOS (Keene, 1989) on a SUN SPARC workstation (Jennings *et al.*, 1992; Jennings, 1992c). An object oriented approach was adopted because of the natural way in which notions such as encapsulation, local control and message passing can be encoded. The case for using this as an implementation paradigm is made more fully in (Briot and Gasser, 1991) and can be observed by the success of languages such as ACTORS (Agha, 1986), ABCL/1 (Yonezawa and Tokoro, 1987) and Mering-IV (Feber and Briot, 1988).

2.4.1 GRATE Agents

By having distributed control, an individual has two distinct roles - as a *team member* acting in a community of cooperating agents and as an *individual*. Much of the early work on Distributed AI concentrated almost exclusively on the former and paid scant regard to the latter. However contemporary DAI, with its greater emphasis on autonomous agents, also highlights the role of the individual (Demazeau and Muller, 1990/91). When designing GRATE, both aspects were taken into consideration, meaning an agent must:

- Direct local problem solving

  decide which tasks to launch and when, their relative priorities and how best to interleave them.

- Coordinate activity with that of others

  when and how to initiate cooperative activity, how to respond to cooperative initiations and which activities require synchronization.

When defining a modular architecture, it is intuitively appealing to reflect this binary distinction directly. However upon deeper analysis, there is a significant class of activities which fall into a grey area between the two. They are concerned with *situation assessment*; for example determining which tasks should be carried out locally and which should be solved with aid of others, what cooperation requests should be honoured and which should not, the relative priority of activities and so on. Therefore to promote a cleaner separation of concerns, GRATE has three main modules in which situation assessment acts as an interface between the local and cooperative control mechanisms (see figure 2.3). The assessment module informs the control which tasks should be performed and their relative priorities; it is then the control module's responsibility to ensure this request is fulfilled. Similarly, the need to initiate social activity is detected by the assessment module and the responsibility for realising it left to the cooperation module.

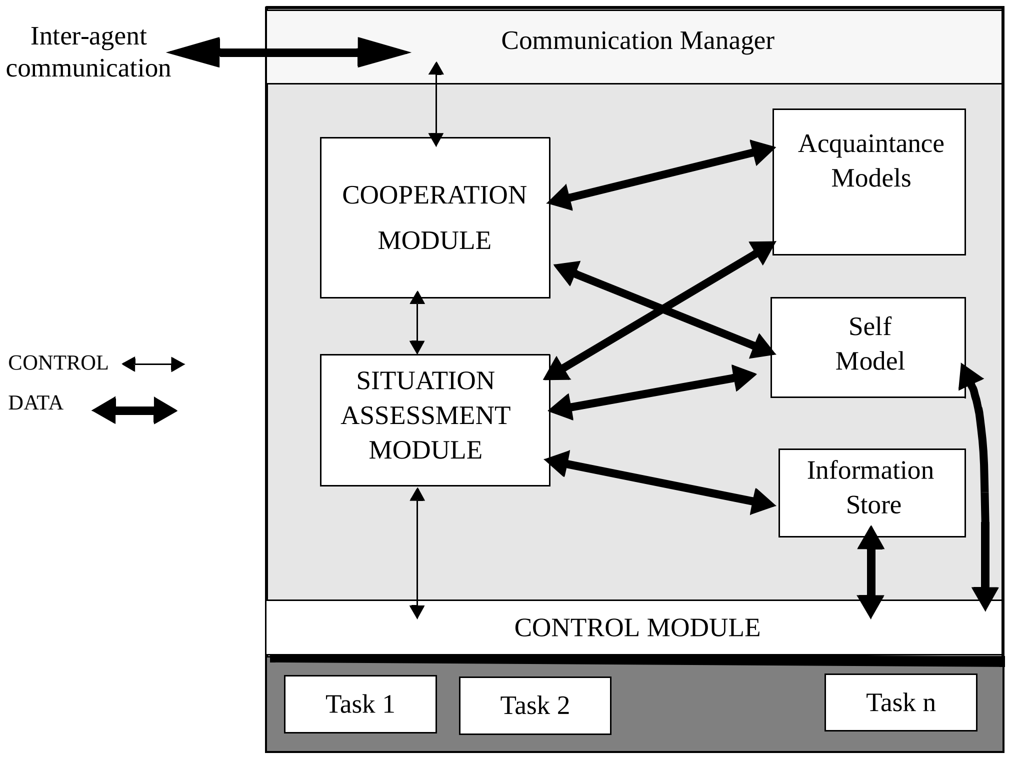

Figure 2.3: Detailed GRATE Agent Architecture

The information store provides a repository for all domain information which the underlying 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 and have been described thoroughly in the previous section.

Each of the three main modules is implemented as a separate forward-chaining, production system with its own inference engine and local working memory[3]. Communication between modules is via message passing, there is no shared memory. Each module's inference engine may be identical, as is the case with GRATE, or may be tailored to its particular the needs. For example, the control module might need to respond rapidly to certain key events and hence require a different inference

[3]. Modules are implemented using the Allegro Common LISP multiprocessing package, meaning they operate asynchronously and their scheduling and interleaving is handled by the UNIX environment.

mechanism to the cooperation module in which events can be handled on a first come first served basis. The generic rules are written in a standard if-then format, as illustrated in section 2.5.

2.4.2 Control Module

The control module is the cooperation and control layer's interface to the domain level system and is responsible for managing all interactions with it. This interaction is controlled through the following set of primitives:

➪From GRATE to the domain level system:

- START task inputs
- KILL task
- SUSPEND task
- REACTIVATE task
- MORE-INFO-AVAILABLE task info

➪ From the domain level system to GRATE:

- RESULTS-PRODUCED task
- DIRECTIVE-EXECUTED directive
- DIRECTIVE-FAILED directive
- FINISHED task

In real-size problems, task execution management requires sophisticated reasoning and control. Tasks cannot be suspended or aborted arbitrarily as this may leave the system in an incoherent state. For instance if a domain level model (eg a network description in electricity transport management) is being updated, then in most circumstances it would be unwise to suspend or abort it. Task interleaving also has to be managed so that results are generated according to their relative priority, bearing in mind the constraints which are caused by the tasks themselves.

GRATE's control rules can be divided into the following main categories; there are rules for dealing with:

- Controlling the execution of tasks based on directives from the situation assessment module (eg stop, suspend and reactivate)

- Ensuring the information required by a task is available. There are three potential sources.

    - The situation assessment module can provide it

    - It may be available locally but not supplied by the assessment module in which case it must be retrieved from the information store

- It may not be available locally in which case the assessment module must be informed. It will then decide how to proceed.

• If a task is waiting for a particular piece of information and it becomes available then pass it on.

• Determining whether task execution can begin. For instance checking that all information essential for task execution is available.

• Detecting when the task has finished. At which point gather the results and pass them to the situation assessment module.

• If optional information becomes available (i.e. information which could be used in task, but which is not mandatory) determine what to do with it.

- If the task has not yet started, add it to the list of information to be passed

- If the task has started then inject it into the process.

2.4.3 Situation Assessment Module

This module serves as a link and balancer between the agent's role as an individual and as a community member. It decides which activities should be performed locally and which should be delegated, which requests for cooperation should be honoured and how, which plans should be activated, what action should be taken as a result of freshly arriving information and so on. It issues instructions to, and receives feedback from, the other two modules. Typical requests to the cooperation module include "get information X" and "get task Y executed by an acquaintance". Requests to the control module are of the form "stop task T1" and "execute task T2". These two perspectives are reflected in the types of rule contained in this module.

Firstly there are rules for assessing and controlling local problem solving:

• If the trigger conditions for a plan are satisfied and it is appropriate for that plan to be started then adopt the intention of completing it.

• If there is a task to perform and it can be started then instruct the control module to proceed.

• If a plan step is completed then perform the next one.

• If an information request is received and it can be satisfied then select which task to execute:

- If there is already an active task which will produce the information then use it

- If no task is currently active, then select the most appropriate one for generating the required information

• If a goal request is received and that task is already active then use the existing task rather than duplicating activity.

• If information is needed before a task can proceed and it is worth generating it and it can be generated locally then adopt the intention of producing it

• If information useful to a task becomes available then pass it onto the control module

• If unrequested information arrives (information which has not been explicitly asked for) then assess what it can be used for:

- If it is a mandatory input for a task which is to become active then pass it onto the control module so that it can be used.

- If it is an optional input for a task which is (or is about to become) active then pass it onto the control module to determine whether it can be used.

- See whether it can act as a trigger for a local plan.

- If it will be of use at a later stage then store it, otherwise discard it.

• If it is necessary to alter the execution of a task (eg suspend, abort, etc.), then inform the control module.

Secondly there are rules for assessing and evaluating social interactions:

• If a task has to be performed and it cannot be achieved locally then instruct the cooperation module to get an acquaintance to do it.

• If a plan has finished then inform the cooperation module of its outcome.

• If information is needed before a task can proceed and it is worth generating it and it cannot be generated locally then ask the cooperation module to find an agent capable of supplying it.

2.4.4 Cooperation Module

The cooperation module is responsible for managing the agent's social activities. The need for such activity is detected by the situation assessment module, but its realisation is left to this module. This is also the module where requests from other agents are first received. Three primary objectives related to the agent's social role have been identified.

Firstly there is establishing new social interactions. This module has to decide how requests can be best satisfied. Two forms of cooperation are presently supported: task and result sharing. The module must decide with which acquaintances the interaction should take place (i.e. which agents to request aid from and which agents to disseminate information to).

- If a need for information has been detected, determine which agents are capable of producing it, then select one of them and send a request.
- Similar reasoning is also used to initiate a request for a task to be carried out by another agent.
- If information has been produced locally (by the execution of a task) then determine which acquaintances (if any) it would be of use to.

Secondly the module has to maintain cooperative activity once it has been established, tracking its progress until successful completion. So, for example, if an agent agrees to do a task because an acquaintance asked it to, then this interaction needs to be monitored. Tracking in this case involves sending any relevant intermediate results and ensuring that upon completion a final report describing the status and results is sent to the originating agent.

- If a social request has been satisfied, inform the originator. Determine whether other agents may benefit from receiving the outcome.
- When an originator of social activity is informed that its request has been completed, pass the results onto the situation assessment module.

Finally the cooperation module has to respond to cooperation initiations from other agents. Typically it cannot determine whether a request should be honoured without referring to the situation assessment module. However if the request is for a service which cannot be performed, then it can be rejected at this level without bothering the situation assessment.

- If a request for a service (task or information requirement) is received:
    - Ascertain whether the service can be provided and if not reject it.
    - If the request is for information which is already available then return this information as the result of the request
    - If the service can be met locally then pass it onto the situation assessment module for evaluation

2.5 Achieving Generality in GRATE

Having specified the generic rules and types of knowledge which reside in the agent models, it is important to see how these two combine to define useful behaviour in a social environment. The underlying assumption is that meaningful behaviour can be described in terms of the *type* of information maintained in the agent models (i.e. the structure) rather than the information itself. Expressed in more conventional terms, the models and the generic rules can be seen as an abstract data type. The information is actually stored in the models and the generic rules provide the procedural semantics. To illustrate this notion, several examples are given; the first group relate to controlling social activity and the second group to controlling the domain level system.

2.5.1 Cooperative Know-How

Within the acquaintance model definition there is a slot which expresses an agent’s interest in particular pieces of information (capability knowledge):

$$\text{INTERESTS: } \{<I_1, \text{Cond}_1>, <I_2, \text{Cond}_2>....\}$$

Intuitively this means if the modelling agent has information Ω and that within the interest slot of an acquaintance model it finds a tuple involving Ω (say $<\Omega, \omega>$), then if ω is true Ω is relevant for that acquaintance. Expressing this in a declarative form leads to the following production rule which is contained in GRATE’s cooperation module:

```
(rule cooperation-module-5
   (IF (INFO-GENERATED ?AGENT ?INFO ?GOAL))
   (THEN (FIND-INTERESTED-ACQUAINTANCES ?AGENT ?INFO ?GOAL ?ACQS)
         (SEND-TO-INTERESTED-ACQUAINTANCES ?AGENT ?INFO ?GOAL
                                                          ?ACQS)))
```

The rule is independent of any application and can be considered relevant for all cooperative situations. It shows that the rule interprets the slot structure and can be defined without actually knowing what Ω or ω are.

Active tasks have a solution progress description which contains the input values, execution status, priority and reason for execution. Cooperation rule 19 makes use of this descriptor to obtain the reason why a task and its associated plan was activated. If it was to satisfy an information request, then a generic rule is needed to ensure that the social activity is terminated properly when the desired information is available. This is part of the cooperation module’s functionality related to the tracking of social activities.

```
(rule cooperation-module-19
   (IF (PLAN-FINISHED ?PLAN ?RESULTS
                           (SATISFY-INFO-REQUEST ?INFO ?ORIG) ?ID))
   (THEN (EXTRACT-DESIRED-VALUE ?RESULTS ?INFO ?DESIRED-VALUE)
         (ANSWER-INFO-REQUEST ?INFO ?DESIRED-VALUE ?ORIG)
         (DELETE-MOTIVATION SELF ?PLAN
                                (SATISFY-INFO-REQUEST ?INFO ?ORIG))
         (SEE-IF-FURTHER-MOTIVATIONS ?PLAN ?RESULTS ?ID)))
```

If the situation assessment module has decided a particular task should be executed then it must ascertain whether it can be performed locally. If it can, the assessment module must determine whether to perform it locally or whether to try and delegate it. If the task cannot be performed locally, then the agent has no choice but to ask an acquaintance if it wants the task completed. Such generic knowledge about social behaviour, requires the agent to have a representation of the activities which can be performed locally - GRATE agents encode this in task descriptors.

```
(rule situation-assessment-14
   (IF(HAS-GOAL ?GOAL ?INPUTS ?MOTIVE ?PRIORITY)
      (CANNOT-PERFORM-LOCALLY ?GOAL))
   (THEN (TELL-MODULE COOPERATION-MODULE GOAL-AID-NEEDED
                                        ?GOAL ?MOTIVE ?PRIORITY)))
```

As all these rules illustrate, representing information in the agent models is only useful if it influences an agent's behaviour. If no action is ever taken on the basis of believing a proposition then there is no point in even deducing it. This observation offers a consistency constraint for the developer of the general rules: all information slots of the agent model must appear in the antecedent of at least one generic production rule

2.5.2 Controlling Domain Level Tasks

One of the most important functions of the control module is to launch tasks in the domain level system. The responsibility for deciding which tasks should be executed rests with the situation assessment module, however their initiation and interleaving is left to the control module.

Control module rules 5 and 6 provide an illustration of generic rules associated with the execution of domain level tasks. Within the agent's self model there is a description of all the tasks which can be performed locally (capability knowledge). This defines the task's inputs, other information which could be exploited and the expected results. Again the role of generic knowledge at this level is to define situations which must always be true before other actions can occur. With regards to starting a domain level task, the following general rules can be specified:

```
(rule control-module-5
   (IF  (EXECUTE-TASK ?Agent ?Task)
        (ALL-INFORMATION-AVAILABLE ?Agent ?Task))
   (THEN (TASK-READY-FOR-EXECUTION ?Agent ?Task)))

(rule control-module-6
  (IF   (EXECUTE-TASK ?Agent ?Task)
        (MISSING-INFORMATION ?Agent ?Task ?Info))
  (THEN (TELL-MODULE SITUATION-ASSESSMENT INFO-MISSING ?Agent
                                                 ?Task ?Info)))
```

These rules place an interpretation on the “required information” slot of the task description (see 2.3.5). They define that a task should be placed on the local agenda if all the necessary information is available (rule 5) or that if there is missing information then a request should be made to the situation assessment module (rule 6). Assessment may mean that the request initiates social activity (eg if the information cannot be produced locally), it may result in a new task being launched locally which is able to supply the desired information or it may result in no action if the task is unimportant.

The assessment module also contains general rules associated with the control of domain level activity. For instance, if an agent accepts a request to provide information for an acquaintance, then it first ought to check whether it is currently undertaking some activity which will produce it (achieved by looking at the intention descriptors). If so, then this activity should be utilised, rather than starting a new one.

```
(rule situation-assessment-30
   (IF  (INFO-REQUEST-MADE ?INFO ?ORIG ?PRIORITY)
        (ACTIVE-INTENTION-PRODUCING-INFO SELF ?INFO ?INTENTION))
   (THEN (ADD-ADDITIONAL-MOTIVATIONS SELF ?INTENTION
                          (SATISFY-INFO-REQUEST ?INFO ?ORIG))))
```

Another important situation assessment activity is to track the execution of tasks in the domain level system. Situation assessment rule 20 illustrates one facet of this tracking. If the control module has reported that a task has finished, but the results were not what was expected, then an alternative means of achieving them should be found. This process may involve trying another local activity if one exists or requesting aid from an acquaintance if not. This rule makes use of the motivation slot for carrying out a goal and the expected outcome slot of the recipe to determine what should have been produced.

```
(rule situation-assessment-20
   (IF (GOAL-FINISHED ?GOAL ?RESULTS ?ORIG)
      (IS-MOTIVATION ?ORIG ?GOAL (PLAN-ACTION ?PLAN-NAME
                                                   ?MARKER ?ID))
      (UNEXPECTED-RESULTS-PRODUCED ?GOAL ?RESULTS))
   (THEN (DETERMINE-WHETHER-ALTERNATIVE-SOURCES ?GOAL ?RESULTS
                                       ?PLAN-NAME ?MARKER ?ID)))
```

2.6 Using GRATE in Electricity Transportation Management

This section describes how GRATE has been applied to a cooperative scenario in the domain of electricity transportation management. The interaction involves the CSI, AAA and BAI and illustrates both task and result sharing. A more extensive list of scenarios can be found in (Aarnts *et al.*, 1991).

If the electricity network is in a stable situation, alarm messages which correspond to changes of state, arrive at the CSI in relatively small numbers. Their arrival triggers a CSI recipe which analyses them to determine whether they correspond to planned maintenance operations or whether they represent a disturbance - the “detect disturbance” recipe is described in Appendix A. In most cases the alarms correspond to planned operations and after a fixed time delay of ten seconds they are grouped together and the plan finishes. The outcome of this plan (i.e. the block of messages and the fact that there is no disturbance) is then passed to the CSI’s cooperation module to see if the results are of use to any other agents in the community. The CSI’s model of the AAA and BAI contains the following information:

```
MODEL OF AAA/BAI: INTERESTS:
      {...<BLOCK-ALARM-MESSAGES, T>,
        <DISTURBANCE-DETECTION-MESSAGE, HAS-VALUE(DISTURBANCE)>....}
```

Upon receipt of this information, the cooperation module invokes rule 5 (see section 2.5.1) for result sharing activity. The block of alarm messages is sent to both acquaintances as unrequested information (since “T” trivially evaluates to true), but the disturbance detection information is not passed on since it does not have value “disturbance”. If, however, the alarms correspond to a disturbance, then the detection message will also be sent as unrequested information since its condition of interest will be true. The following description assumes that a disturbance has been detected.

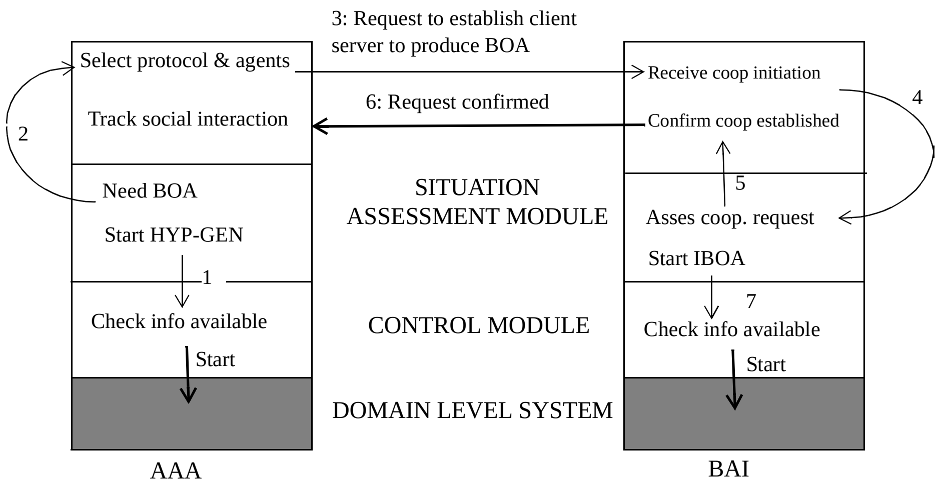

Figure 2.4: AAA Initiating Social Activity

Upon receipt of the notification of a disturbance, the situation assessment module of the AAA determines that the recipe "DIAGNOSE-FAULT" should be started - this process is illustrated in figure 2.4. Further details of this and other recipes are given in appendix A. The plan is retrieved from the self model and the situation assessment module tells the control module to start the first task - hypothesis generation (HYP-GEN) (action 1). Hypothesis generation is a fast but inaccurate task and will typically produce a large number of hypotheses for the network element at fault.

The situation assessment module also notes (from the second action in the recipe) that information concerned with the black out area (BOA) is useful. The reason for this being that the BOA identifies a list of network elements in which the fault is situated. Therefore to be consistent, the individual network element pinpointed by the AAA should also be in the BOA. This is a useful social interaction because any faults which the AAA initially proposes which are not within the BOA can be disregarded - thus, the search space can be significantly reduced. Based on its self model, the AAA is able to deduce that the BOA cannot be produced locally and so using situation assessment rule 14 (see 2.5.1) a request for aid with this goal is sent to the cooperation module (action 2). The cooperation module then examines its acquaintance models and sees that the BAI is the only agent capable of generating this information. It selects the BAI as the agent with which to interact, the client-server as the appropriate protocol and sends a cooperation initiation request (action 3).

The BAI receives the cooperation initiation request. Its cooperation module confirms that it is capable of producing the BOA, through the identify black out area (IBOA) recipe. The request is then passed to the situation assessment module to determine whether and how the request should be honoured (action 4). The situation assessment module confirms the request will be met (as the agent is currently idle) and informs the cooperation module of this fact (action 5). The cooperation module then confirms with the AAA that its cooperative initiation has been accepted (action 6). In parallel with the confirmation process, the BAI's situation assessment module tells the control module to execute the IBOA recipe (action 7).

When the IBOA task finishes (see figure 2.5), the control module is informed and it gathers the information produced (action 1) and passes it onto the situation assessment module (action 2). The assessment module which is monitoring the execution realises that the plan has been successfully completed (i.e. rule 20 (see 2.5.2) is not applicable). It then passes the results onto the cooperation module (action 3). The cooperation module which is tracking the social interaction progress, notes the request has been satisfied and sends an activity complete message back to the originating agent (using rule 19 - see 2.5.1) along with the desired piece of information (action 4). Upon receipt of the BOA, the AAA terminates the social action and passes the desired result onto its situation assessment module (action 5).

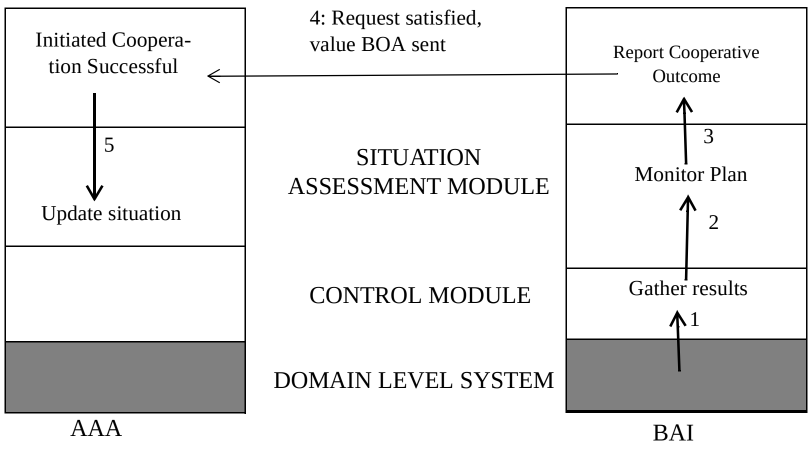
4: Request satisfied, value BOA sent
Initiated Coopera-tion Successful
Report Cooperative Outcome
5
3
SITUATION ASSESSMENT MODULE
Monitor Plan
2
Update situation
CONTROL MODULE
Gather results
1
DOMAIN LEVEL SYSTEM
AAA
BAI

Figure 2.5: Terminating Social Interaction

When the AAA's situation assessment module receives the BOA it must determine what action to take. The third action (i.e. WHILE-WHEN) of the "DIAGNOSE-FAULT" recipe is depicted in figure 2.6 and shows that the next step depends upon the current problem solving context. If it has not yet started validation then the refinement task should be executed before validation is undertaken. If it has started validation, then this should be suspended, hypothesis refinement executed and then validation restarted. This evaluation is performed by the situation assessment module, making use of the current action component of the recipe, and then the appropriate commands (eg SUSPEND, START) are sent to the control module. When the list of validated fault locations have been produced, they are sent to the operator and the SRA so that fault repair can commence.

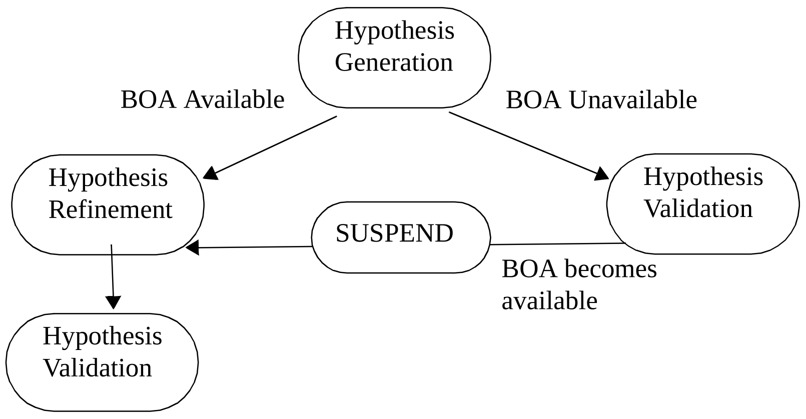

Figure 2.6: AAA’s Diagnose Fault Recipe

The rationale for the social interaction is that validation is a very thorough and time consuming activity, verifying a single fault may take a significant amount of time (Gallastegui *et al.*, 1989). In the worst case, validation may have to work on the large number of approximate hypotheses produced during generation. The refinement task is very fast and involves scanning the list of hypothesis which have yet to be validated and removing any which are not in the black out area. This gives a substantial speed up because fewer hypothesis have to be examined during validation.

2.7 Experiences with GRATE

The stated aim of this aspect of the research was to ease the process of constructing multi-agent systems. The approach adopted was to investigate the feasibility of designing a cooperation framework which contains a significant amount of inbuilt knowledge. This approach has been validated to a certain degree by the fact that such a framework has been constructed and successfully applied to a real world problem in the field of electricity distribution management.

Further evidence to support this claim was obtained when GRATE was applied to the cooperative diagnosis of a particle beam at the CERN laboratories under the auspices of the ARCHON project (Jennings *et al.*, sub). In this experiment two relatively sophisticated expert systems, written in KEE, were controlled by GRATE and were able to cooperate on a number of diagnostic tasks. This application was built in under a week and no extensions to GRATE’s generic knowledge were needed. The developer merely filled in the agent models and implemented the interface with the domain level system. A paper study of the applicability of GRATE for fault detection in telecommunication networks has also been carried out (Whitney, 1992). Again no extensions to GRATE’s generic knowledge were required and the application builder merely had to instantiate the self and acquaintance models.

As well as being a contribution in its own right, the experience gained during this work raised several important issues.

1) How can a community of cooperating problem solvers continue to act in a coherent manner when actions and interactions are fallible?

2) What is the underlying model of social activity upon which the generic cooperation and situation assessment rules are based?

3) How far can the approach of constructing shells with inbuilt knowledge be extended?

4) If domain dependent reasoning is required, how can it be incorporated into the system in a clean manner?

Issues one and two are the subject of the remainder of this thesis. Solutions to issues three and four are sketched in this section, but their implementation is left as further work.

The problem of coordinating activities when actions and intentions are fallible arose because of the significant impact which unanticipated or unpredictable events had on the coherence of the GRATE community. For instance in the electricity transport domain, if the CSI discovered, as a result of receiving new information, that there was not really a fault in the network after indicating that there was, the AAA and BAI continued with their diagnosis. Other events which caused incoherence in this application included: the AAA realising that it is not being supplied with sufficient alarm messages to make a diagnosis, substantial changes occurring in the network which invalidated crucial assumptions during diagnosis and distractions from unforeseen events or agents in the community.

The ease with which the community became uncoordinated was due to the fact that social interactions were only implicit. Agents do not organise themselves into explicit groups to work on particular problems. Rather each one is acting by itself until it receives information from an acquaintance or needs to request a service.

One solution is to *a priori* identify where problems may occur and explicitly encode them in the agent models. For example to specify that the AAA and BAI are interested in the information that the CSI's state has changed from believing there is a fault to believing there is not a fault. However such an approach is *ad hoc* and the designer would need to predict all such conditions in advance. A more principled and elegant solution is to provide agents with a deeper understanding of the collaborative problem solving process and allow them to infer that something has gone awry. Thus rather than working at the domain level, this work concentrated on defining an application independent description (model) of what it means to participate in collaborative activity. In particular the work focused on how joint action may become unstuck. For instance, rather than the CSI believing that its state change is interesting for the other agents; this approach leads to a situation in which it realises that the motivation for the social action of locating faults is no longer present (since there is not really a fault). The general description can then be manipulated by inbuilt knowledge which provides the mechanisms for detecting problematic situations. The model must also provide a clear definition of how the agent should behave in such circumstances. With this approach, the application designer has to provide a mapping from the general descriptions to the particular application (eg the motivation for diagnosing faults is that there is a disturbance in the network) which is a much better structured task and therefore less prone to errors and omissions.

Hence the problem of attempting to ensure coordinated behaviour in dynamic and unpredictable environments led to the need for a deeper understanding and model of collaborative problem solving. The joint responsibility model proposed by this thesis is outlined in chapter four and its implementation in an updated version of GRATE is discussed in chapter five.

The logical extension of GRATE is to present an application builder with a framework in which all the necessary control knowledge is inbuilt. While this appears attractive, it is unlikely that it would be feasible for complex applications because of the need to incorporate domain specific reasoning. For example in GRATE there is already one domain dependent rule which is shown below. The “react to delay” action is specific to a particular application context, it may have no affect, cause the agent to reschedule one of its actions or even lead to the abandonment of an entire recipe.

```
(rule situation-assessment4-25
     (IF (DEPENDENT-ACT-DELAYED ?ACT ?COMPONENT ?DELAY ?MOTIVE))
     (THEN (REACT-TO-DELAY ?ACT ?COMPONENT ?DELAY ?MOTIVE)))
```

The problem then is to find a sufficiently flexible approach which allows the developer to make maximum use of the available generic knowledge, but which facilitates easy integration of domain specific information. GRATE’s generic knowledge and the knowledge required to control activity in a particular application are opposite ends of a spectrum which could be brought to bear in controlling the activity of cooperating agents. The former being applicable to all cooperative scenarios and the latter to one specific problem. In between these two, however, are several others layers which represent varying levels of generality (see figure 2.7).

To minimise development costs and effort, an ideal system would provide built-in knowledge for all but the individual problem level (which must obviously be provided by the application developer). For such an approach to succeed, it must firstly be possible to identify and characterise general areas of problem solving. The feasibility of constructing generic task models has been demonstrated by Chandrasekaran (1983a) and libraries of such tasks have been constructed in the KADS project as a means of simplifying the domain modelling process (Hickman *et al.*, 1989). Similar approaches in conventional AI have been hailed as a mechanism for making software development easier, by supplying programs which solve classes of problems (McDermott, 1990), form the basis of the knowledge sharing vision of building conventional expert systems (Neches *et al.*, 1991) and have been likened to the industrial revolution for software development (Cox, 1990).

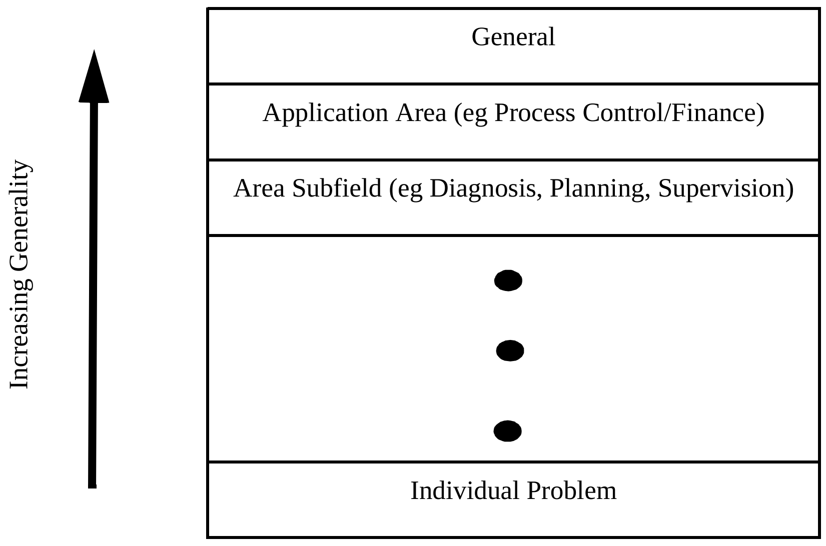
General
Application Area (eg Process Control/Finance)
Area Subfield (eg Diagnosis, Planning, Supervision)
Individual Problem
Increasing Generality

Figure 2.7: Spectrum of Cooperation Knowledge

As such models appear feasible for conventional AI, there is no reason to doubt the possibility of constructing similar descriptions of social interactions. The "application area" knowledge for process control would define typical cooperative scenarios for process control systems and the "area subfield" for diagnosis would provide general models of cooperation between systems working on diagnosis, and so on. As an illustration of such a model, consider the problem of diagnosis. The following are examples of generic, high-level interactions which might take place; agents may:

- divide the problem into non-overlapping parts and each work separately.
- both perform the same diagnosis using different data or problem perspectives.
- cross-check diagnoses of the same problem.
- focus each others problem solving by exchanging highly rated hypotheses.

If such knowledge could be acquired, the developer's task becomes that of configuring appropriate knowledge from the available corpus, providing the information specific to the individual problem and of fine tuning the system's control strategies as illustrated in figure 2.8.

This approach has significant advantages over conventional means of constructing multi-agent systems. It permits problem solving components to be re-used - thus increasing reliability, decreasing risks and development time and making effective use of specialists (Horowitz and Munsen, 1984). It also provides the important ability to fall back on increasingly general knowledge (Lenat and Feigenbaum, 1991). In addition, this methodology follows the lead of other disciplines which engineer complex artifacts (eg planes, cars), in that product development would consist predominantly of assembling components (Stefik, 1986).

.

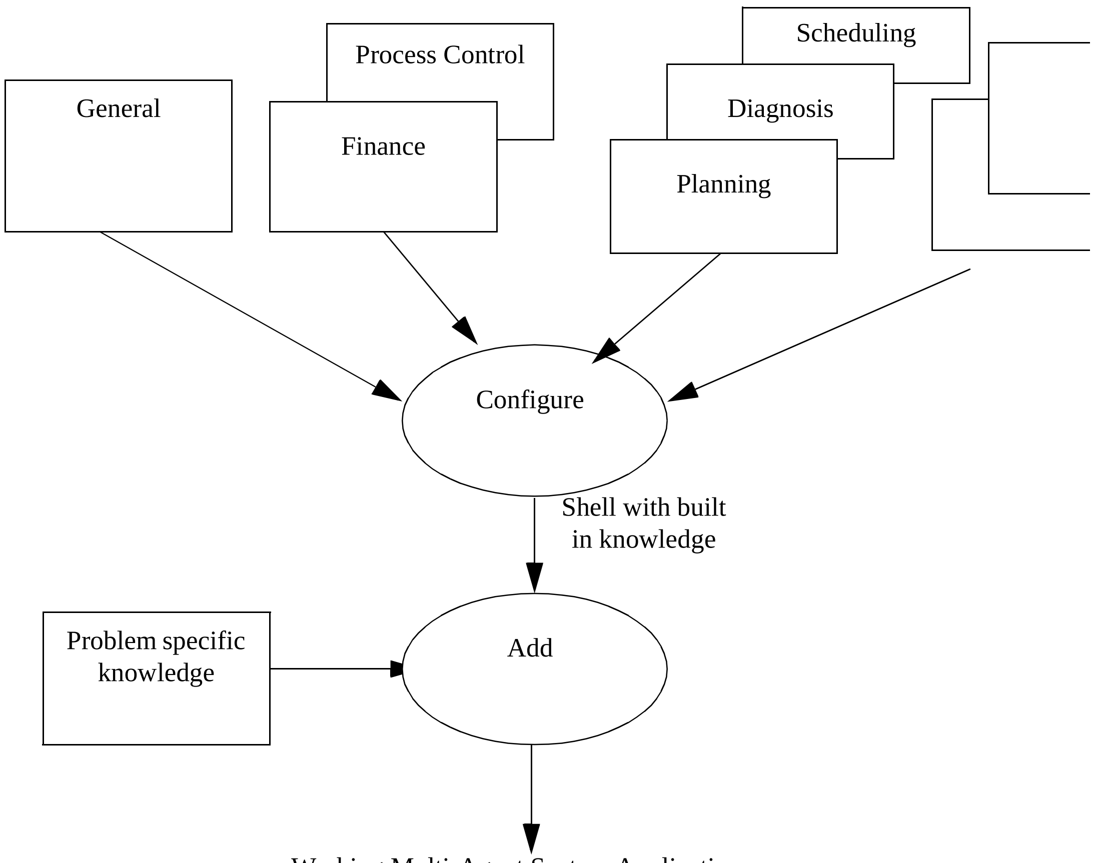
Scheduling
Process Control
General
Diagnosis
Finance
Planning
Configure
Shell with built in knowledge
Problem specific knowledge
Add
Working Multi-Agent System Application

Figure 2.8: Building Multi-Agent Systems Using Levels of General Knowledge (Jennings, 1992c)

3. INTENTIONS AND JOINT INTENTIONS

This chapter builds upon the predominantly intuitive and discursive model of intentions presented in the introduction. The role of intentions in guiding an agent's problem solving is expanded upon and a comparison with reactive systems in which agents just respond to environmental stimuli is undertaken. The conclusion of this evaluation is that reactive and intentional systems are merely opposite ends of a spectrum rather than fundamentally different approaches.

Some extensions to standard first order logic are presented so that a review of formal accounts of intentions can be undertaken. The material presented is considered as being representative of the field and includes the work of Cohen and Levesque, Pollack and Werner. All the reviews presented in this chapter are structured into two parts; a description of the model followed by an evaluation in which important points and criticisms are highlighted.

The view presented in this thesis is that notions of individual intentions are insufficient for describing complex social interactions. The rationale behind this observation is presented before some existing work on joint intentions is described and evaluated. Finally the perceived shortcomings of this existing work are gathered together and presented as a motivation for the development of the model of joint responsibility.

3.1 The Role of Intentions in Problem Solving

There has been much debate on the exact role of intentions in problem solving and indeed whether it is needed at all. This section highlights these arguments and sketches the role of intentions as used in this research. It also investigates the relationship with the apparently contradictory approach of reactive systems in which agents neither maintain nor reason about a representation of their environment, let alone have an explicit notion of intentions.

Two main ways of describing an agent's practical reasoning have been proposed; one approach has no distinct role for future directed intentions whereas the other uses prior intentions as an important input (Bratman, 1984).

Approach 1:	Practical reasoning consists of weighing desire-belief ratings for and against conflicting actions.
Approach 2:	Practical reasoning involves reasoning from a prior intention and relevant beliefs to derivative intentions concerning means, preliminary steps or more specific courses of action.

Bratman (1990) proposes a model of practical reasoning which encompasses both views and emphasises the key role of both prior and future directed intentions. Firstly, prior intentions pose problems which require means-end analysis in order to satisfy them. For example, if I intend to go to Paris on Saturday (the end), I should be motivated to find a way of getting there (the means). In this case, means-end analysis would involve finding out about modes of transport, departure times, ensuring I have a valid passport and so on. Secondly, intentions provide a degree of consistency within which the agent must operate; they should be both internally consistent and consistent with the agent’s beliefs. Consistency with beliefs means 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. Internal consistency means that an agent’s intentions should not conflict with one another. This requirement distinguishes intentions from desires because the latter can be inconsistent - for instance I may desire both a cream cake and to lose weight. This consistency requirement means that intentions provide a “filter of admissability” (Bratman, 1984) for adopting subsequent intentions. An agent should not commit itself to something which will conflict with or endanger its existing intentions.

Thus in Bratman’s unifying proposal, prior intentions provide a framework within which desire-belief rating can occur. This framework, in turn, poses problems for subsequent reasoning and constrains the solutions which can be adopted. This view is expressed in functional terms in his proposal for a belief desire and intention (BDI) architecture (Bratman *et al.*, 1988) which is discussed further in chapter five.

Bratman’s model of practical reasoning requires intentions to have a degree of stability. If this were not the case, and 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 if agents are to respond intelligently to unanticipated events. These desiderata mean that general policies and habits for governing the reconsideration of intentions are needed and also that agents need to monitor the achievement of their intentions. Tracking is necessary because agents care whether their attempts succeed; failure using one approach means they will replan and try another. So, for instance, if I originally intended to fly to Paris and I subsequently discover that all the flights are booked, then I will endeavour to find an alternative mode of transport.

In addition to these functional roles, if an agent intends to achieve an objective p, then the following properties should hold: (Bratman, 1984; Cohen and Levesque, 1990a)

• it believes that p is possible.

• it does not believe that it will not bring about p.

• under certain conditions, it believes it will bring about p.

• it need not intend all the expected side effects of their intentions.

The last point is what Bratman (1984) calls a “package deal.” An agent may intentionally bring about X, even if it does not intend X, so long as it intends something it expects will result in X. Consider an agent selecting one desire from amongst its set of potential desires. By doing so, it chooses to achieve a certain state of affairs, namely those associated with the desire. However the chosen act may have some undesirable side effects which occur as a result of performing the action. As the agent is aware of them, it has brought them about intentionally; however Bratman argues that it did not intend them. The reason for this is that side effects do not play the same role in the planning process as true intentions. They will not be tracked and if the agent does not achieve them, it will not go back and try again.

3.1.1 Reactive Systems

In the real world things do not always go as planned. The assumptions of traditional planning systems, such as STRIPS (Fikes *et al.*, 1971), are that the environment is totally predictable, the internal world model is totally complete and correct and that primitive actions are instantaneous and never fail. These assumptions are often inappropriate - many environments are dynamic, on-going, real time and unpredictable (Brooks, 1991). To cope with this mismatch, some researchers started investigating the idea of reactive systems (Agre and Chapman, 1987; Brooks, 1991; Suchman, 1987). In such systems, agents merely react 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.

This reactive approach appears to be at odds with the observation that intentions play a central role in guiding an agent’s problem solving. As a consequence there has been a vigourous debate between proponents of “reflective” and reactive systems - see Demazeau and Muller (1990/91) for several such papers.

However intentions may provide a link between these seemingly opposing camps. It is hypothesised that reactive agents merely exhibit a special type of intention. Each individual has a number of fixed, simple intentions which are specified by the system designer and are implicitly available to the agent. With this view, when the designer defines an agent’s behaviour he is in fact installing its intentions, the filter of admissability is set through the stimulus conditions. These fixed intentions (or pre-compiled behaviours) are then invoked automatically whenever certain conditions prevail, there is no runtime means-end reasoning. This contrasts with deliberative (non pre-compiled) intentions which are subject to means-end analysis and must pass through a filter of admissability before being invoked. Taking this unifying perspective, 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.

Adherence to this view means there should be problems which can be solved by deliberative 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 such as the towers of Hanoi or the n-puzzle and also encounter problems in environments which contain dead ends (Feber, 1992). Also the observation that coordination among interacting plans must be done by the programmers of reactive systems when they are designing appropriate actions (Lansky, 1989) is further support for this view. This leads to the hypothesis that the problems encountered with reactive systems are caused by the predefined, inflexible nature of their intentions. If richer models of intention are used, then these problems can be overcome.

Despite this intuition, and because of its emphasis on being a theory used by the agents rather than about them, joint responsibility is aimed at reflective agents which have explicit representations of beliefs, desires and intentions. However, in the future, it would be interesting to see whether the responsibility ideas could be embedded into the behaviours of reactive systems as a way of making them more general.

3.2 Logical Preliminaries

Most of the formal accounts of (joint) intentions presented in the subsequent review section are based upon possible worlds semantics. Therefore a brief introduction to this and the related area of dynamic logic is provided.

3.2.1 Possible Worlds and Belief

The “classical” model for reasoning about knowledge and belief is the so-called *possible worlds* model. The basic idea is that besides the actual state of affairs there are a number of other possible states (or worlds). Possible worlds therefore represent not only actually occurring worlds but any logically possible situation (i.e. one in which there are no contradictions).

Possibility is not the same as conceivability, it is a more general notion. For example the statement “London buses are black” is true in some imaginable world, meaning it is possible. This possibility is not ruled out just because it is not actually the case. However a proposition such as “the sky is blue and the sky is not blue” cannot be true in any possible world since it is contradictory (Reeves and Clarke, 1990).

Using this model well formed formulae are no longer true or false absolutely, rather they are true or false with respect to a particular world. For example, it is possible to state that the expression white(snow) is true in world w_0 but false in some other imaginable world (say w_{20}) where snow is black. Another key idea is the notion of accessibility, indicating which worlds are accessible from which others.

To describe beliefs it is necessary to extend the first-order predicates of AND (^), OR (v), NOT (~) and conditional ($\supset$) with a modal operator BEL[1]. BEL takes two

[1]. Belief is related to knowledge, however their exact relationship is still the subject of intense philosophical debate. Here the simplistic view that knowledge is true belief is adopted.

arguments, the first is a term denoting the individual who has the belief and the second a formula representing the statement to be believed. For example the fact that I believe you are enjoying reading this thesis can be expressed in the following manner:

BEL(I, ENJOY-READING(You, Thesis))

In possible world semantics, this statement has the value true with respect to a particular world w_i if and only if ENJOY-READING(You, Thesis) is true in all worlds accessible from w_i[2]. For an agent not to believe the truth of a proposition there must be some possible worlds which are accessible from the current one in which the proposition is true and some in which it is false.

These ideas can best be illustrated by means of an example. A labelled directed graph is used to represent the possible worlds model, nodes represent the possible worlds and arcs the accessibility relations. Figure 3.1 represents a scenario with three possible worlds (s_1, s_2 and s_3), two agents (A and B) and one proposition p which is interpreted as “it is sunny in San Francisco” (Halpern, 1986).

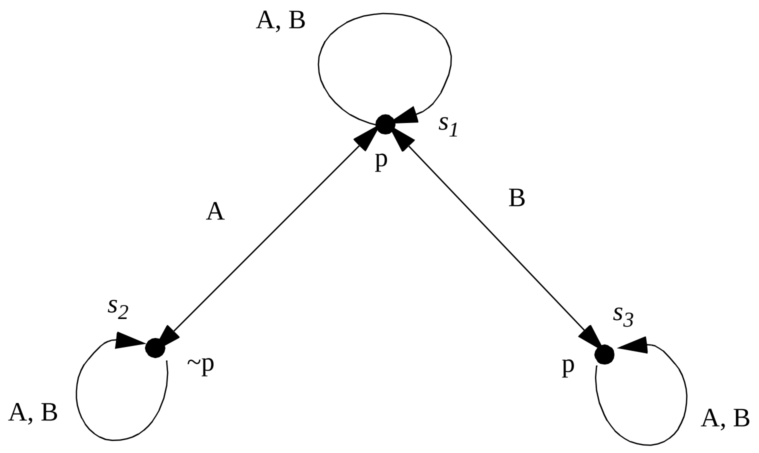

Figure 3.1: Possible Worlds and Accessibility Relations

In state s_1 it is sunny in San Francisco but A does not believe it (since it believes both s_1 and s_2 are possible). A believes that B knows whether or not it is sunny in San Francisco (since in s_1 B believes p is true and in s_2 B can access only ~p). B believes it is sunny in San Francisco but does not know that A does not know this fact because in both s_1 and s_3 A believes p.

3.2.2 Mutual Belief

Mutual belief and common knowledge have also been extensively investigated by

[2]. Like many possible world formulations, this assumes that agents are logically omniscient; they know all the consequences of their knowledge (Genesereth and Nilsson, 1988). This is a somewhat unrealistic assumption for finite agents, but this difficulty is ignored here.

logicians and philosophers. Intuitively, 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. For example for A and B to have mutual belief about a proposition p; A and B must believe p, A must believe that B believes, B must believe that A believes that B believes and so on. Formally, a definition for mutual belief between agents A and B about proposition p can be stated as follows (Barwise, 1988):

$$MB(A, B, p) \equiv BEL(A, p \wedge MB(B, A, p))$$

It has been shown that mutual belief is not attainable in systems where communication is not guaranteed or where there is some uncertainty in message delivery time (Halpern and Moses, 1984). Thus in any practical distributed system, common knowledge is unattainable. This fact can be illustrated by the coordinated attack problem (Halpern, 1986).

> Two divisions of an army are camped on two hilltops overlooking a common valley. In the valley awaits the enemy. If both divisions attack the enemy simultaneously they will win, whereas if just one division attacks it will be defeated. The divisions do not initially have plans for launching an attack and the commanding general of the first division wishes to coordinate one. Neither general will attack unless he is sure the other will join him. The generals can only communicate by means of a messenger. Normally it takes the messenger one hour to get from one encampment to the other. However, it is possible that he will get lost in the dark or, worse yet, be captured by the enemy. Fortunately on this night everything goes smoothly.
>
> How long will it take them to coordinate an attack?

Suppose the messenger sent by general A makes it to general B with a message saying "attack at dawn". Will there be an attack? No, since general A does not know that B got the message he will not attack. So general B sends the messenger back with an acknowledgment. Suppose the messenger makes it. Will there be an attack now? No, because now general B does not know whether A got the message, so he thinks general A may think that he did not get the original message, and thus not attack. So A sends the messenger back with an acknowledgment. But of course this is not enough either. No amount of acknowledgments sent back and forth will ever guarantee agreement.

3.2.3 Temporal and Dynamic Logics (Reeves and Clarke, 1990)

One important use of possible worlds logic is to relate the truth values of a proposition to time. In this case, the accessibility relation can be viewed as connecting states (possible worlds) that are temporally earlier with those which are temporally later.

Let P be a set of possible worlds and R the accessibility relation. To emphasize the fact that R represents an “earlier-later than” relation it will be written using “<“. The fact that a proposition p is true at a particular time t will be represented as t ||= p. There are usually four temporal modalities, extending both into the future and back into the past, however only the future ones are used in this review. The two future modalities which need to be expressed are:

◊p - there is a future time at which p will be true (*eventually p*)

i.e. t ||= ◊p if and only if there is a t’ such that t < t’ and t’ ||= p

❑p - at all future times p will be true (*always p*)

i.e. t ||= ❑p if and only if for all t’, t’ ||= p

The possible worlds model can also be used to express relations between actions within a computation (dynamic logic). This is achieved if the possible worlds are interpreted as states of a computation and the accessibility relation links one state of the computation to another[3]. It is possible to define several primitives which can be used to express the relations between actions in a program (see figure 3.2).

PRIMITIVE	**MEANING**
a; b	do a then do b
b^*	do b a finite number of times
a || b	do a and b in parallel
a | b	non-deterministic choice between a and b
a?	evaluate the proposition a. If it is true then proceed with the evaluation, otherwise fail

Figure 3.2: Dynamic Logic Primitives

Using these basic forms it is possible to construct more expressive statements. The following are used in various formulations both in this and the following chapter.

3. In this case, accessibility is dependent on the program being executed, thus the accessibility relation needs to be indexed by the program. If s and t are states of a computation and we get from s to t by executing program α then $(s, t) \in R(\alpha)$.

(UNTIL a b)	$(\sim a?;\ b)^*;\ a?$
(WHILE a do b)	$(a?;\ b)^*;\ \sim a?$
(WHILE a DO b WHEN c)	$(a?;\ b)^*;\ \sim a?\ \wedge\ c$

3.3 Existing Work on Individual Intentions

Three prominent models of intention are presented in this section, those of Cohen and Levesque, Pollack and Werner. All the authors share the supposition that intentions are a key component for defining agent behaviour but they differ in their emphasis. The models are formal in nature, although in this review they are presented in a predominantly discursive fashion to increase comprehensibility and because of the space limitations.

3.3.1 Cohen and Levesque's Model

Perhaps the most comprehensive attempt to formalise individual intentions is due to Cohen and Levesque (1990a/b). Their formulations are made at two levels of detail; a fundamental one which provides the primitives for a theory of action (eg definitions of beliefs, goals and action) and a second layer which builds upon these concepts to develop a theory of rational action. This review concentrates primarily on the latter, meaning several lower level considerations are given only descriptive semantics.

At the lower level, their account is formulated in a modal language which has the usual connectives of a first order language plus operators for propositional attitudes and for talking about sequences of events.

• BEL(x p) means that x has p as a belief.

• GOAL(x p) means that x has p as a goal.

Goals are also formalised as accessibility relations over possible worlds, where the accessible worlds are those which have been selected as most desirable. Goals are propositions which are true in all these worlds.

• MB(x y p) means that x and y mutually believe that p holds.

• AGT($x_1,...,x_n$ e) means that $x_1,...,x_n$ are the only agents involved in the sequence of events e.

Possible worlds are temporally extended into the past and future; each such world consists of an infinite sequence of primitive events. Each event is of a certain type and can have an agent associated with it.

Action expressions are built up using these primitive components and constructs from dynamic logic. For instance a; b, a | b and a || b are all valid action expressions. (HAPPENED a), (HAPPENING a) and (HAPPENS a) respectively represent the fact that a sequence of events describable by action sequence a has just happened, is happening now or will happen next. Additional syntactic abbreviations are also used:

$$\text{(DONE } x_1,\ldots\ldots,x_n \text{ a)} \equiv \text{(HAPPENED a)} \wedge \text{(AGT } x_1,\ldots\ldots,x_n \text{ a)}$$
$$\text{(DOING } x_1,\ldots\ldots,x_n \text{ a)} \equiv \text{(HAPPENING a)} \wedge \text{(AGT } x_1,\ldots\ldots,x_n \text{ a)}$$
$$\text{(DOES } x_1,\ldots\ldots,x_n \text{ a)} \equiv \text{(HAPPENS a)} \wedge \text{(AGT } x_1,\ldots\ldots,x_n \text{ a)}$$

Turning to higher level considerations, it is now possible to define what it means for agent x to be committed to achieving goal p relative to motivation q. The first definition is that of a persistent goal.

$$\begin{array}{l} \text{(PGOAL x p q)} \equiv \\ \quad \text{1: BEL(x } \sim p) \wedge \\ \quad \text{2: GOAL(x } \Diamond p) \wedge \\ \quad \text{3: (UNTIL} \quad [\text{BEL(x p) v BEL(x } \sim\Box p) \text{ v BEL(x } \sim q)] \\ \quad \qquad\qquad\quad \text{GOAL(x } \Diamond p)) \end{array}$$

This means agent x believes that p is currently false (1), that x wants p to be eventually true (2) and that this state of affairs will continue until x comes to believe either that p is true or that it will never be true or that q is false (3).

This definition captures the pre-theoretic notion of intention stability discussed earlier and also incorporates the idea of irrevocability. Once adopted, an agent cannot drop commitments freely; rather it must keep them at least until certain conditions arise. Persistent goals also capture the notion that agents will continually try and achieve objectives, even if their initial attempts are unsuccessful. An intention can then be defined as a commitment to act in a certain mental state:

$$\begin{array}{l} \text{(INTEND x a q)} \equiv \\ \quad \text{(PGOAL} \quad \text{x} \\ \qquad\qquad \text{(DONE x [UNTIL (DONE x a) BEL(x (DOING x a))]?; a)} \\ \qquad\qquad \text{q)} \end{array}$$

meaning agent x *intends* to do action a if it has the persistent goal to have done that action and, moreover, to have done it believing throughout that it was doing it. This captures the notion that there is a commitment to do it deliberately.

3.3.1.1 Evaluation

This model accords well with many of Bratman's intuitions; indeed Cohen and Levesque explicitly address this link in several of their papers. Their formulation embodies the critical notion of commitment, but also enumerates conditions under which commitment to an objective can be dropped.

There are, however, some shortcomings. Most importantly there is no mention of plan states - meaning there is no link between the objectives and the way in which they are achieved. This is unfortunate because when an agent adopts a goal it may be contingent on being able to achieve it a particular manner. Thus not only do agents need to be committed to the objective, they also need to be committed to the plan states which achieve it.

Another gap in their model is the means by which agents adopt goals and intentions. Like many others, they assume that the only condition which must be fulfilled is consistency with pre-existing goals. However as Galliers (1988) points out, notions of preference and interests also need to be taken into consideration before goals can be adopted. This is especially important for reasoning in situations in which agents may have conflicting beliefs or goals, rather than assuming agents are merely benevolent towards one another.

3.3.2 Pollack's Model

Pollack (1990) starts by distinguishing two common, but potentially confusing, uses of the term "plan". Firstly, there are *recipes* which are sequences of steps known by an agent[4]. For example I may know a recipe for going to America which involves going to a travel agent, booking a flight and so on. Secondly there are plans which an agent adopts and which guide its actions. The latter is akin to "having a plan" and is close to the idea and functional role of intentions. Therefore although this work is phrased in terms of having a plan, in this context it can be thought of as being synonymous with intention.

This distinction is drawn to highlight the fact that most AI research in plan recognition has concentrated on the former notion, aiming to automate the process of recipe production. However Pollack believes that a comprehensive model must also define the structure of the mental attitude of having a plan. This dual consideration is especially important when viewing plans as a force for guiding actions, as she does within the fields of dialogues.

Pollack's model has two components, one related to an agent's beliefs and another to its intentions[5]. For agent A to have a plan to do β that consists of doing actions Π, A must have the following beliefs about the actions which comprise Π.

1: It must believe that executing the actions in Π, in their specified temporal order, will entail performance of β. Whether they actually do is irrelevant, it is only important that the agent believes they will.

> This ensures that if A believes that α is part of what it will do to achieve its goal, then α is in the plan.

[4] The term recipe is used in the description of GRATE's agent models in this context (see 2.3.5).
[5] In this work, intentions are given no finer structure nor formal semantics. It is merely defined in an intuitive manner without reference to notions such as commitment, stability or consistency.

2: It must believe that each α in Π plays a role in the plan. A believes that by doing α *it will do* β or some other action γ that plays a role in its plan, or that doing α will *enable doing* β or some other γ that plays a role in the plan.

> The former corresponds to generator relationships and the latter to enablement. If action α *generates* action β then the agent only need do α and β will automatically be done. However when α *enables* β, the agent needs to do something more than α to guarantee that β will be done. For example, knowing the phone number of a pizza place enables a pizza to be ordered if there is access to a phone, but it does not generate the order. Generator relationships are used in Grosz and Sidner's (1990) work on shared plans and are expanded upon in section 3.5.3.

Pollack claims that these beliefs are necessary but not sufficient conditions to guarantee that doing Π is A's plan to do β. For sufficiency, it is also necessary that A has a certain set of "intentions" with respect to Π. In particular A must intend:

3: To execute each act α in Π in the specified temporal order.

4: To execute Π as a way of doing β. This circumvents the problem of an agent accidently (unintentionally) carrying out an action which succeeds in bringing about the desired outcome.

5: That each act α in Π plays a role in the plan. A must intend by doing α to *do* β or some other γ that plays a role in its plan or by doing α to *enable* doing β or some other γ that plays a role.

There is clearly a close link between conditions 2 and 5 of this formulation. The division between recipes and intentions ensures that having the beliefs described in condition 2 does not imply having the intention described in 5. However it is unclear as to whether having the intention in condition 5 necessarily means holding the beliefs in condition 2. There is contrasting literature to support both views:

> • Plans normally support expectations of their successful execution (Bratman, 1984). Therefore having an intention to do α involves having a belief that it will be done.
>
> • I may intend to make ten legible copies of what I am writing by pressing hard on carbon paper, without believing with any confidence that I may succeed (Davidson, 1980).

Adopting the former view means that 5 directly entails 2, and also that 4 entails 1; adopting the latter requires both aspects to be explicitly present. Pollack adopts the latter case because it is useful when an inferring agent deems an actor's plan invalid to determine whether this is through belief in the plan action per se or whether it is due to incompatible beliefs.

3.3.2.1 Evaluation

This work represents an important contribution to the field of plan inference in that it stresses the central role of mental attitudes alongside plan structure. This concept is very much in keeping with the ideas expressed in this thesis relating to the key role of the mental state of intention. It also emphasises that when considering collaborative activities, it is essential to include notions related to the plan states of the participants (cf Cohen and Levesque's's model).

However there are several flaws within this work. Most importantly, the notion of intention is stated only informally. As already indicated, there is still significant debate about exactly what intentions are; therefore to use them as such an integral part of the formalism, without giving precise semantics can lead to ambiguities.

This work also neglects the fact that a typical plan step involves not only an action but also an expectation of what that action will achieve. Thus executing a plan involves more than doing the plan act (as described here), it also involves ensuring the desired objective is attained. This concept is absent. Also the model does not adequately explain how an agent should behave if things go wrong. For example, if an agent no longer believes it is capable of executing an action, what should it do? Should it give up?, should it replan? This formalism, simply states that the agent no longer has Π as a plan.

3.3.3 Werner's Model

Werner (1988/89) aims to construct a general theoretical framework for designing agents with a communicative and social competence. Agents are represented by triples: R = <I, S, V>. I is the agent's information state (the information it has at a particular time), S is the intentional state (set of strategies that guide its actions) and V the evaluative state (desires which enable alternatives to be distinguished). More formally, an information state is a history that is possible given the available information and an intention is a class of strategies which can be used to map between information states and their alternatives.

He argues that building up mutual intentional states through communication is expensive, therefore it is necessary to have abstraction mechanisms to represent this process. He champions the use of social roles - a similar proposal has also been made by Gasser *et al.* (1989). A social role (rol) is a description of an abstract agent $R_{rol} = <I_{rol}, S_{rol}, V_{rol}>$ that defines state information, permissions, responsibilities and values. When an agent assumes a role, it is internalised by constraining its representational state R_A to $R_A + R_{rol}$. Without its information and evaluative components, a role is an intentional substate of the agent's overall state. For example in master-slave interaction, the role of the master contains an expectation that it will tell the slave what to do and the role of the slave carries the expectation that the slave follows the instructions of the master.

3.3.3.1 Evaluation

Werner sees intentions as a central component in defining a unified theory of communication and social structure. His work has a different emphasis from that described previously and contains several interesting ideas. Firstly, there is an attempt to integrate communication primitives into the formalism, thus providing an account of how messages affect an agent’s intentions. So, for example, sending an assertion has the affect of modifying the recipient’s information state, a command (if accepted) affects the recipient’s information and intentional states and so on for other linguistic primitives. This formalism also explicitly includes evaluative information (cf Cohen and Levesque), thus it provides an account of single agent rational decision making which includes both intentions and desires.

This work is arguably the most comprehensive attempt to try and formalise several aspects of multi-agent systems. With such a broad coverage however, some of the constituent components are lacking in detail. The definition of intentions is particularly weak and does not embody the key notion of commitment. He does not explain what it means for an agent to employ a particular strategy in computational terms. For example when describing the contract net, he states that there are two roles: $\text{rol}_{\text{manager}}$ and $\text{rol}_{\text{contractor}}$ but then proceeds with only a textual description of what the roles are and what it means to adopt a particular role. This aspect needs much greater structuring and detail if this work is to be used as an implementation specification; in particular the constituent components (eg beliefs and goals) and their inter-relationships need to be given. It also needs to be clarified when particular strategies are appropriate and how to behave if a strategy fails.

3.4 Limitations of Individual Intention Approaches

Intentions define individual behaviour and, as such, are insufficiently rich for describing collaborative actions. Social acts need representations which address higher order units of analysis (Gasser and Huhns, 1989). There are two main limitations with the purely individualistic approach. Firstly, joint action is more than just the sum of individual actions, even if the actions happen to be coordinated. To represent the “something else” part, formalisms and structures specifically related to collaborative activity are needed. Some of these notions are investigated in the next section.

Secondly there is a fundamental difference between individuals and groups. This can be illustrated by considering the notion of commitment. A group’s commitment to an objective cannot simply be a version of individual commitment where a team is taken to be the agent because teams may diverge in their beliefs[6] (Cohen and Levesque, 1991). If an individual comes to believe that a particular goal is impossible, then it is rational for it to give it up. The agent can drop the goal because it knows enough to do so. Similarly when an individual finds that the team’s overall goal is impossible, the team must stop trying to fulfill its objective. However in this case the whole team may not know enough to do so. For example one of the agents in a

[6]. This can only happen to schizophrenic individuals!

collaborative lift may observe that the object to be moved is nailed to the floor and therefore the group’s objective cannot be attained. However it cannot be assumed that all the other team members have been also able to make this observation and the corresponding deduction. Hence although there is no longer mutual belief that the goal is achievable, since one agent has seen the nails, there is not yet mutual belief that the goal is unachievable and therefore some parts of the team remain committed.

One means of circumventing this problem is to allow goals to be dropped when there is no longer mutual belief. However this would cause agreements to be dissolved as soon as there was uncertainty about the state of other team members. For instance if one of the agents participating in the collaborative lift saw another stop momentarily, for whatever reason, then it may be unsure of its commitment. There would no longer be mutual belief and the agreement would be dissolved. Such weak criteria for dropping commitment would lead to a tremendous replanning overhead and would quickly ensure that joint action became unprofitable.

3.5 Existing Work on Joint Intentions

Joint action requires an objective the group wishes to achieve and a recognition that they wish to achieve it in a collaborative manner[7]. So in a cooperative lift, all team members must want to lift the table and they must want to do it as a team. The second component of this definition is important because it distinguishes between identical and parallel goals (Conte *et al.*, 1990). For instance if both x and y have the goal to have spaghetti cooked, their goals are identical; but if they both have the goal to eat spaghetti (i.e. x has the goal that x eats spaghetti and y has the goal that y eats spaghetti), they merely have parallel goals. This distinction is important because the two relationships imply different consequences in social interaction. Parallel goals give rise to competition if resources are scarce, whereas identical goals result in cooperation and coordination. This work focuses upon identical goals, which will hereafter be termed common, joint or collaborative.

Once a common goal has been agreed by all team members, individuals become *committed* to achieving it. Therefore the shared objective provides the glue to bind individual’s actions into a cohesive whole. Merely having a common aim is not sufficient for obtaining the objective in a collective manner. Agents also need to agree upon a common solution and base their subsequent actions, related to the attainment of the shared objective, on it. Thus 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 solution which the agents are working under[8]. This solution may be refined or even abandoned in favour

[7]. Singh (1990a) presents a dissenting view, stating a group can intend to achieve something even though only certain of its members really intend it. For example, a house building team may intend to put a roof on after the walls. The wall building crew might just be interested in completing their bit and not care about the remainder of the job. However having a group objective does not mean completing it alone or being involved at all stages. Also if the wall building crew do not care about the rest of the job, or their role in it, then it is difficult to claim that the house is being built as the result of a joint action.

of a new one at a later stage, but nevertheless in normal situations there must be such a frame of reference for the collective action to proceed

Developing or refining the common solution is a complex activity in which participating agents need to augment their individual intentions to comply with those of others. So if one agent wants to lift the table at time 20 and another at time 30, then one or both of them will need to modify their intentions if the joint action is to become viable. Such adaptation is necessary because actions are intertwined, not performed in a vacuum. There are several paradigms which may be used to generate the common plan; 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.

Having a common objective and solution context means participants' actions can be phrased in terms of "doing their bit" (Searle, 1990). In a collaborative lift, an agent lifts at one end as a means of contributing to the overall objective of moving the object. This action is only undertaken because the agent believes that the others are also doing their bit and sticking to the agreed solution. If an individual comes to believe that others are no longer contributing, then it would be irrational for it to continue because it will not achieve the overall objective by working alone.

The research described in this thesis is underpinned by the notion that team problem solving is characterised by the mental state of the participants. A joint activity is one performed by individuals sharing certain specific mental properties. Previous work in this area has addressed various aspects of this mental state and will now be reviewed before the model of joint responsibility is presented.

3.5.1 Group Minds

One school of thought amongst philosophers and social scientists talks about group minds and the collective unconscious. As figure 3.3 illustrates, the group mind is not part of any individual member yet it impinges upon their actions.

Conte (1989) gives several reasons why such notions deserve further attention but homes in on the notion of norms as the most convincing. Norms typically provide solutions in which the resulting net benefit to social actors is greater than that which would be obtained in their absence. They are often customary solutions worked out by interacting social actors, their origins are inscrutable since they do not derive from any deliberate human act. They point to the phenomena of "extramental" regulation, they impinge upon social actions and yet nobody devised them! So, for example, in an institution different agents often do tasks in a particular way, not through codes of practice which have been laid down but because "it is the way we do things around here". Therefore it is the collective mind of the organisation where these procedures are

---

8. 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.

laid down, and individuals are only making use of it.

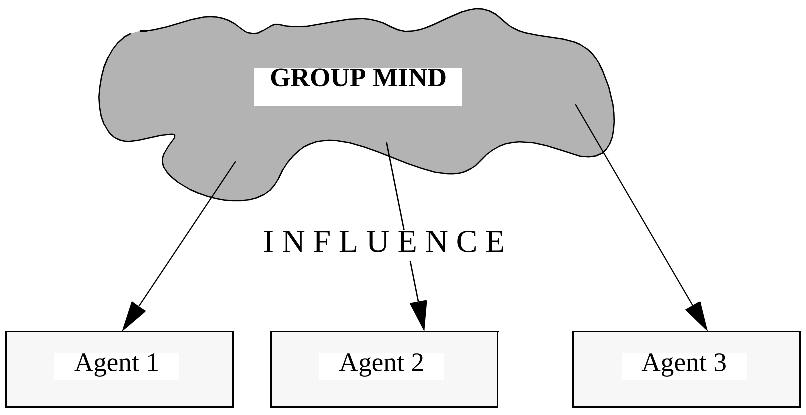

Figure 3.3: Group Minds

3.5.1.1 Evaluation

This approach recognises that there is something more than just individual action in many forms of collaborative activity. The group mind acts as this something extra and provides some knowledge which can be used for coordinating activity, through the use of norms for instance.

Despite providing a simple description of collaborative activity, several important points need to be answered before this approach can proceed. Plausible answers are needed to the questions of how there can be group behaviour which is not just the behaviour of team members? and how there can be group mental states which are not contained within the "brains" of individuals in the team? Because these crucial questions remain unanswered, most studies of collaboration have not adopted this approach. Subsequent models reviewed in this section all acknowledge that something over and above individual action is needed to describe cooperative problem solving. However they all agree that whatever this component is, it is in the mind of the individual actors.

3.5.2 Collective Intentionality

Searle (1990) postulates that collective intentions expressed in the form "we intend to do X" are primitive and cannot be analysed in terms of individual intentions expressed in the form "I intend to do Y". This is the case even if individual intentions are supplemented with (mutual) beliefs about the intentions of other members of the group. To illustrate the point, he proposes the following scenario:

> a group of businessmen are all educated at the same business school where they learn the theory of the hidden hand. Each comes to believe that he can best help humanity by pursuing his own self interest and

> forms a separate intention to this effect. Each has the individual intention "I intend to do my part toward helping humanity by pursuing my own self interest and not cooperating with anybody." Suppose the members also have mutual belief that each of the others intends to help humanity by pursuing their self interest and that these intentions will probably be carried out with success.

This is *not* a case of collective intentions; rather each MBA graduate is merely following his own individual intention. In Conte's (1990) terms they have parallel goals not identical ones. The scenario's objective is to provide a test case for definitions of joint intention. Any model whose conditions are satisfied by this scenario is insufficient. All the subsequent models presented in this review are tested against this scenario as part of their evaluation.

Searle believes the only way of proposing a definition of joint action which will not succumb to his MBA example is to incorporate the notion of an agent "doing its bit" toward the collective aim. Any model which does not embody this principle will be subject to the MBA counter example. However, he argues, including "doing its bit" leads to notions of circularity; collective action being defined in terms of collective actions.

This leads him to the conclusion that collective intentions cannot be reduced to individual ones. The reason being that collective intentions imply the notion of cooperation. But the presence of individual intentions to achieve a goal, which is believed to be the same as that of others, does not entail the presence of an intention to cooperate to achieve that goal. The solution is to simply recognise that there are intentions whose form is "we intend that we perform act A." This means collective acts require constructs specifically related to social aspects of operation. Such an intention can exist in the mind of each individual who is acting as part of the collective, implying that all the intentionality needed for collective behaviour can be possessed by individual agents even though the intentionality in question makes reference to the collective.

3.5.2.1 Evaluation

This work is consistent with the observation that notions of individual intention are insufficient for collaborative scenarios and also highlights the need for developing models which have explicit representations of collectives to express "we intend".

Hobbs (1990) recasts Searle's conclusions in a more traditional AI view; stating an agent can have as goals in its plan, logical formulae whose predicates describe actions that collectives engage in and whose agent argument is such a collective. Agent A can form a plan "A and B push car" with the goal that they push the car together. This decomposes into the sub-goals "A pushes car" and "B pushes car" with the associated temporal relations. In addition, for A to believe that it is engaged in collective action, it has to believe that both it and B mutually believe the same plan. Mutual belief is what gives A the belief that the events B is responsible for will happen at the appropriate time (and vice versa). Whilst this approach is attractive, and is in fact consistent with the model of joint responsibility, it is important that the model of a collective agent permits and accounts for the divergence of beliefs amongst its constituent components.

3.5.3 Shared Plans

Grosz and Sidner (1990) have adapted Pollack's (see section 3.3.2) work on recipes and their associated mental states to collaborative situations, and in particular to dialogue. They propose a special operator, called a SharedPlan, which exists between a group of agents $\alpha_1,..,\alpha_n$ and is with respect to a particular objective σ. For instance SharedPlan(α_1, α_2, Achieve(Clean-Room)) represents a situation in which α_1 and α_2 wish to clean a room together.

A SharedPlan requires that for all sub-components $\sigma_1,..,\sigma_m$ of σ, the following are mutually believed for each σ_i:

• one team member α_i is capable of executing σ_i - EXEC(α_i, σ_i)

• α_i intends to achieve σ_i - INT(α_i, σ_i)

• α_i intends to achieve σ "BY" σ_i - INT(α_i, BY(σ_i, σ))

There also needs to be mutual belief about "*generator relationships*" (Pollack, 1990). Grosz and Sidner consider four such relationships; those involving simultaneous actions by two agents, conjoined (together but not necessarily simultaneously) actions by two agents, sequences of actions by two agents and actions by one agent only. An illustration of a generator relationship for agents α_1 and α_2 performing *simultaneous actions* is as follows:

$$\begin{aligned} \mathrm{MB}(\alpha_1, \alpha_2, [\ & \mathrm{OCCURS}(\delta_i, \alpha_1, \mathrm{T1}) \Leftrightarrow \mathrm{GEN}(\beta_j, \gamma, \alpha_2, \mathrm{T1})\ \& \\ & \mathrm{OCCURS}(\beta_j, \alpha_2, \mathrm{T1}) \Leftrightarrow \mathrm{GEN}(\delta_i, \gamma, \alpha_1, \mathrm{T1})], \mathrm{T0}) \qquad (1) \end{aligned}$$

OCCURS(δ, α, t) is true if and only if an action δ is performed by α during time interval t. The generator relation (GEN) can best be illustrated by means of an example. Sometimes the act of typing "DEL" will generate an act of deleting the current mail message, when I am in an electronic mail system for example. However not every case of typing DEL will achieve this result, for instance I may be in an editor. GEN(β_j, γ, α_2, T1) means that agent α_2 doing β_j at time T1 will generate γ. Clause 1 can be more succinctly expressed in the following manner:

$$\mathrm{MB}(\alpha_1, \alpha_2, \mathrm{GENSimultaneous}[\delta_i\&\beta_j, \gamma, \alpha_1\&\alpha_2, \mathrm{T1}], \mathrm{T0})$$

There are analogous definitions for GENConjoined and GENSequence, however this section focuses exclusively on simultaneous actions. Consider a case in which α_1 and α_2 wish to collaboratively lift a heavy object such as a piano. The time intervals used in the previous definitions have been omitted for reasons of clarity.

SharedPlan(α_1,α_2,lift[piano])

MB(α_1, α_2, EXEC(lift(foot-end), α_1)) ^

MB(α_1, α_2, EXEC(lift(keyboard-end), α_2)) ^

MB(α_1, α_2, GENSimultaneous[lift(foot-end) & lift (keyboard-end), lift(piano), α_1&α_2]) ^

MB(α_1, α_2, INT(α_2, lift(keyboard-end))) ^

MB(α_1, α_2, INT(α_1, lift(foot-end))) ^

MB(α_1, α_2, INT(α_2, BY(lift(keyboard-end), lift(piano)))) ^

MB(α_1, α_2, INT(α_1, BY(lift(foot-end), lift(piano)))) ^

INT(α_2, lift(keyboard-end)) ^ INT(α_1, lift(foot-end)) ^

INT(α_2, BY(lift(keyboard-end), lift(piano))) ^

INT(α_1, BY(lift(foot-end), lift(piano)))

This represents the situation in which α_1 and α_2 have a SharedPlan to lift the piano. It means that the following conditions are mutually believed: α_1 will lift the foot end, α_2 will lift the keyboard end, these acts will be carried out simultaneously and will cause the piano to be lifted, both agents intend to lift their respective ends and that by doing so the overall objective will be attained. In addition to these mutual beliefs, it must actually be the case that both agents intend to lift their respective ends and that by doing so the collaborative lift will be attained.

3.5.3.1 Evaluation

This work emphasises the fact that group-level constructs, SharedPlans in this case, are needed to describe collaborative activity. It stresses the need for joint problem solving to be placed within the context of a common solution and that plan states should be an integral component of any comprehensive model. Generator relationships also highlight the fact that actions within the common solution are interdependent and that their relationships must be upheld if the overall objective is to be attained.

However there are several problems associated with this model. Firstly “BY” is given no formal semantics. This can lead to ambiguous and counter intuitive observations. For example the above definition contains the following clause:

INT(α_2, BY (lift (keyboard-end), lift(piano)))

meaning the piano will be moved by α_2 lifting at the keyboard-end. This is clearly nonsense because the piano will only be lifted if α_1 simultaneously lifts at the foot-end. This problem arises because the authors have tried to map single agent planning constructs onto multiple agent domains. This approach misses one important point; in single agent domains there is no need to specify the agent carrying out the action since there is only one. Whereas in multiple agent domains, the matching of tasks and agents is a crucial consideration.

Secondly, the semantics of $\alpha_1 \& \alpha_2$ in the GENSimultaneous part of the clause are unclear. If it is taken to mean conjunction, what is the conjunction of two agents? (Hobbs, 1990). This approach requires a myriad of GEN relations to be developed for each different action ordering and context which may occur. A SharedPlan in which actions are performed in sequence requires GENSequence to replace GENSimultaneous and if there is a more complex interplay (eg one action is to start after the other has performed two steps) then new generator relationships need to be developed (eg GENStartAfterTwoSteps). It is better to have a description which, though acknowledging the existence of related actions, represents this dependency in a generic way and does not need to be redefined for each particular case.

Thirdly, SharedPlans also fall prey to Searle's MBA counter example despite using "BY" to represent the notion of an agent doing its bit. The MBA's have a SharedPlan even though there is no collaboration. Suppose α_1 and α_2 are two of the MBAs who intend to help humanity by serving their own interest. Satisfying the first two clauses, they mutually believe that α_1 is going to serve his own self interest and that α_2 is going to do the same. They also mutually believe that simultaneously serving their own interests will generate helping humanity and that they each intend to serve their own self interests. They mutually believe that α_2 intends to help humanity by serving his own self interest, and similarly for α_1 and that they each intend to serve their self interest. Finally, α_2 intends that by serving his own interest he will contribute to the helping of humanity, similarly for α_1. Thus, they satisfy all the components of the SharedPlan, but they are not engaged in collective behaviour.

What is missing in this account is what Searle (1990) has tried to capture in his notion of collective intentions and what Hobbs (1990) argues is simply a matter of having as a goal an action whose agent is a collective. There is no notion that *we* do something. Moreover this is a possibility that Grosz and Sidner explicitly reject in their belief that agents can only intend their own actions.

This work has recently been extended and refined. Firstly the troublesome generator relationships have been replaced by mutually known recipes and secondly "BY" is replaced by a "CONTRIBUTES" relation (Lochbaum *et al.*, 1990). Recipes, as described in section 3.3.2, refer to a way of doing something; they encode constraints on constituent acts and following any ordering stipulated by the relations will result in performance of the act. The SharedPlan's generator relationships are replaced by:

$$\text{MB}(\alpha_1, \alpha_2, \text{R:recipe-for-lift-piano})$$

meaning that when agents α_1 and α_2 have a SharedPlan for doing some act, they must have mutual belief about the way to perform it. This use of recipes and their interrelationships removes the need to produce different generator relationships for each ordering and time constraint.

BY relations are replaced by CONTRIBUTES. So, for example, in the collaborative piano lift:

$$MB(\alpha_1, \alpha_2, INT(lift(foot\text{-}end), \alpha_1) \wedge CONTRIBUTES(lift(foot\text{-}end), \alpha_1, lift(piano)))$$

means that α_1 intends to lift(foot-end) as a way of contributing to the overall objective of lift(piano). There are two ways in which an action can contribute to an overall objective, either by performing a primitive action (as is the case above) or by being part of a larger series of actions (Lochbaum *et al.*, 1990).

These modifications remove some of the criticisms of this work whilst retaining the positive aspects of stating the need for a commonly agreed solution and that actions are undertaken as a means of contributing towards an overall objective. However they fail to recognise that with joint action it is usually not possible to state that an individual act contributes *per se*, rather it contributes only when combined with other actions, respecting certain relationships. There is also no “solution-action binding” between the recipe and the actions performed - the formulation merely states there must be mutual belief about a means of achieving the desired state. At no point is it specified that one recipe is selected - thus α_1 and α_2 may have mutual belief in R_1 and R_2 as recipes for achieving the overall objective and α_1 may be working under R_1 and α_2 under R_2.

3.5.4 We Intentions

Tuomela and Miller (1988) propose the notion of we-intentions (eg “we shall do X”) as a means of describing collaborative situations. The motivation for their work is that in order to study social action, it is necessary to have a clear idea about the internalisation of the notion of “group” in its members. They identify the concept of “we-ness” as being central; believing that the sociality inherent in acting together comes from the relevant we-intentions. Thus if an agent intentionally performs a helpful act, but it does not share the common objective then it cannot be considered as part of the joint action. There cannot be joint action if any participant lacks the relevant group-intention expressing the overall common goal. They propose the following structure for we-intentions. Agent A is a member of group G “we intends” to do X if:

- A intends to do its part of X

- A believes that the joint action opportunities for X are true, especially that at least a sufficient number of the full-fledged and adequately informed members of G, as required for the performance of X, will do their part.

  This expresses the conditional nature of the individual intentions. Unless A believes that the others will do their share, A will not do its part.

- A believes there is mutual belief amongst group members to the effect that the preconditions of success mentioned above hold true.

3.5.4.1 Evaluation

The second component of the definition highlights the shortcoming of Lochbaum *et al.*'s (1990) "CONTRIBUTION" relation. It emphasises the fact that actions only contribute to the overall objective when they are considered in conjunction with those of others. There is no absolute contribution to a group action. Like the SharedPlan work, this analysis attempts to reduce collective intentions to individual intentions plus beliefs; there is no attempt to explicitly represent collectives. Indeed Tuomela and Miller explicitly rule out the formulation of we-intention which is along the lines of "I do my part of X". Their reasons are that each agent is supposed to share the group's intention to do X, reflected in its acceptance of the intention-expressing statement "we shall do X" as being true of itself. Thus an agent does not merely accept "I shall do my part of X" as being true of itself but also the stronger statement "we shall do X". To illustrate this point they propose the following example. Consider the case of a musician in an orchestra who intends to perform his part properly, but nevertheless intends to do something which will make the visiting conductor look ridiculous and will spoil the orchestra's overall performance. Such a musician cannot be said to we-intend to play the symphony.

There are several shortcomings of this approach. Firstly there is no representation of collective action. Secondly it falls prey to Searle's MBA example; all the conditions necessary for a we-intention are satisfied and yet there is no collective behaviour. For any graduate A of the business school:

- A intends to pursue his own selfish interests without reference to anybody else and thus intends to his part towards helping humanity.

- A believes that the preconditions of success are true. In particular, that other classmates will also pursue their own selfish interests and thus help humanity.

- Since A knows that his classmates were educated in the same selfish ideology, he believes there is mutual belief among the group's members that each will pursue his own selfish interests and that this will benefit humanity.

Finally joint activity requires a common plan to be agreed by the participants, however this is not explicitly mentioned in the account of we-intentions. Its existence is merely tacitly assumed from the explanation of what it means to have a we-intention.

3.5.5 Social Plans

Rao *et al.* (1991) argue that in order to describe collaborative problem solving, notions such as individual intentions, beliefs and desires need to be augmented with structures describing joint goals and joint intentions. Like Hobbs (1990) they argue that a crucial component of a formulation of social interaction is the ability to describe and represent collections. So, for example, a joint action may be formulated as:

$$(\text{lift(piano)}, \{\text{lifter}_1, \text{lifter}_2\})$$

in which both the lifters are involved in the joint action of lifting the piano. This approach is also consistent with the joint responsibility model of collaborative behaviour which has the concept of social actions (see section 4.2.1).

Plan expressions are a pair consisting of a plan type (an abstract structure that, when executed by an agent, results in the occurrence of an action in the real world) and an agent. More complex expressions are obtained by using operators from dynamic logic. For example a plan to represent a triathlon race in which two agents (a_1 and a_2) have to cycle (e_1), swim (e_2) and run (e_3) in that sequence is:

$$((e_1\ a_1); (e_2\ a_1); (e_3\ a_1)) \parallel ((e_1\ a_2); (e_2\ a_2); (e_3\ a_2))$$

This represents the fact the agents have to synchronise at the start of the race, however the subsequent actions need not be synchronised, just carried out in the specified sequential ordering.

As well as describing the activities of individuals within this formalism, it is also possible to define social agents (teams). For example the notion of a team s_1 performing a relay race in which one agent runs after another can be expressed as follows:

$$(\text{ACHIEVE(run-relay)}, s_1)$$

The body of the plan can be represented by the sequence of actions: $(e_1\ a_1); (e_1\ a_2); (e_1\ a_3)$ in which e_1 is the act of running around the track and a_1, a_2 and a_3 are the agents in the team.

They then go on to define the notions of mutual belief, joint goal and joint intention for a social agent. Mutual belief ϕ of a social agent is defined to be all members of the social agent believing ϕ and all of them believing that it is mutually believed. A joint goal ϕ of a social agent is defined to be all members of the social agent having the goal ϕ and mutually believing that ϕ is held as a joint goal and an analogous definition is also given for joint intentions. Their formulation gives rise to a theorem which states that if a social agent jointly intends an action, the social agent also has it as a joint goal and mutually believes it.

3.5.5.1 Evaluation

This formulation does not succumb to Searle's MBA example because a social agent (all MBA graduates) which jointly intends the action must also have a joint goal to serve humanity. This joint goal could be obtained by some prior form of communication, like all the graduates meeting together and jointly adopting the goal of helping humanity. But as there is no such goal, there can be no joint intention.

The strong points of this work are the rigorous manner in which the planning language is specified and also the central role played by the notion of a social agent. It also provides a formalism and semantics which allows the success and failure of social plans to be taken into consideration.

However it also has several drawbacks. Firstly by opting for dynamic logic as plan operators the expressiveness of the planning language is limited; complex timing constraints such as overlapping and meeting may be difficult to express and non-temporal constraints cannot be defined. Despite recognising that actions may fail, the formulation does not describe conditions under which the joint action may go wrong, nor does it give a prescription of how to behave in such circumstances. This means there are no criteria for guiding the builder of a multi-agent system when constructing the component which monitors the ongoing collaborative activity - this is an important part of the responsibility model and is described in detail in section 5.2.2. Finally the formulation does not make reference to plan states, it is defined exclusively in terms of goal states and so does not express the requirement for a commonly agreed solution to frame the actions of the participants.

3.5.6 Group Intentions

The basic building blocks in this model of collaborative problem solving are the notion of “strategies” which provide an abstract specification of the behaviour of an agent or a group (Singh, 1990a/b)[9]. In addition to strategies which may be adopted, there are those which provide background constraints against which the agent must operate - examples include “survive” and “thrive”.

Strategies can also be used to characterise the behaviour of groups. It does not matter from a theoretical standpoint whether the strategies are hard-wired (as in a reactive architecture) or obtained through planning. A group strategy is treated as a set of strategies of its members. The structure of a group is defined by the interactions amongst its members at the levels of strategies and reactive actions.

- Strategic Interactions

  Some of the abstract interactions among agents involve illocutionary acts (Austin, 1962) such as commands and assertions whilst others involve the establishment of various conditions in the world by some members’ strategies that other members’ strategies rely on.

- Reactive Interactions

  Interactions among the members as they execute their strategies may be implicit in the way in which a particular group acts.

A group can be said to “perform” a strategy if all its members follow the current part of their individual strategies whilst interacting in the right way and, as they complete the current parts, go on to do the next part. This notion can be used to define

[9]. Strategies are similar in nature to “roles” as discussed by Werner (1989) - see section 3.3.3.

a group intention; a group intends all the necessary consequences of the strategy it is currently employing. In otherwords, a group intends to achieve p if it is following a strategy that necessarily means that p will come true.

These concepts are illustrated on the pursuit problem (Gasser *et al.*, 1989), in which four blue agents have to capture (surround) one red agent in a two-dimensional grid. The strategy of one of the blue agents (B1) could be described as “get above the red agent”, and other blue agents could have analogous definitions for ensuring that the red one is surrounded. From this, B1 can be said to have the intention to occupy the location above the red agent. The group of blue agents has a strategy that simply embodies the four substrategies; but when viewed from an external perspective the blue agents can be ascribed the objective of surrounding the red one. Such high-level specifications can be further refined to an arbitrary level of detail in order to provide more concise definitions of the various strategies. In this example, intentions were attributed to the blue agents without regard for whether the individual agents knew of them or planned to achieve them. In fact the group itself was described independently of whether its members knew they had participated in it.

The sense in which a group “has” a strategy determines the sense of group intention being defined. In its most basic sense, the members need not even be aware that they are part of a group. The theory can be extended to describe groups in which different states of awareness are required, including situations in which each member is aware of the strategic intentions of all the others, of the strategies and so on. Thus it is possible to describe groups which range in their awareness from nothing to full mutual belief about the intentions of others.

3.5.6.1 Evaluation

This approach differs from those described earlier in this section. Group intentions are described from an “external” perspective; taking the view of an observer and trying to characterise a system in terms of its behaviour as objectively observed. This contrasts with the “internal” theories which take the view of the agent being analysed or specified. Singh’s theory is meant to be used about groups, rather than by them.

The external viewpoint means that the resulting theory is a considerable distance from an implementation architecture. It is considerably more neutral on this issue than the others presented in this review which are committed to a plan-based view of intelligence embodying notions such as beliefs and goals. Because of its external perspective, the model can be applied both to reactive and reflective architectures.

The models of joint intention presented earlier were also unable to account for the internal structure of groups. By requiring mutual beliefs the groups must be homogeneous in terms of their view of the collaborative act. Singh’s formulation is equally applicable to diverse types of agents, some of which may require mutual belief, others of which have pre-coded stimulus-response pairings.

The use of strategies suffers from many of the drawbacks attributed to roles (as discussed in section 3.3.3). Although there is a deeper specification of strategies than there was of roles, the mapping to actions is still intuitive rather than formal. It is also unclear how individual strategies can be combined to give rise to group strategies, clearly there needs to be something more than simply the union of two sets. For instance the formulation leaves unexplained the means by which the four blue agents strategies combine to give the overall one of surrounding the red agent. Also the interrelationships between the four strategies is left open, this becomes an important failing in situations in which actions are intimately intertwined and complex temporal relationships exist between the various sub-components. Finally, the definition of intention conflicts with Bratman's (1984) idea that an agent need not intend all the expected side effects of its intentions.

Searle's MBA test is not applicable to this formulation because it is based on the hypothesis that mental states are the primary consideration. Singh rejects this fundamental observation and reserves centre stage for the behaviour of agents meaning that the MBA example can be considered a collaborative act.

3.6 Conclusions

This review has highlighted the fact that descriptions of individual intentions are inadequate for describing collaborative activity. Inherently social activities require formalisms which are designed specifically for that purpose. Even amongst the social action formulations there is still a significant range of descriptions and issues addressed; none of the formulations are anything like complete. Some important issues which have been identified include:

- the need to describe collectives as well as individuals.
- agents must agree on a common goal.
- agents must agree they wish to collaborate to achieve their shared objective.
- agents must agree a common means of reaching their objective.
- action inter-dependencies exist and must be catered for in general terms.

However an important area which has been largely neglected, is the provision of criteria against which joint activity can be evaluated and a prescription of how to behave when things go wrong. Assessment criteria and the concomitant causal link to behaviour are especially important when agents are situated in environments in which they possess neither complete nor correct beliefs about their world or other agents; have changeable goals and fallible actions and are subject to interruption from external events. In these complex, dynamic environments it is difficult to ensure that a group's behaviour remains coordinated, because initial assumptions and subsequent deductions may be incorrect or inappropriate. Therefore it is imperative to have a theory which provides a grounded basis from which robust problem solving communities can be constructed.

Evaluation criteria are especially vital when agents decide that they will work together because they believe a team approach is the best means of solving the problem. In these circumstances, the benefits of collaboration must outweigh the overhead associated with coordination activities. One way in which this can be achieved, and hence one of the major 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. This problem is tackled by the work on joint responsibility which in turn builds upon the notion of j*oint persistent goals* (Levesque *et al.*, 1990; Cohen and Levesque, 1991). Responsibility provides criteria for assessing joint problem solving activity and a prescription of how to behave when things go wrong which can be used directly by the designer of a multi-agent system (see section 5.2.2 for further details).

4. JOINT RESPONSIBILITY

The chapter describes the model of joint responsibility (Jennings, 1991b/92a) which is the main theoretical contribution of this research. Firstly the work on *joint persistent goals* (Cohen and Levesque, 1991; Levesque *et al.*, 1990) is described because it forms the foundation upon which the responsibility model is constructed. Joint persistent goals define the conditions under which groups of agents can drop commitment to a shared objective and prescribe how to behave in such circumstances. However the formulation does not explicitly address issues related to plan states nor the make up of the group. Joint responsibility tackles these shortcomings, producing a more comprehensive description of how to operate when engaged in collaborative problem solving.

There were three primary objectives of the joint responsibility work against which its success can be gauged:

- to provide a comprehensive formulation of multi-agent problem solving in complex, dynamic environments.

- to produce a theory which is computationally tractable.

- to have a clear mapping from the theory to its implementation.

These criteria mean that the joint responsibility theory must specify the roles of the individual agents from an internal perspective (i.e. their beliefs, plans and goals) and have a clear link to agent architecture, rather than adopting a behaviouristic approach and being neutral about implementation. Joint responsibility aims to be usable by the agents to guide their behaviour.

Finally, in this chapter, the applicability of the responsibility model to the domain of electricity transportation management is illustrated. The scenario used is the cooperative detection and diagnosis of faults by the AAA, BAI and CSI agents which was first introduced in chapter 2.

4.1 Joint Persistent Goals

One of Cohen and Levesque’s (1991) primary motivations for the work on joint persistent goals was to investigate the ways in which a team is similar to an aggregate agent and to what extent their previous work on individual intentions could be carried over to social situations. They propose a definition of joint intentions which is analogous to their individual intention formulation;

joint intention is a joint commitment to perform a collective act while in a certain shared mental state

The central notion of joint commitment is formalised through the definition of joint persistent goals which, in turn, are based upon the concept of *achievement goals*.

Achievement goals define the state of individuals participating in a team which is working on a collective goal (eg moving a table) with a specified motivation (eg because it is necessary to gain access to a cupboard). Agent α has a *weak achievement goal*, relative to its motivation q, to bring about p if either of the following are true:

- α does not yet believe that p is true and has p being eventually true as a goal (i.e. α has a *normal achievement goal* to bring about p).

- α believes that p is true, will never be true or is irrelevant (q is false), but has as a goal that the status of p be mutually believed by all team members.

Thus a weak achievement goal involves four separate cases: either α has p as a normal achievement goal (it wants the table to be moved); thinks that p is true and wants to make this fact mutually believed (it believes the table has already been lifted); believes that p will never be true (it believes the table is nailed to the floor) and wants to make this fact mutually believed or, finally, believes there is no longer a need to gain access to the cupboard (q is no longer true). Using the language introduced in the previous section, a weak achievement goal (WG) for agents α_1 and α_2 to achieve p relative to condition q can be formalised in the following manner.

$$
\begin{aligned}
&(\mathrm{WG}\ \alpha_1\ \alpha_2\ p) \equiv \\
&\quad [\sim\mathrm{BEL}(\alpha_1\ p) \wedge \mathrm{GOAL}(\alpha_1\ \Diamond p)]\ \mathrm{v} \\
&\quad [\mathrm{BEL}(\alpha_1\ p) \wedge \mathrm{GOAL}(\alpha_1\ \Diamond(\mathrm{MB}\ \alpha_1\ \alpha_2\ p))]\ \mathrm{v} \\
&\quad [\mathrm{BEL}(\alpha_1\ \Box\sim p) \wedge \mathrm{GOAL}(\alpha_1\ \Diamond \mathrm{MB}(\alpha_1\ \alpha_2\ \Box\sim p))]
\end{aligned}
$$

Weak achievement goals form the basis of the definition of *joint persistent goals*[1] (JPGs). A team of agents have a JPG, relative to q, to achieve p if and only if:

$$
\begin{aligned}
&(\mathrm{JPG}\ \alpha_1\ \alpha_2\ p\ q) \equiv \\
&\quad 1\text{: } \mathrm{MB}(\alpha_1\ \alpha_2\ \sim p) \wedge \\
&\quad 2\text{: } \mathrm{MB}(\alpha_1\ \alpha_2\ \mathrm{GOAL}(\alpha_1\ \Diamond p) \wedge \mathrm{GOAL}(\alpha_2\ \Diamond p)) \wedge \\
&\quad 3\text{: } (\mathrm{UNTIL}\ [\mathrm{MB}(\alpha_1\ \alpha_2\ p)\ \mathrm{v}\ \mathrm{MB}(\alpha_1\ \alpha_2\ \Box\sim p)\ \mathrm{v}\ \mathrm{MB}(\alpha_1\ \alpha_2\ \sim q)] \\
&\qquad\qquad \mathrm{MB}(\alpha_1\ \alpha_2\ (\mathrm{WG}\ \alpha_1\ \alpha_2\ p) \wedge (\mathrm{WG}\ \alpha_2\ \alpha_1\ p)))
\end{aligned}
$$

meaning that they mutually believe that p is currently false (eg the table has not been lifted) (1); they mutually believe that they all want p to be eventually true (eg they all want the table to be lifted) (2) and finally until they come to mutually believe either that p is true, that p will never be true or that q is false, they will continue to mutually believe that they each have p as a weak achievement goal relative to q (3).

Thus if a team is jointly committed to achieving p, they mutually believe that they

1. Joint persistent goals are the multi-agent analogy of individual persistent goals (PGOALs) which are defined in section 3.3.1.

each have p as a normal achievement goal initially. However as time passes, team members cannot rely on the fact that they all still have p as a normal achievement goal; they can only assume that they have it as a weak achievement goal. The reason for this is that one team member may have discovered privately that the goal is finished (true, impossible or irrelevant) and be in the process of making this fact known to its associates.

If at some point, it is no longer mutually believed that everybody still has the normal achievement goal, then there is no longer a JPG as not all the agents wish p to be true. In this case, the team is no longer committed to p. However there is still mutual belief that a weak achievement goal will continue to persist which ensures that all team members are informed of the lack of commitment by an individual within the group.

It can be shown that if a team has a JPG to achieve p, then each member also has p as an individual persistent goal (PGOAL). To show this, imagine that agent α is in a situation where it does not believe that p is true nor that p is impossible. As at least α still believes that p is feasible, there is not mutual belief among the whole team that p is untenable, so p must still be a weak goal. Under these circumstances, it must still be a normal goal for α. Therefore p persists as a goal until α believes it to be satisfied or impossible to achieve.

The definition of JPGs also means that during joint actions agents can count upon the commitment of others; firstly to the overall objective (normal achievement goal) and then, if necessary, to the mutual belief of the status of the goal (weak achievement goal). This means that if one of the team members comes to privately believe that the goal is no longer tenable, but that this is not mutually known by all participants, then this agent will be left with a commitment to make the goal's status mutually known. Thus JPGs allow team members to undertake activities in the knowledge that others are working towards the same overall objective and that if they are no longer doing so then they will eventually be informed of this fact. Although joint persistent goal was defined only in terms of a weak goal, a concept which does not in itself incorporate a commitment, a persistent goal follows as a result.

A joint intention (JI) between agents x and y to achieve action a relative to motivation q, is then defined as a joint commitment to the agents' having done a collective action, with the agents of the primitive events as team members and with the team acting in a joint mental state.

```
(JI x y a q) ≡
    (JPG x y
        (DONE x y [UNTIL (DONE x y a) (MB x y (DOING x y a))]?;a)
        q)[2]
```

[2]. DONE and DOING are defined in section 3.3.1.

An important part of this formulation is that it provides criteria with which team members can evaluate their ongoing problem solving activity. JPGs specify that agents must track a goal’s validity and also provide the conditions which must be monitored in order to perform this task. In addition to evaluation criteria the model also provides the causal link to behaviour; when an agent comes to privately believe that the joint act is untenable (or successfully completed) it must endeavour to ensure that all its fellow team members are made aware of this fact. These specifications ensure there is a clear path to an implementable architecture; enumerating some of the tracking which is necessary and defining how the agent should behave in both nominal and unanticipated circumstances.

As the conclusions of chapter three highlighted, most of the previous work has failed to address these issues and has merely concentrating on defining what it means for a joint intention to exist. For example Tuomela and Miller (1988) state that when things go wrong with one agent’s activities, then the other agents in the group will help exert pressure and do whatever they think is necessary for the collective to succeed in achieving its objective. This is a particularly weak statement; it does not specify what it means for something to go wrong nor how to behave in such circumstances. For these reasons, we-intentions do not easily translate into useful guidelines for implementing multi-agent systems.

Application of Searle’s test to JPGs reveals that the necessary conditions are not satisfied. In particular in the definition of weak achievement goals, when the individual MBAs believe that they have helped humanity (i.e. BEL(α p)) they do not adopt the goal of informing their fellow graduates that they have succeeded in their objective (i.e. GOAL(α $\Diamond$(MB α α_2 p)) is not true). Hence there is no weak achievement goal and hence there cannot be a joint persistent goal, nor a joint intention.

However, contrary to the claims of Cohen and Levesque, JPGs are not sufficient for attaining joint action; there are several areas in which their formulation is deficient. Firstly, joint action requires that agents work within the context of a commonly agreed solution; without this there can be no joint action. 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 they are capable of agreeing upon a common plan for achieving their objective, there will never be joint action despite the fact that there is a joint goal.

A comprehensive formulation of joint action must make reference to plan states as well as goal states and explicitly specify the need for a common solution. Joint action requires both a joint goal and adherence to a common solution whilst attaining it. This does not imply that a common solution must be developed before the joint action can commence nor that the solution 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. Including this component places an additional functional role on joint intentions; namely that they will drive the participants' actions and behaviour towards agreement of a common solution.

As plan states should be an integral component in a formulation of joint intentions, it is necessary to define an associated "code of conduct" which specifies how team members should behave with respect to them. So just as joint commitment to a goal was defined, analogous stipulations related to the common solution are required. The formulation must specify what it means for a group of agents to be jointly committed to a common solution, enumerate conditions under which it is rational to drop commitment and prescribe how individuals should behave in such circumstances.

Another important facet of being a team member is the ability to contribute something towards the group's objective. For example if two people are attempting to lift a table and another is sunning himself on a distant beach, then it is somewhat counterintuitive to say that the three agents are involved in the joint action of lifting the table. If the person on the beach had intentionally done something related to the joint action, for example removing nails which bound the table to the floor, then it could indeed be argued that the three agents were collaborating. However if the person on the beach was unaware of the lift and therefore could not have intentionally participated, then this fact should be taken into consideration when describing the joint action.

The rationale for incorporating such a clause is related to the pragmatic motivation behind the responsibility theory's development. A team's communication and coordination overhead is, in general, related to the number of members. The more agents greater the overhead. Therefore if there are "freeloading" agents within the team they are increasing the coordination overhead, without producing any tangible benefit. Other formulations of intention have neglected this facet, mainly because of its pragmatic nature and its link to implementation level concerns (Jennings, 1991b). Indeed some authors go so far as to state that a group can perform an action even if only a few of its members are actually responsible for that action (Singh, 1990a).

However work on public choice theory, a sub-discipline of political economy, identifies free-riding as one of the three most common forms of cooperation related with public goods[3] (de Greef, 1991). It postulates that the state usually has to provide such facilities, because it can cause friction if the payment is met privately and some of the potential benefactors refuse to contribute. Glance and Huberman (1992) also investigated free-riding in collective actions in which community members can either cooperate in the production of the overall objective or free-ride on the efforts of others. They show that in such environments the onset of cooperation can take place in a sudden manner. Likewise, defection (non-cooperation) can appear out of nowhere in previously cooperating groups. These outbreaks mark the end of long transient states

[3]. The most important feature of a public good is that consumption by one individual does not reduce the amount available for consumption by others. For example, once a street light is erected, the fact that one person is enjoying the light does not reduce the amount available for consumption by others.

in which defection or cooperation persists in groups that cannot sustain it indefinitely.

4.2 Joint Responsibility

Joint responsibility stresses the role of joint intentions as “conduct controllers” (Bratman, 1984); specifying how both individuals and collectives should behave whilst engaged in collaborative problem solving. In particular the model addresses the problem of ensuring that groups remain coordinated in the face of unanticipated events or environmental changes. This emphasis offers a clear path from theory to implementation - providing functional guidelines for architecture design, defining criteria against which ongoing problem solving activity can be evaluated and prescribing how to behave when collaborative problem solving becomes untenable. Responsibility subsumes and extends the work on joint persistent goals presented in the previous sub-section. It defines a finer structure for joint commitment which involves both plan and goal states.

The responsibility model addresses two facets of collaborative action - it defines preconditions which must be satisfied before joint problem solving can commence and prescribes how team members should behave once it has started. Both aspects are outlined discursively before the formal definitions are developed.

Before joint problem solving can commence, four conditions need to be satisfied:

• One (or more) agent(s) must recognise the need for collaborative action to solve a particular problem.

• This core group must contact other agents, who it is believed will be able to play an active problem solving role, to see whether they wish to be involved in bringing about the collective aim.

> Signing up new members may be achieved simply by making a request if the agents are benevolent or may involve persuasion if they are more autonomous. An important constraint when carrying out this task is to ensure that all members are able to contribute something to the group’s efforts.

• Once a group which shares a common purpose has been formed, members must agree that they wish to achieve the common objective in a collaborative manner.

> In particular, this involves agreeing they need a common solution which they will abide by. Without acknowledging this principle, there can be no intentional joint action, only unintentional (accidental) interaction. The actual solution need only be developed once all the prerequisites have been satisfied; it may be proposed by an individual or developed collaboratively, it may be developed in one go or in an incremental fashion using whatever mechanisms are most appropriate.

• Agents must agree for the duration of the joint action they will follow a code of conduct with respect to their collaborative activity.

Adherence to this code (the responsibility model in this case) ensures that the group operates in a coordinated and efficient manner even if things do not go according to plan.

The second aspect of the responsibility model is applicable once the above conditions have been met and the joint action has started (i.e. the code of conduct for collaborative problem solving). The model prescribes how individuals should behave in their local problem solving activity and how they should act towards other members of the team. An individual's behaviour can abstractly be described as following the agreed plan of action (honouring its commitments) whilst continuously monitoring its local activity and the information received from other agents to ensure everything is progressing smoothly (tracking the rationality of its commitments).

As agents are situated in dynamic and unpredictable environments, events may occur which affect their commitments. For example, newly acquired information may invalidate previous assumptions or important events may require urgent attention and distract the agent from its agreed course of action. In such circumstances it would be irrational for an agent to steadfastly follow the agreed course of action. Therefore conditions for reneging upon commitments (both to the overall objective and the common solution for obtaining it) need to be enumerated and the agent needs to continuously monitor its activity to detect when these conditions occur. If such circumstances do arise, agents need to reassess their position; this may range from rescheduling actions which were delayed, to replanning if the solution is invalid or unexecutable to dropping the joint objective altogether if the motivation is no longer present.

As with JPGs, when an individual becomes uncommitted, it cannot simply stop its own activity and disregard others. The reason for this being that just because one agent has been able to detect that the overall objective is no longer achievable by the present means it does not mean that the other group members have also been able to do so. Therefore the agent who realises there is a problem must endeavour to inform all its fellow team members of this fact and also the reason for its change of commitment. This ensures team members are kept informed of events which affect their joint work and so when things do go wrong, the amount of wasted resource can be minimised.

Responsibility is specified using the normal connectives of a standard first order language plus operators (BEL, MB, GOAL) for expressing propositional attitudes. Constructs from dynamic and temporal logics are used to describe actions, their interrelationships and how they can be combined into larger units of activity. Joint responsibility incorporates the notion of joint persistent goals described in the previous section; therefore an explicit formulation will not be repeated here. There is one slight adaptation in that responsibility makes use of a collective agent whilst Cohen and Levesque have no such concept. Therefore their separate agents and goals need to be replaced by descriptions of groups (social actions, see below) and correspondingly their definition of execute is replaced by the social action perform operation (again see below for further details).

4.2.1 Describing Actions and Plans

In common with most modern planning systems, the adopted representation formalism defines points in the search space as partially elaborated plans which are traversed using plan transformations[4] (Hendler *et al.*, 1990). Plans are represented as an action ordering in which the actions are strung together with temporal ordering relations. There are two types of action; those which can be undertaken by individuals (*primitive actions*) and those in which groups (at least two) agents work together (*social actions*[5]). So, for example, if two agents collaboratively lift a table then the action is “lift table” and it is social because it involves a team of agents. Social actions ultimately give rise to primitive actions because only individuals have the ability to act. Thus the social act “lift table” may give rise to the primitive actions of agent_1 lifting at end_1 and agent_2 simultaneously lifting at end_2.

Throughout this section, let the set of agents in the community be represented by A, the set of primitive actions which can be performed by some agent in the community by P and the set of social actions which community A can perform by S. Note $P \subseteq S$ since a primitive act can trivially be performed by 2 or more agents.

In joint problem solving the actions of individuals are intertwined, necessitating the synchronisation of actions. The relationship between actions may be temporal (eg BEFORE, DURING and OVERLAPS (Allen, 1984)) or express other types of constraint. Rather than enumerating all possible relationships, using generator relationships (Grosz and Sidner, 1990) or dynamic logic (Rao *et al.*, 1991), it was decided to have a general purpose operator $\Re$. This permits different constraints and partial orderings to be defined without *a priori* restricting the set of possible interactions.

Relationships can involve arbitrary numbers of actions and may be composed entirely of primitive actions, or of social actions or a mixture of the two. So if $\sigma_1, \sigma_2 \in S$; $a_1, a_2 \in P$ and $\Re_{a,\, b}(a, b)$ the relationship between actions a and b; then the following relationship types may exist: $\Re_{\sigma1,\sigma2}(\sigma_1, \sigma_2)$, $\Re_{p1,p2}(p_1,p_2)$, $\Re_{\sigma1,p1}(\sigma_1, p_1)$, $\Re_{p1,\sigma1}(p_1, \sigma_1)$ and $\Re_{p1,\sigma1,\sigma2}(p_1, \sigma_1, \sigma_2)$. Actions are independent if $\sim\Re_{a,b}(a, b)$. All actions are subject to at least one relationship, the tautology $\Re_{a,a}(a, a)$. Relationships between actions represent coordination points and are, therefore, as important as the actions themselves. For example, when moving an object in which all parties are required to lift at the same time, failing to satisfy the relationship

4. Examples of plan transformations include expansion of an action to a greater level of detail or the inclusion of an additional ordering constraint to resolve some interaction between effects of unordered actions.

5. The desirability of social actions has also been noted independently by Hobbs (1990) and Rao *et al.* (see 3.5.5) and is consistent with social psychology work in which the society is considered prior to the individual (Mead, 1934).

SIMULTANEOUSLY means the move will not occur.

Both primitive and social actions can be combined into finite sequences to specify more complex interactions. Sequences are composed of at least one action and may contain mixtures of primitive/social actions and related/independent actions. A sequence Σ containing 4 actions (3 social [$\sigma_1, \sigma_2, \sigma_3 \in S$], 1 primitive [$p_1 \in P$]); two of which are related (σ_2 and σ_3) and two of which are not (p_1, σ_1) can be written as follows: $\Sigma = \{p_1, \sigma_1, \Re_{\sigma 2, \sigma 3}(\sigma_2, \sigma_3)\}$. For our purposes, it is assumed that primitive actions are solved by action sequences of length one (i.e. action $p \in P$ is solved by the sequence $\Sigma = \{p\}$[6]). In goal directed systems, actions are carried out in order to attain particular objectives; so Σ_σ represents the situation in which action sequence Σ is executed in order to fulfill objective σ.

At any particular instant in time the action sequence is likely to be partial. It may be *temporally partial*; the exact ordering between some plan elements may not be specified because it does not affect the plan (Pollack, 1990). For example, an agent may have a plan to set a table that includes carrying the place mats, plates and glasses to the table. The agent may believe that the first act in the list must be performed before the others but that the remaining ones can be performed in any order with respect to one another and that they can be interleaved. Action sequences can also be *structurally partial*. Agents frequently decide upon objectives they wish to achieve, leaving open for later deliberation questions about means to achieve those ends. Unlike traditional hierarchical planners, where partial plans are only used as intermediate steps in the plan formation process, it is often useful in multi-agent systems to act on the basis of such partial plans (Bratman, 1984).

In multi-agent planning, it is often useful to distinguish the actions which need to be carried out from the agents who will actually perform them. This separation allows the mechanisms for deciding the actions (determining the recipe) to be independent of task and resource allocation mechanisms. Thus, for example, a centralised agent may develop the recipe and pass it onto a task allocation agent who defines which actions will be performed by which agents. Once an action sequence has been defined, the agents who will actually perform it need to be decided upon - actions and action sequences must be *instantiated*.

Primitive Action Instantiation

$<\alpha, a>$: agent $\alpha \in A$ is involved in performing primitive action $a \in P$.

Social Action Instantiation

$<\{\alpha_1,..,\alpha_n\}, \sigma>$: agents $\alpha_1,..,\alpha_n \subseteq A$ $(n > 1)$ are involved in performing social action $\sigma \in S$.

6. In reality such actions may be composed of several sub-components, but these will be internal to the agent and need not concern us at this level.

Action Sequence Instantiation

A sequence of primitive and social action *instantiations* which specifies the agents who will perform each component as well as the actions themselves. If Σ_σ is an action sequence, then its instantiation is represented by Σ'_σ. An example being: $\Sigma'_\sigma = \{\Re_{\sigma1,\sigma2}(<\{\alpha_1, \alpha_2\}, \sigma_1>, <\{\alpha_2, \alpha_3\}, \sigma_2>), <\alpha, a_1>, <\{\alpha_4, \alpha_5\}, \sigma_3>\}$.

The participants of an action must be known to all team members if effective coordination is to be achieved. So in $<\{\alpha_4, \alpha_5\}, \sigma_3>$ both α_4 and α_5 must mutually believe that they are engaged in a social action to achieve σ_3. Imagine an orchestra conductor faced with a room full of musicians or a film director shooting a scene; if they and the other participants do not know which musicians (actors) are involved, coordination becomes impossible. Also if the team is ill-defined then it may be possible for non-team members to infiltrate the group and sabotage its activities. Other predicates associated with actions ($a \in P$, $\sigma \in S$) are listed below. Those described using social actions have analogous definitions for primitive acts, those described using primitive acts are only applicable to this type.

• EXECUTE(α, a) and EXECUTED(α, a)

agent α will execute primitive action a next and has just executed a respectively.

• MOTIVE($<\{\alpha_1,..,\alpha_n\}, \sigma>$)

the reason why agents $\alpha_1,..,\alpha_n$ wish to achieve σ. This will typically represent a goal-subgoal hierarchy with the root node giving the rationale for carrying out the action. So in the electricity transportation domain, the motivation for collaborative activity is that a fault has been detected and the reason for carrying out specific actions such as monitoring the network is that they are a part of the overarching joint act of fault diagnosis.

• RELATION-OK

indicates that the relationship between a number of actions in the same action sequence instantiation is satisfied. If the group of actions are unrelated then this predicate is satisfied.

$$\text{RELATION-OK}(<\{\alpha_m... \alpha_n\}, \sigma_i>,...., <\{\alpha_o... \alpha_p\}, \sigma_j>, \Sigma'_\sigma) \equiv ?\Re_{\sigma i,.....,\sigma j} \vee \sim\Re_{\sigma i,....,\sigma j}$$

The simplest type of action whose execution can be described is a primitive act. For agent $\alpha \in A$ to undertake primitive action $a \in P$ within the context of action sequence Σ'_σ; all its associated relationships in Σ'_σ must be satisfied.

$$\text{PERFORM}(<\alpha, a>, \Sigma'_\sigma) \equiv (\forall <\{\alpha_w...\alpha_x\}, \sigma_i> \in \Sigma'_\sigma)\ \text{RELATION-OK}(<\alpha, a>, <\{\alpha_w...\alpha_x\}, \sigma_i>, \Sigma'_\sigma)?; \text{EXECUTE}(\alpha, a)$$

Similarly before a group of agents ($\{\alpha_1,..,\alpha_n\} \subseteq A$, $n \geq 2$) can execute social action $\sigma_i \in S$ within the context of action sequence Σ'_σ; all relationships involving σ_i must be satisfied.

$$\begin{aligned}
&\text{PERFORM}(<\{\alpha_1,..,\alpha_n\}, \sigma_i >, \Sigma'_\sigma) \equiv \\
&\quad (\forall <\{\alpha_w... \alpha_x\}, \sigma_j> \in \Sigma'_\sigma) \\
&\qquad\qquad \text{RELATION-OK}(<\{\alpha_1,..,\alpha_n\}, \sigma_i >, <\{\alpha_w... \alpha_x\}, \sigma_j>, \Sigma'_\sigma)?; \\
&\qquad\qquad\qquad \text{PERFORM}(<\{\alpha_1,..,\alpha_n\}, \sigma_i >, \Sigma'_{\sigma i})
\end{aligned}$$

Analogously the predicate PERFORMED indicates whether a joint action has been carried out and uses EXECUTED instead of EXECUTE.

4.2.2 Solution Commitment

Having given the syntax and semantics of the plan description language it is now possible to proceed with the definition of solution commitment. Solution commitment defines the mental attitudes for joint action which are connected with plan states; it incorporates both prerequisites and a prescription of how agents should behave with respect to the commonly agreed solution. The first aspect is that all team members $\{\alpha_1,..,\alpha_n\}$ acknowledge the *principle* that a common solution Σ'_σ (or action sequence instantiation) is needed in order to achieve the desired objective σ:

$$\begin{aligned}
&\text{NEED-COMMON-SOLUTION}(<\{\alpha_1,..,\alpha_n\}, \sigma>) \equiv \\
&\qquad (\Diamond\, \exists \Sigma'_\sigma\ \text{PERFORM}(<\{\alpha_1,..,\alpha_n\}, \sigma>, \Sigma'_\sigma))?;\ \sigma
\end{aligned}$$

Many of the formulations of joint intention fail to express even this basic requirement (Jennings, 1992a). In some cases this is because plan states are not considered (eg Group Minds, Collective Intentionality, We-Intentions and Social Plans). However even those which do represent plan states, either in an explicit manner (eg SharedPlans) or in an implicit one (eg strategies), fail to capture this essential notion.

This does *not* imply that such a solution must be fully developed before joint action can commence since action sequence instantiations may be partial. Rather this condition stipulates that at the onset, the group must believe that they will arrive at and work within a common context. All the members must be convinced that the group is capable of reaching an agreement and also they must be confident that if all goes well this solution will be adhered to and the desired objective will ensue as a consequence of the actions taken. There is no constraint on how or when the solution is developed.

By explicitly recognising that joint action requires a common solution, it is possible to couch subsequent relations in its context. In particular, it is necessary to enumerate conditions under which commitment to a particular solution as a means of obtaining the collective goal should be dropped. This aspect is not tackled in any of the formulations presented in chapter three and extends the basic principles of joint persistent goals to plan states. Enumerating such conditions shapes the joint action tracking component by defining situations which need to be detected (see section 5.2.2). If they arise, they need to be recognised at the earliest opportunity and the agent should halt its associated processing.

It is irrational for an agent to remain committed to the common solution if:

• the expected outcome of an action is already available, either because the action has already been performed or the outcome has been produced by a different sequence of actions.

$$\text{OUTCOME-AVAILABLE}(<\{\alpha_1,..\alpha_n\}, \sigma>, \Sigma'_\sigma) \equiv \qquad \{\alpha_w,..,\alpha_x\} \subseteq \{\alpha_1,..,\alpha_n\}$$
$$(\exists <\{\alpha_w,..\alpha_x\}, \sigma_i> \in \Sigma'_\sigma)\ \text{PERFORM}(<\{\alpha_w,..\alpha_x\}, \sigma_i>) \wedge ?\sigma_i$$

• following the agreed action sequence does not achieve the desired action.

$$\text{INVALID}\ (<\{\alpha_1,..\alpha_n\}, \sigma>, \Sigma'_\sigma) \equiv \text{PERFORMED}(<\{\alpha_1,..,\alpha_n\}, \sigma >, \Sigma'_\sigma);\ \sim\sigma?$$

• one of the specified actions cannot be carried out.

$$\text{UNATTAINABLE}\ (<\{\alpha_1,..\alpha_n\}, \sigma>, \Sigma'_\sigma) \equiv \qquad \{\alpha_w,..,\alpha_x\} \subseteq \{\alpha_1,..,\alpha_n\}$$
$$(\exists <\{\alpha_w,..,\alpha_x\}, \sigma_i> \in \Sigma'_\sigma)\ \Box{\sim}\text{PERFORM}(<\{\alpha_w,..,\alpha_x\}, \sigma_i>\ \Sigma'_\sigma)$$

• one of the agreed actions has not been carried out.

$$\text{VIOLATED}\ (<\{\alpha_1,..\alpha_n\}, \sigma>, \Sigma'_\sigma) \equiv {\sim}\text{PERFORMED}(<\{\alpha_1,..,\alpha_n\}, \sigma>, \Sigma'_\sigma)$$

The above represent situations in which an individual team member can detect, for itself, that the agreed common solution is no longer sustainable.

$$\text{DETECT-PROBLEM-LOCALLY}(\alpha, <\{\alpha_1,..\alpha_n\}, \sigma>, \Sigma'_\sigma) \equiv \qquad \alpha \in \{\alpha_1,..,\alpha_n\}$$
$$\begin{aligned}\text{BEL}(\alpha,\ \ & \text{OUTCOME-AVAILABLE}(<\{\alpha_1,..\alpha_n\}, \sigma>, \Sigma'_\sigma)\ \text{v} \\ & \text{INVALID}(<\{\alpha_1,..\alpha_n\}, \sigma>, \Sigma'_\sigma)\ \text{v} \\ & \text{UNATTAINABLE}(<\{\alpha_1,..\alpha_n\}, \sigma>, \Sigma'_\sigma)\ \text{v} \\ & \text{VIOLATED}(<\{\alpha_1,..\alpha_n\}, \sigma>, \Sigma'_\sigma))\end{aligned}$$

Because of the very nature of group problem solving, if one participant stops contributing because it has detected a problem then the whole activity may be jeopardised. Therefore if an agent comes to believe that a fellow team member is no longer committed to the solution, then it also needs to reassess its position. In this case the potential problem has been detected externally to the agent by a fellow team member.

$$\text{PROBLEM-DETECTED-NONLOCALLY}(\alpha, <\{\alpha_1,..\alpha_n\}, \sigma>, \Sigma'_\sigma) \equiv \qquad \alpha_i \neq \alpha, \alpha \in \{\alpha_1,..,\alpha_n\}$$

$$\text{BEL}(\alpha, (\exists \alpha_i \in \{\alpha_1,..,\alpha_n\} \; \text{DETECT-PROBLEM-LOCALLY}(\alpha_i, <\{\alpha_1,..\alpha_n\}, \sigma>, \Sigma'_\sigma)))$$

Combining these two facets, it is possible to define the situations under which agent α can drop commitment to common solution Σ'_σ for group action $<\{\alpha_1,..\alpha_n\}, \sigma>$.

$$\text{DROP-SOLUTION-COMMIT}(\alpha, <\{\alpha_1,..\alpha_n\}, \sigma>, \Sigma'_\sigma) \equiv \qquad \alpha \in \{\alpha_1,..,\alpha_n\}$$
$$\text{DETECT-PROBLEM-LOCALLY}(\alpha, <\{\alpha_1,..\alpha_n\}, \sigma>, \Sigma'_\sigma) \vee$$
$$\text{PROBLEM-DETECTED-NONLOCALLY}(\alpha, <\{\alpha_1,..\alpha_n\}, \sigma>, \Sigma'_\sigma)$$

It is insufficient for an agent (α) to simply disregard a joint action once it is no longer committed to the agreed solution, because its accomplices may not have been able to detect the problem for themselves. Therefore to ensure such information is disseminated as widely as possible, α must endeavour to inform all other team members of the fact that it is no longer committed to the common solution. This enables them to reassess the agreed solution and in particular the actions involving α. Thus if the common solution needs to be abandoned or refined, the amount of wasted resource is kept to a minimum because futile activities are stopped at the earliest opportunity.

Individual solution commitment (ISC) encodes this high level description of how each team member should behave in its own problem solving and towards others with regard to the common solution. For each action that an agent is involved in, it should believe that it is going to perform that action and also actually endeavour to perform the action at the specified time. This mental state continues until the agent has good cause not to follow the agreed solution; whereupon it aims to disseminate its lack of commitment to all the others.

```
$\text{ISC}(\alpha, <\{\alpha_1,..\alpha_n\}, \sigma>, \Sigma'_\sigma) \equiv$                         $\alpha \in \{\alpha_1,..,\alpha_n\}$
  **WHILE** $\sim\text{DROP-SOLUTION-COMMIT}(\alpha, <\{\alpha_1,..\alpha_n\}, \sigma>, \Sigma'_\sigma)$ **DO**
      $(\forall <\{\alpha, \alpha_w,..\alpha_x\}, \sigma_i> \in \Sigma'_\sigma)$          $\{\alpha, \alpha_w,..,\alpha_x\} \subseteq \{\alpha_1,..,\alpha_n\}$
            $\text{BEL}(\alpha, \text{PERFORM}(<\{\alpha, \alpha_w,..\alpha_x\}, \sigma_i>, \Sigma'_{\sigma i})) \wedge$
            $\text{PERFORM}(<\{\alpha, \alpha_w,..\alpha_x\}, \sigma_i>, \Sigma'_{\sigma i}))$
  **WHEN**
      $\text{GOAL}(\alpha, \text{MB}(\{\alpha_1,..\alpha_n\},$
                 $\text{DROP-SOLUTION-COMMIT}(\alpha, <\{\alpha_1,..\alpha_n\}, \sigma>, \Sigma'_\sigma)))$
```

An important reason for explicitly distinguishing between goal and plan states can be uncovered 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 goal 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 invalid, the agents may retry their activity in the hope that the objective may be achieved this time or they may try a different action which produces the same result. Similarly if the plan is violated, the agents would typically 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. With the plan, the group would typically enter a replanning phase to amend the existing solution or develop a new one, with goal states the joint act is finished. 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 (see section 5.2 for further details of the functional architecture produced from the responsibility model).

Combining the results of this section, there are two facets related to plan states in a social group; agreeing to the principle of a common solution and defining how individuals should behave once the solution has been chosen.

$$\begin{aligned}&\text{SOLUTION-COMMITMENT}(<\{\alpha_1,..,\alpha_n\}, \sigma>) \equiv\\&\quad \text{MB}(\{\alpha_1,..,\alpha_n\}, \text{NEED-COMMON-SOLUTION}(<\{\alpha_1,..,\alpha_n\}, \sigma>)) \wedge\\&\quad \text{MB}(\{\alpha_1,..,\alpha_n\}, (\forall\ \alpha_i \in \{\alpha_1,..\alpha_n\}\ \text{ISC}(\alpha_i, <\{\alpha_1,..\alpha_n\}, \sigma>, \Sigma'_\sigma)))\end{aligned}$$

4.2.3 Ready, Willing and Able To Contribute

An important aspect of collaborative problem solving, which is seldom considered, is the make-up of the cooperating group (Jennings, 1991b). One illustration of this is the tacit assumption that all the participants are able to contribute to some aspect of the group's objective (i.e. there are no free-loaders). This property can be regarded as the minimum requirement for entry into the group and must be satisfied by all team members. There are two ways in which an agent can contribute to a social goal; by performing an action which is part of the agreed solution (*positive contribution*) or by refraining from performing one which would interfere with it (*non-negative contribution*[7]). To illustrate these concepts, imagine a team of agents trying to stack blocks B1, B2 and B3; a positive contribution could be putting B2 onto B1, a non-negative one not unstacking B2 once it is in place (because it is known that doing so would have a detrimental effect on the stacking blocks objective).

Action $\sigma_i \in S$ is a component of action sequence Σ_σ if and only if Σ_σ contains σ_i

[7]. Negative plan interactions include interfering plan steps or incompatible use of a shared resource. If such interactions occur, they may cause the joint action to fail. Hence deliberately avoiding them is a meaningful contribution and warrants an agent's inclusion in the group.

or there is an action σ_j which is a part of Σ_σ for which σ_i is a component of $\Sigma_{\sigma j}$. Thus an action is a component of a solution if it occurs at any node of the plan transformation search space, irrespective of its depth.

$$
\begin{aligned}
&\text{COMPONENT-OF}(\sigma_i, \Sigma_\sigma) \equiv \\
&\qquad (\sigma_i \in \Sigma_\sigma) \vee \\
&\qquad (\exists\, \sigma_j \in \Sigma_\sigma)\ (\exists\, \Sigma_{\sigma j}\ \text{COMPONENT-OF}(\sigma_i, \Sigma_{\sigma j}))
\end{aligned}
$$

The next stage is to formalise what it means for an agent to have the potential of being able to contribute to a group action. Positive contribution requires a solution to the group goal (σ) in which the agent (α) is capable of performing at least one action.

$$
\begin{aligned}
&\text{POSITIVE-CONTRIBUTION}(\alpha, \sigma) \equiv && \alpha \in \{\alpha_1,..\alpha_n\} \\
&\qquad (\exists \Sigma'_\sigma\ \text{PERFORM}(<\{\alpha_1,.....,\alpha_n\}, \sigma>, \Sigma'_\sigma);\ \sigma?) \wedge && \{\alpha_x,..,\alpha_y\} \subseteq \{\alpha_1,...,\alpha_n\} \\
&\qquad (\exists <\{\alpha, \alpha_x,..,\alpha_y\}, \sigma_i> \in \Sigma'_\sigma\ \text{COMPONENT-OF}(\sigma_i, \Sigma'_{\sigma i}))
\end{aligned}
$$

Non-negative contribution means an agent deliberately abstains from performing an action which if carried out would jeopardise the group's efforts. An action is harmful to another if it stops it from being realised by the means currently being pursued.

$$
\begin{aligned}
&\text{DISRUPT}(<\{\alpha_x,..,\alpha_y\}, \sigma_{disrupt}>, <\{\alpha_1,.....,\alpha_n\}, \sigma>, \Sigma'_\sigma) \equiv \\
&\qquad (\exists\, \Sigma'_{disrupt}\ \text{PERFORM}(<\{\alpha_x,..,\alpha_y\}, \sigma_{disrupt}>, \Sigma'_{disrupt})) \supset \\
&\qquad\qquad \text{PERFORM}(<\{\alpha_1,.....,\alpha_n\}, \sigma>, \Sigma'_\sigma);\ {\sim}\sigma?
\end{aligned}
$$

An agent makes a non-negative contribution if it deliberately avoids carrying out an action because it realises that by doing so it enables the group action to proceed as planned.

$$
\begin{aligned}
&\text{NON-NEGATIVE-CONTRIBUTION}(\alpha, \sigma) \equiv && \alpha \in \{\alpha_x,..\alpha_y\} \\
&\qquad (\exists \Sigma'_\sigma\ \text{PERFORM}(<\{\alpha_1,.....,\alpha_n\}, \sigma>, \Sigma'_\sigma));\ \sigma? \wedge \\
&\qquad (\exists <\{\alpha_x,..,\alpha_y\}, \sigma_{disrupt}> \\
&\qquad\qquad \text{DISRUPT}(<\{\alpha_x,..,\alpha_y\}, \sigma_{disrupt}>, <\{\alpha_1,.....,\alpha_n\}, \sigma>, \Sigma'_\sigma)) \wedge \\
&\qquad {\sim}\text{PERFORM}(<\{\alpha_x,..,\alpha_y\}, \sigma_{disrupt}>, \Sigma'_{disrupt})
\end{aligned}
$$

Combining these two facets, an agent is able to make a contribution to the group's objective if it can carry out a positive act or it refrains from carrying out a negative one.

$$
\begin{aligned}
&\text{CAN-CONTRIBUTE}(\alpha, \sigma) \equiv \\
&\qquad \text{POSITIVE-CONTRIBUTION}(\alpha, \sigma) \vee \\
&\qquad \text{NON-NEGATIVE-CONTRIBUTION}(\alpha, \sigma)
\end{aligned}
$$

CAN-CONTRIBUTE defines whether an agent has the *potential* to contribute to a group action; however merely possessing this capability does not imply the agent will actually use it. For various reasons, such as it is too busy or it is not interested in the group's objective, the agent may not wish to participate. If an agent (α) is willing to take part in collaborative activity σ, there is a possibility that a group involving α may form a joint persistent goal of achieving their shared aim.

$$\text{WILL-PARTICIPATE}(\alpha, <\{\alpha_1,..,\alpha_n\}, \sigma>) \equiv \qquad \alpha \in \{\alpha_1,..\alpha_n\}$$
$$\text{JPG}(<\{\alpha, \alpha_1,..,\alpha_n\}, \sigma>, \text{MOTIVE}(<\{\alpha, \alpha_1,..,\alpha_n\}, \sigma>))$$

Having identified what it means for an individual to be both capable and inclined towards contributing, conditions which must be met before it can be considered as a potential collaborator must be defined. The definition given below requires an agent to convince all the others of its capability and willingness before it can be considered; a very rigid entry requirement. At the other extreme, individuals may be admitted on the basis that they alone believe they satisfy the conditions (in which case BEL would replace MB[8]).

$$\text{POTENTIAL-CONTRIBUTOR}(\alpha, <\{\alpha_1,..,\alpha_n\}, \sigma>) \equiv \qquad \alpha \in \{\alpha_1,..\alpha_n\}$$
$$\text{MB}(\{\alpha_1,..,\alpha_n\}, \text{CAN-CONTRIBUTE}(\alpha, <\{\alpha_1,..,\alpha_n\}, \sigma>)) \wedge$$
$$\text{MB}(\{\alpha_1,..,\alpha_n\}, \text{WILL-PARTICIPATE}(\alpha, <\{\alpha_1,..,\alpha_n\}, \sigma>))$$

This membership function may be dependent on the application. In some cases it may be desirable to form very tight knit groups, in which case the initial criteria for inclusion may be very strict. Whereas in other cases it may be preferable to have a large pool of potential contributors and only finalise the group's exact make-up at the last possible moment.

4.2.4 Joint Responsibility Defined

The mental state of joint responsibility which a group of agents $\{\alpha_1,..,\alpha_n\}$ must adopt if they are to collaboratively attain a shared objective (σ) can now be defined.

$$\text{JOINT-RESPONSIBILITY}\ (<\{\alpha_1,..,\alpha_n\}, \sigma>) \equiv$$
$$\text{MB}\ (\{\alpha_1,..,\alpha_n\}, \text{JPG}(<\{\alpha_1,..,\alpha_n\}, \sigma>, \text{MOTIVE}(<\{\alpha_1,..,\alpha_n\}, \sigma>))) \wedge$$
$$\text{MB}\ (\{\alpha_1,..,\alpha_n\}, \text{SOLUTION-COMMITMENT}\ (<\{\alpha_1,..,\alpha_n\}, \sigma>)) \wedge$$
$$\text{MB}(\{\alpha_1,..,\alpha_n\},$$
$$(\forall \alpha_i \in \{\alpha_1,..,\alpha_n\}\ \text{POTENTIAL-CONTRIBUTOR}(\alpha_i, <\{\alpha_1,..,\alpha_n\}, \sigma>))$$

This formulation fulfils an important desiderata of joint intentions; "rational agents need general policies that govern the reconsideration of prior intentions and plans....... this non-reconsideration should be treated as the default, over-rideable by special kinds of problems" (Bratman, 1984). Default behaviour occurs when there is

[8]. It is also possible to have other points within this continuum; so individuals may be admitted if they can convince more than a third, half, two-thirds and so on of the members.

no reason to drop commitment to either the collective goal or the means of attaining it. The exceptional conditions have been defined and there is a specification of how the agent should behave in such circumstances. In addition to these general policies, the model also makes explicit the conditions which must be satisfied before cooperative problem solving can commence.

4.3 Responsibility in Electricity Transport Management

The purpose of this section is to illustrate that the model of joint responsibility is appropriate and useful for describing collaborative problem solving in dynamic and unpredictable environments. To this end, the electricity transport domain scenario involving the AAA, BAI and CSI performing a collaborative diagnosis will be used. In this case the CSI detects the disturbance and monitors the network for significant changes, whilst the AAA and BAI attempt to locate the fault.

Once the need for collaborative action has been detected and all three agents are aware of this fact and have agreed to participate, a social goal exists:

$$\sigma = <\{AAA, BAI, CSI\}, DIAGNOSE\text{-}FAULT>$$

When the specified preconditions have been met, the actual solution can be developed. The responsibility framework is independent of any particular planning paradigm; so it may be derived by a centralised planning or from a collaborative exercise involving several agents. Assuming the planning process is successful, a common solution will be developed[9].

$$\begin{aligned}\Sigma'_{\text{diagnose}} = \{\textit{SIMULTANEOUSLY}\ (\quad & <\{CSI\}, MONITOR\text{-}NETWORK> \\ & <\{AAA, BAI\}, LOCATE\text{-}FAULT>)\}\end{aligned}$$

This action sequence instantiation has two components; one primitive action (monitor-network) and one social action (locate-fault) which are related by the SIMULTANEOUSLY relationship. As interaction proceeds, the social action locate-fault will need to be further specified until it results in an action sequence which contains just primitive actions.

$$\begin{aligned}\Sigma'_{\text{locate-fault}} = \{\quad & \textit{SIMULTANEOUSLY}\quad <\{BAI\}, PRODUCE\text{-}BOA> \\ & \qquad\qquad\qquad\qquad <\{AAA\}, INITIAL\text{-}DIAGNOSIS>), \\ & \textit{AFTER}(\quad <\{BAI\}, PRODUCE\text{-}BOA>, \\ & \qquad\qquad <\{AAA\}, FINAL\text{-}DIAGNOSIS>), \\ & \textit{AFTER}\ (\quad <\{AAA\}, INITIAL\text{-}DIAGNOSIS>, \\ & \qquad\qquad <\{AAA\}, FINAL\text{-}DIAGNOSIS>)\}\end{aligned}$$

Having established the common solution at the appropriate level of detail, responsibility requires each agent to carry out its agreed part whilst committed to the joint action and the chosen means of attaining it (i.e. $\Sigma'_{\text{diagnose}}$). Thus each of the

[9]. Note that the subscripts on the relationships have been dropped to aid readability.

agents will start their initial actions; the CSI will monitor the network and the AAA and BAI will start locating the fault. If everything goes smoothly and this first round of actions is completed satisfactorily, the AAA will proceed to its final diagnosis phase and the shared objective will be met by the production of the elements at fault. When the AAA produces this final result, the collective's objective is satisfied and there is no longer a joint persistent goal to produce the list of faults. In these circumstances, the AAA must endeavour to inform its fellow team members of its success and of the reason why it is no longer committed. Assuming the AAA is successful in this activity, the BAI and CSI will realise that the shared aim has been realised and they can drop the goal of producing the list of faulty elements. Only at this stage are all team members aware that the joint action has been successful.

However because of the environmental dynamics and the system's inherent uncertainty and unpredictability, many events may disrupt the planned joint action. There are two main categories of problem; those with respect to the overall goal of diagnosing the fault and those with respect to $\Sigma'_{diagnose}$ the common solution for attaining it. Considering the joint persistent goal of diagnosing the fault which produces the result ELEMENTS-AT-FAULT.

JPG(<{AAA, BAI, CSI}, DIAGNOSE-FAULT>, FAULT-IN-NETWORK)

• the motivation for the objective is no longer present.

Representation: ~FAULT-IN-NETWORK

Example: CSI realises that the group of alarms which it originally interpreted as a fault either represented a transient fault or no fault at all. Therefore the AAA and BAI must be informed so that they do not continue attempting to locate a nonexistent fault.

• the objective will never be attained.

Representation: $\Box$~ELEMENTS-AT-FAULT

Example: AAA realises that it is not being supplied with sufficient alarm messages to make a diagnosis. Therefore ELEMENTS-AT-FAULT cannot be produced because the AAA's FINAL-DIAGNOSIS task is the only means of producing them.

Problems may also arise with the agreed solution and thus affect the individual solution commitment to $\Sigma'_{diagnose}$

• following the agreed common plan does not lead to the desired outcome.

Representation: INVALID(<{AAA, BAI}, LOCATE-FAULT>, $\Sigma'_{locate\text{-}fault}$)

Example: CSI detects a substantial change in the network, meaning that the models being used by the AAA and BAI are so inaccurate that any ensuing diagnosis will be worthless.

• one (or more) of the actions cannot be executed.

Representation: ❑~PERFORM(<{CSI}, MONITOR-NETWORK>,
MONITOR-NETWORK)

Example: CSI is no longer receiving information from the network and so is unable to monitor its status. This is different from the goal being unattainable because the AAA and BAI may decide to attempt the diagnosis without having the CSI monitor the network status.

• one of the agreed actions has not been performed correctly.

Representation: VIOLATED(<{BAI}, PRODUCE-BOA>, PRODUCE-BOA)

Example: BAI has been distracted by an unplanned task and cannot produce the black out area at the agreed time. This means the AAA cannot start its detailed diagnosis because it is unable to compare its preliminary hypothesis with the black out area in order to ensure consistency.

As this scenario highlights, collaborative fault diagnosis in the electricity transport domain is fraught with opportunities for failures. When the joint action does run into problems, it is usually detected by only one team member; the others have to rely on being informed. Without a prescription of how to behave or criteria against which to evaluate joint problem solving, the team may perform in an inefficient or uncoordinated manner, as was the case with GRATE communities (see section 2.7). For example, if after having detected the fault is transient, the CSI merely stopped its local activity without informing the others then they would continue to expend resources on diagnosing a fault which does not exist. Responsibility ensures that the CSI tries to inform the AAA and BAI that the motivation for their joint action is no longer present.

4.4 Conclusions

There were two key constraints on the model of joint responsibility which distinguish it from previous work. Firstly it should have a clear path to implementation level systems. The importance of this link has been made by several researchers, indeed some go as far as to state that AI will not advance as a science until the gap between researchers who construct models and those who build systems is closed (Cohen, 1991). To this end, the responsibility model aids the application designer by providing a description of the required agent functionality, specifying the mental states of the collaborating agents and defining how agents should behave in nominal and exceptional circumstances. This strand of work is elaborated upon in section five where a high-level architecture is given and a deep representation of an agent’s joint action tracking component is outlined. The joint responsibility description of behaviour addresses the dual roles which team members have to play; specifying how they should behave with respect to their individual problem solving and also with respect to their fellow team members. This contrasts with many of the models reviewed in the previous chapter which aim to provide a plausible psychological or philosophical account of community behaviour from an external perspective.

A second distinguishing characteristic of this work is that it was targeted specifically at the problem of ensuring robust, coordinated behaviour in dynamic and unpredictable environments (such as industrial control). The importance of this problem was highlighted by Gasser (1991) who observed that a theory of DAI ought to account for how aggregates of agents can achieve joint actions that are robust and continuable despite intermediate foul-ups and inconsistencies. As the empirical investigations of section six highlight, responsibility offers a step towards this objective; providing procedures for controlling activities in dynamic and unpredictable environments, whilst retaining a degree of generality and predictability.

The fact that notions of intention should embody more than just beliefs about the state and intentions of other agents is an important contribution of this research. In addition to the mental state framework, the concomitant behavioural rules for dealing with unanticipated events form an integral part of this formulation. These rules ensure intentions play the central role in coordinating collaborative problem solving which has been proposed earlier in this thesis.

Responsibility is at a sufficiently abstract level to be equally applicable to both human and artificial problem solvers. This is especially useful as it means in communities in which the two types of problem solver coexist (eg participant systems (Chang, 1988) or human computer cooperative work (Steiner *et al.*, 1990)), responsibility can be viewed as a common basis for interaction.

Finally, joint responsibility also represents an important step forward in the field of distributed and multi-agent planning. Presently these systems do not maintain a principled model of joint problem solving activity and their control knowledge is compiled-down and represented implicitly within the planning formalism. Joint responsibility offers an explicit meta-level representation which is independent of any particular planning paradigm but can easily be mapped down to this level. Hence the causal and motivational links for many of the important actions and interactions within the planners can be explained by referring to the responsibility model.

5. GRATE*: A COOPERATION KNOWLEDGE LEVEL SYSTEM

This chapter outlines the case for a new computer level specifically for collaborating problem solvers. The *cooperation knowledge level* (Jennings, 1992a/b) describes rich and explicit models of common social interactions. The rationale and benefits of the new level are explained and examples of existing DAI work which falls into this category are given. The model of joint responsibility is used as an illustration of a knowledge level description for collaborative activity.

Following the knowledge level approach, a broad functional architecture for an agent with a social competence is presented. The proposal describes the key role of joint intentions; particular emphasis is given to the component which tracks the progress of joint actions and a deep model of collaborative problem solving is given.

Finally, the realisation of this high-level architecture in the GRATE* system is discussed. It is shown how the responsibility model's prerequisites for collaboration are satisfied, how the tracking component works and how commitments are made and honoured.

5.1 Defining the Cooperation Knowledge Level

Sophisticated problem solving is based upon knowledge. In advanced systems, typified by expert systems, this knowledge can be divided into two distinct categories; knowledge about the domain (eg blood disorders (Buchanan and Shortcliffe, 1984) and system configuration management (McDermott, 1981)) and knowledge about problem solving per se (i.e. the inference structures). First generation expert systems, despite some degree of success, had many important drawbacks, as outlined in section 1.1, which were attributed to their sole use of “surface knowledge” (Steels, 1985).

To overcome these problems, second generation expert systems use as rich and explicit models of knowledge as possible - so called deep knowledge (Steels, 1985). These systems have explicit models of the domain and the inference calculus operating over the models. Examples of components which are made explicit include: the inference structure (eg within the field of heuristic classification there are those making abstractions of the data, those matching the data with a solution class and those refining this solution to the actual solution (Clancy, 1985)); the problem solving method (eg cover and differentiate, propose and revise (McDermott, 1988)) and the generic task (eg classification, interpretation and diagnosis (Chandrasekaran, 1983a)). Similar approaches which emphasize the key role of explicit representations have also been proposed within the planning community (Cesta *et al.*, 1989). They note that advances in plan synthesis and recognition will require the representation of knowledge and strategies for its management to be made explicit. Structures of explicit knowledge must include both planning knowledge and deep causal theories referring to the domain of application.

All of the above processes concentrate on making explicit the knowledge which is brought to bear during problem solving and clearly distinguish it from representational issues. Such knowledge-level (Newell, 1982) approaches have enabled significant advances to be made in single agent problem solving. At this time, Distributed AI systems are suffering from many of the problems which were associated with first generation expert systems; including difficulties operating in dynamic environments, weak explanation of social activity, poor software reusability and lack of a structured framework within which development can proceed (Jennings, 1992a). It is believed that these failings are symptoms of a more fundamental problem; namely the dearth of high level principles about collaborative problem solving. Therefore just as recognition of the existence of a knowledge level allowed high-level principles of the individual problem solving process to be recognised; it is believed that a similar set of principles should be developed for multi-agent problem solving. This development will be enhanced by separating out issues of collaborative problem solving from those of individual problem solving and representing them in a distinct computer level.

The cooperation knowledge level concentrates on developing principles and explicit models of various social phenomena (such as cooperation, coordination, hostility, competition and conflicts) as well as the reasoning processes which control them. In Newell’s taxonomy of computer system levels, the cooperation level would be directly above the knowledge level. Like the others, the cooperation knowledge level can be reduced to the level directly below it; ultimately being expressed in terms of single agents and individual goals, actions and knowledge states.

The benefits of developing cooperation knowledge level systems are similar to those experienced when transferring from first to second generation expert systems. They include better explanations of the objectives and means of social interactions, greater system flexibility in terms of the mechanisms and knowledge brought to bear during joint problem solving and finally a more structured knowledge acquisition process.

Explanation is enhanced because the group’s activities can be described at a meta-level rather than at a task or message level. For instance it is possible to provide a helpful abstraction of a protracted series of message exchanges if there is an explicit representation of the social activity being undertaken. So with cooperation knowledge level systems it would be possible to state that agent A1 is trying to persuade A2 to do X or A1 and A2 are engaged in a conflict about Y which they are trying to resolve by Z; rather than merely indicating that message M1 has been sent from A1 to A2, A2 has responded with M2 and so on. These advances are important if DAI systems are to leave the laboratory and especially so in situations in which the user plays an active problem solving role (i.e. participant systems (Chang, 1988) or human computer cooperative work (Steiner *et al.*, 1990)).

System generality is improved by clearly distinguishing between the domain-independent principles on which behaviour is based and the domain-dependent knowledge which is used as data during the reasoning process. The generic component, which represents the cooperation knowledge level description, can be isolated and coded as an off the shelf software system. It can be applied to many different problems merely by substituting in the appropriate domain knowledge.

Finally, the multi-agent system developer is aided by having a focused set of questions, strategies and options with which to confront the organization who commissioned the system. This framework allows development to proceed more rapidly because its inherent structure means much of the groundwork has already been carried out; it also helps avoid omissions by providing a comprehensive view of the problem being tackled. This issue is discussed at length in section 5.2.2 where the structure imposed by the responsibility model is examined and its benefits in constructing GRATE* assessed.

The cooperation level differs from the knowledge level in at least two important ways; in terms of the *system* and in the *laws which govern component behaviour.* The cooperation level has societies as the system, whereas the single agent knowledge level has individuals. Thus formalisms are needed to describe the actions of collectives, as well those of the individuals - joint responsibility allows collectives to be represented by means of social actions. Since it is individuals who have the ability to act, work must be done on the means of mapping descriptions of collective actions into those of the individuals and of combining related individual acts into a single supra-activity. That is, a two-way mapping between the cooperation and individual knowledge levels is required.

The other major area of difference between the individual and cooperation knowledge levels is in the laws which determine how system behaviour depends upon component behaviour and the structure of the system. Within the single agent knowledge level the behaviour law is that of rationality: if an agent knows an action will lead to the satisfaction of one of its goals, it will select that action (Newell, 1982). However the joint responsibility model stipulates that even after an agent has discovered that there will be no personal benefit in pursuing the social activity, it must still endeavour to inform the other team members of this fact. The advantage of team rationality over individual rationality is highlighted in section 6.3. As this activity requires computational resource which the agent could be spending on potentially beneficial activities it is an irrational act when viewed from the position of the individual agent. Therefore notions such as individual rationality or self interest, even when tempered with notions such as “shadow of the future” (Axelrod, 1984), are not sufficient for defining the behaviour of participants engaged in cooperative problem solving. What is required is *team rationality*, an embodiment of intuitive notions such as “team spirit” and “being a good team member.” As behavioural laws are a key component in the definition of any level, a comprehensive theory of collective problem solving must provide appropriate definitions for them. As the review of chapter three highlighted, present formulations of joint intention are completely lacking in this area; concentrating predominantly on what it means for an intention to exist. Joint responsibility is a step in the right direction, providing prescriptions of behaviour for nominal and exceptional cases which apply to both the individual and its interactions with others.

Further support for the desirability and feasibility of the approach advocated here comes in an independent proposal made by Malone and Crowston (1990). They attempt to define a new discipline called *coordination theory* which aims to “derive a body of principles about how actors can work together”. They propose that work from organisational theory, sociology, social psychology, anthropology, linguistics, law and political science should be drawn together in an attempt to develop general principles about coordination. In their view, such a theory should consider how goals can be identified, activities can be ordered and assigned to actors, resources allocated and activities synchronised.

Although not explicitly advocating a cooperation knowledge level, some recent work in contemporary DAI can be viewed as moving in this direction. The most prevalent illustration is with respect to communication. In speech act theory, the effect of utterances are explicitly represented and can be reasoned about by the sender in order to try and induce specific mental states in the receiver (Austin, 1962; Cohen and Perrault, 1979; Searle, 1969; Werner, 1989). Other illustrations of explicit reasoning models occur in conflict resolution (Klein and Baskin, 1990; Lander *et al.*, 1990) in which resolution strategies are categorized and selected according to the desired objective and prevailing circumstances; in persuasion and negotiation (Castelfranchi, 1990; Sycara, 1989) in which agents reason about how to induce greater cooperativeness in other community members; in the definition of hedonic states, likes, goals and values based on physical dynamics (Kiss and Reichgelt, 1991) and in game-theoretic approaches to the selection of individual actions based on notions of utility function evaluation (Rosenschein and Genesereth, 1985).

The research described in this thesis concentrates on defining an explicit and deep model for one particular form of social interaction; namely the solving of a common problem by a team of agents. A complete knowledge level description of this process would have to cover the following aspects; the detection of when team problem solving is required or beneficial, what organizational form the team will take (will there be a single controller, a controlling committee or will all members be equal?, will decisions require unanimous or majority support?), who should be in the team (is it best to have small teams with each member doing significant amounts of processing related to the joint act or larger teams with less active members?), how to recruit community members to the team (will individuals join out of benevolence or will they need convincing?, if so how?), how to construct the team plan (single planner or multiple partial planners?), how to divide the labour within the team, how to behave once team activity has begun and how team activity should be terminated. Fortunately, not all of these issues need to be addressed every time a social action is needed. In many situations, such issues will be constrained by the structures already present in the environment; however the decisions need to have been taken originally or emerged as common practice.

As indicated previously, joint responsibility places particular emphasis on defining a model which incorporates how collaborative activity can falter and in prescribing how the individual agents should behave in such circumstances in order to retain the maximum level of group coherence. Addressing such issues is especially important in the domain of industrial control where, because of the complexity and dynamicity of the problem, planned collaborative acts frequently run into problems (Jennings, 1991a). As it is difficult to predict the exact type of problem which may be encountered in advance, it is important that agents have a general representation of collaborative action which is rich enough to cope with a wide variety of circumstances and also that they are able to manipulate this representation in a sufficiently flexible manner. These desiderata are met in GRATE* by providing a deep representation of collaborative activity and a sophisticated joint action tracking mechanism; both of which are expanded upon in the following subsection.

5.2 Implementing Joint Responsibility

One of the primary motivations for developing the responsibility model was to provide a theoretically grounded description of collaborative problem solving based on accepted joint agreements, rather than reactive and *ad hoc* adjustments. By basing the implemented system on a firm footing, it is easier to predict the range of agent responses, verify that the agent has been implemented correctly and provide a clear boundary on the types of situation in which the agent can be expected to successfully operate.

Joint responsibility had to be implementable in a computer system which could be used to control real-size industrial applications; therefore it was decided to adopt a rule-based approach rather than a modal theorem prover. A rule based interpretation of the model should consume less computational resource and hence leave the domain level system with sufficient time with which to operate effectively. Following this approach meant that several assumptions were required in order to provide a mapping with the formal description. Firstly it was assumed that communication is fool proof and that the message delay time is known to all agents. Secondly although mutual belief has many appealing theoretical properties, it is not a concept which can easily be used in implementable systems. Thus a computational approximation was required. In GRATE* this approximation is one level of nesting greater than “everybody knows” (Halpern, 1984), since agents maintain representations of the beliefs of other community members through their acquaintance models. Thirdly in order to carry out coordination activities, agents share a global clock reference[1]. Finally agents are able to accurately predict the time taken, in terms of the global clock, to execute each domain level task. This facilitates the coordination process and enables agents to make and honour commitments in a controllable manner.

A further key aim of the responsibility model is that it should be capable of ensuring coordinated group behaviour even when things do not go according to plan. In industrial applications this can occur with a relatively high frequency; because of unanticipated events or because predictions based on limited information turn out to be incorrect or inappropriate. Joint responsibility, therefore, has to provide *criteria*

[1]. This is a simplifying assumption. In real-time, distributed systems it has been shown that the concept of a unique global clock is not meaningful in the context of a distributed system of self-contained parallel agents (Agha, 1986; Hewitt and Baker, 1977).

against which ongoing collaborative activity can be evaluated as well as *procedures* for ensuring the community acts coherently in spite of possible inconsistencies and foul-ups.

With regard to implementable systems, the responsibility model can be used at two distinct levels. Firstly, at a broad level, it implies an architecture for collaborative problem solving based on the central notion of joint and individual intentions. Secondly, the model provides a detailed, but implementation independent, description of the process by which agents should track the execution of their collaborative actions. Both of these modes are examined in further detail in the remainder of this section; although it is in the latter area where the proposals for a cooperation knowledge level are most vividly illustrated.

5.2.1 Broad Functional Architecture

The responsibility model suggests a high-level BDI architecture for collaborative, multi-agent behaviour in which intentions play a central role. Intentions are used both to coordinate actions (future directed intentions) and to control the execution of current ones (present directed intentions). Adoption of the responsibility model therefore places certain constraints on the agent architecture, it is **not** neutral in terms of implementation. However the descriptions are at a suitably high level to ensure there is still significant leeway in how the concepts will be realised. This is in the spirit of Newell’s (1982) characterisation of the knowledge level; knowledge is to be characterised entirely functionally, in terms of what it does, not structurally in terms of physical objects with particular properties and relations. The responsibility model does not specify how intentions are represented (eg there may be strategic intentions to represent long term goals and tactical intentions to represent short term ones as in Burmeister and Sundermeyer (1991) or intentions may be represented as goals as in MACE (see section 2.3.2)), how commitment is described, what mechanisms are used to obtain agreements nor how to develop the common solution.

Figure 5.1 shows this high-level architecture. The rectangular boxes correspond to processes and the ovals to data stores. To aid clarity, 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. This framework represents an advance on other BDI architectures (eg Bratman *et al.*, 1988; Burmeister and Sundermeyer, 1991) in that it is specifically designed for collaborative problem solving; other proposals have concentrated on building single agents that are able to interact with others but purely from an individualistic standpoint. This proposal also incorporates joint intentions; it specifies their role in the problem solving process and their relation to individual intentions.

The structure of the key data stores of recipes and intentions (as they are realised in GRATE*) are given in section 2.3.5; joint intentions are illustrated in figure 5.4. Implementation of the processes in GRATE* is by production rules; a full listing together with the associated mapping to these processes is given in appendix B. The remainder of this subsection concentrates on elaborating the detail of this architecture

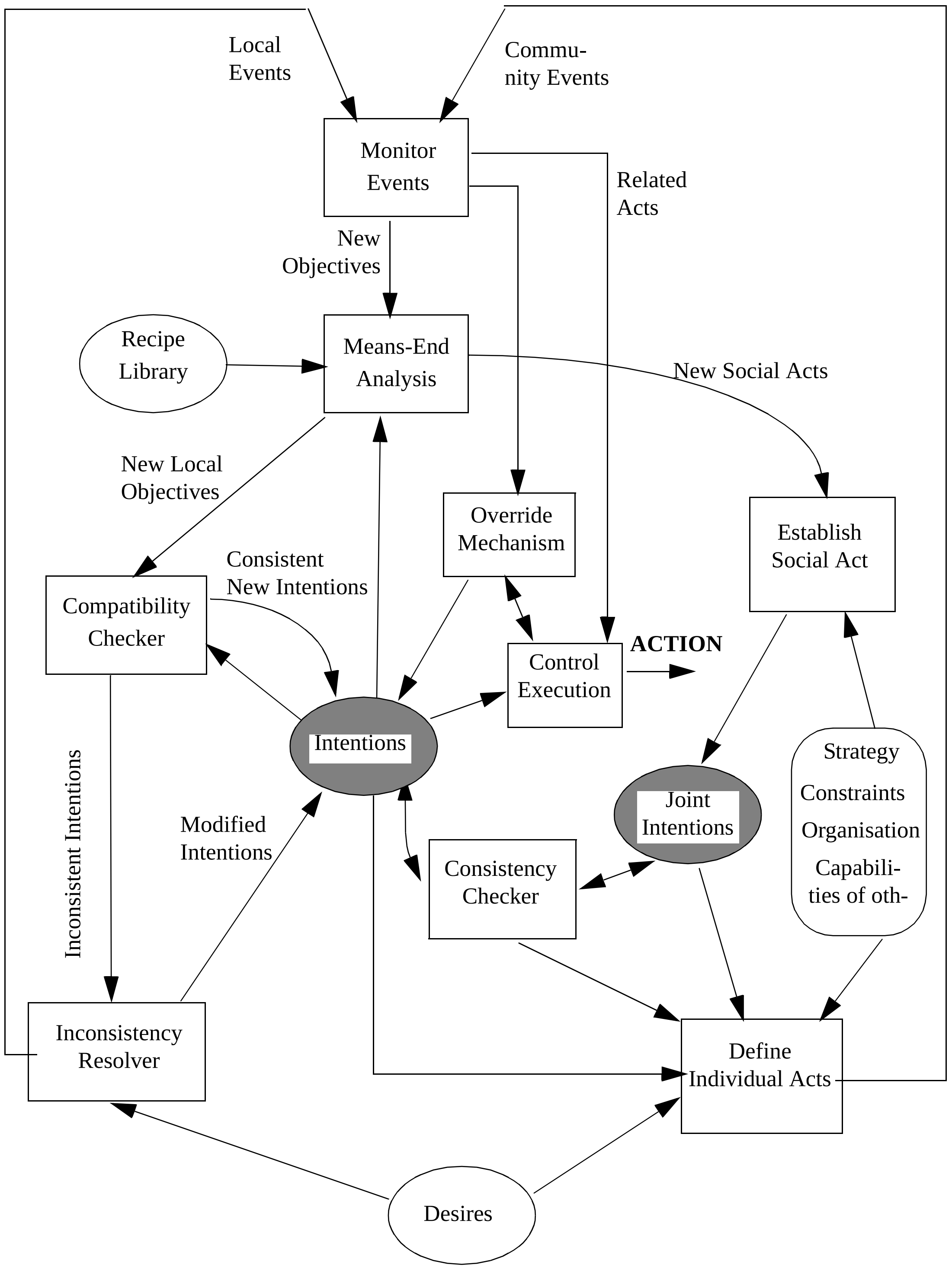

Figure 5.1: Agent Architecture suggested by Responsibility Model

proposal.

As figure 5.1 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. The objective of the event monitor process is to identify situations in which any of the following arise:

• a potential new objective is raised

> For instance in the electricity transport scenario; an agent may realise there is a fault in the network and that it ought to start a diagnosis process or that a new network description has been produced and that it should update its topology description or that a request to perform a task has been received from an acquaintance.

• an event which is related to a local action occurs

> An agent may be waiting for an acquaintance to provide a particular piece of information or to enter a specific problem solving state; when this event occurs the agent can continue with its processing. In particular, social actions are usually composed of individual actions which need to be synchronised. Such points must be detected by the agent so that it can coordinate its actions with those of others within the community and that relationships between actions can be upheld.

• an existing commitment is overridden

> Commitments are not irrevocable, therefore an agent must detect events which invalidate commitments so that it does not pursue fruitless activities.

A particular event instance may fall into several of these categories. For example in the electricity transport scenario, the event of realising that a fault is transient may act as an override for existing commitments to fault detection activity and may also necessitate a new objective of updating the network topology representation. Hence it falls into both the first and last categories.

Assume an event which signifies the need for fresh activity is 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. In these circumstances, the rationale for the new objective should be added to the list of reasons for carrying out the existing intention, rather than needlessly duplicating activity in the domain level system (see section 2.3.5).

When deciding whether to adopt a new objective, the agent must consider its library of recipes 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 sequence of actions which can all be performed locally and which achieve the objective), whether it necessitates social activity (eg 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 (social) fashion.

If the agent decides to pursue the objective locally, it must ensure the intention is compatible with its existing intentions. 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 a different task from time 9 to time 14. GRATE* only considers temporal inconsistencies (see 5.3.3); however in the general case consistency could be defined with respect to any scarce or non-shareable resource (such as communication bandwidth, environmental states, physical space and so on). If there are no inconsistencies, the new objective is added to the list of individual intentions and the agent commits itself to performing it so long as it remains rational to do so.

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 (see 5.3.3 for a description of this process in GRATE*). An important consideration in such situations is the agent’s preferences or desires. If the new task is less important (desirable) than existing ones, then it should be the one which is 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.

If as a consequence of its means-end reasoning the agent decides the new objective is best met collaboratively, then a social action must be established. In this context, establishment of a social action means that a group of agents have to agree to work together to achieve a common objective and that for the duration of this activity they will obey the responsibility code of conduct. The means by which this is achieved is not specified at this level. So it may be possible for one agent to dictate to the others that they should participate (i.e. a hierarchically controlled system) or it may require a stage of negotiation 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 a hierarchical? and will the solution be decided by one agent or a group of them?), the strategy for group formation (eg large or small groups?) and constraints on group membership (eg everybody must be able to contribute). Such considerations are important even at this early stage, because they may affect an agent's willingness to participate. For example, an autonomous agent may not wish to participate unless the solution is developed collaboratively or agent X may not wish to be involved if agent Y is. Once the social action has been established, a joint intention exists and an appropriate representation is set up (see figure 5.4 for an example from GRATE*).

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 actions. However there is a causal link between individual and joint intentions - 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 10 to 20, this must be consistent with any of the related actions in the joint intention. Coherency between individual and joint intentions is ensured by the consistency checker. This process 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.

Once joint intentions have been established they must be mapped into actions which individuals within the group can undertake. This process is carried out bearing in mind the behaviour influencing factors such as constraints, desired organisation and so on which were discussed earlier. The specification of individual acts may be done in an incremental fashion (eg a social action of five agents may be split into two social acts with two and three agents respectively) 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 task 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. 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.

As well as providing a means for coordinating actions, intentions also act as a guide for task execution. It is the responsibility of the control execution 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 - provided it is still rational to do so. The controller may also suspend or abort tasks or reschedule them for completion at a later time. This process 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. This component is the only one which interacts with the domain level system.

The final functional role suggested by the responsibility model is that of monitoring commitments. This is necessary because intentions are not irrevocable. In the time between a commitment being made and its execution, the environment or the agents beliefs may have radically altered; meaning that it may no longer be rational to perform the action. In order to ascertain when such conditions arise this process receives feedback from the task execution controller and also any other events, either local or from elsewhere in the community, deemed to influence the situation. The process can influence task execution directly, via the control execution process, or indirectly through the intention representation component (eg change the start time from time 10 to time 20).

5.2.2 Tracking the Execution of Joint Problem Solving

The second level at which the responsibility model can be used is in the design and implementation of the joint action tracking and monitoring component. The conceptualisation of figure 5.2 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.

Extant DAI systems do not have such an explicit representation, their monitoring is *ad hoc* with each and every case having to be discovered and enumerated separately, rather than having broad and structured categories such as “plan invalid” and “goal motivation lacking”. In terms of figure 5.2, the *ad hoc* approach represents a direct link between the matching of environmental conditions and associated actions - there is no principled model of why the actions are taken nor what alternatives are available. Some sample rules for monitoring joint problem solving which do **not** embody a principled representation are given below.

```
R1:  if executing a plan with input i and
        receive more recent value for i
     then restart the plan with more recent value

R2:  if task in recipe does not produce expected result
     then select another recipe to produce result
```

```
R3:  if detect that fault is transient
     then inform AAA and BAI agents
```

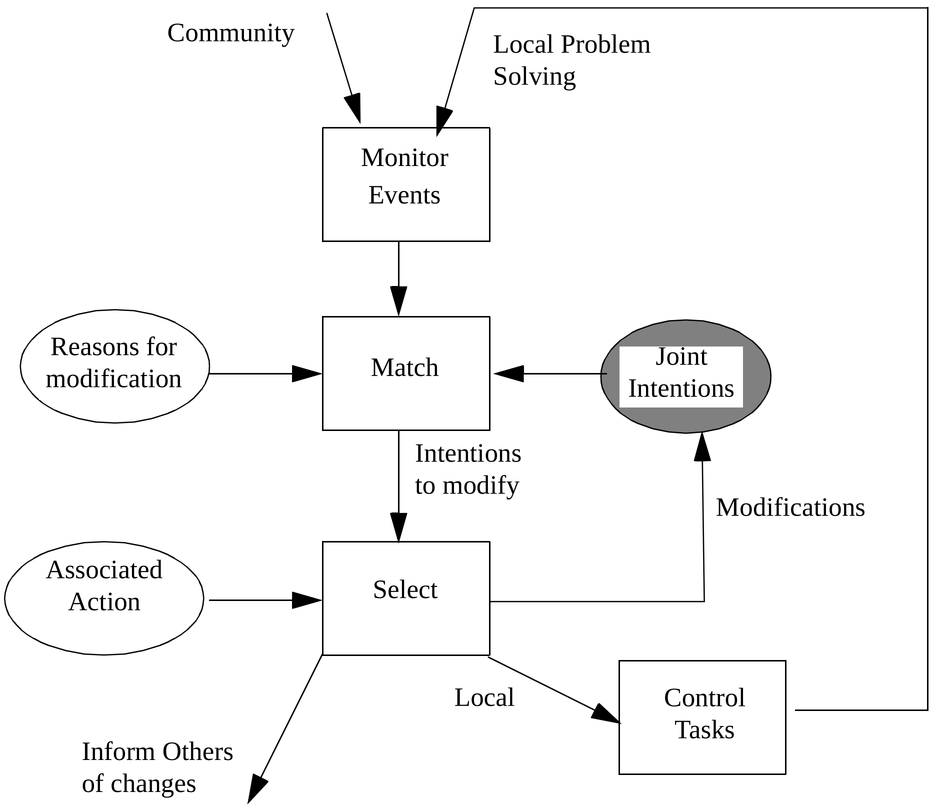

Figure 5.2: Schemata for Monitoring Joint Problem Solving

The principle underlying R1 is that subsequent information related to variable i has emerged which invalidates execution of the plan with the old value; executing the plan does not produce the desired or best result. As the plan is invalid, the agent’s commitment to it must be re-examined; the chosen remedial action is to restart it with the more recent data. R2 also deals with invalid plans, since plan execution did not produce the expected result. In this case, the remedy is to see whether the action can be redone. Finally R3 could be embodied in the CSI agent of the electricity transport scenario. The knowledge underpinning the rule is that the joint action of diagnosing faults assumes there is actually a fault to detect. If the CSI comes to believe this is no longer the case then the motivation for the diagnosis is no longer present, hence all team members engaged in this activity should be informed so that they can cease their associated activities.

All three rules exhibit the characteristics of first-generation expert system knowledge, they encode surface knowledge which is predominantly associational (de Kleer, 1984). They embody just enough knowledge to have the effect of the required inference, but nothing of the underlying domain knowledge. There is no explicit model of collaborative problem solving to which references for failure can be linked. Rather the rules represent a straight association between an agent’s problem solving state and its actions; the knowledge of collaborative problem solving on which such decisions are based is lost.

Following the approach advocated in figure 5.2, the application builder has to write the match process. This involves identifying those events which cause any of the reasons for joint action failure identified by the responsibility model to happen (see figure 5.3). For example the responsibility model stipulates that the agent must detect when the commonly agreed solution is no longer valid (i.e. following the solution does not produce the desired objective). The system designer has to identify all the events which may lead to this situation in the application being built and then encode them. Examples of such events include the fact that the initial assumptions were invalid, that one of the recipe actions does not produce the expected result and that the relationship between two actions needs to be stricter (eg SIMULTANEOUSLY rather than PARALLEL).

Using the responsibility model as a guide, the task of building the monitoring process is structured and simplified. The model specifies all the situations which can cause the collaborative act to falter. The application designer then has to identify the domain-level events which led to the reasons for failure occurring. Adopting this approach also ensures it is easier to produce an implementation which is able to cope with a wider spectrum of eventualities, since the decision making process is explicit and also structured into the categories prescribed by the responsibility model. Software reusability is enhanced because the domain independent component of the model can be re-used in many different scenarios.

In addition to aiding the process of constructing multi-agent systems, the explicit model of joint action tracking offers better explanation facilities than surface level systems. For example rather than stating that the joint action terminated because one of the agent’s initial assumptions proved to be incorrect; the agent can respond with the fact that the commonly agreed solution is no longer deemed likely to give the desired result since one of its initial premises turned out to be inappropriate.

Once the match process has identified joint intentions which need to be re-examined, the agent must decide which remedial actions, if any, must 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 becomes uncommitted 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. There is a link between the high level descriptions of reasons for terminating joint action and the associated actions which need to be performed.

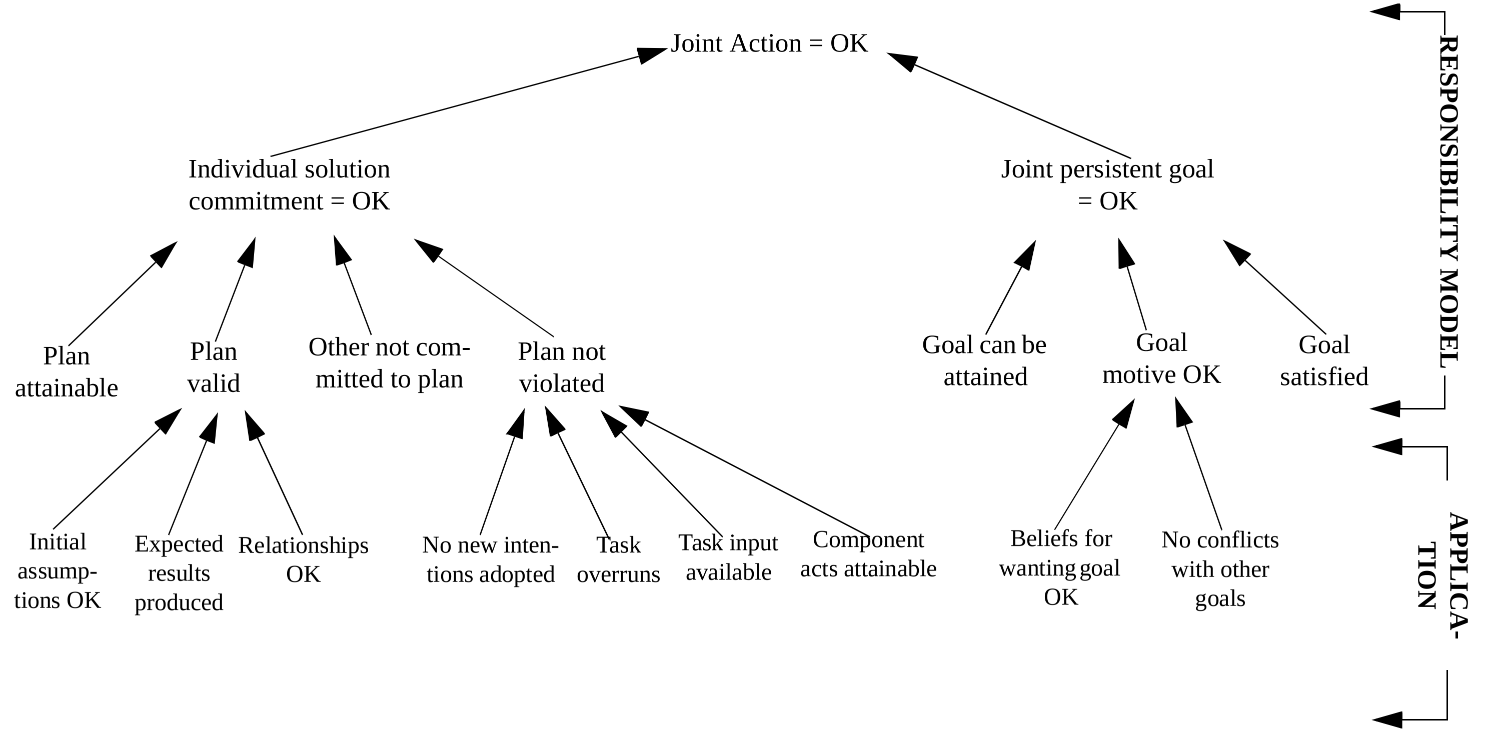
RESPONSIBILITY MODEL
APPLICA-TION
Joint Action = OK
Individual solution commitment = OK
Joint persistent goal = OK
Plan attainable
Plan valid
Other not committed to plan
Plan not violated
Goal can be attained
Goal motive OK
Goal satisfied
Initial assumptions OK
Expected results produced
Relationships OK
No new intentions adopted
Task overruns
Task input available
Component acts attainable
Beliefs for wanting goal OK
No conflicts with other goals

Figure 5.3: Deep Model of Collaborative Problem Solving in form of causal network

Note that because of space limitations only some of the application designer level tasks are filled in

Again this simplifies the application builder’s task and enforces the rigour of a sound theoretical grounding for agents’ actions.

5.3 The GRATE* Implementation

Any system which adheres to the functional architecture and specification of behaviour of figures 5.1 and 5.2 for collaborative action can be viewed as an implementation of the responsibility model. Whether the system uses message passing or blackboards for communication; has a centralised or distributed planner or has benevolent or autonomous agents is left as a design option. Hence there are a number of realisations which can be reverse engineered using this model, GRATE* is just one example of an explicit implementation of it.

The broad requirements of the responsibility model can be mapped into the architecture and separation of concerns used in the GRATE system (see section 2.4). That is, an agent architecture with three main problem solving modules - cooperation, control and situation assessment - which utilise acquaintance and self models for data and knowledge representations. Although the basic architectures are identical, implementing the responsibility model required several extensions to the original system. Firstly, the rules of the situation assessment and cooperation modules were completely rewritten; a specification of the new rules is given in appendix B. GRATE’s *ad hoc* rules were replaced with those which encode the responsibility model. Thus, an agent monitors the situation with respect to the conditions for dropping commitment, ensures any new commitments are consistent with existing ones and so on. Secondly the agent models had to be extended, so that they could represent joint intentions (see figure 5.4), the notion of commitment and an agent’s preferences. To indicate these significant enhancements, the new system is called GRATE*. The following description of GRATE* is broken into three parts; indicating how the prerequisites for joint problem solving are met, how the monitoring phase is implemented and finally how commitments are honoured and consistency maintained.

5.3.1 Prerequisites for Joint Problem Solving

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 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, determining when the actions will be performed and matching the team members and the actions to 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 situation assessment module of the organiser agent. As figure 5.1 illustrates, the organiser’s first concern is to determine whether it will be able to complete the act alone or whether it requires help from others. The organiser takes this decision by computing the number of tasks which it 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 but not sufficient to warrant a full-scale joint action. If n is the number of actions for which assistance is required, R the recipe and m a threshold value set by the system designer[2], then:

- $n = 0$: R is purely local
- $1 \leq n \leq m$: R is predominantly local, but it will generate some result or task sharing interactions.

  The rationale for having this intermediate level is that it may not be worth incurring the full overhead of distributed planning for just n actions, where n is less than the threshold value m. As is indicated later in this section, simpler forms of social interaction can make use of some aspects of responsibility without incurring the overhead of distributed planning.

- $n > m$: R should be solved using joint action.

  The full responsibility protocol and distributed planning mechanisms must be invoked because significant assistance is needed from other community members.

Once the need for joint action has been ascertained, the responsibility model requires the following conditions to be fulfilled before a social action can commence; other community members who are willing to participate and are able to contribute something must be identified, the fact that a common solution is required needs to be acknowledged, participants must agree to obey the responsibility code of conduct and finally the common solution by which the group’s objective will be attained must be developed.

Rather than having a protracted protocol in which each of the joint action pre-conditions 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 the responsibility pre-conditions. The first phase establishes 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 concrete basis for the discussions of this section, the concepts will be illustrated using a slight adaptation of the electricity transport scenario of section two. The CSI will detect faults and the AAA and BAI will cooperatively diagnose them as before. However in this, and subsequent, cases the CSI will also monitor the state of the network after a fault has been indicated to ensure it does not change so radically that the diagnosis

[2]. The setting of m will depend on the individual application designer and can be set to best reflect the fine tuning for a particular domain or problem.

process is invalidated.

After establishing the desirability, the organiser instantiates a representation of the social action (joint intention) in its self model (see figure 5.4). The motivation slot indicates the reason for carrying out the joint intention. Figure 5.4 shows only one motive, the fact that there is a fault in the network, but in practice there may be several such reasons (eg to honour an information or task request, because it is part of a larger plan, etc.). The recipe is a series of actions which need to be performed together with some temporal, partial ordering constraints which will produce the desired outcome. It 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 unfilled at this stage - such issues are only finalised in the second phase of the protocol.

```
Name: (DIAGNOSE-FAULT)
Motivation: ((FAULT-IN-NETWORK))
Recipe[3]:
   (((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) ?))
```

3. The plan directives START and START-OFF define relationships between actions. Using the notation of the previous section, the recipe $\Sigma_{\text{diagnose-fault}}$ would be expressed as follows:
{ *PARALLEL*(IDENTIFY-INITIAL-BOA, GENERATE-TENTATIVE-DIAGNOSIS, MONITOR-DISTURBANCE),
AFTER(GENERATE-TENTATIVE-DIAGNOSIS, PERFORM-FINAL-HYPOTHESIS-DIAGNOSIS)}

The synchronisation based on the black out area being supplied by the BAI agent (see figure 2.6) is carried out in the perform final diagnosis action.

```
        (CSI ((MONITOR-DISTURBANCE) ?)))
```

Figure 5.4: Representation of Social Action (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. In figure 5.4 the intrinsic priority of the diagnose fault recipe is 10 (see section 2.3.5) - this value is fixed by the user and is persistent. The second component is used to provide flexibility and takes into account the agent's dynamic problem solving state (eg whether it is working on a local action, a joint action or a social request). Each distinct type of action is assigned a value by the system developer (eg local act = 5, joint action = 10, social request = 2) which is then combined with the recipe's 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 (eg part of a joint goal and part of a social action request from a separate agent), 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). Agents have a common frame of reference with respect to priority. If an agent requests a task which it rates as intrinsic value 10, the recipient, if it undertakes the task, will use the value supplied even if it places a different worth on the task.

The status slot of the joint action description refers to the current phase of the distributed planning protocol and has 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 action, there is one team organiser (AAA) and two other potential team members (BAI and CSI). The AAA has agreed that it will participate in the collective act, 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 and so is not discussed here.

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 wishes 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, see figure 5.5, is to ascertain which acquaintances are ready and able to participate. Firstly the organiser's situation assessment module informs its cooperation module that a social act is required (action 1).

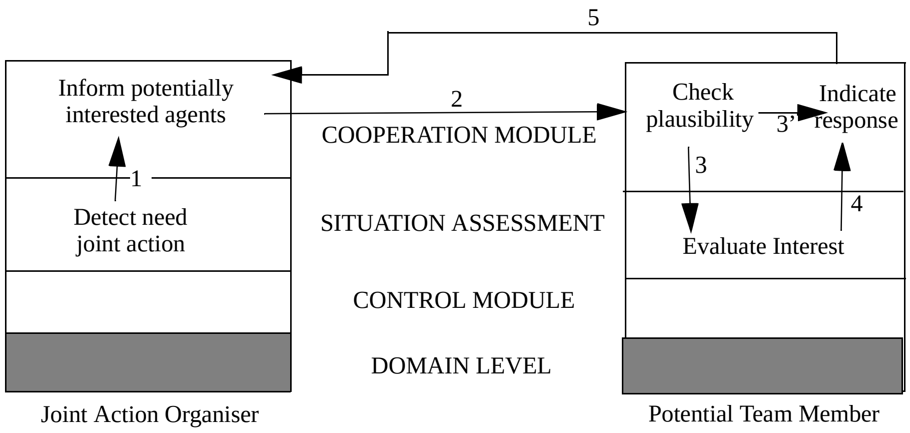

Figure 5.5: Establishing a Joint Action

The cooperation module identifies those acquaintances who it believes can contribute to actions which are part of the 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 message requesting their participation in the joint action (action 2). This message, see figure 5.6, indicates that the sender wishes to establish a joint action involving the recipient, states the reason why this joint action is needed, the other agents who are being asked to participate, the team organiser's priority for the task and its expected outcome.

Sender: AAA

Receiver: CSI

Type: PROPOSE-GROUP-FOR-JOINT-ACTION

Contents:

```
( (NAME (JOINT-ACTION DIAGNOSE-FAULT))
  (MOTIVATION (SATISFY-GOAL (DIAGNOSE-FAULT) AAA))
  (OTHER-PARTICIPANTS BAI)
  (PRIORITY . 20)
  (OUTCOME VALIDATED-FAULT-HYPOTHESES)              )
```

Figure 5.6: Group Proposal Message

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 cooperation 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 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 5.3.3. The result of this evaluation, either yes or no, is returned to the cooperation module (action 4) which returns the answer (see figure 5.7) to the organiser (action 5). If the agent does wish to participate, it sets up a joint intention description in its self model, similar to that described in figure 5.4.

Sender: BAI
Receiver: AAA
Type: GROUP-INITIATION-RESPONSE
Contents: ((JOINT-ACTION DIAGNOSE-FAULT)
(RESPONSE YES))

Figure 5.7: Group Proposal Response Message

By answering affirmative to the group initiation response, the agent tacitly acknowledges that it is interested in participating in the collaborative fault diagnosis, that a common solution is needed to solve the problem and that it will obey the responsibility code of conduct 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 first phase of the protocol satisfies the constraint that only those agents who, in the opinion of the team organiser, are potentially able to contribute something useful are admitted to the group. This is achieved by the organiser only contacting a chosen subset of the community about specific activities. This approach works well in situations in which 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 (Hewitt, 1985) 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. The organiser updates the joint action's status slot to developing-solution and modifies the participants slot to reflect the fact that participants are now in the solution planning phase. The purpose of this phase is to agree upon the fine details of the common solution, including the exact action timings.

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, and also because the coordination overhead is reduced 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 team leader then makes an initial proposal for the timings of the individual actions, expressed in the bindings slot, and fills in the joint action’s duration and its start and end times.

Bindings: (

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

The timing proposal takes into account the fact that some time is required to agree the solution, work cannot commence immediately. The formula for calculating the time lag is given below. It takes the following factors into consideration; for each action which needs to be performed by an acquaintance at least two messages must be sent to establish its start time; an agent takes time to process a message and that an acknowledgment message must be sent to all community members when the final timings are agreed upon.

Start-Time = Current-Time +
2 * (Number-Non-Local-Acts * Communication-Delay) +
(Number Actions * Estimated Time to Process Message) +
Communication Delay

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 (Simon, 1957). 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 behaviour 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 and the AAA's rating of the action's desirability. Then the AAA is contacted about the generate tentative diagnosis task and a time agreed, and so on for each of the actions. The proposals sent to each contributor indicate the action to be performed, the time at which it should be started and provides the team leader's current proposal for its context (see figure 5.8).

Sender: AAA

Receiver: BAI

Type: CONTRIBUTION-PROPOSAL

Contents:

```
    ((JOINT-ACTION DIAGNOSE-FAULT)
    (CONTRIBUTION IDENTIFY-INITIAL-BOA)
    (CONTRIBUTION-TIME . 19)
    (INITIAL-PROPOSAL (BAI (IDENTIFY-INITIAL-BOA) 19)
                      (AAA (GENERATE-TENTATIVE-DIAGNOSIS) 20)
                      (CSI (MONITOR-DISTURBANCE) 21)
                      (AAA (PERFORM-FINAL-HYPOTHESIS-DIAGNOSIS) 36)))
```

Figure 5.8: Contribution Proposal Message

Upon receipt of the proposal, the team member evaluates it to see whether it is acceptable; refer to section 5.3.3 for further details of this process. If there is no conflict, the agent adopts the intention and sets up an individual intention description[4] in its self model (see figure 5.9). It then returns a message indicating its acceptance to the team organiser (see figure 5.10).

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 (20/4)
Status: PENDING	**Outcome**: (Black-Out-Area)

Figure 5.9: Individual Intention Representation for BAI Agent

Sender: BAI

Receiver: AAA

Type: CONTRIBUTION-PROPOSAL-RESPONSE

Contents: ((JOINT-ACTION DIAGNOSE-FAULT)
(CONTRIBUTION IDENTIFY-INITIAL-BOA)
(CONTRIBUTION-TIME . 19)

[4]. As illustrated in section 2.3.5, 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.

```
(RESPONSE OK))
```

Figure 5.10: Contribution Proposal Acceptance Message

If the suggested time is unacceptable the prospective team member proposes a 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 (see figure 5.11). If the modified time is acceptable to the organiser, it will make appropriate adjustments to the solution bindings and proceed with the next action. If the modified time proposal is unacceptable, the organiser will look for a new actor to perform the action from its list of proposed contributors.

```
Sender: BAI
Receiver: AAA
Type: CONTRIBUTION-PROPOSAL-RESPONSE
Contents: ( (JOINT-ACTION DIAGNOSE-FAULT)
            (CONTRIBUTION IDENTIFY-INITIAL-BOA)
            (CONTRIBUTION-TIME . 22)
            (RESPONSE NOT-OK))
```

Figure 5.11: Modified Proposal Message

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 informs all the team members of the final solution (see figure 5.12) - the joint action is now 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.

```
Sender: AAA
Receiver: CSI
Type: JACT-SOLUTION-AGREED
Contents:
  (  (JOINT-ACTION DIAGNOSE-FAULT)
     (SOLUTION   (BAI (IDENTIFY-INITIAL-BOA) 19)
                 (SELF (GENERATE-TENTATIVE-DIAGNOSIS) 20)
                 (CSI (MONITOR-DISTURBANCE) 21)
                 (SELF (PERFORM-FINAL-HYPOTHESIS-DIAGNOSIS) 36)))
```

Figure 5.12: Notification of Start of Joint Action

To summarise, the GRATE* distributed planning algorithm can be stated in the following manner.

```
Detect need for Joint Action to achieve recipe R
Forall actions in R do
  begin
      select agent A to carry out action $\alpha$
                                        (criteria: minimize number group members)
      calculate time ($t_\alpha$) for $\alpha$ to be performed based on temporal orderings of R
      and the time lag to agree the common solution
      send ($\alpha$, $t_\alpha$) proposal to recipient A
      A evaluates proposal against existing commitments (C's):
        if no-conflicts ($\alpha$, $t_\alpha$) then create commitment $C_\alpha$ for A to ($\alpha$, $t_\alpha$)
        if conflicts(($\alpha$, $t_\alpha$), C) $\wedge$ priority($\alpha$) > priority(C)
          then create commitment $C_\alpha$ for A to ($\alpha$, $t_\alpha$) and reschedule C
        if conflicts(($\alpha$, $t_\alpha$), C) $\wedge$ priority($\alpha$) < priority(C)
          then find free time ($t_\alpha + \Delta t_\alpha$), note commitment $C_\alpha$ and return
               updated time to leader
        Return acceptance or modified time to team organiser
      if time proposal modified, then update remaining action times by $\Delta t_\alpha$
  end-forall
```

Figure 5.13: GRATE* Distributed Planning Algorithm

5.3.2 Monitoring Joint Action

Once a joint action has been established, the tracking aspect of the responsibility model comes to the fore. As depicted in figure 5.2, there are two main components of the tracking. The first is the match process in which reasons for terminating joint action are identified by the situation assessment module (i.e. the application specific part of figure 5.3 is constructed). Such circumstances may arise as a result of local problem solving:

```
 if  a local task has finished execution and
     it has produced the desired 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 (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 respectively). 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 joint action failure against the possible courses of action and picking the most appropriate remedial action. 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 that the joint action has
      successfully finished
      suspend all associated local activities

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

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

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.

```
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 this fact and also of
      remedial actions undertaken
```

The joint responsibility model can also be used for tracking simpler forms of social interaction such as an agent requesting a task to be performed or asking for a particular piece of information to be supplied. With these forms of social interaction it may not be worth incurring the full distributed planning protocol overhead, but there are benefits to be accrued from using a principled tracking mechanism.

A simple form of joint action can be established by an agent making a request, say for task t, to an acquaintance which accepts it. In this case, the joint action is composed of one action (task t) and involves two agents (the agent originating the request and the acquaintance accepting it). The commonly agreed solution is that the acquaintance performs task t. The contribution constraint needs to be relaxed in this case because the requesting agent is doing nothing towards the overall objective.

The acquaintance will eventually execute task t and can undertake the tracking functions associated with a full blown joint action. If t is successfully completed, this fact will be detected by the situation assessment module and will be passed back to the originating agent, via the cooperation module, using the following rule.

```
if objective of joint action has been satisfied
then  inform cooperation module that the joint action has
      successfully finished
      suspend all associated local activities
```

A rule which has a similar effect was also present in the original version of GRATE. This rule was as follows:

```
if   an agent finishes a task and
     the task was done because an acquaintance requested it
then  inform the acquaintance that the task has finished and of
      the results produced.
```

Although serving the same purpose, the two rules are very different in nature. The first was written because the responsibility model stipulates that when a joint action has successfully completed, the agent will become uncommitted and so must inform all of its fellow team members. It is part of a principled model of collaborative problem solving. However the GRATE rule makes no reference to a model of collaborative problem solving; it was included merely because the designer’s intuitions or experience indicated that it would be useful.

This is a clear illustration of one of the major advantages of the responsibility model and of cooperation knowledge level solutions in general - the designer’s decision making process is better structured. Hence building systems is easier since the high level interactions are prescribed by the responsibility model (as shown in figure 5.3); also the chance of missing a major class of activities is substantially reduced.

If something does go wrong with the joint action and task t cannot be completed as agreed, the responsibility model ensures that the originating agent is informed at the earliest opportunity. Remedial action can then be decided upon.

5.3.3 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 (see figure 5.1). 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 5.14. They are consistent because the times at which they are to be carried out, 19-29 and 38-44, do not overlap.

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)

Figure 5.14: BAI’s Existing Intentions

Assume a new intention, say identify the initial black out area, is proposed by the BAI’s situation assessment module (see figure 5.15). The situation assessment’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)

Figure 5.15: 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 forms the intention to identify the initial black out area from time 20 to 35; this means the intention to check the restoration premises has been violated and must be modified.

The inconsistency resolver process will propose the means to remedy 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; 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 producing the black out area is more desirable - priority value 6 versus 3. Thus it commits itself, all other things being equal, to check the restoration premises from time 36 to time 46. The updating network model 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 is no conflict 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)

Figure 5.16: Final Consistent Intentions

5.4. Conclusions

This chapter has highlighted the needs and benefits for a new computer level. The cooperation knowledge level sits above the individual knowledge level and makes use of its components. The cooperation knowledge level differs from the individual knowledge level in that it describes the actions of collectives, not just individuals, and also the notion of individual rationality is insufficient for describing the behaviour of groups. The responsibility model can be viewed as defining one aspect of this new computer level. It describes the actions of groups in terms of social actions and defines one aspect of team rationality; namely how to behave when things do not go according to plan.

The functional architecture suggested by the responsibility model was then outlined. This description identifies the key processes and knowledge required to support the model. It is pitched at a suitably high level to leave the application designer with sufficient leeway to implement the concepts as he sees fit. Hence GRATE* is but one realisation of the model. Special consideration is given to the architectural component which tracks joint actions. A principled model of tracking is described which makes reference to an explicit representation of collaborative problem solving.

GRATE* is loosely based on the earlier GRATE system, but with enhancements to the problem solving modules and to the knowledge representation components. The mappings with the high level architecture and the principled model of collaborative problem solving suggested by the responsibility model are discussed and the process for maintaining consistent intentions and honouring commitments is outlined.

GRATE* was also designed with the idea of controlled experimentation in mind - thus it embodies some features which are typical of experimental testbeds (see section 2.1). All the asynchronous operations and communications of GRATE were replaced by a discrete event simulator. This allows communication delays to be deterministic, a limit to be placed on the number of GRATE* rules which can be fired at any time instant (in order to vary the level of resource boundedness), events to occur at predictable times and so on. Discrete timing also simplifies the reasoning about time which needs to be undertaken by the agents in order to coordinate their actions.

6. EXPERIMENTAL EVALUATION OF THE RESPONSIBILITY MODEL

This chapter reports on an empirical investigation of the model of joint responsibility and its implementation in GRATE*. The main benefits of the model are stated as testable hypotheses and then a series of controlled experiments are undertaken in order to evaluate the claims. The scenario used to illustrate the concepts is the now familiar electricity transport domain, although the results obtained are believed to be generalisable.

6.1 Organisational Forms and some Conjunctures

Monitoring the activities and progress of group members is an essential aspect of all forms of joint problem solving, but it becomes especially pertinent if the domain is subject to frequent changes or agents have to take decisions using limited and possibly unreliable information. In such instances, it is difficult to keep a group of agents acting together in a coherent manner because individuals' beliefs and desires are constantly changing as more information becomes available. One of the primary motivations for developing the responsibility model was to help retain group coherence in such complex and dynamic environments; it provides criteria which need to be checked for and prescribes how to behave in both nominal and exceptional circumstances.

To evaluate the effectiveness of the responsibility model, a series of controlled experiments were undertaken; the objective of which was to test the following hypotheses.

(1) The responsibility model facilitates the coherent collaboration of groups of agents in complex and dynamic environments.

(2) The responsibility model's utility is correspondingly greater in domains in which there is significant scope for foul-ups and inconsistencies during joint action.

Manifestations of this type of domain include a high loading of tasks for each agent, new activities emerging at unpredictable times, interactions lasting over a prolonged period of time and agents having to base decisions on poor quality information.

(3) The responsibility model is implementable in a manner which does not compromise the agent's ability to operate in environments where the resources available for coordination are finite.

This is important because the aim is to use the model to control agents in real world environments. In such situations it is impractical for the resources required by the coordination mechanism to significantly dwarf those needed for the domain level system. The rationale being that the domain level system is actually carrying out the desired work, the coordination mechanisms are merely organising it.

These conjectures were tested on the cooperative fault diagnosis scenario for electricity transport management. The agents involved were the AAA, BAI and CSI; each was given an equal priority (time slice) and could carry out identical amounts of reasoning about the process of coordination. Three different types of agent community were constructed in order to carry out comparisons. They were all implemented in GRATE*; but varied in their inbuilt knowledge, to give the desired problem solving characteristics, and the information contained in their agent models. The first community was organised using the model of *joint responsibility* and the GRATE* distributed planning protocol outlined in the previous chapter.

In the second type of community, agents interact with one another but they do not undertake explicit collaborative activity; the groups are *implicit*. Cooperative behaviour is seen as “emerging” (Steels, 1991) through interactions. Agents do not participate in a collaborative planning phase to decide who will perform which action and no *joint* intentions are formed. Agents do however form individual intentions, related to their own actions, and plan which local tasks to carry out at what times. Information exchange is based on knowledge contained in acquaintance models; in particular on representations of the domain level data which others could benefit from receiving. Recipes are triggered solely on the basis of events the local agent observes or local planning which it undertakes; not as a result of some prior agreement with others. From an external (behaviouristic) perspective, many of the interactions appear similar to those in the responsible community. The true difference only emerges when the mental states of the participants are examined. In the implicit organisation, agents only consider the ramifications of actions and events from a local perspective; they do not consider the effects of their actions on others nor do they reason about how others will react to their activities. When agents undertake social interactions they do so out of benevolence (eg information is sent because it is believed to be useful), not through a deep desire to coordinate their actions with others.

The final type of community is composed of *selfish problem solvers*. They are similar to the responsible ones in many ways; forming groups, developing joint intentions and agreeing solutions using the GRATE* distributed planning protocol. Differences only emerge when the joint action runs into difficulty. The agent who detects the problem stops its associated local processing immediately, drops commitment to any subsequent activities related to the joint action *but does not inform the others of its lack of commitment*. This is selfish because the agent who detects a problem and realises that the joint action is doomed to failure does not expend further resources informing others, since doing so brings it no direct benefit with respect to the doomed social action. Such agents can be viewed as using the principle of individual rationality (Newell, 1982) to guide their behaviour.

A number of different types of experiment were carried out to test the predictions about the responsibility model. Firstly, in sections 6.2 and 6.3, the effect of joint action unsustainability on the group of collaborative problem solvers was examined. Comparisons of the three organisation forms were undertaken. The objective of these tests was to provide an analysis of the level of coherence of the different types of community in correspondingly more complex and dynamic environments. The second set of experiments, section 6.4, assessed the effects of varying the time taken for interagent communication on the community's behaviour in both successful and unsuccessful joint problem solving. The final group of experiments, section 6.5, examined the effect of limiting an agent's reasoning about coordination. The aim being to gain an insight into the resource overheads required to sustain each of the organisational forms

6.2 Varying Chance of Error In Joint Activity

This set of experiments evaluate the effect, on the process of collaborative fault diagnosis, of events which cause the joint action to become untenable. The behaviour of each organisational form is described along with an indication of the level of coherence within the community when unsustainability occurs at various stages of interaction. All the reasons for joint action faltering identified by the responsibility model are presented.

Coherence is measured in terms of wasted effort; the number of processing cycles from when the event causing the problem occurs to the time when an agent stops performing domain-level actions associated with the collaborative activity. So, for example, if the CSI detects that a fault is transient at time 10 and the AAA stops its detailed diagnosis at time 15, after having spent 20 units of activity on it, the wasted effort is 5 (15 - 10). The figure does **not** include the effort spent before the action ran into problems (eg 20 units); up to this point it was perfectly rational to carry out the activity and hence the community was performing coherently.

The graphs' x-axis represent the time for which the joint action has been active when it falters - varying from the very instant the activity started (0) to the time it finished (68). Agents had infinite processing power to carry out their reasoning about coordination and the delay in interagent communication is one processing cycle.

6.2.1 Goal Motivation Lacking

The event which causes the overall goal motivation to be dropped is that the CSI realises the fault, which the group is working on, is only transient or never even existed. In such circumstances the AAA and BAI should abandon their diagnosis tasks since there is not really a fault in the network.

In the community organised using the responsibility model, the effort wasted by each agent exhibits a periodic pattern (figure 6.1). This is because the CSI can only detect that the fault is transient at the end of its monitoring task, an activity taking 7 units of time and which is repeated continuously until the list of faulty elements is available. So even if the information indicating that the fault is transient arrives at the CSI just after a cycle has started, it only realises the significance after it has completed its analysis for that cycle.

In all cases, the AAA wastes more effort than the CSI because it cannot detect the problem for itself. The difference between the two corresponds to the time taken for the message indicating that the fault was transient to be sent from the CSI to the AAA. The BAI's role in the joint action is comparatively small, it is only active for the first 15 time units. If a problem occurs in this time, the BAI relies on being informed by the CSI and so there is a time lag in stopping its activity, for the same reason as the AAA. However if the problem arises after the BAI's period of activity, it is unaffected because it is not actively carrying out processing related with the social goal. The whole joint action takes 68 cycles to complete and so any event causing unsustainability after this time means no wasted effort, since there is no associated processing taking place.

In this instance, the implicit group behaves identically to the responsible one from an external perspective (figure 6.2). This is because the CSI is aware, through its acquaintance models, that both the AAA and BAI are interested in the state of the electricity network when there is a change, since this is what triggers their diagnosis activities in the first place. Therefore if the CSI comes to believe that there is not a fault after all, network state change from faulty to normal, this fact is sent to the AAA and BAI as unrequested information. Within their local, individual intention representations, the AAA and BAI store the reason why they are carrying out their diagnosis activities (i.e. because there is a fault). Receipt of the CSI's message undermines this motivation and because they are rational agents they stop performing related activities.

The community of selfish problem solvers behaves differently from the other two organisational forms (figure 6.3). When the CSI realises the fault is transient it abandons its associated local processing, as before, since it is no longer rational for it to monitor the network for a non-existent fault. However as it is selfish, the CSI does not want to expend additional effort on the joint activity. So it fails to inform the others that there is not a fault in the network because sending the information would require it to undertake reasoning and hence use resources. The pattern of coherence is markedly different as a consequence. The effort wasted by the AAA and BAI is not periodic because they never realise the fault was transient, they just continue until their activities are finished. Therefore problems with the joint action which occur near the beginning have significantly greater impact on the community's coherence than those near the end.

6.2.2 Goal Unattainable

The overall goal becomes unattainable if the AAA realises that too much of the data which should be provided by the CSI is missing for it to make a diagnosis. As the AAA is the only one capable of pinpointing the element at fault, the overall objective cannot be met in such circumstances. This realisation may occur either in the AAA's preliminary or detailed diagnosis phase. As before, recognising that the results cannot be produced only occurs when the task completes, even if there is insufficient data available near the beginning.

With the responsibility organisation there can be a substantial amount of wasted effort, both by the AAA and CSI due to the time lag in detecting the problem (figure 6.4). This phenomena is exacerbated still further if the realisation occurs in the very long detailed diagnosis phase. The amount of effort wasted by the CSI, and the BAI for the initial period, lags behind that of the AAA because they both rely on being informed of the problem since they are unable to detect it for themselves. Because of the time taken for information to reach them, the CSI and BAI cannot stop their associated activities as early as the AAA.

In the implicit group organisation (figure 6.5), the external behaviour of the AAA is the same as with the responsibility model because it is the one who can detect the problem and is able to stop its associated processing. However it does not inform the other two; since no information has been generated and it is unaware that the others are interested in its local problem solving state. Therefore the CSI continues with its monitoring, oblivious to the fact that it is ultimately pointless. Originally the condition for the CSI terminating its monitoring activity was that the diagnosis had been completed. However in this case the CSI would loop forever as the condition would never be met. Therefore for practical purposes, a time-out facility of 80 units was introduced and it is this which terminates activity.

The behaviour of the selfish problem solvers is identical to that of the implicit group (figure 6.6). The reason being that in both cases no information is exchanged related to the overall objective’s unattainability; hence the activities of both the BAI and CSI are left to run their course, while the AAA who can detect the problem stops.

6.2.3 Plan Invalid

The event which causes the common plan to become invalid is that the CSI detects a critical change in the network topology. This means the diagnoses being undertaken by the AAA and BAI are based on invalid assumptions, their network models being flawed. As a consequence, following the agreed sequence of actions will not lead to the desired outcome of identifying the faulty network elements.

In the responsible community the pattern of wastage is again periodic, 7 time units, because of the time taken by the CSI agent to identify that a substantial change has occurred (figure 6.7). Whenever a critical change occurs the CSI always takes some time to identify it, hence the wasted effort is never zero. Again the AAA and BAI lag behind because they cannot detect the problem for themselves and so rely on being informed - the time required to send a message produces the time delay.

The behaviour of the implicit organisation mirrors that of the responsible one (figure 6.8). This is because the information which causes the AAA and BAI to detect a faltering of the joint action (i.e. a critical network change) is sent out as unrequested information by the CSI. It is spread by the CSI because it is of interest to the other community members; in fact it is a trigger for urgently updating their network model. The first act of this recipe is to kill all tasks which are making use of the outdated model. Therefore from the joint action perspective, the diagnosis activities of the AAA

and BAI are halted by local tasks for local reasons. From a behaviouristic viewpoint both communities appear the same, the CSI stops its activity because there is no point in monitoring the network using an incorrect model and an indication that a substantial change has occurred is sent to the AAA and BAI whereupon they stop their diagnoses. The difference occurs in the mental state of the CSI; in the responsible case it sends a message to the others because it knows that the joint action is unstable and it wants its fellow participants to be aware of this fact. In the implicit group, the CSI sends the same information as unrequested data, it does not know (or care) what the AAA and BAI will do with it.

In the selfish group (figure 6.9), the CSI behaves as before since it is able to detect when the joint action has gone awry. However it does not inform the others, since this would waste resources, and as they cannot detect the problem for themselves they continue in vain. The amount of effort wasted by the AAA can be substantial, that of the BAI is more restricted because of its limited participation in the social action. As with the previous selfish groups, there is a significant difference to coherence if the problem with the joint action occurs near the beginning or the end of the activity; the responsibility model produces periodic patterns of wastage, so such differences do not arise.

6.2.4 Plan Unattainable

The event which causes the common solution to become unattainable is that the system feeding information about the network to the CSI becomes damaged. In such instances, the CSI has no information on which to base its monitoring duties. In general, a component of a joint action solution becoming unattainable does not imply the overall objective is also unattainable. Since, for example, the agent may have an alternative means, another agent or procedure, of producing the necessary information. In this case, the AAA may decide to carry out a cooperative diagnosis with the BAI and not bother with having the network monitored in the hope that nothing important will change.

With the responsibility organisation (figure 6.10) the pattern of wasted effort is similar to that of several of the earlier interactions. The CSI detects the problem at the end of its looping cycle and the AAA and BAI (while it is still active) lag behind because of the communication delay involved in them being informed.

In the implicit group (figure 6.11), the behaviour of the CSI is identical to that of the responsibility model because it is the one who can detect the problem. However it does not inform the other two agents, since the problem manifests itself in terms of a problem solving state not in terms of information which can be sent as unrequested data. Typically reports of problem solving state are not exchanged by agents since they are too numerous. Therefore the AAA and BAI are left to continue with their activities, oblivious to the fact that they are wasting their time.

The selfish problem solvers are identical to the implicit group (figure 6.12). In both cases no information is exchanged related to the unattainability of the agreed

common solution. Therefore the agents which cannot detect the problem waste resources on fruitless activities.

6.2.5 Plan Violation

The plan violation experiments differ from the others in this subsection because the group of agents actually try to repair the problem which is causing the joint action to falter. The group then continues with the remedial action in place, rather than simply abandoning the social activity. The common plan may be violated if unanticipated events occur, during the course of the joint action, which cause an agent to be distracted from its commitments. For instance, the BAI may have agreed to calculate the initial black out area between times 10 and 25. However, after this commitment has been made, circumstances may require the agent to perform a higher priority (more desirable) task at this time, meaning the intention to calculate the black out area has to be put back. Thus the BAI has violated its commitment to produce the black out area by time 25.

In order to cause plan violation, varying numbers of new tasks were assigned to the agent community *during the lifetime of the joint action*. These additional tasks could all be solved locally, could be carried out without the need for coordination, and were all of the same relatively short duration (3 time units). The agent which has to undertake them is uniformly distributed over the community, as is their arrival time. Their priority is randomised, using the value of the joint action components as a mean. The results shown in figures 6.13 and 6.14 are averaged over 100 test runs.

To create a greater interdependence between the actions the scenario was also slightly modified. *This change applies to this set of experiments only*. The CSI’s task of monitoring the network is assumed to be dependent on the AAA’s initial approximate diagnosis. In terms of the scenario, this means the CSI’s monitoring is focused; concentrating on potential problem areas indicated by the AAA not the network as a whole. The task also becomes of deterministic duration rather than looping continuously. The relationship between the AAA and BAI was made firmer; in this case the black out area is a prerequisite for the detailed analysis. Until now there has been a weaker link in that the AAA could continue with its detailed diagnosis without the black out area (see figure 2.6), it would just take longer. This modified scenario was used to simulate the effects of plan violation on closely related actions. Therefore any differences between the responsibility and implicit organisations highlighted by these experiments would be less pronounced in the original scenario.

If a domain level task is running when the coordination mechanism decides it should be rescheduled, then it must be suspended before any new activity can start. When such tasks are restarted, they carry on from where they left off. For example, if a task takes 15 time units to execute and it is suspended after 5, upon restart it will require a further 10 (15 -5) units of processing to complete.

In the community guided by the responsibility model (figure 6.13), if an agent delays an action which is being performed in the context of a social goal then it

informs all the other team members because the common solution has been violated. The agent indicates the action and the time by which it has been delayed. When such a message is received, the recipient evaluates whether the change will affect its processing. So, for example, if the BAI delays its production of the black out area, the AAA’s detailed diagnosis will be adversely affected. If the delay is relevant, the agent reschedules its local activities ensuring that any relationships are upheld. It then sees if any of its other actions can be moved into the vacuum which has been created so that it does not lie idle during this period. It attempts to bring as many tasks forward as possible, they are selected in priority order until either the time has been allocated or no tasks will fit into the remaining time. Hence if the AAA decides it needs to delay its detailed diagnosis by 10 units; it attempts to bring forward one or more tasks which, in total, will take 10 or less units of time to complete. Care must be taken that other relationships are not broken during this rescheduling and so it is predominantly local tasks which are moved. If such rescheduling is impossible, the agent remains idle until its next intention becomes active.

In the implicit organisation (figure 6.14), when agents delay activities they do not inform their fellow team members, since it only reflects an internal change of problem solving state. The agent where the delay occurs, undertakes the type of rescheduling described above, but its fellow team members are unable to do so because they are not aware of any changes. This means agents may be waiting for substantial lengths of time for synchronization events to occur (eg the AAA for the black out area to be produced), unaware that the task has been put back. The time spent waiting is completely unproductive as during this time the agent is not performing any useful processing.

Comparison of figures 6.13 and 6.14 highlights the superior performance of the responsibility model along two important directions. Firstly, the delay in finishing the joint action[1] rises less steeply and to a lower value for the responsible organisation. Therefore by indicating when delays have (or will) occur, team members are able to make better use of their resources. Rather than simply idling until synchronization events occur, responsible agents are able to intertwine other activities into spare moments when they know acquaintances have delayed actions. Agents in the implicit group are unaware of such delays and so suffer prolonged periods of inactivity.

Secondly the average effort expended on the joint action rises more steeply for the implicit organisation. In particular the differences between an agent which has an action dependent on another (eg the CSI) and an agent which has not (eg the BAI) are highlighted. In this case both the BAI’s and the CSI’s tasks actually take the same time to execute, but in the implicit organisation the effort spent by the CSI is considerably greater than that of the BAI. The amount of effort spent by the AAA also rises at a greater rate for the implicit organisation because its detailed diagnosis is dependent on the BAI producing the black out area. The responsible community, on the otherhand, remains virtually impervious to delays and rescheduling activities. The effort

[1]. Calculated from the time at which the joint action would finish if there were no interruptions; the time spent executing the additional tasks is subtracted from this figure.

expended by each agent varies very little and for the community as a whole it is significantly less than that of the implicit group. These results mean that in the implicit community the joint action is taking longer to carry out and is more costly, because team members are spending significant amounts of time in an unprofitable manner (eg waiting for events and information from a team member who has put back some actions).

In both organisational forms the BAI’s effort is virtually constant. The explanation for this is that producing the black out area is not dependent upon the actions of others; it is related to the task of producing the detailed diagnosis but is unaffected by delays in the processing of others. However both the CSI and AAA are dependent on activities performed by others; the CSI on the AAA’s approximate diagnosis and the AAA on the BAI’s black out area. Therefore both of them are more adversely affected in the implicit groups, because they are spending time waiting for synchronisation events to occur. The responsibility model ensures all team members are made aware of any relevant changes and hence both agents perform much better.

6.3 Comparison of Organisational Forms

This set of experiments compare the performance of problem solving communities, organised according to the three forms described earlier, when there are varying chances of the collaborative action becoming unsustainable. Environments tested range from those in which it is guaranteed that the joint action will not falter (0% chance of problem) to those in which a problem is assured (100% chance of problem).

An increased chance of unsustainability corresponds to a more dynamic environment or one in which decisions are based on less stable data. In all cases, the cause of the problem is uniformly distributed over the conditions identified by the responsibility model. The time at which the problem occurs is also uniformly spread over the joint action’s duration. The wasted effort on one run is summed over all the agents in the team. So if the AAA wastes 2 units, the BAI 1 and the CSI 10, the total wasted effort is 13. Averages are produced by repeating runs 100 times. The communication delay is again one time unit and the agents have infinite processing power for coordination activities.

The community of problem solvers using the responsibility model as a guide has a steady rise in the amount of wasted effort as the chance of a problem occurring increases (figure 6.15). The effort wasted by the BAI is minimal, due to its limited participation, and that of the AAA and CSI are virtually identical. However the resource wastage is kept relatively small even in the most hostile environments. This provides a good indication that the responsibility model fulfills one of its main objectives of ensuring coherence in difficult environments.

With the implicit organisation, the amount of wasted resource also rises in line with the increasing chance of the joint act becoming unsustainable (figure 6.16). Again the BAI has negligible wasted effort. In contrast to the responsible organisation,

there is a greater divergence in the performance of the AAA and CSI, the latter being more expensive on average. The reason for this is that although the CSI can detect many of the problems with the joint action for itself it usually informs the AAA by sending it unrequested information. Based on this data, the AAA is able to conclude that the activity is no longer sustainable and stops its associated actions. Hence although the AAA will waste more effort, it will not waste significantly more. However when the AAA detects problems with the joint action, their manifestation is in terms of its local problem solving state (not results) and so is not communicated to others. In an important number of cases, the CSI is unaware of any problems and so wastes significant amounts of effort.

In the selfish problem solving community the opposite is true; the AAA performs worse than the CSI (figure 6.17). The reason being that it is the CSI who detects problems with the joint action in most circumstances. Under the selfish regime, the CSI merely stops its local activity and does not inform the others; hence the AAA carries on with ultimately unrewarding (wasteful) activities. So in most cases of unsustainability the amount of wasted effort per agent is of approximately the same order of magnitude; the dominant factor being the number of problems each agent is able to detect. Again the BAI has only a short amount of processing to do and so the amount of resource wasted is minimal; for the same reasons as the AAA it performs slightly better in the implicit group than with the selfish problem solvers.

Figure 6.18 summaries the overall performance of the communities for each organisational strategy. It shows the average wasted effort for the responsibility model is significantly less than the other two, confirming the hypothesis that it facilitates robust group behaviour in harsh environments. The implicit group performs better than the selfish one because agents exchange information, based on known interests stored in their acquaintance models, which can lead to the recipient realising that some of its intended actions are no longer appropriate and need to be abandoned. This informal interchange means that in the case of unsustainability, an agent which cannot detect a problem itself may inadvertently be supplied with sufficient information to enable it to drop commitments by an agent who can. In the selfish group such informal communication paths were deliberately avoided since calculating agents interested in a piece of information requires resources - thus agents which were unable to detect problems were left to complete (wastefully) all of their actions.

These results show that when claiming that self interest is the basis for cooperation (Durfee *et al.*, 1988b; Axelrod, 1984), it is important to note that it should ***not*** be used as a criteria for defining agent behaviour once the social action has started. Participation in group problem solving requires some element of compromise, meaning self interest needs to be tempered with consideration for the group as a whole. This is one of the reasons why, for cooperation knowledge level descriptions, individual rationality is an inappropriate law for governing the behaviour of agents and a notion of team rationality is required (see section 5.1).

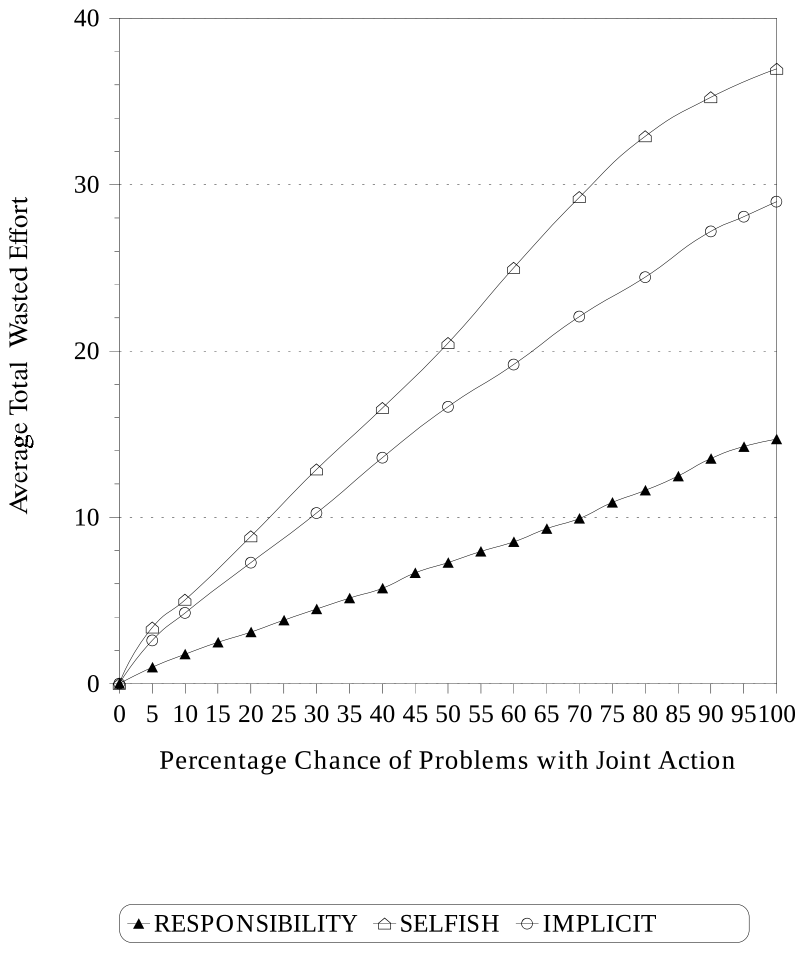

Figure 6.18: Varying Chance of Failure - Summarised Results

6.4 Varying the Delay in Inter-Agent Communication

The set of experiments outlined in this section, describe the effect of varying the communication delay for interagent message passing. Intra-agent communication between the problem solving modules is not affected. The influence of this parameter on both successful and unsuccessful joint actions is evaluated. Two organisational forms were compared; the responsibility model (figure 6.19) and the implicit group (figure 6.20). The selfish problem solving organisation was not considered because it is identical to the responsible one for successful joint problem solving. All agents have infinite processing resources for coordination activities.

In the responsibility organisation, the start time for each of the agents is agreed by all participants, hence it is the same for each team member. As such agreements are deliberately avoided by the implicit organisation, individuals start activities which are

part of the joint action at different times. The CSI always initiates the activity and this occurs at the same time (7), the AAA and BAI then follow when they receive the CSI's information - meaning they start their activities simultaneously. The difference between the starting time of the CSI and the other two is purely due to the message delay involved in indicating that a fault has arisen and the information being available to the AAA and BAI. It rises in proportion as the time delay increases from one time unit to fifteen. The responsibility framework always takes longer to start (19 versus 7 for the minimum delay, 131 versus 7 for a delay of 15). This is despite the fact that the events which initiate the diagnosis process occur at the same time in both simulations.

The reason for this disparity is that the responsible community constructs groups and common solutions afresh each time joint action is required. This process needs communication and is, therefore, more adversely affected by any delays; indicated by the sharply increasing joint action start time. In the implicit organisation, much of the coordination process is carried out by the application builder at design time. Hence many of the issues which need to be agreed at run time in the responsibility model are already predetermined and need not be re-examined. In the implicit model there is less to be settled at runtime and so there are reduced communication needs. The communication delay also has an impact upon the time at which the joint action ends. In both cases the BAI finishes before the other two agents since it has only a small amount of processing to undertake. The difference in end times between the CSI and AAA is due to the fact that the AAA detects that the joint action has been successfully completed and that it has to inform the CSI - thus it is due to communication.

Figures 6.21 and 6.22 show the effect on coherence of varying communication delays in the responsibility model when something goes awry with the joint action. The communication delays are shown in the legend in brackets against the CSI and AAA respectively. The BAI is not included because of its limited role. In both cases, the amount of wasted effort increases in line with the delay in interagent communication. This is because once the joint action becomes unsustainable the agent which detects the problem must ensure the others are aware of this fact, a state only achievable by communication. Thus the differences between the agents are solely as a consequence of communication, as the delay increases so will the effort wasted by the agent who relies on being informed of the problem. Similar patterns of wasted effort being proportional to communication delays would appear in the implicit organisation for those cases in which joint action is terminated. In the other cases, communication delays have no affect.

6.5 Varying the Amount of Processing Power for Coordination

The final set of experiments examine the effects of agents having to achieve coordinated activity with varying amounts of computational resource at their disposal. The aim being to compare the resource requirements of the responsibility model with those of the implicit group. Resource boundedness was simulated by setting a limit on the number of rules which could be fired by the agent's coordination mechanism on each processing cycle - it was varied from one rule per agent per cycle to an infinite

number. When an agent's limit was reached, no more rules were fired even if there were some which were waiting. The detailed algorithm for this process is given in appendix B.

Figure 6.23 shows the effect, on the agents' finishing times, of decreasing processing power from fifteen rules per agent per cycle down to one during successful joint problem solving. For these purposes 15 represents infinite processing power, since at no time during the simulation did any agent fire more in a single cycle. The results show that apart from severely resource bounded situations, less than 5 rules per time unit, there is very little difference between the amount of processing required for the responsibility model and that for the implicit organisation. With severe limitations, the responsibility model performs worse than the implicit organisation; meaning it requires slightly more processing power to operate effectively.

Figure 6.24 summarises this information in terms of the total delay, to the joint action compared with infinite processing power. The total delay for the responsible community rises more steeply and to a higher value than that of the implicit organisation. The graph also indicates the differences in starting times for the two agents remains virtually constant except for very resource limited situations, less than five rules per cycle. So taken together with the results of figure 6.23, the responsibility model does not require substantially more processing - either to maintain joint action once it has started or to initiate it than in the first place. This result was somewhat surprising since it was envisaged that the responsibility model would require significantly greater processing power that of the implicit group.

6.6 Discussion of Results

The aim of this empirical evaluation was to provide an insight into the behaviour of communities of cooperating agents which use the model of joint responsibility to guide their local and social actions. In order to make a comparative analysis, two other types of agent community were constructed; one in which the group interaction is implicit and one in which the agents behave in a selfish manner.

The experiments did **not** aim to produce absolute evaluations of the various aspects of the responsibility model. So, for example, it cannot be concluded that the problem of a goal becoming unattainable will lead to greater incoherence than if the reason for faltering was that the commonly agreed solution is no longer valid. Rather these experiments aim to identify underlying trends which would expect to be observed in other instances of collaborative action.

Relating the results of the experiments back to the initial conjectures it can be seen that.

> **(1) The community guided by the responsibility model performs either the same or in a more coherent manner for all types of faults than the other two organisational forms.**

The implicit group is sometimes able to match the performance of the responsible one, in cases where the manifestation of the joint action falter is in terms of data produced. In certain instances this information may be volunteered to acquaintances, but without the knowledge that it will enable them to halt their associated local problem solving.

(2) In situations in which there is a high chance of the joint action running into difficulties, the responsibility model is able to keep the amount of wasted resources to a minimum.

The other two organisational forms behave far worse in harsh environments; having a far steeper rise in the level of uncoordinated activity within the community.

(3) Except in situations where the amount of processing is severely limited, the responsibility model requires no more computational resources than the other two organisational forms.

(4) Communities using the GRATE* distributed planning protocol take longer to get started on the joint action and are more adversely affected by increases in communication time.

As well as highlighting the many benefits of the responsibility model, these experiments also indicate a potential drawback. The GRATE* distributed planning protocol is heavy on communication and it takes a significant amount of time to establish a joint action. Therefore it could be argued that the model is inappropriate for cases in which agents are situated in time-critical environments. However, this characteristic is not an inherent property of the model; rather it is indicative of the planning protocol used to implement the model. The GRATE* protocol built up all social actions from scratch, making no use of previous events or interactions. The group, the actions to be performed, the agent who will perform them and timings all have to be agreed upon at run time.

However as construction of the implicit community indicates, it is possible for the designer to make use of prior knowledge about the agents and their interactions to considerably shorten this process. Utilising such knowledge would bring about significant reductions in the communication overheads of the model and the time needed to establish joint actions. Whatever improvements are made it is unlikely that the delays could be reduced to the levels of the implicit groups, since they perform no run-time planning. Therefore when evaluating the responsibility model, the designer is faced with a tradeoff between the ability to ensure robust behaviour even in the most hostile environments and the overhead associated with constructing the group and agreeing a common solution. The choice will be dependent on the problem being tackled; if joint actions rarely run into difficulties then the overheads associated with the responsibility model may be prohibitive. If there is scope for dynamic change and unpredictability then, because of its ability to maintain coordination, the model of joint responsibility would seem a good choice.

7. CONCLUSIONS AND GENERAL DISCUSSION

This chapter summarises and draws some conclusions from the research described in this thesis. The main achievements are highlighted and the methodology used to guide this research is commented upon. Finally some areas for further work are presented; in particular one advance that combines generic rules with reactive mechanisms is described in detail.

7.1 Summary and Methodological Issues

This research started with the aim of constructing a general-purpose environment which would ease the task of building multi-agent system applications. This objective was attained by constructing a system which embodied significant amounts of inbuilt generic knowledge related to cooperation and control. The application designer then builds upon this pre-existing base of knowledge, rather than constructing the system completely from scratch. This approach is beneficial because many routine interactions can be encoded by the generic rules, enabling them to be viewed as high-level inbuilt services. In previous work, all interactions are coded afresh for each new application. Thus in many cases, designers continuously re-implement the same decision making process.

To help substantiate the claim that GRATE is indeed general-purpose and that the knowledge is generic; the system has been applied to three different industrial applications. Firstly, and most extensively, to the domain of electricity transport management (Jennings *et al.*, 1992). The other applications tested were cooperative diagnosis of a particle accelerator beam controller (Jennings *et al.*, sub) and detecting and dealing with focused overloads in telecommunications networks (Whitney, 1992). In all cases, the designer was able to construct a working multi-agent system in a relatively short space of time and did not need to augment the inbuilt knowledge. Whilst the applications share certain common characteristics, such as involving predominantly expert systems and being related to the task of diagnosis, they are sufficiently diverse to give credence to the claims that GRATE is general purpose and that its inbuilt knowledge is generic.

Although GRATE was successfully applied to these real-world domains, the experience gained during this process led to a fundamental rethink of the method of approach. GRATE communities did not cope well with unpredicted events or dynamically evolving situations. In many cases it was impossible to predict how individual agents or the community as a whole would react to certain events and as a consequence coherent behaviour was difficult to sustain. These experiences offer further support for the “difficult problem hypothesis” (Lenat and Feigenbaum, 1991). This states that systems should be tested on realistic problems rather than contrived scenarios because the latter often fail to highlight deficiencies. By applying GRATE to these demanding applications with their real world constraints, it became apparent that the GRATE approach was not sufficient; using abstract scenarios this observation may never have happened or it may have taken much longer to realise there was a problem.

GRATE’s poor performance and lack of predictability was attributed to the absence of an explicit model of collaborative problem solving on which its actions could be based. GRATE’s inbuilt knowledge was developed in an *ad hoc* manner and consequently had little structure and no means of ascertaining the variety of situations under which agents could operate successfully. Existing literature was then scanned to see if a suitable model of joint problem solving had already been proposed. This review identified several theoretical models; very few of which had been applied to realistic problems, let alone implemented. Also most of the implemented systems were developed in a similar way to GRATE; lacking a principled model of collaboration on which to base their behaviour. As a consequence of these findings, a new model of collaborative problem solving was required; one which could cross the great divide between model (theoretical) and system oriented approaches (Cohen, 1991) and which would be useful for controlling cooperation in industrial applications.

Joint responsibility was devised as a model of cooperative problem solving for an implementable multi-agent system; particular emphasis was placed on maintaining coherence in environments which are complex and dynamic. It is based on the notion of joint commitment and defines how individuals within a team should behave, in their local problem solving and towards others, when engaged in collaborative activities. It specifies how to behave when the joint action is progressing according to plan and when disruptive events occur.

The model of joint responsibility suggests a high-level architecture for social interaction based on joint intentions. The proposal outlined in chapter five clearly identifies the functional role of joint intentions and their relationship with individual intentions. It provides a specification of the process of tracking joint action which separates circumstances in which collaboration may be jeopardised from the choice of how to recover from such situations. It also prescribes how to behave towards other agents involved in the social act.

The responsibility model and its associated architecture proposal were then transformed into an updated version of GRATE called GRATE*. Unlike its predecessor, GRATE* is based on a principled model of collaborative problem solving. This means it is possible to make predictions about the behaviour of agents, to verify these hypotheses by empirical analysis and to clearly delimit the types of situation to which the agent can be guaranteed to respond. Building systems which have a firm theoretical grounding has been widely advocated in the AI community. Indeed the lack of both predictions and empirical analysis has been lamented as a major failing of current AI research and posited as one of the reasons for the discipline’s slow advancement (Cohen, 1991; Lenat and Feigenbaum, 1991).

As the joint responsibility model was designed to ensure groups of agents remain coordinated even in the most dynamic and unpredictable environments, it was hypothesised that GRATE* communities would exhibit high levels of coherence. To evaluate this claim, a series of comparative experiments were undertaken in which groups of collaborating agents were organised using different problem solving models. Three different forms were compared - the responsibility model, an implicit group model and groups of selfish problem solvers. As predicted, jointly responsible communities performed more coherently than the other two; this difference being especially noticeable as the domain became more dynamic and unpredictable (i.e. the chance of joint action unsustainability increased). This gain in performance was achieved with negligible extra processing requirements for the coordination mechanisms.

Several issues uncovered in the course of this research have not been adequately addressed in this thesis. In chapter two, the desirability and possibility of different levels of generic knowledge was proposed. Thus rather than having just two levels of knowledge, completely general and application specific as there are in both GRATE and GRATE*, there may be intermediate categories such as fault diagnosis or industrial applications. However the means of determining what these other levels should be and how to elicit knowledge for them is not covered. An associated point, which is also left unanswered, is how the different levels may be integrated into a consistent whole; what happens in the case of conflicts between the levels and how can such inconsistencies be detected? For hierarchically organised knowledge the answer may be to use inheritance in a similar manner to object-oriented programming. However for levels which are orthogonal, there is no obvious technique which can be suggested at this stage.

Although GRATE* embodies the model of joint responsibility in its production rules, the gap from the theoretical specification (eg mutual beliefs, joint intentions, etc.) to its computational realisation is quite large. A more satisfactory situation would be to allow the responsibility model to be expressed in a language which could be treated as an executable specification (after the appropriate translations have been performed). This approach has been followed in the work on Agent Oriented Programming (AOP) (Shoham, 1990) and in Rosenschein's work on situated automata (Rosenschein and Kaelbling, 1985). The basic idea behind AOP is to use mentalistic notions (such as beliefs, intentions, etc.) in a declarative programming regime. There will be three components which together constitute a complete AOP system: a formal system for defining the mental state of agents, an interpreted programming language for programming agents and an "agentification" process for converting agent programs into low-level executable objects. Rosenschein and Kaelbling's work aims to analyse the knowledge content of states of automata. They define an epistemic temporal logic which is used to specify what a designer would have a machine "know". This intentional description is then compiled down to a gate-level description of a digital machine, which satisfies the properties expressed in its intentional description.

Performance problems may occur if GRATE was applied to domains in which speedy responses and inferences were essential. At present the inference control strategy is inflexible, just being a loop, and GRATE agents may not respond rapidly to key events. There are two main ways in which this problem can be alleviated. The first is to develop a more flexible control regime which is able to identify and respond to important events in a timely fashion - an excellent illustration of such an approach is the control blackboard proposal made in (Hayes-Roth, 1985). In this system, knowledge related to control objectives is made explicit and reasoned about in order to make the system more responsive to key events. A second technique is to speed up the actual reasoning process. This can be achieved by pre-compiling knowledge and reasoning, in a manner similar to the reactive system approach described in section 3.1.1. This method is being pursued within the ARCHON project and is described further in the next subsection.

The case for a new computer level specifically for multi-agent problem solving has been described in this thesis. A small component of this cooperation knowledge level, for collaborative problem solving, is given by the model of joint responsibility. However substantial work remains to further define other aspects of this level, both for team problem solving and for other forms of social action. Of particular importance are more explicit models of negotiation and conflict resolution. In GRATE* negotiation techniques would have been useful for defining the cooperating group and the common solution they should adhere to. The intention consistency checking component of GRATE* has limited conflict resolution properties, however further work is needed to incorporate inter-agent conflicts into the system - again a principled model of negotiation would be useful in this context.

Finally, as the empirical investigations of section six highlighted, the distributed planning algorithm used by GRATE* is slow; the process of forming groups and agreeing a common solution is time-consuming. Improvements could be made if the planning was less sequential in nature and several agents were carrying out processing, rather than just the organiser. Also the philosophy of constructing groups which have the smallest number of members may not always be the most appropriate (eg if the tasks to be performed are time consuming and independent, the maximum amount of parallelism should be exploited). For greater flexibility, agents should have other organisational strategies available and the associated knowledge which allows them to select the most appropriate one at run-time.

7.2 A Hybrid Approach for Power and Generality

The mechanisms and techniques required to construct general-purpose, multi-agent systems powerful enough to handle complex interactions in real-size, time-critical domains is still an open research problem. Based on the experiences obtained by applying GRATE to industrial applications, it is felt that generic knowledge should be an integral component of the eventual solution. Detailed examination of the GRATE approach reveals the following features which enhance generality, adapted from (Kiss, 1991).

• Explicit Representations

Explicit representations of the world and an agent’s actions allow high level descriptions of behaviour to be formulated. Such mechanisms enable the agent to operate in a wide range of situations as it can work with general principles, rather than be tightly coupled to specific situations.

• Generic Structures (eg self and acquaintance models)

> By using structures which are meaningful in a variety of different contexts to represent domain specific information, the associated reasoning mechanisms obtain a degree of generality. Behaviour can be based on the generic structure rather than the idiosyncracies of a particular problem. This issue is discussed in depth, with respect to GRATE's agent models, in section 2.5.

One potential drawback of the generic rules is their inability to make a complex series of inferences rapidly. This is caused by their need to maintain a representation of the world, then perform inferences on it, then update the models and so on until the series is complete. Hence they may lack the *power* (Kiss, 1991) to be used as the sole control mechanism in certain time-critical domains. Also many of GRATE's lower-level rules, such as those dealing with local control of tasks, encode sequences of actions - so clear patterns of rule firings can be observed. To overcome this limitation and enhance the system's power, well defined sequences of actions may be compiled down and activated as a single unit when the appropriate stimulus in received. Such units of activity are very similar in nature to the reactive mechanisms described in section 3.1.1. By compiling knowledge and reasoning, processing power is greatly improved (Kiss, 1991). Rather than reasoning about which actions should be taken, an agent merely has to recognise the situation and carry out the associated actions. This means the system can carry out complex reasoning very effectively and with minimal computational overhead. The drawback of this approach, however is that such reasoning is invariably tightly coupled to the environment and consequently agents' behaviour is highly domain dependent and inflexible. Agents are only able to respond to situations for which they have a predefined stimulus-response pair.

As a result of these insights, a hybrid approach encompassing both a reflective component (the generic rules) and reactive mechanisms is now being followed within the ARCHON project (Jennings, 1991a; Jennings and Wittig, 1992). Such an approach is deemed necessary because ARCHON aims to construct a general cooperation environment which is applicable to real-size applications for the domain of industrial control (Wittig, 1992). ARCHON agents have three main functionalities; control of local and social activities are distinct components (modules) of the architecture, whilst situation assessment is shared by these two modules. The characteristics of each type of activity are given below and the chosen representation formalism stated and then briefly justified (Jennings and Wittig, 1992).

• Local Control

> Requires a fast response to numerous situations arising in the domain-level system. Such control varies between applications and is difficult to generalise because of the diverse nature of the systems being controlled. Therefore it is well suited to being encoded in a reactive

formalism.

• Social Activities

The number of cooperative requests occurring are relatively few in number. Agents spend most of their time engaged in problem solving, rather than in communicating information, and are capable of solving a substantial proportion of their problems alone.

It is also relatively easy to identify the types of cooperation request which may occur and the reasons behind them. For example, agents may request information or ask for tasks to be performed because they cannot be realised locally or because it is deemed better to ask for assistance. As such activities can be described at a relatively high (general) level and because they are relatively few in number, such knowledge is well suited to a reflective reasoning mechanism. In fact GRATE's rules for cooperation and situation assessment form the basis of this module.

• Situation Assessment

Assessment activities are intermediate in number and in terms of their generality. Some assessment is general, whereas some is specific to particular applications. In terms of knowledge representation, this functionality has been implemented as a mixture of generic rules and reactive mechanisms.

As well as being used in separate modules, it is also possible to combine reflective and reactive reasoning within a single module. Consider the following example taken from the social activity module (Jennings and Wittig, 1992). One of this module's tasks is to decide upon a particular cooperation protocol once the need for social interaction has been established. At present, two protocols are available; *client server* and *contract net* (Smith, 1980). In the former, the agent initiating the cooperation (the client) decides with whom the interaction ought to be (the server), based on information contained in its acquaintance models. It then sends a request directly to the chosen server. This protocol has the advantages of a low communication overhead and that interaction can start immediately. However the disadvantage is that the chosen server may be unsuitable; for example it may be just about to embark on a time consuming, high priority task - ensuring the request will not be honoured in a timely fashion.

The contract net protocol is more protracted. The agent initiating a social activity sends out a request to all its acquaintances, waits for bids from each of them which it evaluates before deciding which of them should be awarded the activity. Compared with client server mode, this protocol has a higher communication overhead and takes a longer time for the interaction to be established. However the agent eventually chosen is likely to be better suited to tackling the problem, since it's bid states the time by which the activity will be completed. The characteristics and decision making ability associated with each protocol are encoded in generic rules.

```
(rule select-protocol1
   (IF (COMMUNICATION-RESOURCES-OVERBURDENED ?Agent))
   (THEN (ESTABLISH CLIENT-SERVER-COOPERATION ?Agent ?Task)))

(rule select-protocol2
   (IF (AND (IMPORTANT ?Agent ?Task) (FIXED-DEADLINE ?Agent ?Task)))
   (THEN (ESTABLISH CONTRACT-NET-COOPERATION ?Agent ?Task)))
```

The first rule states that if communication resources are currently being used to near capacity, then the client-server protocol is more appropriate because of its lower communication overhead. The second rule states that if a task is considered important and has a deadline which must be adhered to, but which is not too close, then the contract net protocol is appropriate because it provides a more reliable estimate of completion time.

It is also possible to have protocol selection based on meta-level considerations. The following rule states that if the client server mode of cooperation was selected initially, but that its execution failed (eg the intended server was too busy to respond) then a contract net ought to be established. The rationale for this is that the contract net reaches a greater number of agents, hence there is an improved chance of finding a suitable acquaintance.

```
(rule select-protocol3
   (IF (FAILED (ESTABLISH CLIENT-SERVER-COOPERATION ?Agent ?Task)))
   (THEN (ESTABLISH CONTRACT-NET-COOPERATION ?Agent ?Task)))
```

All three rules are generic in that they can be used in any application in which client server and contract net protocols are available. It would also be possible to encode the protocol control mechanism in a similar manner since, by its very nature, it follows a well-defined ordering of steps. So with a task announcement in the contract net protocol, the originator has to decide the agents to whom the request should be sent, give the contract a unique identifier and send out the announcement. It then has to wait until all the bids have been received before deciding to whom the contract should be awarded. When a decision has been reached, the originator informs the chosen agent that it's bid was successful. However because of the static sequence of actions and the minimal associated reasoning, it was decided that the protocol could be more efficiently encoded by a pre-compiled behaviour; leaving the reflective component free for more sophisticated reasoning.

```
TASK-ANNOUNCEMENT (TASK REP) {[1]
   TASK-DESCRIPTION (TASK) ((?AGENTS ?TIME ?CANDIDATES))
   CONTRACT-ID () (?CONTRACT-ID)
   BROADCAST(CANDIDATES ‘ANNOUNCE CONTRACT-ID TASK)
   GET-BID(CONTRACT-ID AGENTS TIME) (?LIST-TO-AWARD)
   BROADCAST (LIST-TO-AWARD CONTRACT-ID TASK)}
```

This behaviour differs from most of the others in the system in that it is domain independent. It can, therefore, be used in other applications in which the contract net protocol is desirable.

[1]. This behaviour is written in ALAN (Loingtier, 1991). Underlining signifies reference to other behaviours

REFERENCES

Aarnts, R. P., Corera, J., Perez, J., Gureghian, D., and Jennings, N. R., (1991), "*Examples of Cooperative Situations and their Implementation*", Journal of Software Research, 3 (4), pp 74- 81.

Abraham, M. F., (1982), "*Modern Sociological Theory*", Oxford University Press.

Agha, G., and Hewitt, C. E., (1985), "*Concurrent Programming Using ACTORS: Exploiting Large Scale Parallelism*", AI Memo 865, MIT.

Agha, G., (1986), "*ACTORS: A Model of Concurrent Computation in Distributed Systems*", MIT Press.

Agre, P. E., and Chapman, D., (1987), "*Pengi: An Implementation of a Theory of Activity*", Proc. 6th National Conference on AI, pp 268-272.

Allen, J. F., (1984), "*Toward a General Theory of Time and Action*" Artificial Intelligence, 23, pp 123-154

Austin, J. L., (1962), "*How to do Things With Words*", Harvard University Press.

Avouris, N. M., Liedekerke, M. H. V., and Sommaruga, L., (1989), "*Evaluating the CooperA Experiment: The Transition from an Expert System Module to a Distributed Artificial Intelligence Testbed for Cooperating Expert Systems*", Proc. of 9th Workshop on Distributed AI.

Axelrod, R., (1984), "*The Evolution of Cooperation*", Basic Books.

Barwise, J., (1988), "*Three Views of Common Knowledge*" Proc. of Second Conference of Reasoning About Knowledge, Morgan Kaufmann.

Benyon, D., Innocent, P., and Murray, D., (1987), "*System Adaptivity and the Modelling of Stereotypes*", Proc. INTERACT 1987, (Eds H.J.Bullinger and B.Shackel), pp 245-253.

Bisiani, R., Alleva, F., Forin, A., Lerner, R., and Bauer, M., (1988), "*The Architecture of the Agora Environment*", in Distributed Artificial Intelligence (Ed M.N.Huhns), pp 99-119 Pitman Publishing.

Bond, A. H., and Gasser, L., (eds) (1988), "*Readings in Distributed Artificial Intelligence*", Morgan Kaufmann.

Bond, A. H., (1989), "*Commitment: Some DAI insights from Symbolic Interactionist Society*", Proc. of 9th Workshop on Distributed Artificial Intelligence.

Bouron, T., Feber, J., and Samuel., F., (1990), "*MAGES: A Multi-Agent Testbed for Heterogeneous Agents*", Proc. Modelling An Autonomous Agent in a Multi-Agent

World, pp 221-239, Saint-Quentin en Yveslines, France.

Brandau, R., and Weihmayer, R., (1989), "*Heterogeneous Multi-Agent Cooperative Problem Solving in a Telecommunication Network Management Domain*", Proc. of 9th Workshop on Distributed Artificial Intelligence, pp 41-57.

Bratman, M. E., (1984), "*Two Faces of Intention*", Philosophical Review 93, pp 375-405.

Bratman, M. E., (1990), "*What is Intention?*", Intentions in Communication, (Eds P.R.Cohen, J.Morgan and M.E.Pollack), pp 15-33, MIT Press.

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

Briot, J. P., and Gasser, L., (1991), "*Connections Between Object Based Concurrent Programming and Distributed Artificial Intelligence*", Proc. International Joint Conference on AI Workshop on Objects and AI.

Brooks, R. A., (1991), "*Intelligence without Representation*", Artificial Intelligence, 47, 1-3, pp 139-159.

Buchanan, B. G., and Shortcliffe, E. H., (1984), "*Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*", Reading, MA, Addison Wesley.

Bürckert, H. J., Muller, J., (1990), "*RATMAN: Rational Agents Testbed for Multi-Agent Networks*", Proc. Modelling An Autonomous Agent in a Multi-Agent World, pp 243-257, Saint-Quentin en Yveslines, France.

Burmeister, B., and Sundermeyer, K., (1991), "*Cooperative Problem Solving Guided by Intentions and Perception*", Proc. Modelling An Autonomous Agent in a Multi-Agent World, Kaiserslautern, Germany.

Cammarata, S., McArthur, D., and Steeb, R., (1983), "*Strategies of Cooperation in Distributed Problem Solving*", Proc. International Joint Conference on AI, pp 767-770.

Castelfranchi, C., (1990), "*Social Power: A Point Missed in Multi-Agent, DAI and HCI*", in Decentralized AI, (Eds Y.Demazeau and J.P.Muller), pp 49-62, Elsevier.

Cesta, A., Castelfranchi, C., Conte, R., and Miceli, M., (1989), "*Deep KR About Goal Achievement: Motivations for Pragmatic Structures*", Computational Intelligence I, pp 143-149.

Chandrasekaran, B., (1983a), "*Generic Tasks in Knowledge Based Reasoning: High Level Building Blocks for Expert System Design*", IEEE Expert, 1 (3), pp 23-30.

Chandrasekaran, B., (1983b), "*Towards A Taxonomy of Problem Solving Types*", AI

Magazine 4, pp 9-17.

Chang, E., (1988), "*Participant Systems for Cooperative Work*", in Distributed Artificial Intelligence (Ed M.N.Huhns), pp 311-339, Pitman Publishing.

Chin, D. N., (1986), "*User Modelling in UC, The UNIX Consultant*", Proc. CHI, (Eds M.Mantei and P.Orbiton).

Clearwater, S. H., Huberman, B. A., and Hogg, T., (1991), "*Cooperative Solution of Constraint Satisfaction Problems*", Science, 254, pp 1181-

Cohen, P. R., (1991), "*A Survey of the Eighth National Conference on Artificial Intelligence: Pulling Together or Pulling Apart*", AI Magazine, 12, 1, pp 16-41.

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

Cohen, P. R., and Levesque, H. J., (1990b), "P*ersistence, Intention and Commitment*" in Intentions in Communication (Eds P.R.Cohen, J.Morgan and M.E.Pollack), pp 33-70, MIT Press.

Cohen, P. R., and Levesque, H. J., (1991), "*Teamwork*", SRI Technical Note 504.

Cohen, P. R., Morgan, J., and Pollack, M. E., (eds), (1990), "*Intentions in Communication*", The MIT Press, Cambridge, Massachusetts.

Conte, R., (1989), "*Institutions and Intelligent Systems*", in Operational Research and the Social Sciences (Ed M.C.Jackson, P.Keys and S.A.Cooper), pp 201-206, Plenum Press, New York.

Conte, R., Miceli, M., and Castelfranchi, C., (1990), "*Limits and Levels of Cooperation: Disentangling Various Types of Prosocial Interaction*", Proc. Modelling an Autonomous Agent in a Multi-Agent World.

Coulouris, G. F., and Dollimore, J., (1988), "*Distributed Systems*", Addison Wesley.

Cox, B. J., (1990), "*Planning the Software Industrial Revolution*", IEEE Software, Nov. 90, pp 25-33.

Davidson, D., (1980), "*Essays on Actions and Events*" Oxford University Press.

Davis, R., and Smith, R. G., (1983), "*Negotiation as a Metaphor for Distributed Problem Solving*", Artificial Intelligence, 20, pp 63-109.

Dawkins, R., (1976), "*The Selfish Gene*", Oxford University Press.

Decker, K. S., (1987), "*Distributed Problem Solving Techniques: A Survey*", IEEE Trans. on Systems Man and Cybernetics, 17, 5, Sept. 87, pp 729-740.

Deen, S. M., (Ed) (1990), “*Cooperating Knowledge Based Systems 1990*” Springer Verlag

de Greef, P., (1991), “*Analysis of Human Computer Cooperative Work*”, IMAGINE Deliverable, D-I.2

de Kleer, J., (1984), “*How Circuits Work*”, Artificial Intelligence 24 (1-3), pp 205-281.

Demazeau, Y., and Muller, J. P., (eds) (1990), “*Decentralized AI*” Elesevier Science Publishers.

Demazeau, Y., and Muller, J. P., (eds) (1991), “*Decentralized AI Vol II*” Elesevier Science Publishers.

Dennett, D. C., (1987), “*The Intentional Stance*”, Bradford Books / MIT Press.

Doran, J., Carvajal, H., Choo, Y. J., and Li, Y., (1990), “*The MCS Multi-Agent Testbed: Developments and Experiments*” in Cooperating Knowledge Based Systems 1990, (Ed S.M.Deen), pp 240-254, Springer Verlag.

Durfee, E. H., and Lesser, V. R., (1987), “*Using Partial Global Plans to Coordinate Distributed Problem Solvers*”, Proc. International Joint Conference on AI, pp 873-883.

Durfee, E. H., Lesser, V. R., and Corkill, D. D., (1987), “*Coherent Cooperation among Communicating Problem Solvers*”, IEEE Trans. on Computers, 36, pp 1275-1291.

Durfee, E. H., Lesser, V. R., and Corkill, D. D., (1988a), “*Cooperation through Communication in a Distributed Problem Solving Network*”, in Distributed Artificial Intelligence (Ed M.N.Huhns), pp 29-59, Pitman Publishing.

Durfee, E. H., Lesser, V. R., and Corkill, D. D., (1988b), “*Coherent Cooperation Among Communicating Problem Solvers*”, IEEE Transactions on Computers, 36, pp 1275-1291.

Engelmore, R., and Morgan, T., (eds) (1988), “*Blackboard Systems*”, Addison Wesley.

Erman, L. D., and Lesser, V. R., (1975), “*A Multi-Level Organisation for Problem Solving using Many Diverse Cooperating Sources of Knowledge*”, Proc of International Joint Conference on AI, pp 483-490.

Feber, J., and Briot, J. P., (1988), “*Design of a Concurrent Language for Distributed Artificial Intelligence*”, Proc. International Conference on Fifth Generation Computer Systems, Vol 2, pp 755-762, ICOT, Tokyo, Japan.

Feber, J., (1992), “*Representing and Planning in the Eco Problem Solving Framework*”, in Distributed Artificial Intelligence: Theory and Praxis, (Ed L.Gasser and N.M.Avouris), Kluwer Academic Press (forthcoming).

Feigenbaum, E. A., McCorduck, P., and Nii, H. P, (1988), “*The Rise of the Expert Company*” Times Books.

Fikes, R. E., Hart, P. E., and Nilsson, N., (1971), “*STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving*”, Artificial Intelligence, 2, pp 189-208.

Fuchs, J., Skarek, P., Varga, L., and Malandain, E., (1992), “*Distributed Cooperative Architecture for Accelerator Operation*” Proc. Second International Workshop on Software Engineering, Artificial Intelligence and Expert Systems for High Energy and Nuclear Physics.

Gallastegui, I., Laresgoiti, I., Perez, J., Amantegui, J., and Echavarri, J., (1990), “*Operating Experience of an Expert System for Fault Analysis in Electrical Networks*”, Proc International Working Conference on Expert Systems in Electrical Power Systems, Avignon.

Galbraith, J., (1973), “*Designing Complex Organizations*” Addison-Wesley.

Galliers, J. R., (1988), “*A Strategic Framework for Multi-Agent Cooperative Dialogue*”, Proc. European Conference on AI, pp 415-420.

Galliers, J. R., (1989), “*The Positive Role of Conflict in Cooperative Multi-Agent Systems*”, Proc. First European Workshop on Modelling an Autonomous Agent in a Multi-Agent World.

Gasser, L., (1991), “*Social Conceptions of Knowledge and Action: DAI Foundations and Open System Semantics*”, Artificial Intelligence 47, pp 107-138.

Gasser, L., Roquette, N., Hill, R. H., and Lieb, J., (1989), “*Representing and Using Organisational Knowledge in DAI Systems*”, in Distributed Artificial Intelligence Vol II, (Eds L.Gasser and M.N.Huhns), pp 55-78, Pitman Publishing.

Gasser, L., Braganza, C., and Herman, N., (1988), “*MACE: A Flexible Testbed for Distributed AI Research*”, in Distributed Artificial Intelligence (Ed M.N.Huhns), pp 119-153 Pitman Publishing.

Gasser, L., and Huhns, M. N., (eds) (1989), “*Distributed Artificial Intelligence Volume II*”, Pitman Publishing.

Genesereth, M. R., Ginsberg, M. L., and Rosenschein, J. S., (1988), “*Cooperation Without Communication*”, in Readings in Distributed Artificial Intelligence (Eds A.H.Bond and L.Gasser), pp 220-226, Morgan Kaufmann.

Genesereth, M. R., and Nilsson, N. J., (1988), “*Logical Foundations of Artificial Intelligence*” Morgan Kaufmann.

Ginsberg, M. L., (1988), “*Decision Procedures*”, in Distributed Artificial Intelligence

(Ed M.N.Huhns), pp 29-58, Pitman Publishing.

Glance, N. S., and Huberman, B. A., (1992), “*The Outbreak of Cooperation*”, Technical Report, Dynamics of Computation Group, Xerox Palo Alto Center, Palo Alto, California.

Guha, R. V., and Lenat, D. B., (1990), “*CYC: A Mid Term Report*”, AI Magazine, pp 32-59, 11 (3).

Green, P. E., (1988), “*AF: A Framework for Real-Time Distributed Cooperative Problem Solving*”, in Distributed Artificial Intelligence (Ed M.N.Huhns), pp 153-177, Pitman Publishing.

Grosz, B. J., and Sidner, C. L., (1990), “*Plans for Discourse*”, Intentions in Communication, (Eds P.R.Cohen, J.Morgan and M.E.Pollack), pp 417-444, MIT Press.

Halpern, J. Y., (1986), “*Reasoning About Knowledge: An Overview*”, Proc. Theoretical Aspects of Reasoning About Knowledge, (Ed J.Y.Halpern), pp 1-17, Morgan Kaufmann.

Halpern, J. Y., and Moses, Y. O., (1984), “*Knowledge and Common Knowledge in a Distributed Environment*”, Proc. of the 3rd ACM Conference on Principles of Distributed Computing, pp 50-61.

Hayes-Roth, B., (1985), “*A Blackboard Architecture For Control*”, Artificial Intelligence Journal, 26, pp 251-321.

Hayes-Roth, F., (1980), “*Towards a Framework for Distributed AI*”, SIGART Newsletter pp 51-52.

Hayes-Roth, F., Erman, L. D., Fouse, S., Lark, J. S., and Davidson, J., (1988), “*ABE: A Cooperating Operating System and Development Environment*”, in Readings in Distributed Artificial Intelligence (Eds A.H.Bond and L.Gasser), pp 457-490, Morgan Kaufmann.

Hendler, J., Tate, A., and Drummond, M., (1990), “*AI Planning: Systems and Techniques*”, AI Magazine, pp 61-77

Hewitt, C. E., and Baker, H., (1977), “*Actors and Continuous Functionals*”, Proc. of IFIP Working Conference on Formal Descriptions of Programming Concepts, pp 367-387, August 1977.

Hewitt, C. E., (1985), “*The Challenge of Open Systems*”, BYTE, 10, 4, pp 223-244.

Hewitt, C. E., and Kornfield, W. A., (1980), “*Message Passing Semantics*”, SIGART Newsletter, pp 48.

Hickman, F. R., Killin, J. L., Land, L., Mulhall, T., Porter, D., and Taylor, R. M., (1989), "*Analysis for Knowledge Based Systems: A Practical Guide to the KADS Methodology*", Ellis Horwood.

Hobbs, J. R., (1990), "*Artificial Intelligence and Collective Intentionality*", Intentions in Communication, (Eds P.R.Cohen, J.Morgan and M.E.Pollack), pp 445-460, MIT Press.

Horowitz, E., and Munsen, J. B., (1984), "*An Expansive View of Reusable Software*", IEEE Trans. Software Engineering, 10 (5), pp 477-487.

Huhns, M. N., (Ed) (1988), "*Distributed Artificial Intelligence*", Pitman Publishing.

Huhns, M. N., Mukhopadhyay, U., Stephens, L. M., and Bonnell, R. D., (1988), "*DAI for Document Retrieval: The MINDS Project*", in Distributed Artificial Intelligence (Ed M.N.Huhns), pp 249-284, Pitman Publishing.

Jennings, N. R., (1991a), "*Cooperation in Industrial Systems*" Proc. ESPRIT Conference, pp 253-263, Brussels.

Jennings, N. R., (1991b), "*On Being Responsible*", Proc. Modelling Autonomous Agents in a Multi-Agent World, Third European Workshop, Kaiserslautern, Germany (also appearing in Decentralized AI Volume 3, (ed E.Werner & C.Castelfranchi), Elesevier Science Publishers).

Jennings, N. R., (1992a), "*Towards a Cooperation Knowledge Level for Collaborative Problem Solving*", Proc. 10th European Conference on Artificial Intelligence, pp 224-228, Vienna, Austria.

Jennings, N. R., (1992b), "*A Knowledge Level Approach to Collaborative Problem Solving*", Proc. AAAI Workshop on Cooperation amongst Heterogeneous Intelligent Agents, pp 55-64, San Jose, California.

Jennings, N. R., (1992c), "*Using GRATE to Build Cooperating Agents for Industrial Control*", Proc. IFAC/IFIP/IMACS International Symposium on Artificial Intelligence in Real Time Control, pp 691-696, Delft, The Netherlands.

Jennings, N. R., and Mamdani, E. H., (1992), "*Using Joint Responsibility to Coordinate Collaborative Problem Solving in Dynamic Environments*", Proc. 10th National Conference on AI, pp 269-275, San Jose, California.

Jennings, N. R., Mamdani, E. H., Laresgoiti, I., Perez, J., and Corera, J., (1992), "*GRATE: A General Framework for Cooperative Problem Solving*" IEE-BCS Journal of Intelligent Systems Engineering, Volume 1, Issue 2.

Jennings, N. R., and Wittig, T., (1992), "*ARCHON: Theory and Practice*", in Distributed Artificial Intelligence: Theory and Praxis, (Ed L.Gasser and N.M.Avouris), Kluwer Academic Press (forthcoming).

Jennings, N. R., Varga, L. Z., Aarnts, R. P., Fuchs, J., and Skarek, P., (sub), "*Transforming Standalone Expert Systems into a Community of Cooperating Agents*", submitted to Int. Journal of Engineering Applications of Artificial Intelligence.

Jones, G. W., (1990), "*Software Engineering*", Wiley

Kass, R., and Finin, T., (1987), "*Rules for Implicit Acquisition of Knowledge About the User*", Proc. of 6th National Conference on AI, pp 295-300.

Kass, R., and Finin, T., (1989), "*The Role of User Models in Cooperative Interaction Systems*", International Journal of Intelligent Systems, 3, pp 329-354.

Keene, S. E., (1989), "*Object Oriented Programming in COMMON LISP*", Addison-Wesley.

Khoui, H., and Jennings, N. R., (1990), "*Efficiency of Modelling Other Agents with a Distributed Problem Solving Network*", MSc Project, Dept. Electronic Engineering, Queen Mary and Westfield College.

Kiss, G., (1991), "*Varying Coupling of Agents to their Environment: Combining Situated and Symbolic Automata*" Proc. Modelling An Autonomous Agent in a Multi-Agent World, Kaiserslautern, Germany.

Kiss, G., and Reichgelt, H., (1991), "*Towards A Semantics of Desires*" Proc. Modelling An Autonomous Agent in a Multi-Agent World, Kaiserslautern, Germany.

Klein, M., and Baskin, A., (1990), "*A Computational Model for Conflict Resolution in Cooperative Design Systems*", in Cooperating Knowledge Based Systems (Ed S.M.Deen), pp 201-222, Springer Verlag.

Lander, S., Lesser, V. R., and Connell, M. E., (1990), "*Conflict Resolution Strategies for Cooperating Expert Agents*" in Cooperating Knowledge Based Systems (Ed S.M.Deen), pp 183-200, Springer Verlag.

Lansky, A. L., (1989), "*A Perspective on Multi-Agent Planning*", SRI International, Technical Note 474, Center for Study of Language and Information, Stanford University.

Lenat, D. B., (1975), "*BEINGS: Knowledge as Interacting Experts*", Proc. International Joint Conference on AI, pp 126-133.

Lenat, D. B., and Feigenbaum, E. A., (1991), "*On the Thresholds of Knowledge*", Artificial Intelligence, 47, pp 185-250.

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

Lesser, V. R., and Corkill, D. D, (1987), "*Distributed Problem Solving*", Encyclopedia of Artificial Intelligence (Ed S.C.Shapiro), pp 245-251, John Wiley and Sons.

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

Leveson, N. G, (1990), "*The Challenge of Building Process Control Software*", IEEE Software, pp 55-62.

Levesque, H. J., Cohen, P. R., and Nunes, J. H., (1990), "*On Acting Together*", Proc. 9th National Conference on AI, pp 94-99.

Lochbaum, K. E., Grosz, B. J., and Sidner, C. L., (1990), "*Models of Plans to Support Communication*", Proc. 9th National Conference on AI, pp 485-490.

Loingtier, J. M., (1991), "*ALAN: An Agent Language for Cooperation*", Proc. International Joint Conference on AI Workshop on Objects and AI, Sidney, Australia.

Luce, R. D., and Raiffa, H., (1957), "*Games and Decisions: Introduction and Critical Survey*", John Wiley and Sons.

Malone, T. W., (1987), "*Modelling Coordination in Organizations and Markets*", Management Science, 33, pp 1317-1332.

Malone, T. W., and Crowston, K., (1990), "*Toward an Interdisciplinary Theory of Coordination*" Technical Report CCS TR 120, Centre for Coordination Science, Sloan School of Management, Cambridge, Massachusetts.

McArthur, D., Steeb, R., and Cammarata, S., (1982), "*A Framework for Distributed Problem Solving*", Proc. 1st National Conference on AI, pp 181-184.

McClelland, J. L., and Rumelhart, D. E., (1986), "*Parallel Distributed Processing*", MIT Press.

McDermott, J., (1981), "*R1: A Rule Based Configurer of Expert Systems*", Artificial Intelligence, 19.

McDermott, J., (1988), "*A Taxonomy of Problem Solving Methods*" Automating Knowledge Acquisition for Expert Systems, (Ed S.Marcus), pp 225-256, Kluwer Academic Press.

McDermott, J., (1990), "*Developing Software is Like Talking to Eskimos about Snow*", Proc. 9th National Conference on AI, pp 1130-1133.

Mead, G. H., (1934), "*Mind, Self and Society*", University of Chicago Press, Chicago.

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

Newell, A., (1982), “*The Knowledge Level*” Artificial Intelligence 18, pp 87-127.

Nii, P. H., (1986a), “*Blackboard Systems (PART I)*”, AI Magazine, 7, 2, pp 38-53.

Nii, P. H., (1986b), “*Blackboard Systems (PART II)*”, AI Magazine, 7, 3, pp 82-106.

Nilsson, N. J., (1980), “*Two Heads are Better than One*”, SIGART Newsletter, 73, pp 43.

Novak, M. A., and Sigmund, K., (1992), “*Tit for Tat in Heterogeneous Populations*”, Nature, 355, pp 250-253.

Partridge, D., (1987), “*The Scope and Limitations of Future Generation Expert Systems*”, Future Generations Computer Systems, 3, 1, pp 1-10.

Parunak, H. V. D., (1988), “*Manufacturing Experience with the Contract Net*”, in Distributed Artificial Intelligence (Ed M.N.Huhns), pp 285-310, Pitman Publishing.

Parunak, H. V. D., (1989), “*Distributed AI and Manufacturing Control: Some Issues and Insights*”, Proc. First European Workshop on Modelling an Autonomous Agent in a Multi-Agent World.

Pollack, M. E., (1990), “*Plans as Complex Mental Attitudes*”, in Intentions in Communication (Eds P.R.Cohen, J.Morgan and M.E.Pollack), pp 77-105, MIT Press.

Power, R., (1984), “*Mutual Intention*”, Journal for the Theory of Social Behaviour, 14, pp 85-105.

Rammamritham, K., and Stankovic, J. A., (1984), “*Dynamic Task Scheduling in Hard Real-Time Distributed Systems*”, IEEE Software pp 65-75.

Rao, A. S., Georgeff, M. P., and Sonenberg, E. A., (1991), “*Social Plans: A Preliminary Report*”, Proc. Modelling An Autonomous Agent in a Multi-Agent World, Kaiserslautern, Germany.

Rissland, E., (1987), “*Ingredients of Intelligent User Interfaces*”, in Readings in Human Computer Interaction (Eds R.M.Baecker and W.A.S.Buxton), pp 703-708, Morgan Kaufmann.

Rich, E., (1979), “*User Modelling via Stereotypes*”, Cognitive Science, 3, pp 329-354.

Reeves, S., and Clarke, M., (1990), “*Logic For Computer Science*”, Addison Wesley.

Roda, C., and Jennings, N. R., (1991), “*The Impact of Heterogeneity on Cooperating Agents*”, Proc. AAAI Workshop on Cooperation among Heterogeneous Intelligent Systems, Anaheim, Los Angeles, USA.

Rosenschein, J. S., (1982), “*Synchronization of Multi-Agent Plans*”, Proc. 1st National

Conference on AI, pp 115-119.

Rosenschein, J. S., and Genesereth, M. R., (1985), "*Deals Among Rational Agents*", Proc. International Joint Conference on AI, pp 91-99.

Rosenschein, S. J., and Kaelbling, L., (1985), "*The Synthesis of Digital Machines with Provable Epistemic Properties"* in Proc. of the 1986 Conference on Theoretical Aspects of Reasoning About Knowledge, (ed J. Y. Halpern), Morgan Kaufmann.

Searle, J. R., (1969), "*Speech Acts: An Essay in the Philosophy of Language*", Cambridge University Press.

Searle, J. R., (1983), "*Intentionality: An Essay in the Philosophy of Mind*", New York, Cambridge University Press.

Searle, J. R., (1990), "*Collective Intentions and Actions*", in Intentions in Communication, (Eds P.R.Cohen, J.Morgan and M.E.Pollack), pp 401-416, MIT Press

Shoham, Y., (1990), "*Agent Oriented Programming*", Technical Report STAN-CS-1335-90, Dept. of Computer Science, Stanford University, California.

Singh, M. P., (1990a), "*Group Intentions*", Proc. of 10th International Workshop on Distributed Artificial Intelligence, Texas, MCC Technical Report ACT-AI-355-90.

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

Simon, H. A., (1957), "*Models of Man*", New-York, Wiley.

Smith, D., and Broadwell, M., (1988), "*The Pilot's Associate: An Overview*", Proc. SAE Aerotech Conference, Los Angeles, CA.

Smith, R. G., (1980), "*The Contract Net Protocol: High Level Communication and Control in a Distributed Problem Solver*", IEEE Trans. on Computers, 29, 12, pp 1104-1113.

Smith, R. G., and Davis, R., (1981), "*Frameworks for cooperation in Distributed Problem Solving*", IEEE Trans. on Systems Man and Cybernetics, 11, 1, pp 61-70.

Sommerville, I., (1992), "*Software Engineering*", Fourth Edition, Addison Wesley.

Sridharan, N. S., (1987), "*1986 Workshop on Distributed AI*", AI Magazine, Fall, pp 75-85.

Steels, L., (1985), "*Second Generation Expert Systems*", Future Generations Computer Systems, 1, 4, pp 213-221.

Steels, L., (1990), "*Components of Expertise*" AI Magazine, 11 (2), pp 29-49.

Steels, L., (1991), "*Toward a Theory of Emergent Functionality*" in From Animals to Animats, (Ed J.Meyer and S.Wilson), pp 451-461, Bradford Books.

Steiner, D. D., Mahling, D. E., and Haugeneder, H., (1990), "*Human Computer Cooperative Work*" Proc. 10th Int. Workshop on Distributed AI, Austin, Texas.

Stefik, M., (1986), "*The Next Knowledge Medium*", AI Magazine 7 (1), pp 34-46.

Suchman, L., (1987), "*Plans and Situated Actions: The Problem of Human-Machine Communication*", Cambridge University Press.

Sueyoshi, T., and Tokoro, M., (1991), "*Dynamic Modelling of Agents for Coordination*" in Decentralised AI Vol II (Ed Y.Demazeau and E.Werner), Elesevier Science.

Sycara, K. P., (1989), "*Argumentation: Planning Other Agents' Plans*" Proc. International Joint Conference on AI, pp 517-523

Trenouth, J., and Ford, L., (1989), "*Computational Models of Computer Users: A Review, Classification and Case Study*", Dept. of Computer Science, Exeter University.

Tuomela, R., and Miller, K., (1988), "*We-Intentions*", Philosophical Studies 53, pp 367-389.

Wahlster, W., and Kobsa, A., (1986), "*Dialogue Based User Models*", Proc. of IEEE, 74, 7, pp 948-960.

Wesson, R., Hayes-Roth, F., Burge, J. W., Stasz, C., and Sunshine, C. A., (1981), "*Network Structures for Distributed Situation Assessment*", IEEE Trans. on Systems Man and Cybernetics, 11, 1, pp 5-23.

Werner, E., (1988), "*Social Intentions*", Proc. European Conference on AI, pp 719-723.

Werner, E., (1989), "*Cooperating Agents: A Unified Theory of Communication and Social Structure*", in Distributed Artificial Intelligence Vol II (Ed L.Gasser and M.N.Huhns), pp 3-36.

Whitney, C., (1992), "*Cooperating Intelligent Agents: A Study of GRATE*", BT Report MAIN-WP1008.

Wittig, T., (1992), "*ARCHON: An Architecture for Multi-Agent Systems*", Ellis Horwood.

Yang, J. D., Huhns, M. N., and Stephens, L. M., (1985), "*An Architecture for Control and Communications in Distributed Artificial Intelligence*", IEEE Trans. on Systems Man and Cybernetics, 15, 3, pp 316-326.

Yonezawa, A., and Tokoro, M., (1987), "*Object-Oriented Concurrent Programming*", MIT Press.

APPENDIX A

SPECIFYING GRATE RECIPES

This appendix describes the language used for specifying recipes in GRATE and GRATE*. Recipes are predetermined sequences of action together with an ordering and are stored in the agent models (see section 2.3.5). A recipe's actions may themselves be recipes or they may be tasks which can be executed by the domain level system. Recipes can be composed of activities which an agent can be perform locally or actions which only acquaintances can perform. They do not identify the agents who will undertake actions, this is left to the cooperation module to decide at run-time.

Recipes consist of a name for identification purposes, a set of trigger conditions which indicate when the recipe should be invoked, the actions which make up the recipe and the results which should be produced. The following example is taken from the CSI agent; its function is to group alarms messages into meaningful chunks so that network models can be updated and also to indicate whether a disturbance has taken place.

```
NAME: (DETECT-DISTURBANCE)
TRIGGERS: (     (INFO-AVAILABLE BLOCK-ALARMS-ARRIVED-IN-IS)
                (NO-DISTURBANCE-AT-PRESENT) )
ACTIONS: ( (    (START (ALARM-MESSAGES-ACQUISITION
                                        (> BLOCK-ALARM-MESSAGES)))
               (START (PROVISION-OF-SNAPSHOTS (> SNAP-SHOT)) ))
           ( (START (DISTURBANCE-DETECTION (< BLOCK-ALARM-MESSAGES)
                                   (> DISTURBANCE-DETECTION-MESSAGE))))
OUTCOME: (DISTURBANCE-DETECTION-MESSAGE
          BLOCK-ALARM-MESSAGES
          SNAP-SHOT)
```

The name can be any LISP S-expression. The triggers are LISP functions which are part of an implicit AND-graph. They are evaluated to determine whether the recipe should be activated and must return boolean values. The functions are evaluated by placing the current agent object, the CSI in this case, as the first parameter to the function specified. So to evaluate whether the DETECT-DISTURBANCE recipe is to be invoked the following function call is made:

(INFO-AVAILABLE CSI BLOCK-ALARMS-ARRIVED-IN-IS)

For the recipe to be activated all of its trigger conditions must return non-nil results. Some general functions are provided (eg INFO-AVAILABLE is a request to GRATE's information store), however others such as NO-DISTURBANCE-AT-PRESENT are domain dependent and so have to be written by the application developer.

If all the trigger conditions are satisfied, the recipe is ready for activation. Activation involves creating a new recipe instantiation; there can be any number of such instantiations and each has its own problem solving context. The action ordering is given by the position of the brackets in the expression and the control directive. In the DETECT-DISTURBANCE recipe, the first action has two components which are to be started concurrently - the tasks alarm-message-acquisition and provision of snapshots. When BOTH of these actions have been completed, the next one is performed.

Recipe actions usually have parameters to specify the inputs they need in order to start. So ALARM-MESSAGES-ACQUISITION needs a block of alarms and PROVISION-OF-SNAPSHOTS needs a snap shot. These inputs are prefixed by a “>” and are bound to a particular value for a particular recipe instantiation. The values are purely local to that instantiation and disappear when it is terminated. Actions can produce results which can be assigned names (using the “<” symbol) and used elsewhere within the recipe. So the disturbance detection action takes a block of alarms as input and produces an indication of whether a disturbance has been detected as an output.

In addition to “START” the following control commands can be used within GRATE recipes:

DIRECTIVE	DESCRIPTION
(**START** task)	Start a domain level system task. The next action in the recipe is not started until this one has finished.
(**START-OFF** task)	Start a task as above, but proceed with next action immediately.
(**STOP** task)	Kill a task. After performing this action the task cannot continue from the state where it was killed, it must be restarted.
(**SUSPEND** task)	Temporally stop a task, this task can be continued from its present state using the reactivate command.
(**REACTIVATE** task)	Reactivate a task which has been suspended, starting from the state it was in when the suspend command was issued.

(**WHILE** p **DO** q)	While condition p is true do actions q.
(**WHILE-WHEN** p **DO** q **WHEN** r)	While condition p is true do actions q. When (if) it becomes false do r. The overall action finishes when q stops (if p does not become true) or when r stops (if p does).
(**GET-INFO** x)	Generate a request for a piece of information

A more complex recipe used in the electricity transport domain is the AAA's procedure for diagnosing faults. The recipe is activated when a disturbance detection message is available and when this message indicates that a fault has occurred in the network. The recipe is composed of two actions; the first is a composite and comprises starting the hypothesis generation task and indicating that information about the initial elements out of service is required. The second is the WHILE-WHEN action in which the detailed diagnosis is performed (see figure 2.6).

```
NAME: (DIAGNOSE-FAULT)
TRIGGER: (    (INFO-AVAILABLE DISTURBANCE-DETECTION-MESSAGE)
              (FAULT-DETECTED))
ACTIONS:( (   (START (HYPOTHESIS-GENERATION (> FAULT-HYPOTHESES)))
              (GET-INFO INITIAL-ELTS-OUT-OF-SERVICE)
          ((WHILE-WHEN (BLACK-OUT-AREA-NOT-AVAILABLE) DO
               (((START (HYPOTHESIS-VALIDATION
                            (< FAULT-HYPOTHESES) (> VALIDATED-HYPOTHESES)))))
              (WHEN   ((SUSPEND (HYPOTHESIS-VALIDATION)))
                      ((START (HYPOTHESIS-REFINEMENT
                            (< FAULT-HYPOTHESES) (> REFINED-HYPOTHESES))))
                      ((REACTIVATE (HYPOTHESIS-VALIDATION
                                          (< REFINED-HYPOTHESES)))))))
OUTCOME: (VALIDATED-HYPOTHESES)
```

When the plan is activated the first two actions are started concurrently. The situation assessment module instructs the control module to start the hypothesis generation task and the cooperation module to realise the need to produce the initial elements out of service (i.e. the blackout area). When the hypothesis generation task has completed, the second action related to the detailed diagnosis is started.

The action to be performed at this stage depends upon the current problem solving context. The choice is determined by evaluating the function BLACK-OUT-AREA-NOT-AVAILABLE[1]. If the function returns non-nil, the HYPOTHESIS-VALIDATION task is started. A daemon is set up to continually evaluate the black out area not available condition. If the validation task completes before the condition becomes false then the overall WHILE-WHEN action is considered finished.

However if the condition becomes false during the WHILE part (i.e. the black out area becomes available), then the current actions are abandoned and the actions of the when part started. If the condition was false initially, the WHEN actions are started immediately and no daemon is set up.

Assuming the BAI does provide the black out area during the WHILE-WHEN action, the WHEN actions are started. Firstly the currently active hypothesis validation task is suspended, this is achieved by the situation assessment module informing the control module of the desired state and the control module executing it. Once the task has been successfully suspended, the hypothesis refinement task is then started. When this finishes, the validation task is then restarted. When the validation task finishes, the WHILE-WHEN overall action is finished and the recipe is complete.

[1]. This is a LISP function defined by the application developer which either returns T or NIL and indicates whether the agent has received the black out area or not.

APPENDIX B

DETAILED DESCRIPTION OF GRATE*

CONTROL OF KNOWLEDGE BASES

GRATE* has three independent production rule systems which comprise the main problem solving modules of control, situation assessment and cooperation. Each knowledge based system has its own inference engine and local working area. Modules communicate by writing messages into each others working storage area. The situation assessment module is connected to both the others, however the cooperation and control modules can only communicate with the situation assessment module, not directly with each other.

GRATE* has a discrete event time simulator[1]. The move away from uncontrollable asynchronous behaviour was necessary because of the desire to perform repeatable experiments (see section six). The top level of the simulation can be expressed as follows.

```
loop
    deliver-messages
    simulate agent environment[2]
        forall agents in community do
            begin
                fire appropriate number of cooperation and control level rules
                do one unit of domain level system processing
            end
    update simulator clock
until end simulation.
```

In each unit of simulated time, there is a fixed number, determined by the user, of GRATE* coordination rules which can be fired. Each module has equal priority; thus it will fire approximately one third of the total number of rules. However if one of the modules is unable to use all of its allocation, any slack may be taken up by another module. Execution of the modules is not interleaved; the situation assessment module is activated first, then the cooperation module and finally the control module. In a particular module, each rule is examined in turn; starting from rule 1. If the maximum number of firings for the module is met before all rules have been examined, processing of that module is terminated prematurely.

Once a rule set has been cycled through once, the next module is invoked even though there may still be rules in the current module which could be fired because of

1. This differs from GRATE which used LISP light-weight processes to implement each of the modules, the domain level system and inter-agent message exchange.
2. Simulates events occurring in the environment with which the domain level system is interacting. Examples from the electricity transport domain include faults occurring in the network, more telemetry arriving and so on.

new deductions. Once each of the three modules has been processed, the next cycle of rule firings is undertaken and each module is again processed in sequence. This continues until either there are no more rules to fire or the maximum number of firings is met. Expressing this control algorithm more concisely:

```
Rules-Left:= *Number-Rule-Firings-Per-Time-Unit*          ;Set by system designer
Progress-Made:= True
while (Rules-Left > 0) AND Progress-Made do
    Rules-Per-Module:= if (Rules-Left > 3)
                          then TRUNC(Rules-Left / 3) else 1
    Current-Rules-Left:= Rules-Left
    for Module = (Situation-Assessment, Cooperation, Control) do
      for rule (Module) = 1 to Max-Rules(Module) do
         if can-fire(rule (Module)) AND (Rules-Left > 0)
            then  fire (rule(Module)))
                  Rules-Left:= Rules-Left - 1
    Progress-Made:= (Current-Rules-Left = Rules-Left)
end-while.
```

FUNCTIONAL SPECIFICATION

Section five highlights the main functions which need to be supported in GRATE*. Here these requirements are mapped into the rules which implement them. SARx indicates situation assessment rule number x and CRy indicates cooperation module rule y. x and y refer to the rule numbers in the listings which follow this categorization. Control module rules are not written out in full because they are not part of the main emphasis of this work - they are similar to the ones listed in section 2.4.2. Note that some rules are assigned to more than one category, this is because they cross the boundary of the two sections. The first six tasks are related to the overall functional architecture depicted in figure 5.1, the last three are specifically concerned with the process of controlling and tracking joint actions which is given in figure 5.2.

Means-End Reasoning: SAR13, SAR14, SAR17, SAR18, SAR28, SAR29, SAR30, SAR31, SAR32, SAR33, SAR35, SAR36, SAR38, CR31, CR32, CR33, CR35, CR36, CR37, CR38.

Compatibility Checker: SAR1, SAR2, SAR37

Inconsistency Resolver: SAR3, SAR4, SAR6, SAR7, SAR8, SAR9, SAR12

Executing Commitments: SAR15, SAR16, SAR19, SAR23, SAR26, SAR40, SAR41, SAR42, CR13, CR19, CR30, CR34

Establishing Social Act: CR1, CR2, CR3, CR4

Define Individual Acts: SAR2, CR5, CR6, CR7, CR8, CR9, CR10, CR11, CR12

Reasons for Terminating Joint Action: SAR20, SAR24, SAR27, SAR37, SAR39, CR14

Recovery Actions: SAR5, SAR6, SAR7, SAR10, SAR21, SAR25, SAR34, CR15, CR16, CR17, CR18, CR20, CR21, CR22, CR23, CR24, CR25, CR26, CR27, CR28, CR29

Control Domain Tasks: SAR8, SAR10, SAR11, SAR21, SAR22, SAR25.

SITUATION ASSESSMENT RULES

Rule 1

if receive a proposal to participate in a joint action

then evaluate the proposed contribution to see if it is consistent with existing commitments

Rule 2

if new solution proposal is consistent with existing commitments

then make commitment to proposed action

inform cooperation module that proposal is acceptable

Rule 3

if new solution proposal is inconsistent with existing commitments and

the new action has a lower priority

then find new time slot for action

make commitment to new action at new time

inform cooperation module of refined contribution proposal

Rule 4

if new solution proposal is inconsistent with existing commitments and

the new action has a higher priority

then commitment to new action

existing commitment has been broken

inform cooperation module that contribution proposal is acceptable

Rule 5

if social action component has been delayed and

delayed component is related to local act

then reschedule local activity

joint action plan violation has occurred

Rule 6

if commitment to a currently active joint action component is broken
then joint action has been violated

reschedule joint action to new time

Rule 7

if commitment to a future joint action is broken
then joint action has been violated
reschedule joint action component to new time

Rule 8

if a joint action component has been violated and
it has been successfully rescheduled
then suspend any current activities associated with the joint action
reset the description and timings of the joint action
inform the cooperation module that the joint action has been violated

Rule 9

if commitment to a local goal is broken
then reschedule local goal and update its description

Rule 10

if the joint action common solution is invalid
then stop all currently active actions related to the joint action
update the social action description to indicate that solution is invalid
inform cooperation module that commonly agreed solution is invalid.

Rule 11

if have to suspend all activities associated with a joint action
then check all current and future activities associated with the joint action and suspend them

Rule 12

if have to suspend local activity

then check all current and future activities associated with the action and suspend them

Rule 13

if want to start a plan and

plan should be solved by a joint action

then inform cooperation module that joint action is needed to achieve the plan

Rule 14

if joint action becomes established

then identify components which are performed locally

start any which should begin immediately

note the start times of the other local actions

Rule 15

if time to begin action that has been committed to and

commitment is still valid

then update status of joint action

start initiating the action

Rule 16

if have a local plan and

it is appropriate for it to be executed

then start initiating the action

Rule 17

if should execute a task and

task can be performed locally

then tell the control module to execute the task

Rule 18

if should execute a task and
task cannot be performed locally
then inform the cooperation module that help is needed with this task

Rule 19

if task has finished executing
then update plan for which the task is a component

Rule 20

if task which is part of a joint action plan has finished executing and
it has produced the outcomes of the overall joint action
then the objective of the joint action has been met

Rule 21

if objective of joint action has been satisfied
then inform the cooperation module that the outcome has been attained
suspend all associated local activities

Rule 22

if task has finished and
it has not produced the results which were expected
then determine whether there are alternative means by which the expected results can be produced

Rule 23

if task has finished and
task was executed as an alternative means of producing some information from an original plan
then update the original plan and continue working on it

Rule 24

if there are no alternative means for producing the desired information
then the action objective is unattainable

Rule 25

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

Rule 26

if a plan has finished successfully
then inform the cooperation module of the outcome

Rule 27

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

Rule 28

if an information request has been made and
no intentions which have been identified are capable of producing the information
then select a plan which is capable of servicing the request
set up intention to perform that plan

Rule 29

if information request made and
active intention is producing desired information
then additional motive for carrying out intention

Rule 30

if a task request is made and
the task is currently being executed by the agent and
the two tasks are compatible
then additional motive for carrying out task

Rule 31

if a task request is made and
there are no compatible tasks which are currently active
then adopt intention of fulfilling request

Rule 32

if information needed and
it cannot be produced locally
then inform the cooperation module that help is needed to provide the information

Rule 33

if information is needed and
it can be produced locally
then information should be produced locally

Rule 34

if a request for information has been honoured and the information is available
then tell the control module that the requested information is available

Rule 35

if receive unrequested information from an acquaintance
then see if it can be used to answer any outstanding local requests for information

Rule 36

if receive unrequested information and
it can be used as optional input for a task
then tell the cooperation module that some useful information is available

Rule 37

if receive unrequested information
then check it does not invalidate existing intentions

Rule 38

if receive unrequested information
then see if it can be used to trigger any new plans

Rule 39

if existing intentions no longer have valid motivations
then suspend and note the affected intentions

Rule 40

if decide to reactivate task
then inform control module that the task should be reactivated

Rule 41

if decide to suspend task
then inform control module that task should be suspended

Rule 42

if decide to abort task
then inform control module that task should be aborted

COOPERATION MODULE RULES

Rule 1

if joint action needed

then identify the components of the action

select a group of agents such that for each component action there is an agent which is capable of performing it

create a representation of the social action

checks whether the other agents are interested in participating

Rule 2

if receive a joint action proposal and

do not want (or are not able) to participate

then indicate unwillingness to contribute

Rule 3

if receive a joint a proposal and

are willing (able) to participate

then indicate willingness to contribute

Rule 4

if receive indication that an agent is willing to participate in a joint action

then update joint action model

see if all group members have responded

Rule 5

if all group members have indicated their willingness (or not) to participate

then enter next phase of planning protocol

match willing contributor’s to component actions

Rule 6

if generated an initial proposal

then start agreeing proposal with each team member

Rule 7

if next joint action component is to be performed locally

then request situation assessment module to evaluate its consistency

Rule 8

if receive a contribution proposal from the team leader

then note changes in status of joint action

request situation assessment module to evaluate its consistency

Rule 9

if have a statement about participation in a joint action from the situation assessment module

then pass this onto the team leader

Rule 10

if receive a contribution proposal response indication from a team member and the response is positive

then update description of joint action

see if all members have now responded

Rule 11

if receive a contribution proposal response indication from a team member and the response is negative

then modify proposals to take into account the non-participation of that team member on the specified task

Rule 12

if the common solution has been agreed by all participants

then update description of joint action to indicate solution execution phase

inform team members of joint action establishment

Rule 13

if receive notification that joint action has been established

then update description to indicate solution execution phase

inform situation assessment module that execution of joint action can begin

Rule 14

if a receive notification that joint action is completed
then inform situation assessment component

Rule 15

if local joint action violation has occurred
then inform other team members of this fact and of any remedial actions taken

Rule 16

if solution commitment failure for the agreed solution is reported
then update associated actions timings

Rule 17

if joint action has successfully finished
then inform all other team members of successful completion
see if results should be disseminated to agents outside the team

Rule 18

if goal request has finished
then inform originator that request has been satisfied
see if results should be disseminated to other agents

Rule 19

if local plan has finished
then see if results should be disseminated to other agents

Rule 20

if information request has finished
then inform originator that request has been satisfied
see if information should be disseminated to other agents

Rule 21

if receive indication that joint action has successfully completed
then tell situation assessment module to suspend associated activities

Rule 22

if receive local indication that joint action motivation is not present
then inform other team members of need to abandon joint act

Rule 23

if acquaintance indicates that joint action no longer has a valid motivation
then tell situation assessment module to suspend associated processing

Rule 24

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

Rule 25

if acquaintance indicates that joint action is unattainable
then tell situation assessment module to suspend associated processing

Rule 26

if receive local indication that the agreed common solution is invalid
then inform others of need to abandon joint solution

Rule 27

if acquaintance indicates that common solution is invalid
then tell situation assessment module to suspend associated processing

Rule 28

if receive local indication that the common plan is unattainable
then inform others of need to abandon joint solution

Rule 29

if acquaintance indicates that the common plan is unattainable
then tell situation assessment module to suspend associated processing

Rule 30

if need to acknowledge that a task has reached a coordination point
then inform team members that coordination point has been reached

Rule 31

if help with a task is required
then select and identify acquaintance who is capable of performing the task
make request to chosen agent
update acquaintance model of chosen agent to indicate its new intention

Rule 32

if goal request is received and
are potentially capable of honoring the request
then inform situation assessment that goal request has been made

Rule 33

if goal request is received and
are not capable of honouring it
then inform originating agent that request cannot be honoured

Rule 34

if receive acknowledgment that a redone task has been completed by an acquaintance
then inform situation assessment component that request has been honoured

Rule 35

if information is needed
then identify acquaintances who are potentially capable of supplying it
select and make request to chosen agent

Rule 36

if receive a request for information and
relevant information is already available to the agent
then return available information to the requesting agent

Rule 37

if receive a request for information and
relevant information is not currently available and
the agent can potentially produce the information
then inform situation assessment module that request for information has been made

Rule 38

if receive a response to a request for information
then inform situation assessment component that requested information is available