An Agent Architecture for Distributed Medical Care

Jun Huang[1], N. R. Jennings[2] and John Fox[3]

[1] Dept. of Computing, University of Central Lancashire, Preston PR1 2HE, UK.

[2]Dept. of Electronic Engineering, Queen Mary & Westfield College, Mile End Road, London E1 4NS, UK.

[3]Advanced Computation Laboratory, Imperial Cancer Research Fund, 61 Lincoln's Inn Fields, London WC2A 3PX, UK.

Abstract. This paper describes the design and implementation of a layered agent architecture for decision support applications in general and for distributed medical care in particular. Three important characteristics which shaped the agent design are identified: distribution of data and control, information uncertainty, and environment dynamism. To provide appropriate decision support in these circumstances the architecture combines a number of AI and agent techniques: a symbolic decision procedure for decision making with incomplete and contradictory information, a concept of accountability for task allocation, commitments and conventions for managing coherent cooperation, and a set of communication primitives for inter-agent interaction.

1. Introduction

Artificial Intelligence and knowledge based systems are assuming an increasingly important role in medicine for assisting clinical staff in making decisions under uncertainty (eg diagnosis decisions, therapy and test selection, and prescribing). Furthermore, many medical procedures now involve several individuals, in a number of specialist departments, whose decisions and actions need to be coordinated if the care is to be effective and efficient [1, 2, 3]. For example, a general practitioner (GP) may suspect that his patient has breast cancer. However, as he neither has the knowledge nor the resources to confirm this hypothesis, he must refer the patient to a hospital specialist who can make a firm diagnosis. Having confirmed the presence of breast cancer, the specialist must devise a care programme for treating the patient - this typically involves hospital nurses, the patient's GP, and a home care organisation jointly executing a series of interrelated tasks.

To provide the appropriate software support for such coordinated health care management it was decided to adopt an agent-based approach. This decision was based on three main observations about the medical care management domain (given below) and the properties of autonomy, social ability, reactivity and proactiveness which are normally associated with intelligent agents [4]. The first relevant domain property is the fact that there is a significant physical distribution of information, problem-solving capabilities, resources, and responsibilities which need to be brought together in a consistent and coherent fashion by the distributed ‘agents’ who jointly execute a care programme (here agent is defined as an integrated entity involving a computer system and its user). Secondly, the combination of the aforementioned decentralisation and the high cost of obtaining a comprehensive overview means that many different, partial and overlapping problem solving models exist within the community - hence the ability to reason with contradictory and incomplete information and knowledge is essential. Finally, as the environment is highly dynamic and unpredictable the problem solvers need to exhibit intelligent goal-oriented behaviour yet still be responsive to changes in their circumstances - plans to achieve particular goals need to be devised and whilst these plans are being executed they need to be continuously monitored (and perhaps refined) in the light of changes in information and problem solving state.

Given these domain properties and previous experience with medical care management systems it was possible to identify the main problems which the new agent-based system had to overcome and the additional functionalities which needed to be offered to provide the desired degree of decision support. Firstly, the system needs explicit communication management (dealing with both syntax and semantics) so that the sender and receiver of a message have a common understanding of its meaning and purpose (previously messages were often misinterpreted during extensive interactions because of ambiguities in the communication structures). Secondly, appropriate mechanisms and structures are needed to ensure that tasks are delegated to the most appropriate agents (previously tasks could be allocated to the wrong agents and thus delays in the delivery of care occurred - a serious concern when time is a critical factor in care administration). Thirdly, a decision making mechanism which is able to reason with contradictory and incomplete information is needed (previously the popularly used decision methods, especially those based on probabilistic theory, could not tolerate conflicting or incomplete information). Finally, to ensure coherent care in spite of the dynamic and unpredictable environment an explicit set of procedures for monitoring an agent’s goals and plans needs to be specified and adopted by all the agents (previously no explicit procedures existed and changes in goals and care plans were managed largely in an *ad-hoc* and ineffective manner).

The paper is structured in the following manner: section 2 describes the agent architecture and the inter-agent communication structures which were developed in the course of this work. The architecture itself is based on a three-layer knowledge organisation (domain layer, inference layer and control layer) and is informed by work on The Oxford System of Medicine [5] and the KADS model of expertise [6]. The remainder of the paper deals in turn with each of the key agent functionalities described above - section 3 with the decision making mechanisms, section 4 with the task allocation mechanisms, and section 5 with the cooperation management mechanisms. Finally, section 6 compares the developed agent architecture with related work.

2. The Agent Architecture

The agent architecture comprises multiple layers of knowledge, a working memory, a communications manager and a human-computer interface (see figure 1). To be successful in this domain, the agent needs to exhibit both deliberative behaviour (eg plan selection, task decomposition, and task allocation) and reactive behaviour (eg respond in a timely manner to the arrival of new data, to changes in existing data, and to varying agent commitments). Within the proposed architecture the deliberative behaviour is achieved by the incorporation of decision rules for plan selection, task management rules for task decomposition and allocation, and cooperation rules for formulating commitments. The reactive behaviour is achieved by the control layer which responds to changes in the working memory (e.g. the arrival of new task results, goals, or messages or changes in existing data, goals, agent commitments or task states). The remainder of this section describes each of the main components of the architecture in more detail.

The three *layers of knowledge* which form the key part of the agent architecture are as follows:

- Domain knowledge - includes: a knowledge base covering specific medical domains such as breast cancer, a knowledge base of clinical management plans (known as *clinical protocols* [7]), a database of patient records, and a database of resource availability.

- Inference knowledge - in the form of generic, declarative inference rules which apply domain knowledge to specific patient cases to infer new data. Inference rules represent the core of the agent architecture and are subdivided into those for decision making under uncertainty (section 3), those for task management (section 4), and those for managing agent cooperation (section 5).

- Control knowledge - which applies the inference knowledge to the domain knowledge in order to generate inferences whenever new data is added to the working memory.

In more detail, the domain knowledge base simply states information and facts about the domain. It says nothing about how the knowledge is to be used. For example, it states that a task called ‘1st-cycle EMV chemotherapy’ contains two subtasks, ‘inject cytotoxic drugs’ and ‘measure patient temperature’:

```
component(`1st-cycle EMV chemotherapy',
                   'inject cytotoxic drugs').
component(`1st-cycle EMV chemotherapy',
                   'measure patient temperature').
```

The inference knowledge base contains rules (implemented as declarative schemas) that specify the inference relations between domain-level knowledge and possible new information. For example, the task management module contains a declarative inference schema for managing task state transitions:

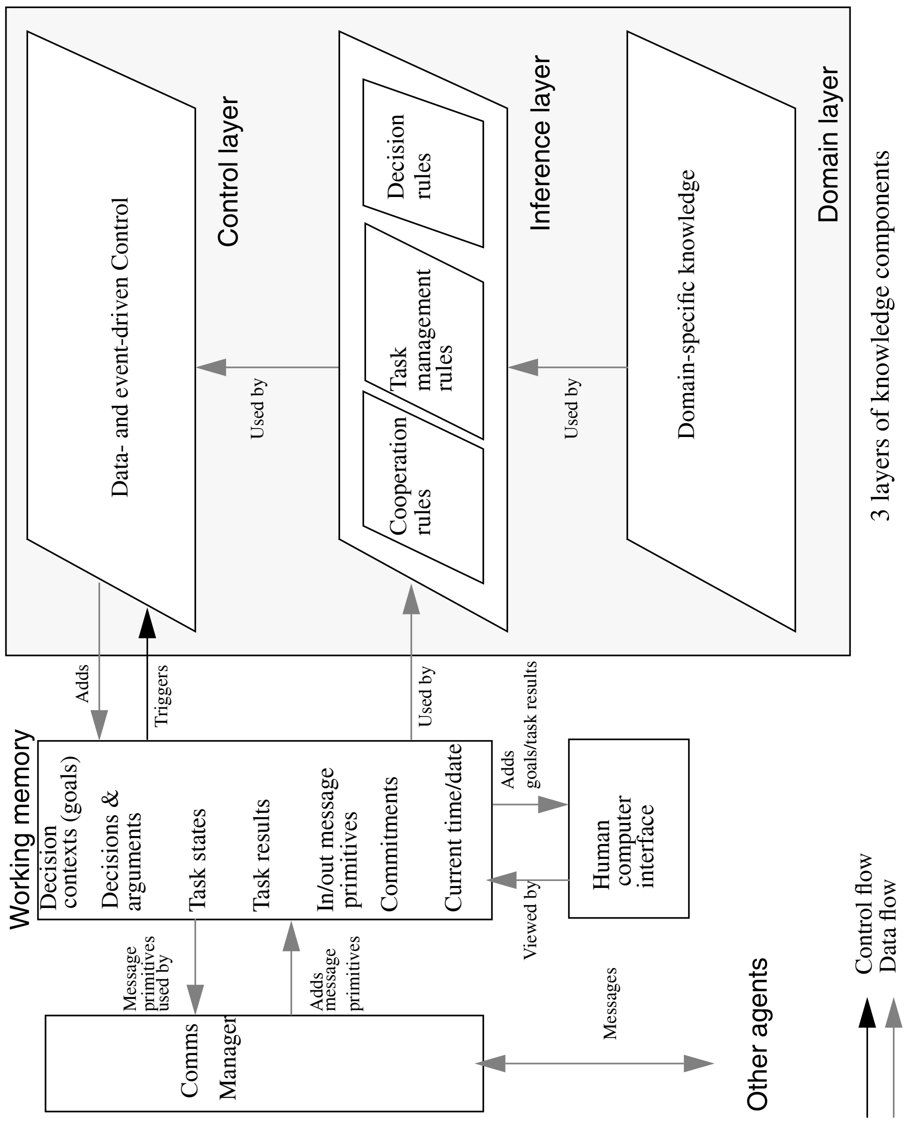

Figure 1: Agent architecture for distributed medical care

```
schema(conditions(component(Task, SubTask) and
                  state(SubTask, started) ),
            conclusions(state(Task, started) ).
```

However, it is only at the control level that the actual execution of the inference rules is carried out and new data is added into the working memory:

```
If schema(Conditions, Conclusions) and
   all_true(Conditions)
then add(Conclusions)
```

There are two main reasons for adopting this functional and logical separation of domain, inference and control knowledge. Firstly, it simplifies the representation, reuse and maintenance of knowledge. Inference knowledge for decision making, task management and cooperation is represented independently of medical domains and can therefore be reused; control knowledge is represented independently of the inference knowledge and so the same control rules can be applied to the different groups of inference rules. Furthermore, modifications to domain knowledge can be made independently of inference and control knowledge. The second main reason for such a separation is that it provides a convenient basis for knowledge elicitation: domain knowledge can be acquired and modified independently of inference and control knowledge.

The *working memory* stores temporary data generated by the control layer, the user, or the communications manager. Examples of the types of information which need to be stored include: goals to be achieved, control states of tasks that are currently active, results of completed tasks, incoming and outgoing messages, and current commitments. Its function is similar to that of a global blackboard, on to which new information (or any change which triggers reactions by the control layer) can be added.

The *communications manager* composes the messages to be sent to the other agents from the primitives produced by firing task management rules or cooperation rules (see sections 4 and 5 for respective examples). The primitives each have a type and a content as well as a specified effect on the recipient - they include: request, accept, reject, alter, suggest, inform, query, cancel and acknowledge [8]. Request, accept, reject and alter are used during the allocation of tasks and the formulation of agreed courses of action. A suggest act may be the result of a query. Inform usually follows an accepted request to perform a certain task and is typically used to return the results of the task back to the originator. Cancel is included because in certain circumstances agents may not wish to adhere to the agreed plan of action (see section 5.2). Finally, all messages must be acknowledged. The communication manager also converts messages which arrive from other agents into primitives that may be used by the cooperation manager.

The *human computer interface* defines a scheme for interaction between the system and its user. The approach is as follows: the computer is capable of performing various functions (i.e. decision making, task management, communication and cooperation) but may not act autonomously on all of these capabilities. In general, the computer informs the user of the results of its inferences and the user must then endorse or authorise them before they can be communicated to external agents. For example, the system may recommend to a GP that he refers a patient to a hospital consultant for a particular course of treatment, but the GP may have a personal preference for another consultant and therefore be unwilling to make such a referral. In this case, the system will not send an electronic request to the consultant and will instead offer the GP an alternative solution.

3. Decision Making Under Uncertainty

The purpose of the decision rules is to choose among alternative options (e.g. potential diagnoses of a patient's illness, potential clinical protocols which could be used to treat the patient, etc.). As well as being used to decide which course of action to start, these rules may also be embedded as a decision point within the body of an action - eg whilst executing a particular clinical protocol there may be a crucial decision to be made which needs to make use of the decision making know-how contained in this rule group (see section 4 for an illustration of this point with respect to prescribing).

In this application, the decision making is often complicated by the presence of incomplete or even conflicting information. For example, a drug may be very effective for eliminating a tumour, but the patient may be unwilling to tolerate its side effects such as loss of hair. To facilitate decision making in such a context, a domain-independent decision procedure is abstracted and separated from domain-specific knowledge: the same set of decision rules can then be used to make decisions in varying medical domains (such as cancer, diabetes, cardiology etc.). Such a separation also permits formalisation of the decision knowledge.

The starting point of a decision making session is a goal, represented as a *decision context*, which is either given by the user or generated by the task management rules (section 4). By way of an example, the agent could have a goal of deciding which clinical protocol to select to treat a patient with breast cancer. Given this context, there are several distinct components of the decision procedure. The primary component activities are *proposing* candidate decision options, *refining* candidates, *arguing* the pros and cons of the options in view of the available evidence (argument generation), and *aggregating* the arguments to determine the preferred option (argument aggregation). For instance, the use of chemotherapy and radiotherapy may be proposed to treat the breast cancer of an old-aged patient (proposing). These options may then be refined to specific chemotherapy and radiotherapy treatments (refining). Arguments supporting the use of a particular chemotherapy treatment may include its effectiveness for removing the cancer, but there may also be arguments against its use (e.g. the level of toxicity associated with the drug may be too high for this particular patient due to her age). The pros and cons for each proposed option are finally combined to give a most preferred decision - e.g. the decision to use FB1 chemotherapy (argument aggregation).

The decision procedure is based on a simple but flexible method of reasoning under uncertainty for argument generation and aggregation, called *argumentation* [9], which avoids the necessity for precise quantification of uncertainty. Argumentation involves two simple ideas. First, one may know that some piece of information increases one's belief in a diagnosis, or preference for an action, though one may not be able to put a precise number on the change. Arguments for options can be constructed that are qualitatively labelled to indicate this change - for example, "confirm", "support", "weaken" or "exclude". Arguments of this sort are similar to Cohen's endorsements [10], but in this work a more sophisticated set of aggregation functions are used to combine collections of arguments to yield a preference ordering on the decision options. This method is versatile, conceptually intuitive, easy to implement and simplifies some of the problems of knowledge acquisition and maintenance. Second, the grounds of arguments for and against decisions are explicitly represented - this means they can serve a variety of functions including truth maintenance and explanation.

Details of this approach to decision making are given elsewhere - for example, [11] describes it within a general context of qualitative reasoning and [12] gives a more formal, declarative specification of the decision procedure and discusses its application in medical decision making - and therefore will not be elaborated upon here. The emphasis in this paper is on the use of this generic decision knowledge alongside task management and cooperation knowledge in an integrated agent architecture for coordinated care. For example, the decision rules select an appropriate clinical protocol, which is then decomposed, allocated and monitored by the task management and cooperation rules.

4. Task Management

Once the decision procedure has selected a particular clinical protocol to achieve the agent's goal the task manager component is responsible for its decomposition into subtasks, the allocation of subtasks to appropriate agents, and the management of task state transitions. Each of these activities is described in turn in the remainder of this section.

The structure of a generic clinical plan (e.g. for treating breast cancer) is determined by experts in authority, and is precisely defined in a clinical protocol. The task management rules decompose such a protocol into subtasks according to the predefined plan structure. Subtasks at the bottom of the plan hierarchy may be primitive actions for humans or machines to perform (such as measuring a patient's temperature) or they may be decision tasks (such as choosing the right cytotoxic drug for a breast cancer patient). In the latter case, the decision procedure is used to perform such a task, as explained in the previous section.

To facilitate task allocation, there are two roles associated with each (sub)task within the system - there is one agent who *manages* the execution of the task (i.e. ensures that it gets executed by somebody within the system and that the result of the execution will be sent back to the originator) and one agent who is actually responsible for *performing* the task (the contractor). Task allocation is, therefore, the process by which the manager of a task finds the most appropriate contractor to perform it. The key structure in making such decisions is that of *accountability.*

Accountability is a static relationship which defines what and to whom an agent is responsible. It is expressed by the following relation: `accountable(Agent1, Agent2, TaskType)` which means that Agent1 is accountable to Agent2 for performing tasks of type TaskType. For example, a hospital nurse may be accountable to one or more doctors for monitoring patient data such as temperature and blood pressure. The task manager component uses its accountability relations, together with the generic inference rule given below, to pick the most appropriate contractor for a given task. The underlined term "request" represents a primitive which is sent to the communications manager when this task management rule is fired (as described in section 2).

```
IF Task is necessary &
   Task is of type TaskType &
   Acquaintance is accountable to Agent for tasks of
         TaskType &
   Agent prefers to interact with Acquaintance
         concerning TaskType
THEN request (Agent, Acquaintance, perform (Task))
```

All tasks within the system have a state (either scheduled, cancelled, started, completed, or abandoned). The management of the transitions between these states needs to be carefully controlled by the agents because such transitions need to be documented in patients' care records: for example, when a task was scheduled, when it was started, when it was completed and when (why) it was abandoned. Transition management is complicated by the complex structure of the care plans. For example, the following two task management rules specify that when a composite task is cancelled, its started subtasks become abandoned and the subtasks that are scheduled but not yet started become cancelled:

```
schema(conditions(state(Task, cancelled), and
                  component(Task, SubTask) and
                  state(SubTask, started) ),
      conclusions(state(SubTask, abandoned) ).

schema(conditions(state(Task, cancelled), and
                  component(Task, SubTask) and
                  state(SubTask, scheduled) ),
      conclusions(state(SubTask, cancelled) ).
```

A distinction is made between the states of cancelled and abandoned because a corrective action is usually needed for an abandoned task (e.g. when the patient has to stop taking a certain drug which he has already been taking for a period) whereas such action is not normally necessary for a cancelled task.

5. Managing Agent Cooperation

The underlying mechanisms on which cooperative interactions are based are those of *commitment* (pledge to undertake a specified course of action) and *convention* (means of monitoring commitments in changing circumstances) [13]. The former means that if an agent agrees to undertake a task then it will endeavour to execute it at the appropriate time - this implies both that the agent is able to perform the task and that it has the necessary resources. Conventions are needed because commitments are not irrevocable: agents' circumstances may change between the making and the execution of their commitments, and agreed actions may turn out to be undesirable or even impossible to perform. Conventions, therefore, define the conditions under which an agent can drop its commitments and how to behave with respect to other agents in the cooperating group when such circumstances arise.

Given that cooperation is founded on commitments and conventions, two key issues need to be addressed: (i) what is involved in establishing a commitment? (section 5.1); and (ii) what type of convention is appropriate for monitoring commitments in the given care organisation? (section 5.2).

5.1 Establishing commitments

Accountability alone does not guarantee commitment: to commit to a specified task, an agent must also have the necessary resources (temporal and material) which are required to perform that task[1]. For example, a hospital specialist may be accountable to patients for breast cancer surgery, but will not become committed to surgery on a specific patient until the time (temporal resource) and equipment (material resource) are available to perform the operation. Although agents know what resources are available to themselves, they do not generally have information about the resources of their acquaintances. Therefore a task may have to be iteratively delegated to a number of acquaintances until a specific agent becomes committed.

When an agent accepts a request it becomes committed to performing it (i.e. it takes on the role of contractor) and informs the manager that the task has been accepted using the following inference rule:

```
IF   Acquaintance is requested by Agent to perform Task &
     Acquaintance accountable to Agent for TaskType tasks&
     Task is of type TaskType &
     Task requires Resources &
     Resources are available to Acquaintance
THEN   Acquaintance becomes committed to Task, AND
       accept(Acquaintance, Task, for(Agent) )
```

[1] In addition, an agent may also have a local policy governing the acceptability of a requested task. For example, a hospital may specify the following internal policy: a patient can only be admitted to the hospital if his/her GP is suitably registered with the hospital (so that the hospital can be paid more quickly). The capture and use of these policies remains a challenge to computer-assisted care and so, for the sake of simplicity, it is assumed here that availability of the appropriate resources is the only requirement for an agent to commit to a task.

Commitment to the role of contractor also entails an additional responsibility - when the task has been completed the contractor must inform the manager about it and any results which have been generated - again this behaviour is encoded in a generic inference rule:

```
IF  Task is completed and it produces Results &
    Acquaintance is committed to Agent for Task
THEN   inform (Acquaintance, Agent, performed(Task),
                         results-produced(Task, Results))
```

Note that in both cases, the underlined term represents primitives sent to the communication manager when the appropriate inference rules are triggered (as described in section 2).

5.2 Adaptive management of commitment changes

In most cases, when an agent commits itself to perform a task then that task will indeed be executed. However in certain well-defined circumstances it may be appropriate for an agent to renege upon its commitment. There may be an unforeseen lack of resources (e.g. unrelated emergencies may arise), the need for the task may cease to exist (e.g. because of the unexpected death of the patient), or it may no longer be feasible to execute a given task (e.g. a planned chemotherapy may have to be withdrawn because the patient has a high temperature resulting from the toxic effect of the drug). Having detailed the conditions under which commitments can be cancelled, the convention must also specify how to manage this change both locally and within the wider context of the cooperating group. The latter is important because it ensures that the cooperating care agents will behave coherently in the face of dynamic and unpredictable changes in the network [14]. Figure 2 details the convention embodied in the cooperation manager in this application.

6. Related Work

In this section our architecture is briefly compared with some of the well-known architectures and systems in the literature.

GRATE [15] is also a layered architecture that provides a generic cooperation module, situation assessment module, control module, and application-specific module. However the GRATE framework lacks an uncertainty management mechanism which is essential for medical decision making. Also GRATE’s layers are functionally separated rather than logically separated - the additional benefit of this logical separation is that it provides a convenient basis for declarative specification and for logical verification and validation of the various layers of knowledge (eg in a formal language such as ML^2 [16]). The same two observations can be made of similar layered architectures presented in this volume, such as INTERRAP [17] and TouringMachines [18].

```
REASONS FOR RE-ASSESSING COMMITMENTS TO A TASK:

      • Task is no longer necessary
      • Resources for Task become unavailable
      • Commitment to the super-task of Task is dropped

ACTIONS:

R1: IF Manager of Task believes Task is no longer necessary
   THEN  request (Manager, Contractor,
                             drop-commitment(Contractor, Task))

R2: IF Contractor for Task believes Task is no longer necessary
            for a certain Reason
   THEN inform(Contractor, Manager, unnecessary(Task, Reason))

R3: IF Contractor for Task drops commitment to Task, AND
       Task has a SubTask
   THEN request (Contractor(Task), Contractor(SubTask),
                drop-commitment(Contractor(SubTask), SubTask))

R4: IF Resources allocated to Task become unavailable
   THEN   Contractor for Task drops his commitment to Task &
          inform (Contractor, Manager(Task),
                    drop-commitment(Contractor, Task, Reason))

R5: IF Manager of Task is informed that Contractor for Task is
          no longer committed to Task, AND
       Manager believes that Task is still necessary, AND
       Manager has another accountable Acquaintance for Task
   THEN request (Manager, Acquaintance, perform (Task))
```

Figure 2. Convention for Adapting Commitments

Coordinator [19] is a conversational system for coordinated action which is based on Searle's speech act theory [20]. However, whilst the generation and monitoring of speech acts and commitments are centralised in Coordinator, our architecture distributes both of these functions (thus helping to reduce the communication bottleneck). Also the functionality of Coordinator is limited to coordination alone through the generation of speech acts and commitments, whereas our architecture accommodates additional functions such as a generic decision module for decision making under uncertainty.

Our proposal also bears certain similarities to a standard blackboard architecture [21]. In both cases the working memory is changed through the application of functionally separated modules of inference rules. However, in addition to functional separation, our architecture also emphasises the logical layering of knowledge for reasons stated above and provides a set of generic knowledge modules for cooperation and decision making.

7. Conclusions

This paper has described the design and implementation of an agent architecture for distributed medical care management (an application area which is representative of an important class of real world problems). A prototype system has been developed for the specific application of distributed management of cancer patients among general practices, hospitals, home care organisations and pharmacies. PROLOG is used for the representation of the domain- and inference- layer knowledge, and a production-rule language, implemented in PROLOG, is used for the data-driven control. A standard email system and server is used for message passing among the care agents.

Preliminary evaluation of this prototype indicates that in real clinical application settings where exact probabilities and utilities are difficult to obtain the built-in symbolic decision procedure is more effective than conventional numerical methods [22]. Also a senior oncologist manager and a senior cardiologist manager concluded that the cooperation strategy would provide useful guidance for clinicians jointly executing a care programme.

Acknowledgements

The first and third authors are grateful to the EC Advanced Informatics in Medicine (AIM) programme for funding the DILEMMA project, in which this work was carried out. The role of the second author has been to assist in the design and specification of the cooperation and communication mechanisms. All the authors would like to thank Richard Thomson and the two anonymous reviewers for their helpful comments on earlier drafts of this paper.

References

[1] Pritchard, P., (1992) “The role of computers in referral”, in *Referrals to Medical Outpatients* (eds. Hopkins, A. & Wallace, P.*)*, Royal Colleges of Physicians and General Practitioners, pp 79-89

[2] Reeves, P., Rickards, T., and Carniel, B., (1993) “Requirements for Shared Care Decision Support in Cardiology”, Technical Report, Royal Brompton National Heart and Lung Hospital, London, England.

[3] Renaud-Salis, J. L., Lagouarde, P., Gordon, C., and Thomson, R., (1992), “Requirements for Decision Support in Cancer Shared Care”, Technical Report, Fondation Bergonie, Bordeaux, France.

[4] Wooldridge, M. J., and Jennings, N. R., (1995) “Agent Theories, Architectures and Languages: A Survey” In this volume.

[5] Fox, J., Glowinski, A., Gordon, C., Hajnal, A., and O'Neil, M., (1990) "Logic Engineering for Knowledge Engineering: Design and Implementation of the Oxford System of Medicine", *Artificial Intelligence in Medicine* 2, pp 323-339.

[6] 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, Chichester.

[7] Gordon, C., Herbert, S. I., Jackson-Smale, A., and Renaud-Salis, J. L., (1993) "Care protocols and healthcare informatics", *Proc. of Artificial Intelligence in Medicine Europe 93*, Munich, Germany, pp 289-309.

[8] Huang, J., Jennings, N. R., and Fox, J., (1994) "Cooperation in Distributed Medical Care", *Proc. of Second Int. Conf. on Cooperative Information Systems*, Toronto, Canada, pp 255-263.

[9] Krause, P., Amble, S., and Fox, J., (1993), "The Development of a Logic of Argumentation", in: *Advanced Methods in Artificial Intelligence*, Lecture Notes in Computer Science Series, Springer-Verlag, Berlin

[10] Cohen, P. R., (1985) "*Reasoning about Uncertainty: An Artificial Intelligence Approach*", Pitman, London.

[11] Fox, J., and Krause, P., (1992) "Qualitative frameworks for decision support: lessons from medicine", *Knowledge Engineering Review* 7, pp 19-33.

[12] Huang J., Fox J., Gordon C., and Jackson-Smale A., (1993) "Symbolic decision support in Medical Care", *Artificial Intelligence in Medicine* 5, pp 415-430.

[13] Jennings N. R., (1993) "Commitments and conventions: the foundation of coordination in multi-agent systems", *Knowledge Engineering Review 8*, pp 223-250.

[14] Jennings, N. R., (1995) "Controlling Cooperative Problem Solving in Industrial Multi-Agent Systems using Joint Intentions" *Artificial Intelligence* 74 (2).

[15] Jennings N. R., Mamdani E. H., Laresgoiti I., Perez J., and Corera J., (1992) "GRATE: a general framework for cooperative problem solving", *Intelligent Systems Engineering* 1(2) pp 102-114.

[16] van Harmelen, F., (1992) "ML^2: a Formal Language for KADS Models of Expertise", *Knowledge Acquisition* 4(1).

[17] Muller, J. P., Pischel, M., and Thiel, M., (1995), "A pragmatic approach to modelling autonomous interacting systems: a preliminary report" In this volume.

[18] Ferguson, I. A. (1995), "Integrated control and coordinated behaviour: a case for agent models" In this volume.

[19] Winograd, T., and Flores, F., (1986) "*Understanding Computers and Cognition: a New Foundation for Design"* Ablex Publishing, Norwood.

[20] Searle, J. R., (1969) "*Speech Acts: an Essay in the Philosophy of Language"* Cambridge University Press.

[21] Engelmore, R., and Morgan, T., (1988) "*Blackboard Systems"* Addison-Wesley.

[22] Walton, R., and Randall, A., (1992), "Clinical Decision Analysis", *British Medical Journal* 301, p.301