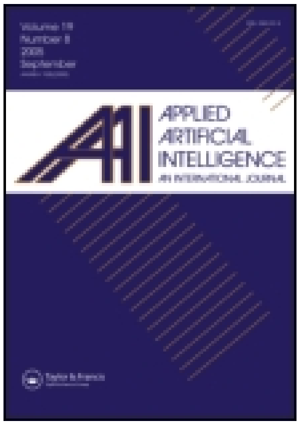

Applied Artificial Intelligence: An International Journal
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/uaai20

# AGENT-BASED APPROACH TO HEALTH CARE MANAGEMENT

JUN HUANG [a] , N. R. JENNINGS [b] & JOHN FOX [c]

[a] Department of Computing, University of Central Lancashire, Preston, United Kingdom

[b] Department of Electronic Engineering, Queen Mary and Westfield College, London, United Kingdom

[c] Advanced Computation Laboratory, Imperial Cancer Research Fund, London, United Kingdom

Version of record first published: 15 May 2007.

To cite this article: JUN HUANG , N. R. JENNINGS & JOHN FOX (1995): AGENT-BASED APPROACH TO HEALTH CARE MANAGEMENT, Applied Artificial Intelligence: An International Journal, 9:4, 401-420

To link to this article: http://dx.doi.org/10.1080/08839519508945482

# AGENT-BASED APPROACH TO HEALTH CARE MANAGEMENT

JUN HUANG
Department of Computing, University of Central Lancashire, Preston, United Kingdom

N. R. JENNINGS
Department of Electronic Engineering, Queen Mary and Westfield College, London, United Kingdom

JOHN FOX
Advanced Computation Laboratory, Imperial Cancer Research Fund, London, United Kingdom



*The provision of medical care typically involves a number of individuals, located in a number of different institutions, whose decisions and actions need to be coordinated if the care is to be effective and efficient. To facilitate this decision making and to ensure the coordination process runs smoothly, the use of software support is becoming increasingly widespread. To this end, this paper describes an agent-based system that was developed to help manage the care process in real-world settings. The agents themselves are implemented using a layered architecture, called AADCare, which combines a number of AI and agent techniques: a symbolic decision procedure for decision making with incomplete and conflicting information, a concept of accountability for task allocation, the notions of commitments and conventions for managing coherent cooperation, and a set of communication primitives for interagent interaction. The utility of this approach is demonstrated through the development of an application prototype for the clinical process of cancer treatment.*

Artificial intelligence and knowledge-based systems are assuming an increasingly important role in medicine for assisting clinical staff in making decisions under uncertainty (e.g., diagnosis decisions, therapy and test selection, and drug prescribing). Furthermore, many medical procedures now involve several individuals, in a number of specialist institutions (or departments), whose decisions and actions need to be coordinated if the care is to be effective and efficient (Pritchard, 1992; Reeves et al., 1993; Renaud-Salis et al., 1992). 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 program for treating the patient.

The first and third authors are grateful to the EC Advanced Informatics in Medicine (AIM) program for funding the DILEMMA project in which this work was carried out. The role of the second author has been to assist in the cooperation and communication aspects of the work.

Address correspondence to Dr. Nick Jennings, Department of Electronic Engineering, Queen Mary and Westfield College, Mile End Road, London E1 4NS, UK. E-mail: N.R.Jennings@qmw.ac.uk.

Applied Artificial Intelligence, 9:401–420, 1995

0883-9514/95 $10.00 + .00

This effort typically involves the hospital, the patient's GP, and a home care organization jointly executing a series of interrelated tasks. In addition to this interorganization coordination, there is also a need to ensure that the activities within an organization are effectively and coherently managed. In a hospital, for instance, care typically involves execution of interrelated tasks by doctors, nurses, pharmacy, laboratories, and resource management departments.

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 (Wooldridge & Jennings, 1995). The first relevant domain property is the fact that there is a significant physical distribution of information, problem-solving capabilities, resources, and responsibilities that need to be brought together in a consistent and coherent fashion by the distributed "agents" who jointly execute a care program (here, "agent" is defined as an integrated entity involving a computer system and its user). Second, the combination of the aforementioned decentralization and the high cost of obtaining a comprehensive (complete) overview means that decisions often have to be made with incomplete information (e.g., diagnosis may be proposed without exhaustive laboratory investigation). Finally, as the environment is dynamic and unpredictable, the problem solvers need to exhibit intelligent goal-oriented behavior 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 the problem-solving state.

Given these domain properties and previous experience with medical care management systems, the essential features of an agent-based system for this application area can be defined. First, the agents need explicit communication management procedures (dealing with both syntax and semantics), so that the sender and receiver of a message have a common understanding of its meaning and purpose (in nonautomated systems, human-to-human messages were often misinterpreted during extensive interactions because of ambiguities in the communication structures). Second, appropriate mechanisms and structures are needed to ensure that tasks are delegated to the most appropriate agents (previously, tasks were allocated to the wrong agents and, thus, delays in the delivery of care occurred—a serious concern, as time is a critical factor in care administration). Third, the agents require a decision making mechanism that is able to reason with contradictory and incomplete information (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, the agents need to specify and adopt an explicit set of procedures for monitoring their goals and plans (previously, no explicit procedures existed, and changes in goals and care plans were managed largely in an ad hoc and ineffective manner).

The remainder of this article is structured in the following manner: the following section presents a real-world clinical scenario of distributed medical care that is used throughout the remainder of the article to illustrate the key agent concepts. Then the agent architecture, called AADCare is described. AADCare is based on a three-layer knowledge organization (domain layer, inference layer, and control layer) and is informed by work on the Oxford System of Medicine (Fox et al., 1990) and the KADS model of expertise (Hickman et al., 1989). This section deals, in turn, with each of the key agent features that were identified above. Finally, AADCare is compared with related work.

## CLINICAL SCENARIO OF DISTRIBUTED CARE

The scenario presented in this section is based on an actual clinical case provided by Foundation Bergonie of Bordeaux, France (Renaud-Salis et al., 1992). All of the interactions described herein have been implemented using AADCare agents.

When an oncologist in a cancer hospital has to treat a patient's breast cancer, the first decision he has to make concerns the treatment plan that will be adopted. To assist him in making this decision, the oncologist consults one of his decision support systems. This system has a built-in decision procedure that is able to deal with incomplete or intuitively inconsistent information, such as evidence in favor of a choice and evidence against the choice (see section below on symbolic decision making for more details). Having weighted the pros and cons of using various treatment options, the system recommends the use of a particular chemotherapy protocol called "CT1 protocol." The oncologist authorizes use of this protocol and requests the computer system to support him in carrying out the treatment.

The CT1 protocol consists of a number of treatment stages, stage 2 of which is shown in Figure 1. According to the protocol, stage 2 is decomposed into a sequence of three subtasks: admit patient to hospital, administer drugs and monitor patient, and discharge patient. The support system recommends that the oncologist carries out the first task because it knows that he is formally responsible for the admission of his patients. This recommendation is endorsed by the oncologist, and consequently, he takes on the role of managing and actually performing the activity. On the recommendation of his support system, the oncologist then decomposes "admit patient to hospital" into two parallel subtasks: "allocate bed" and "obtain patient consent." The machine recommends, and the oncologist accepts, that "allocate bed" should be performed by the hospital's resource management department and "obtain patient consent" should be carried out by the oncologist. With respect to the former subtask, the oncologist sends an electronic request to the resource department to see whether they are willing to take on the responsibility for performing it. Assuming

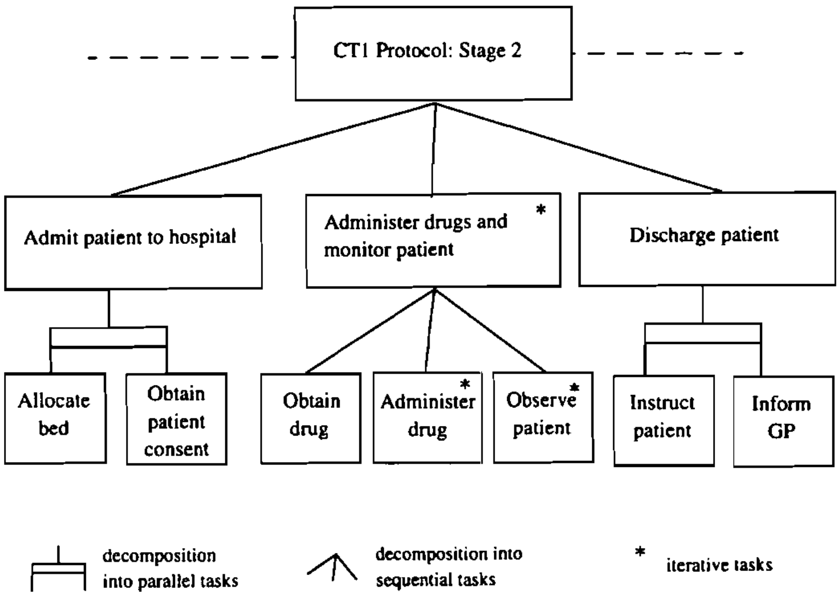


**Figure 1.** Part of CT1 protocol for treating breast cancer.

they are and that the task is successfully completed, the patient will be allocated a bed. Once a bed is available and the patient agrees to be admitted to the hospital, the support system recommends that the task "administer drugs and monitor patient" is allocated to a hospital nurse. Assuming she accepts, the protocol dictates that the task should be decomposed into the sequential subtasks of "obtain drug," "administer drug," and "observe patient"—all of which the nurse takes responsibility for. She may subsequently decompose the "observe patient" subtask still further, for example, into "measure body temperature," "take blood samples," and "analyze blood samples." This decomposition may well involve generating a request to a laboratory to test patient indicators, such as white blood cell count. However, for the sake of simplicity, this level of decomposition is not described here. Finally, the machine recommends that the third subtask of stage 2, "discharge patient," and its two subtasks, "instruct patient" and "inform GP" should be carried out by the oncologist.

In this scenario, information is transferred according to the following pattern: the resource management department must inform the oncologist about the outcome of "allocate bed" (i.e., either that a bed has been allocated as requested or that no bed is available for the requested date), and the nurse has to inform the oncologist of the results of "administer drugs and monitor patient" (e.g., drug has been administered and all patient indicators are normal). Accompanying this information exchange is a concomitant flow of control, in terms of commitments and expectations (Jennings, 1993), between the agents: the oncologist expects the resource management department to perform the activity "allocate bed" once it has agreed to; similarly, he expects the nurse to perform the "administer drugs and monitor patient" task on time, once she has consented to execute it.

## AADCARE: AN AGENT ARCHITECTURE FOR DISTRIBUTED MEDICAL CARE

The AADCare agent architecture encompasses multiple layers of knowledge, a working memory, a communications manager, and a human-computer interface (see Figure 2) (Huang et al., 1995). To be successful in this domain, the agent needs to exhibit both deliberative behavior (e.g., plan selection, task decomposition, and task allocation) and reactive behavior (e.g., 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 behavior is achieved by the incorporation of decision rules for plan selection, task management rules for task decomposition

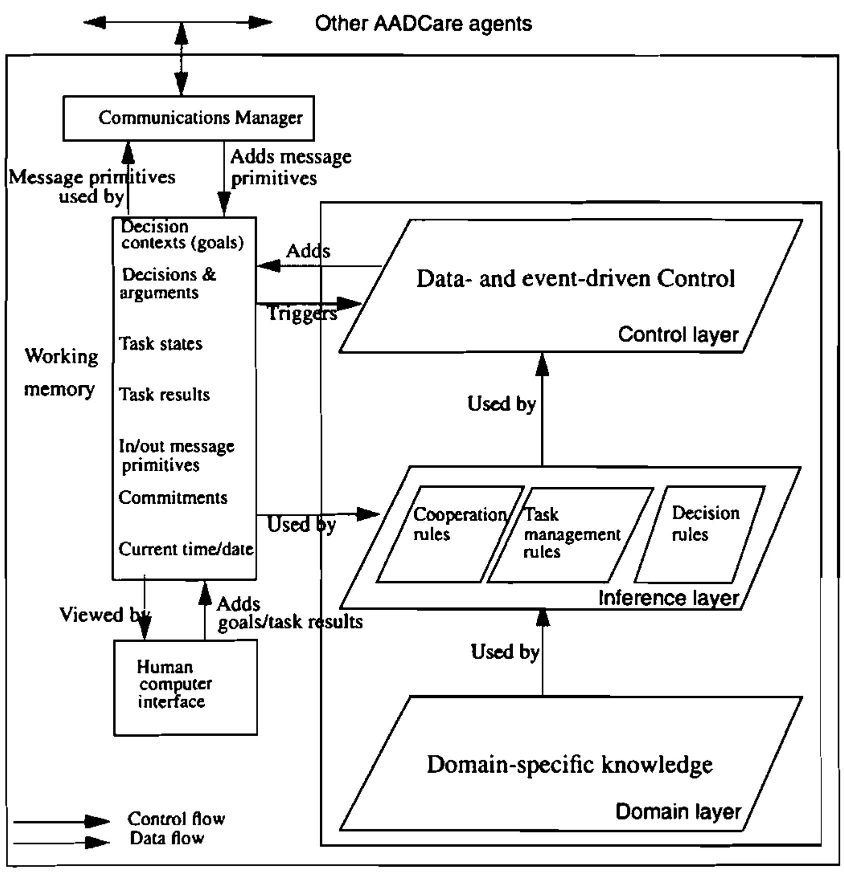


Figure 2. AADCare agent architecture.

and allocation, and cooperation rules for formulating commitments. Reactive behavior 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 three layers of knowledge that form the key part of the AADCare architecture are as follows.

- Domain knowledge includes, for example, a knowledge base covering specific medical domains such as breast cancer, a knowledge base of clinical management plans (known as *clinical protocols* (Gordon et al., 1993)), a database of patient records, and a database of resource availability.
- Inference knowledge is knowledge in the form of generic, declarative inference rules, which specify inference relations between domain knowledge, existing patient information, and possible new data. Inference rules represent the core of the agent architecture and are subdivided into those for decision making under uncertainty, those for task management, and those for managing agent cooperation, all discussed in the sections below.
- Control knowledge applies the inference knowledge to the domain knowledge in order to generate new inferences whenever new data are added to the working memory. Logically, this layer is a metalevel that controls the execution of inference rules and domain facts.

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 the second stage of the CT1 protocol for treating breast cancer contains three subtasks: "admit patient to hospital," "administer drugs and monitor patient," and "discharge patient":

```
component('CT1 stage 2', 'admit patient to hospital')
component('CT1 stage 2', 'administer drugs and monitor
  patient')
component('CT1 stage 2', 'discharge patient')
```

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 following inference schema specifies that the state of a task becomes "started" once the state of one of its subtasks becomes "started":

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

In the context of CT1 protocol, the above schema implies that the task "CT1 stage 2" should become "started" when one of its subtasks (e.g., "admit patient to hospital") becomes "started" (see section below on task management for more details of the management of task state transitions).

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

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

For example, within the context of CTI protocol, once the data `state ('admit patient to hospital', started)` is asserted to the working memory, the above control rule applies the given inference schema and domain knowledge to add a new piece of data into the working memory: `state ('CT1 stage 2', started)`.

Bringing all of this together, a sample working session of an oncologist agent is as follows. The oncologist first specifies an initial goal of finding an appropriate protocol for treating a particular patient's breast cancer. The goal triggers the control layer to apply the decision rules (see section below on symbolic decision making), medical domain knowledge, and patient case data to arrive at a decision (i.e., a suggested treatment protocol, called "CT1 protocol") and associated arguments. The oncologist endorses that decision and requests the machine to assist him in managing the execution of the protocol. This will, in turn, trigger the control layer to apply the task management rules (see section on task management) to decompose the protocol into constituent tasks, propose task allocations to appropriate agents, and generate the corresponding communication primitives. Once the oncologist accepts the proposal, the communications manager will convert the primitives into complete messages (see the following section) and send them to the specified agents. Once the chosen agents inform the oncologist of their acceptance of the task requests, they and the oncologist enter into a dynamic, cooperating agent community. Agent commitments are generated and monitored through triggering the control layer to apply cooperation rules (see section below on managing agent cooperation). Cooperation among the agents continues until the entire protocol is in a terminable state (e.g., completed or abandoned). In the meantime, the oncologist may enter into another agent community and accept task requests from other agents.

There are two main reasons for adopting this functional and logical separation of domain, inference, and control knowledge. First, it simplifies the representation, reuse, and maintenance of knowledge. Inference knowledge for decision making, task management, and cooperation can be 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 three 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.

AADCare's working memory stores temporary data generated by the control layer, the user, or the communications manager. Examples of the types of information that 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 blackboard, onto which new information (or any change that 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 or cooperation rules (see sections below for respective examples). It also converts messages that arrive from other agents into primitives that may be used by the cooperation manager. More details of this module are given in the following section.

The human computer interface defines a scheme for interaction between the support system and its user. The approach is as follows: the computer can perform 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 authorize them before they can be communicated to external agents. For example, the system may recommend to the oncologist that he ask a particular nurse to perform the drugs administration subtask; however, the oncologist may have a personal preference for another nurse and may, therefore, be unwilling to make such a referral. In this case, the system will not send an electronic request to the original nurse, but will instead offer the oncologist an alternative solution.

## Communication Management

After an extensive analysis of the interactions that can occur in cooperative care organizations, a set of communication primitives, based on speech act theory (Searle, 1969), have been defined (Table 1). Each primitive has a type (illocutionary force) and a content (propositional content), as well as a certain effect on the receiver (perlocutionary force). Having a well-defined set of primitives is important because it means that the ambiguity in message interchange is substantially reduced; each primitive has a clear meaning and must be responded to in a predictable way. There has been similar work on communication primitives elsewhere (e.g., in the Contract Net Protocol (Smith, 1980), the message perceptor in OFFICE (Winograd & Flores, 1986), IMAGINE's cooperative primitives (Lux et al., 1993), and the AGENT0

Table 1. Communication primitives

| Type | Content | Effect on receiver |
|---|---|---|
| *request* | task; [provisional schedule]; priority: urgent or not; response_by date | ReceiveAgent evaluates whether to accept the request and informs SendAgent of decision. If recipient decides to accept the request, it becomes committed to the task. |
| *accept* | task; [accepted schedule] | ReceiveAgent knows SendAgent is committed to the request and that SendAgent will inform it of the outcome of executing the task. SendAgent becomes the contractor for the task and ReceiveAgent the manager. |
| *reject* | task; [provisional schedule] | ReceiveAgent has to request someone else to perform the task on the provisional schedule. |
| *alter* | task; provisional schedule; acceptable schedule | ReceiveAgent to evaluate the acceptable schedule and decide whether to replace the provisional schedule with the acceptable schedule. If so, it sends SendAgent a new request. Otherwise, ReceiveAgent has to send the original request to someone else. |
| *propose* | task; [proposed schedule] | ReceiveAgent may or may not adopt the proposal. |
| *inform* | any information: data, domain knowledge, or partial plans | ReceiveAgent may use the information for local problem solving. |
| *query* | a question: what, how, whether, and so on | ReceiveAgent must answer the query, possibly involving extensive local problem solving (e.g., diagnosis and investigation). A reply may be of type *propose*. It may also be *inform*, possibly giving the answer "unknown" to the query. |
| *cancel* | any message of the above types | ReceiveAgent should ignore the earlier message. |
| *acknowledge* | any message of the above types; all messages need to be acknowledged except acknowledgment messages themselves | ReceiveAgent is aware of the successful transmission of the message. |

programming language (Shoham, 1993)). However, none of these systems offered an appropriate set for the domain of distributed health care management.

The primitives request, accept, reject, and alter are used during the allocation of tasks and the formulation of agent commitments. Using the example from the scenario above, the oncologist may allocate the task "administer drugs and monitor patient" to a nurse by requesting her to perform it during a period of 10 days starting on the following day. The nurse may *accept* the task exactly as specified by the oncologist, or she may *reject* it because she is too busy during the next few days (insufficient resources to honor the commitment). Alternatively, the nurse may indicate that she cannot start the task the following day, but she could start it 2 days later (*alter*).

A *suggest* act may be the result of a query, for instance, suggesting treatment protocol CT1 after being asked how to treat breast cancer. *Inform* usually follows an accepted request to perform a certain task and is mainly used to disseminate results. *Inform* may also accompany a request to provide relevant information for the contractor. *Cancel* is included as a primitive type because in certain circumstances agents may modify their commitments (as discussed in the section below on adaptive management of commitment changes). Finally, all messages must be *acknowledged*.

Knowledge about the semantics of the different types of primitives is incorporated in the task management and cooperation rules, which can dynamically generate message primitives when executed by the control layer. The communications manager then converts these primitives into complete, structured messages using a communication protocol that defines the syntax of interagent messages (Huang et al., 1994). Note that the superscript asterisk denotes repeated entries and that `PRIMITIVE_CONTENT` is as defined in Table 1.

<message>::=<sender><receiver><date><time><patient><transaction_primitive>*
<sender>::=<sender_name><contact_address>
<sender_name>::=<first_name><surname>
<first_name>::=NAME
<surname>::=NAME
<contact_address::=<email_address>|<postal_address>|<telephone_number>|
<fax_number>
<email_address>::=EMAIL_ADDRESS
<postal_address>::=POSTAL_ADDRESS
<telephone_number>::=NUMBER
<fax_number>::=NUMBER
<receiver>::=<receiver_name><contact_address>
<receiver_name>::=<first_name><surname>
<date>::=<day><month><year>
<day>::=NUMBER
<month>::=NUMBER
<year>::=NUMBER
<time>::=<hour><minute>
<minute>::=NUMBER
<hour>::=NUMBER
<patient>::=<patient_name><date_of_birth>
<patient_name>::=<first_name><surname>
<date_of_birth>::=<year><month><day>
<transaction_primitive>::=<primitive_type><primitive_content>
<primitive_type>::=REQUEST | ACCEPT | REJECT | ALTER | PROPOSE |
INFORM | QUERY | CANCEL | ACKNOWLEDGE
<primitive_content>::=PRIMITIVE_CONTENT

By means of an illustration, agent Jean-Louis Penn may receive the following message from agent Tony Burg, which includes an urgent request to treat Mary Taylor's breast cancer, as well as some patient data (the most recently measured tumor size and location) that is thought to be relevant:

```
message(from('Tony Burg', 'tb@acl.icrf.ac.uk'),
 to ('Jean-Louis Penn', 'jlp@fb.y-net.fr'),
 date('1993 06 01'), time('12 00'),
 patient ('Mary Taylor', '1925 10 30'),
 request(task('treat breast cancer'), priority ('urgent'),
   response_by ('1993 06 10')),
 inform(date('1993 05 30'), finding ('tumor size',
   '10x 5')),
 inform(date('1993 05 30'), finding ('tumor location',
   'left breast')))
```

Jean-Louis Penn's communications manager converts this message into three primitives (i.e., *request, inform,* and *inform*) and adds them into its working memory. The arrival of these new primitives then triggers the control layer to apply the cooperation rules to evaluate task requests and generate commitments where appropriate. The result is that Jean-Louis Penn agrees to undertake the requested task. Consequently, an accept primitive is generated and then composed by the communications manager into a complete outward message to Tony Burg:

```
message( from ('Jean-Louis Penn', 'jlp@fb.y-net.fr'),
 to('Tony Burg', 'tb@acl.icrf.ac.uk'),
 date('1993 06 02'), time('10 00'),
 patient('Mary Taylor', '1925 10 30'),
 accept(task('treat breast cancer')))
```

## Symbolic Decision Making

The purpose of the decision rules is to choose among alternative options (e.g., potential diagnoses of a patient's illness and potential clinical protocols that could be used to treat the patient). In addition to 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. For example, whilst executing a particular clinical protocol, there may be a crucial decision to be made that needs to make use of the decision-making know-how contained in this rule group (see the following section for an illustration of this point with respect to prescribing).

In this application domain, 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 tumor, but the patient may be unwilling to tolerate its side effects. 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, and cardiology). Such a separation also permits formalization 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 (see the following section). 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 elderly 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 the most preferred decision, e.g., the decision to use CT1 chemotherapy (argument aggregation).

This decision procedure is based on a simple but flexible method of reasoning under uncertainty for argument generation and aggregation, called *argumentation* (Krause et al., 1993), 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 labeled to indicate this change, for example, "confirm," "support," "weaken," or "exclude." Arguments of this sort are similar to Cohen's endorsements (Cohen, 1985), but in this work a more sophisticated set of aggregation functions is used to combine collections of arguments to yield a preference ordering on the decision options. This method is versatile, conceptually intuitive, relatively easy to implement, and simplifies some of the problems of knowledge acquisition and maintenance. The second idea is that the grounds of arguments for and against decisions are explicitly represented, meaning that they can serve a variety of functions, including truth maintenance and explanation.

An example inference schema in the decision procedure specifies that if a decision candidate is proposed for a decision context and supported by a known clinical or nonclinical finding, then a supporting argument for the candidate is derived for the decision context:

```
schema ( conditions ( proposed(Candidate, Context),
   support(Finding, Candidate, Context),
   known(Finding)),
 conclusions (argument(supported, Candidate, Finding,
   Context)))
```

To take an example, suppose that the CT1 protocol has been proposed as a possible treatment protocol for breast cancer. Given the following patient data (patient is old) and domain knowledge (the CT1 protocol is known to be appropriate for treating breast cancer in elderly patients):

```
known((age, old))
support((age, old), CT1_protocol, context(treat,
   breast_cancer))
```

the following argument is derived using the above inference schema:

```
argument(supported, CT1_protocol, (age, old),
   context (treat, breast_cancer))
```

Further details of this approach to decision making are given elsewhere; for example, Fox and Krause (1992) describe it within a general context of qualitative reasoning and Huang et al. (1993) give a more formal, declarative specification of the decision procedure and discuss its application in medical decision making. These aspects, 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.

## 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., CT1 for treating breast cancer) is determined by experts in authority and is precisely defined in a clinical protocol (see Figure 1). The task management rules decompose such a protocol into subtasks according to the predefined plan structure (as described in the scenario above). Subtasks at the bottom of the plan hierarchy may be primitive actions for humans or machines to perform (such as "allocate bed" and "administer drugs") or they may be decision tasks (such as choosing the right drug for a 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 is 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 AADCare for making such decisions is that of *accountability.* Accountability is a static relationship that defines for 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 term *request* represents a primitive that is sent to the communications manager when this task management rule is fired (as described above).

```
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 AADCare 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 nested 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 that he has already been taking for a period), whereas such action is not normally necessary for a cancelled task.

## Managing Agent Cooperation

In AADCare 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) (Jennings, 1993). The former means that if an agent agrees to undertake a task, then it will endeavor 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 (discussed below) need to be addressed: (1) what is involved in establishing a commitment and (2) what type of convention is appropriate for monitoring commitments in the given care organization?

### *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) that are required to perform that task. (In addition, an agent may also have a local policy governing the acceptability of a requested task. For instance, 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 remain 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.) For example, a hospital specialist may be accountable to patients for in-hospital breast cancer treatment but will not become committed to an actual treatment course on a specific patient until the time (temporal resource) and a bed (material resource) are available to perform the treatment. Although agents know what resources are available to themselves, they do not generally have information about the resources of their acquaintances. Therefore, an agent may have to propose the same task to several acquaintances before an acceptable contractor and, hence, commitment can be found (made).

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))
```

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 that have been generated. Again, this behavior 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 italicized term represents primitives sent to the communication manager when the appropriate inference rules are triggered (as described in the section above on communication management).

*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 (Jennings, 1995). Figure 3 details the convention embodied in the AADCare cooperation manager for the breast cancer treatment prototype.

```
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 3.** Convention for adapting commitments in AADCare.

## RELATED WORK

In this section, AADCare is briefly compared with some of the well-known architectures and systems in the agent literature.

GRATE (Jennings et al., 1992) 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 logical separation is that it provides a convenient basis for declarative specification and for logical verification and validation of the various layers of knowledge, e.g., in a formal language such as $ML^2$ (van Harmelen, 1992). The same two observations can be made of other, similar layered architectures such as INTERRAP (Muller et al., 1995) and TouringMachines (Ferguson, 1995).

Coordinator (Winograd & Flores, 1986) is a conversational system for coordinated action that is based on Searle's speech act theory (Searle, 1969). However, whilst the generation and monitoring of speech acts and commitments are centralized in Coordinator, AADCare 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 AADCare accommodates additional functions such as a generic decision module for decision making under uncertainty.

AADCare also bears certain similarities to a standard blackboard architecture (Engelmore & Morgan, 1988). In both cases the working memory is changed through the application of functionally separated modules of inference rules. However, in addition to functional separation, AADCare also emphasizes the logical layering of knowledge for reasons stated above and provides a set of generic knowledge modules for cooperation and decision making.

## CONCLUSIONS

Numerous techniques and systems have been developed in the medical informatics community for tackling isolated aspects of medical decision making. However, despite a well-documented need for supporting an integrated range of different functions (including uncertainty management, task management, and coordination), there has been very little prior work that attempts to provide comprehensive procedures and integrated decision support for these different aspects of health care. AADCare, therefore, represents an important first step toward providing this integrated support. It gives a novel coupling of a decision-making procedure and DAI techniques for task management, cooperation, and communication. Such a coupling is essential if the full potential of automation is to be attained in the important real-world domain of health care management.

A prototype AADCare system has been developed for the specific application of distributed management of cancer patients among general practices, hospitals, home care organizations, 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 (Microsoft Mail) and server is used for message passing among the care agents. This system has been installed on a network of PCs running LPA-Prolog and MAPI (messaging applications interface written in C) under Microsoft-Windows 3.1 for Workgroups.

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 (Walton & Randall, 1992). 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 program. Desirable extensions to the current work would be to interface the prototype system to existing patient information systems and electronic healthcare information networks. Once these interfaces are established, it is envisaged that AADCare technology will be used in operational settings to greatly improve the delivery of effective, efficient, and globally coherent patient care programs.

## REFERENCES


Cohen, P. R. 1985. *Reasoning about uncertainty: an artificial intelligence approach.* London: Pitman.

Engelmore, R., and T. Morgan. 1988. *Blackboard systems.* Reading, Mass.: Addison-Wesley.

Ferguson, I. A. 1995. Integrated control and coordinated behaviour: a case for agent models. In *Intelligent agents*, eds. M. J. Wooldridge and N. R. Jennings. *Lecture Notes in Artificial Intelligence.* Vol. 890. New York: Springer-Verlag.

Fox, J., A. Glowinski, C. Gordon, A. Hajnal, and M. O'Neil. 1990. Logic engineering for knowledge engineering: design and implementation of the Oxford system of medicine. *Artificial Intelligence in Medicine* 2:323–39.

Fox, J., and P. Krause. 1992. Qualitative frameworks for decision support: lessons from medicine. *Knowledge Engineering Review* 7:19–33.

Gordon, C., S. I. Herbert, A. Jackson-Smale, and J. L. Renaud-Salis. 1993. Care protocols and healthcare informatics. In *Artificial Intelligence in Medicine*, eds. S. Andreassen, R. Rengelbrecht, and J. Wyatt. IOA Press, 289–309.

Hickman, F. R., J. L. Killin, L. Land, T. Mulhall, D. Porter, and R. M. Taylor. 1989. *Analysis for knowledge based systems: a practical guide to the KADS methodology.* Chichester, England: Ellis Horwood.

Huang, J., J. Fox, C. Gordon, and A. Jackson-Smale. 1993. Symbolic decision support in medical care. *Artificial Intelligence in Medicine* 5:415–30.

Huang, J., N. R. Jennings, and J. Fox. 1994. Cooperation in distributed medical care. In *Proceedings of second international conference on cooperative information systems*, 255–63. Toronto: University of Toronto Press.

Huang, J., N. R. Jennings, and J. Fox. 1995. An agent architecture for distributed medical care. In *Intelligent agents*, eds. M. J. Wooldridge and N. R. Jennings, 219–32. *Lecture Notes in Artificial Intelligence.* Vol. 890. New York: Springer-Verlag.

Jennings, N. R. 1993. Commitments and conventions: the foundation of coordination in multi-agent systems. *Knowledge Engineering Review* 8:223–50.

Jennings, N. R. 1995. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence* 74(2).

Jennings, N. R., E. H. Mamdani, I. Laresgoiti, J. Perez, and J. Corera. 1992. GRATE: a general framework for cooperative problem solving. *Intelligent Systems Engineering* 1(2):102–14.

Krause, P., S. Amble, and J. Fox. 1993. The development of a logic of argumentation. In *Advanced methods in artificial intelligence. Lecture Notes in Computer Science Series*, eds. B. Bouchon-Meunier, L. Valverde, and R. R. Yager. 682:109–118. New York: Springer-Verlag.

Lux, A., P. de Greef, F. Bomarius, and D. Steiner. 1993. A generic framework for human computer cooperation. In *Proceedings of international conference on intelligent and cooperative information systems (ICICIS-93), Rotterdam, the Netherlands*, 89–97.

Muller, J. P., M. Pischel, and M. Thiel. 1995. A pragmatic approach to modelling autonomous interacting systems: a preliminary report. In *Intelligent agents*, eds. M. J. Wooldridge and N. R. Jennings. *Lecture Notes in Artificial Intelligence.* Vol. 890. New York: Springer-Verlag.

Pritchard, P. 1992. The role of computers in referral. In *Referrals to medical outpatients*, eds. A. Hopkins and P. Wallace, 79–89. London: Royal Colleges of Physicians and General Practitioners.

Reeves, P., T. Rickards, and B. Carniel. 1993. Requirements for shared care decision support in cardiology. Technical Report. Royal Brompton National Heart and Lung Hospital, London, England.

Renaud-Salis, J. L., P. Lagouarde, C. Gordon, and R. Thomson. 1992. Requirements for decision support in cancer shared care. Technical Report. Fondation Bergonie, Bordeaux, France.

Searle, J. R. 1969. *Speech acts: an essay in the philosophy of language.* New York: Cambridge University Press.

Shoham, Y. 1993. Agent-oriented programming. *Artificial Intelligence* 60(1):51–92.

Smith, R. G. 1980. The contract net protocol: high-level communication and control in a distributed problem solver. *IEEE Transactions on Computers* C-29(12):1104–13.

van Harmelen, F. 1992. $ML^2$: a formal language for KADS models of expertise. *Knowledge Acquisition* 4(1):127–161.

Walton, R., and A. Randall. 1992. Clinical decision analysis. *British Medical Journal* 301:301.

Winograd, T., and F. Flores. 1986. *Understanding computers and cognition: a new foundation for design*. Norwood: Ablex Publishing.

Wooldridge, M. J., and N. R. Jennings. 1995. Agent theories, architectures and languages: a survey. In *Intelligent agents,* eds. M. J. Wooldridge and N. R. Jennings, 1–32. *Lecture Notes in Artificial Intelligence*. Vol. 890. New York: Springer-Verlag.