FELINE — A Case Study in the Design and Implementation of a Co-operating Expert System

Michael Wooldridge

Department of Computing
Manchester Metropolitan University
Chester Street, Manchester
United Kingdom

EMAIL: M.Wooldridge@doc.mmu.ac.uk

Greg O'Hare Rebecca Elks

Department of Computation
UMIST, P.O. Box 88
Sackville St., Manchester
United Kingdom

EMAIL:{greg, relks}@sna.co.umist.ac.uk

Abstract

There has recently been considerable international interest in the possibility of building second generation expert systems as groups of co-operating problem solvers. In this paper we relate our experiences with such a system. FELINE is a co-operating expert system composed of five autonomous intelligent agents. These agents co-operate through communication to identify the causes of anaemia in cats. The paper addresses three key issues: development methodologies for building co-operating expert systems, making expert systems co-operative, and inter-agent problem solving techniques. We propose a tentative development methodology for co-operating expert systems and describe how it was applied in FELINE. We also present a detailed, domain independent account of the inter-agent problem solving paradigm devised for use in FELINE. We conclude with a brief critical assessment of our work.

1 Introduction

IN RECENT YEARS there has been considerable interest in the possibility of building complex problem solving systems as groups of co-operating experts. Distributed Artificial Intelligence (DAI) is the study of how such systems might be built. Historically, empirical research in DAI has focussed on three main areas: blackboard architectures [9], systems based on negotiation [19], and multi-agent planning systems [7]. More recently, *co-operating expert systems* have emerged as a research area of some importance (see e.g. [4]).

A co-operating expert system is composed of a group of *agents*, each of which contains an autonomous knowledge based system. Typically, agents will have expertise in distinct but related domains. The whole system is capable of solving problems which require the cumulative expertise of the agent community. In order to achieve this, agents must co-operate with one another, usually by exchanging messages.

There are many potential advantages of such systems: re-usability of knowledge, partitioning of problem domains, and performance improvement through the exploitation of parallelism are just three (see [1, 14] for detailed discussions). However, there are also considerable problems to be overcome. It is possible to identify three key issues:

1. *How is inter-agent problem solving to proceed?* This question has received considerable attention within the DAI community. Previous research has been based around two key paradigms: negotiation [19] and plan exchange [7] (see [5] for a survey of techniques).

2. *How is intra-agent problem solving to proceed?* Given an existing expert system, how may it be incorporated into a community of experts?

3. *What software tools and development methodologies are required to build co-operating expert systems?* Various software testbeds for DAI systems have been reported, (e.g. MACE [12, 13], *CooperA*, [21] and MCS/IPEM [6]). However, these systems have suffered from well documented shortcomings (primarily lack of flexibility.)

The research described in this paper addresses each of these questions. The various theoretical strands of DAI (e.g. [22]) will ultimately form a theory of co-operative problem solving. However, we believe that experimentation must play a crucial role in the development of such a theory. To this end, we have implemented FELINE, a co-operating expert system that operates in the domain of diagnosing the causes of anaemia in cats. FELINE was implemented as part of an ongoing research programme in DAI. There were three main research objectives in developing FELINE:

- investigate inter-agent problem solving techniques for co-operating expert systems;
- investigate development methodologies for co-operating expert systems;
- develop a prototype teaching tool/experts assistant that operates in the domain of diagnosing the causes of anaemia in cats.

FELINE is intended to start from the suspicion that the cat under investigation is anaemic. By a process of questioning, it leads a user through the appropriate chain of reasoning to a conclusion concerning the cause of the anaemia.

In this paper, we describe our experiences with FELINE. In the next section, we briefly review related empirical research. We then outline the key characteristics of the problem domain, before describing the development methodology devised for FELINE, and the co-operative problem solving paradigm used. We then describe how the development methodology was applied in FELINE, and subsequently present an example case history. To conclude, we present a brief critical review of our work. A more detailed account of FELINE may be found in [8].

2 Related Research

Early research on multi-agent problem solving systems focussed on the blackboard architecture [9]. Agents (or *knowledge sources*) in a blackboard system communicate by writing on a globally accessible data structure called a blackboard. The problem solving protocol is defined by a central control mechanism that determines which agent has access to the blackboard at any given time. Later research recognised the importance of having autonomous agents capable of engaging in peer-to-peer communication. The *contract net* [19] proposed negotiation as a basis for co-operative problem solving. More recently, Durfee [7] developed a still more powerful multi-agent problem solving paradigm called *partial global planning (PGP)*. Durfee's agents are sophisticated social entities capable of generating and exchanging plans of proposed activity in order to efficiently allocate problem solving resources.

More recently still, the ESPRIT project ARCHON was initiated with the stated aim of producing an architecture and methodology for developing co-operating expert systems [23]. Agents in ARCHON have four key components: a communications manager (enables agents to communicate by sending messages), a monitor (intra-agent control), an agent acquaintance model (a representation of an agent's beliefs about itself and its environment), and an underlying intelligent system (domain expertise).

Other researchers have described DAI testbeds: *CooperA* [21] was to implement a multi-agent chemical emergencies manager; MACE [12, 13] is a well known DAI testbed that has been used to implement multi-agent systems of varying degrees of size and complexity; Feyter [11] described RTEX, a software platform for real-time multi-agent industrial applications; Doran [6] has described the MCS/IPEM testbed, in which each agent is a PROLOG-based planner.

3 The Problem Domain

Anaemia in cats is a clinical sign associated with a number of conditions, rather than a diagnosis in itself. Anaemia is a serious problem in feline medicine: experience has shown that up to 10% of cats presented at clinics are anaemic. Of those, up to 80% may die in the eighteen months following presentation [10]. Unfortunately, the range of conditions which cause anaemia makes diagnosis frustratingly difficult.

The term *anaemia* simply indicates that a cat's blood haemoglobin concentration has fallen below the range considered normal for healthy cats. As pointed out above, recognising that a cat is anaemic is not a diagnosis in itself, but is a symptom associated with a number of conditions. Anaemias in cats may be divided into five causal categories:

- *Haemorrhagic Anaemias.* This type of anaemia is relatively rare in cats. It occurs when the cat has undergone massive blood loss. The cause is usually self evident, but occasionally severe internal bleeding may be the cause.
- *Haemolytic Anaemia.* Excessive destruction of a cat's red blood cells will result in haemolytic anaemia. The commonest cause of haemolytic anaemias is infection by the feline leukaemia virus (FeLV). Certain oxidative substances may also cause a haemolytic anaemia; one such substance is paracetomol. It is not uncommon for misguided owners to administer paracetomol to their cat.
- *Dyserythropoitic Anaemias.* In healthy cats, red blood cells are produced by bone marrow to replace those cells that die normally. If this replacement mechanism becomes disorganised, a dyserythropoitic anaemia will occur.
- *Anaemias Associated with Marrow Failure.* If the bone marrow replacement mechanism fails, then anaemia will occur. A number of conditions can cause marrow failure.
- *Anaemias Secondary to Systemic Disease.* Diseases of the 'whole body' (systemic diseases) can cause anaemia as a secondary effect. Anaemias of this type are particularly difficult to diagnose correctly, as the symptoms are likely to be extremely confusing.

Although most anaemias will fall into one of the above categories, many other anaemias cannot be so easily classified. It should be clear that speculative treatment is not a viable option. The principle functional objective of FELINE is to assist in the task of identifying the cause of a feline anaemia.

4 The FELINE System

This section deals with the design and implementation of FELINE. We begin by introducing the development methodology devised for FELINE. We then describe the inter-agent problem solving paradigm used, and go on to describe how the development methodology was applied in the construction of FELINE.

4.1 Development Methodology

Earlier in this paper we observed that there is no accepted methodology for developing co-operating expert systems (although the DAI community has recognised the need for such techniques [3, 23].) In the absence of any existing techniques, we have devised a tentative methodology adapted from the five stage knowledge acquisition cycle proposed by Buchanan *et al* [2]. In this section, we outline the methodology: in Sections 4.3–4.5 we show how it was applied in the development of FELINE.

Recall that Buchanan's development cycle consists of five stages: **(1) Identification:** Specify the domain of the system; **(2) Conceptualization:** Recognition of key concepts in domain and relationships between them; **(3) Formalization:** Map concepts developed during (2) to formal representations; **(4)**

Implementation: Map formal representations to target environment; **(5) Testing:** Identify any errors; feedback to earlier stages.

Our methodology retains each of these key stages. However, we propose the introduction of a *decomposition* stage between conceptualization and formalization. Additionally, we recognise the need for an *integration* step during implementation, and an altered testing stage.

Decomposition concerns the partitioning of the problem domain into agents. This partitioning must be performed before it is possible to begin formalizing the domain. The key problem lies in deciding how the domain is to be partitioned; it is possible to identify at least two techniques for achieving this.

First, a domain might be divided into sub-domains reflecting natural partitions in the overall knowledge space. There is no clear-cut method for deciding what constitutes a 'natural partition'; a decision can only be reached by extensive consultation with the domain expert. A clear analogy may be drawn with software engineering, where the issue of what constitutes a module must frequently be addressed. Parnas [16] attempted to identify criteria for the decomposition of software systems; similar work needs to address this issue in DAI. As yet, the problem of delimiting agent boundaries is largely dependent on the intuitions of the domain expert.

Second, a domain might be divided into partitions which exploit different knowledge representation techniques. This would provide an elegant method for incorporating disparate reasoning styles into a single system. Since there is no attempt to build a single agent incorporating a variety of knowledge representation techniques, the problems inherent in using hybrid reasoning schemes are avoided.

In addition to the decomposition stage, there must necessarily be a point in the implementation stage where the various agents are fused into a single system. How easily this integration is achieved will depend on how well the domain was decomposed. In the next Section, we look at how inter-agent problem solving is achieved in FELINE.

4.2 Inter Agent Problem Solving in FELINE

Each agent in FELINE maintains a data structure representing its beliefs about itself and its environment. This data structure is called the *environment model*. It contains an entry for the modelling agent and each agent that the modelling agent might communicate with (its *acquaintances*.) Each entry contains two important attributes:

- *Skills.* This attribute is a set of identifiers denoting hypotheses which the agent has the expertise to establish or deny. The skills of an agent will correspond roughly to root nodes of the inference networks representing the agent's domain expertise.

- *Interests.* This attribute is a set of identifiers denoting hypotheses for which the agent requires the truth value. It may be that an agent actually has the expertise to establish the truth value of its interests, but never the less is 'interested' in them. The interests of an agent will correspond roughly to leaf nodes of the inference networks representing the agent's domain expertise.

An agent's environment model serves to delimit the co-operative problem solving process. Before describing inter-agent problem solving in detail, we describe the types of message that agents may send.

First, we define a message as a triple of *sender, receiver,* and *contents*. In FELINE, the contents field is also a triple, containing *message type, attribute*, and *value*. Agents communicate using three message types:

- *Request*. If an agent sends a request, then the attribute field will contain an identifier denoting a hypothesis. It is assumed that the hypothesis is one which lies within the domain of the intended recipient. A request is assumed to mean that the sender wants the receiver to derive a truth value for the hypothesis.

- *Response*. If an agent receives a request and manages to successfully derive a truth value for the hypothesis, then it will send a response to the originator of the request. The attribute field will contain the identifier denoting the hypothesis; the value field will contain the associated truth value.

- *Inform*. The attribute field of an inform message will contain an identifier denoting a hypothesis. The value field will contain an associated truth value. An inform message will be unsolicited; an agent sends one if it thinks the recipient will be 'interested' in the hypothesis.

We now look in detail at the co-operative problem solving technique used. First, consider goal-driven problem solving in a normal rule-based system. Typically, goal driven reasoning proceeds by attempting to establish the truth value of some hypothesis. If the truth value is not known, then a recursive descent of the inference network associated with the hypothesis is performed. Leaf nodes in the inference network typically correspond to questions which are asked of the user. Within FELINE, this well known scheme is augmented by the following principle. When evaluating a leaf-node, if it is not a question, then check the environment model to see if any other agent has the node as a 'skill'. If there is some agent which lists the node as a skill, then send a request to that agent requesting the hypothesis. Wait until a response is received; the response will indicate the truth value of the node. This technique is an instance of the general problem solving technique called *task sharing* [20], since evaluation tasks are explicitly delegated on the basis of the skills entry in the environment model.

We now turn to data-driven problem solving. Typically, data-driven problem solving proceeds by taking a database of facts (hypotheses and associated truth values), and a set of rules, and repeatedly generating a conflict set of new facts. These new facts are then added to the database, and the process begins again. If a hypothesis follows from a set of facts and a set of rules, then this style of problem solving will eventually generate a result. In FELINE, the basic scheme is augmented as follows. Whenever a new fact is generated, the environment model is consulted to see if any agent has the hypothesis as an 'interest'. If it does, then an 'inform' message is sent to the appropriate agent, containing the hypothesis and truth value. Upon receipt of an 'inform' message, an agent adds the fact to its database and enters the forward chaining cycle. This technique is an instance of *result sharing*, [20] since agents share results that they believe may be of interest to other agents.

We may usefully compare this technique with the contract net [19]. In a contract net, agents share tasks by broadcasting task-announcements to the entire agent community. Other agents may then 'bid' for the task, though only one will be 'awarded the contract'. The contract net is thus well suited to dynamic domains, where the community of agents is variable; agents need little information about their peers, as messages are broadcast to the entire community. However, the broadcast nature of the contract net makes great demands on the communications medium. In contrast, agents in FELINE maintain models of their peers which they use to direct requests and informs; there is thus a tradeoff between the volume of communication traffic and the size and complexity of the internal model. Our technique therefore seems appropriate for static domains, where the community of agents remains fixed, or where all agent interdependencies are known at development time.

Although the technique described above is simple, it demonstrates that a hybrid task sharing/result sharing system can be incorporated into communities of existing expert systems with comparative ease. The technique is opportunistic, since the use of 'inform' messages means that results are exploited as they become available. It is also goal directed, since agents can explicitly delegate tasks (using 'requests') based on beliefs about agent skills.

4.3 Identification & Conceptualization

The first task at this stage was to develop an 'outline' system. This system served three purposes: (1) To help to define the extent of the intended domain more precisely (2) To help identify any natural partitions in the domain and recognise their chief characteristics (3) To enable the domain expert to

become acquainted with the reasoning tools used (see *Software Development Tools*, below). The key feature of the domain that emerged during the conceptualization stage was the importance of the problem solving protocol. At the start of a FELINE investigation, there are a large number of potential candidates for the causative disease. Most of these diseases have relatively shallow inference networks associated with them. The key problem when attempting to diagnose the cause of an anaemia is therefore to navigate a path through the hypothesis space as efficiently as possible. This navigation is achieved by the use of meta-rules [18].

The outline system enabled us to develop a coarse definition of the expert's protocol that would be used in the final system. Approximately two person-months of effort were devoted to the outline system. It contained approximately one hundred rules, meta-rules, and questions.

The conceptualization stage resulted in the identification of the following general protocol:

1. *Get initial clinical data from user.* It is usual practice when carrying out a clinical investigation to record a number of standard pieces of data concerning the client. For example, whatever the presenting illness, a vet would almost certainly record age, sex, etc.

2. *If no anaemia appears to be present, or the cause of the anaemia is obvious, then report the appropriate result and quit.* It is rarely the case that the cause of an anaemia can be identified from initial clinical signs alone. However, the cause is occasionally obvious: the classic example being haemorrhagic anaemia caused by severe surface wounds. There are also some cases where an initial suspicion of anaemia turns out to be unsubstantiated. It is sensible to identify these cases as quickly as possible and inform the user of the result.

3. *Perform blood tests.* If anaemia is confirmed, and the cause is not immediately apparent, then it will normally be necessary to carry out blood tests. On the basis of these tests, it is normally possible to form a tentative hypothesis concerning the category of the anaemia.

4. *On the basis of the tentative hypothesis, investigate the cause in detail.* The actual cause of the anaemia may be deduced from a sequence of questions, once the category is known.

4.4 Decomposition

On the basis of the above protocol and the experience gained with the outline system, the domain was decomposed into five agents: initial, blood, haemorrhagic, haemolytic, and aplasia. Each agent reflects a natural partition in the domain knowledge space.

The Agent initial

The initial agent performs steps (1) and (2) in the protocol described above. It gets initial clinical data from the user, and on the basis of this data eliminates some 'obvious' causes. Also, if anaemia is not indicated by the clinical signs, then the agent reports this and quits. In all other cases, the agent requests the blood agent to perform blood tests.

The Agent blood

The blood agent is an expert in the domain of blood tests. It requests the user to perform some tests, and gets the results. On the basis of these tests, the blood agent is able to either immediately identify the cause of the anaemia, or else generate a tentative hypothesis concerning the cause of the anaemia. In order to generate a hypothesis, the blood agent is often forced to request the initial agent for clinical data.

When a tentative hypothesis has been formed, the blood agent matches the category of the anaemia against the agents it knows about, and requests the appropriate agent to investigate the actual cause.

The Agents haemorrhagic, haemolytic, and aplasia

Each of these agents operates in essentially the same way. Each is an expert in one type of anaemia: thus the haemorrhagic agent is an expert in haemorrhagic anaemias. Each starts from the point of being requested to identify the actual type of anaemia present, and thus from the assumption that it has the necessary domain expertise to identify the cause of the anaemia. Each agent will request further data from the blood agent and the initial agent.

4.5 Formalization, Implementation and Testing

4.5.1 Software Development Tools

Despite the immense interest in DAI, at the time of writing there are no commercially available software tools to assist in the development of co-operating expert systems. Moreover, comparatively few applications have been reported. FELINE was implemented as part of an ongoing research project in DAI. It has served as a test application for a set of tools developed for this project. In this section, we briefly review the facilities provided by the Multi-Agent Development Environment (MADE), which was used to develop FELINE. See [24] for a complete account of MADE.

MADE has five principle components:

- Extensions to four programming languages to provide inter-agent communication primitives. Currently, agents may be implemented in any of four languages: Common LISP, PROLOG, POP-11, and C.
- Run time facilities to support inter-agent communication. A group of agents may be instantiated on a single physical machine, or distributed over a network. The network is hidden from agents, who only need to know the *agent identifier* of the agent they wish to communicate with.
- System agents to provide run-time facilities for application agents
- The MASC (Multi-Agent System-description Compiler) Program, a compiler that translates a high level definition of a MADE system into the sequence of instruction necessary to instantiate the system. The semantics of the MASC source language are based upon a simple formal model of organisation structure. MASC was inspired by EFIGE [17].
- The MP (Model Parser) Program, a compiler that translates an abstract model of an agent's environment model (see 4.2) into a concrete data structure in the required destination language.

MADE was implemented across a network of Sun 3/60 workstations operating under UNIX. Agents communicate by buffered, asynchronous message passing.

Finally, we consider the reasoning tools used by agents. It was initially hoped to use a commercial expert system shell (e.g. SAVOIR [15]) as the main reasoning component of agents. However, commercial shells and knowledge engineering environments have associated efficiency overheads, and do not lend themselves to adaptation. A shell package was therefore implemented for use in FELINE. This shell supports forward and backward chaining, as well as several hybrid reasoning styles, and allows meta-rules to guide the problem solving process.

4.5.2 Formalization, Implementation, and Testing

Even under ideal circumstances, building expert systems is not a trivial undertaking. Building a co-operating expert system is an entirely new type of problem; it follows that experimentation featured prominently in the implementation of FELINE.

Each agent was implemented as a stand alone expert system, so that the knowledge engineer would have the advantage of only having to implement a fragment of the entire domain knowledge at any one

time. The advantages of implementing FELINE in fragments outweighed the difficulties of integration. At this stage, inter-agent dependencies were modelled as questions for the user. Agents were tested in isolation. Each agent in FELINE contained approximately one hundred rules, meta-rules, and questions.

When all agents had been tested and debugged in isolation to the satisfaction of the knowledge engineer and domain expert, the integration step began. The ease with which integration is performed depends largely on the success of the decomposition stage. In FELINE, integration consisted of the following process: First, each agent's environment model was constructed using MP. The entire system was modelled using MASC. The 'virtual' inter-agent links inserted during single agent construction were then removed. FELINE was then assembled by adding progressively more agents. As each agent was included, the system was tested as rigorously as possible. When integration was complete, system testing began.

System testing was performed by using real case histories. In the following Section we present one such case history, and illustrate how FELINE was able to deduce the cause of the anaemia in this case.

5 A Case History

FELINE was validated against a number of real case histories. In this section, we present an example case history, and describe how FELINE was able to deduce the cause of the anaemia. The presenting signs were as follows:

> The patient is a six week old male kitten. The presenting symptoms are lethargy and pallor of the mucous membranes. No other abnormalities can be detected. Standard blood tests show a pack cell volume (PCV) of 31.0%, a red blood cell count of 6.96×10^{12}/l, a blood haemoglobin concentration of 7.3g/dl, and a white blood cell count of 12.8×10^{9}/l.

The initial agent was unable to identify the cause of the anaemia on the basis of the clinical signs. It therefore requested the blood agent to get blood test results from the user and generate a tentative hypothesis. The blood agent subsequently deduced that the cat had a normocytic, hypochromic anaemia. Since it had found a new fact, it consulted its environment model to see if any agent listed the fact as an interest. It found that the initial agent was interested in normocyctic, hypochromic anaemias. It therefore informed initial of the result. The blood agent was unable to generate any further hypotheses on the basis of the inconclusive test results.

The initial agent now had sufficient information to recognise (through meta-rules) that a likely cause of the anaemia was dietary iron deficiency. It attempted to confirm the hypothesis by asking the user whether the cat had been recently unweaned. In this case the cat had. The hypothesis was confirmed, and FELINE informed the user of its (correct) diagnosis.

6 Discussion

Our experiences with FELINE are reviewed with reference to our research aims: First, consider inter-agent problem solving. We have described a hybrid task sharing/result sharing technique that extends traditional intra-agent reasoning to multiple agents. The technique is simple to understand and implement, and may be incorporated into communities of pre-existing agents. The technique has been extensively tested in FELINE.

Second, a tentative development methodology for co-operating expert systems has been proposed. The technique is adapted from Buchanan's well known five stage knowledge acquisition cycle. The methodology retains all the stages from Buchanan's approach, but recognises the need for an explicit decomposition stage that must be performed before formalization may begin. Criteria were proposed for performing decomposition. Additionally, the methodology recognises the need for an implicit integration step, and exploits the potential for stand alone testing of agents.

Our final research aim was to build a working teaching tool/experts assistant. It was never an objective of this project to build a production system; it comes as no surprise to learn that in its current state FELINE would not succeed in this respect. However, anecdotal evidence suggests that the performance of FELINE compares favourably with human experts.

Our experience with FELINE has clearly demonstrated that co-operative problem solving techniques may be successfully applied to complex real-world applications. Although the techniques described are comparatively simple, they have nevertheless been used to construct an application of appreciable sophistication. The key limitations of the techniques lie in their scope. Dynamic problems will necessarily require more sophisticated inter-agent problem solving techniques. Such techniques will be investigated during the next phase of our research.

To conclude, our ultimate goal is to develop practically applicable theories of multi-agent problem solving. FELINE has proved to be a valuable first step in this research programme.

Acknowledgements

M. Wooldridge was supported by a SERC PhD studentship; R. Elks was supported by a SERC MSc studentship. The authors are grateful to Dr. Gruffyd-Jones of Bristol University Dept. of Veterinary Science for his generous assistance.

References

[1] A. Bond and L. Gasser (eds), *Readings in Distributed Artificial Intelligence*, Morgan-Kaufmann, 1988

[2] B. G. Buchanan *et al*, 'Constructing an Expert System', in F. Hayes-Roth, D. A. Waterman and D. Lenat (eds), *Building Expert Systems*, Addison-Wesley, 1983

[3] B. Burmeister and K. Sundermeyer, 'COSY: A Project for the Methodology of Multi-Agent Systems', in [4]

[4] Proceedings 1st International Working Conference on Co-operating Knowledge Based Systems, Keele University, UK, October 3–5th, 1990

[5] K. S. Decker, 'Distributed Problem Solving Techniques: A Survey', *IEEE Trans on Systems, Man, and Cybernetics*, Vol SMC-17, 1987

[6] J. Doran, 'The MCS Multi Agent Testbed', in [4]

[7] E. H. Durfee, *Co-ordination of Distributed Problem Solvers*, Kluwer Academic Press, 1988

[8] R. Elks, *FELINE: A Co-operating Expert System Implemented Using MADE*, MSc Dissertation, Dept. of Computation, UMIST, 1990

[9] R. S. Engelmore, T. Morgan (eds), *Blackboard Systems*, Addison-Wesley, 1988

[10] R. Evans and T. Gruffyd-Jones, 'Anaemia in Cats', *In Practice*, __ , __

[11] A. R. Feyter, 'RTEX: An Industrial Real-Time Expert System Shell', *Proc. Avignon 90 (Vol 1)*, 1990

[12] L. Gasser, C. Braganza and N. Herman, 'MACE: A Flexible Testbed for Distributed AI Research', in [14]

[13] L. Gasser, C. Braganza and N. Herman, 'Implementing Distributed AI Systems Using MACE', *Proc. 3rd IEEE Conference on AI Applications*, 1987 (reprinted in [1])

[14] M. N. Huhns (ed), *Distributed Artificial Intelligence (Vol I)*, Pitman, 1987

[15] *The SAVOIR Manual*, ISI Ltd, 1986

[16] D. L. Parnas, 'On the Criteria to be used in Decomposing Systems into Modules', *Comms ACM* Vol 5 No 12, 1972

[17] H. E. Pattison, D. E. Corkhill and V. R. Lesser, 'Instantiating Descriptions of Organization Structures', in [14]

[18] E. H. Shortliffe, *Computer Based Medical Consultation: MYCIN*, Elsevier, 1976

[19] R. G. Smith, 'The Contract Net Protocol', *IEEE Trans. on Computers*, Vol C-29 No 12, 1980 (reprinted in [1])

[20] R.G. Smith and R. Davis 'Frameworks for Co-operation in Distributed Problem Solving', *IEEE Trans on Systems, Man, and Cybernetics*, Vol SMC-11 No 1, 1981 (reprinted in [1])

[21] L. Sommaruga, N. M. Avouris and M. N. Liedekerbe, 'An Environment for Experimentation with Interactive Co-operating Knowledge Based Systems', Joint Research Centre of the CEC, CITE, KBS Lab, 21020 Ispra (VA), Italy, 1989

[22] E. Werner, 'Co-operating Agents: A Unified Theory of Communication and Social Structure', in L. Gasser and M. N. Huhns (eds), *Distributed Artificial Intelligence (Vol II)*, Pitman, 1989

[23] T. Wittig, 'ARCHON — Co-operation of Heterogeneous On-line Systems', Wissenbasierte Systeme, Proc. 3rd International Congress, Springer-Verlag, 1989

[24] M. J. Wooldridge, 'The Architecture of Co-operating Intelligent Agents', PhD Transfer Report, Dept. of Computation, UMIST, October 1990