Multi-Agent Techniques for Fault Tolerance in Computer Control Systems

A. Stothert and I.M. MacLeod

Department of Electrical Engineering, University of The Witwatersrand,
Johannesburg South Africa
email: {stoth,macleod}@odie.ee.wits.ac.za

27 November 1997

Abstract

By giving agents in a multi-agent system a notion of role, self structuring multi-agent systems become possible. This complex behaviour can be used to provide multi-agent control systems with fault-tolerance characteristics. This article describes fault-tolerance characteristics that have been observed in three implemented multi-agent control systems. Theoretical motivation for the observed characteristics is given.

1 Introduction

Control engineers are mainly concerned with ensuring that a system adheres to predefined behaviours. In order to achieve this the control engineer focusses on modelling the system to be controlled. It has long been recognised that conventional modelling techniques such as differential equations are not ideally suited to model the large scale, time varying, stochastic, uncertain[1] systems that abound in industrial applications [9, 22]. To this end the ideal of an autonomous controller has driven recent control research, and in order to achieve autonomy, techniques from artificial intelligence have been introduced into control engineering [33, 2]. In this paper we report on the use of ideas from distributed artificial intelligence in solving control problems. In particular we discuss three example controllers based on multi-agent techniques and report on fault-tolerance characteristics observed in these systems. The techniques discussed, while at this time lacking rigorous proof techniques, have the potential to complement low-level hardware fault-tolerance and provide system wide operational/behavioural fault-tolerance.

Distributed artificial intelligence [6, 8, 20] has resulted in the development of so called multi-agent systems [3, 32]. An agent is an autonomous entity (program) in a distributed computer system (computing nodes connected by a communication network) which communicates with like agents to perform tasks on their behalf or in cooperation with them. A multi-agent system is simply a set of agents that performs a particular task — in our case control of a system. The advantage of a multi-agent system is its ability to capture diverse knowledge representations and facilitate cooperation between various representations. The

[1]Uncertain both in terms of the system itself and the environment in which it operates.

key idea is one of a community of agents, each performing a specialised task but interacting with other agents to fulfill a global objective. Some researchers refer to a swarm intelligence [11] where there are multitudes of similar simple agents that group together to form specialised tasks or problem solving. The important aspect is that the swarm or community (i.e. the global objective) is not dependent on one agent for its functionality but, rather that it is the collection of agents that results in the required system behaviour. In order to achieve such behaviour it is necessary to replicate knowledge, or more correctly, to give each agent knowledge of itself and knowledge of the community [19].

Once such a system is in place the agents sense the environment (either directly or from information gained from other agents) and react by forming groups of cooperation/ coordination and providing actuation back to the environment (again either directly or via other agents). From this discussion it is evident that the groups of cooperating agents, or indeed the number of agents in the distributed system can change dynamically. A structural change (e.g., number of agents in a cooperating group) is required when the *reasons* for which the system acquired a certain structure *change*. Returning to the control problem — a controller structural change is required (and possible, if a correctly designed multi-agent system is used) when operating conditions change, e.g., movement from one region of operation to another or a plant fault develops. The ability to recover or work around such plant faults is investigated here. To this end we develop a multi-agent control system which self-structures depending on environmental conditions. In addition such a controller employs progressive reasoning techniques, provides fallback solutions and can degrade gracefully.

The rest of this paper is arranged as follows. In section 2 work related to fault tolerance and multi-agent systems is discussed, section 3 discusses the multi-agent model in some detail, section 4 describes the resulting fault-tolerant characteristics and section 5 discusses experimental work highlighting these characteristics. Conclusions are drawn in section 6.

2 Related work

While the multi-agent and distributed artificial intelligence literature includes references applying multi-agent *systems* for fault identification, monitoring and diagnosis (see for example [15, 12]), the authors know of no reference indicating the use of multi-agent *techniques* themselves to provide fault tolerance and recovery. The difference is subtle but important. In the first case static agents are defined for particular tasks, e.g., [15] defines the following agents: collect alarms agent, fault hypothesis agent, validate hypothesis agent. In the second case the agents reorganise themselves to contain and recover from a fault. Similarly, there are references to use of agents for modular redundancy but no references describing structural change as a result of an identified fault.

While negotiation protocols [5, 14] are widely used/described in the DAI and multi-agent literature for dynamically forming groups of cooperating agents, the use of negotiation for fault tolerance has not been discussed — possibly because the agents have not been given knowledge of the global objective as is done in our model. Finally, note that the dynamic agent system described in section 3 has a close parallel in actor systems as described in [1].

The basic fault tolerant characteristic provided by a multi-agent system is conceptually similar to "dynamic or standby redundancy" as described in [21],

> Once the presence of a faulty processor is detected, a fault location or diagnosis procedure is used. The faulty processor is now replaced by a spare processor through reconfiguration techniques.

In a multi-agent system the idea is that once an agent identifies a fault the multi-agent system restructures itself so as to either recover the fault or accommodate it. Restructuring is achieved by changing who individual agents cooperate with and the number of agents in the distributed system (possible since the agents react to changes in the environment). At the lowest level if no cooperation/communication is possible the agents are designed to perform "safe" actions, sometimes called tacit cooperation.

3 Multi-agent model

In this section the multi-agent model is described and some implementation issues are discussed, for more detailed descriptions see [29, 26]. Before discussing the model the underlying principles of a multi-agent system are restated.

The basic premise of a multi-agent system is the so-called community metaphor [4, 13]. In principle the metaphor suggests that complex problems (large problems) are solved by a collection of independent "researchers" who divide the problem into areas of expertise that can be tackled by individual researchers. In this framework the key difficulty lies with partitioning the problem and coordinating results. Partitioning the problem is achieved by problem decomposition and assignment of agents for each partition e.g. [30, 16]. Agents require autonomy (to solve areas of expertise independently) and a means of coordinating results. Coordination is made possible by inter-agent communication which is restricted to message passing. Autonomy can be designed into the agents by taking into account the following required characteristics

- An ability to recognise changes in operating conditions, including changes in goals and/or constraints of the control system.
- An ability to act with and in an imprecise or uncertain environment.
- An ability to utilise background knowledge and/or gain experience.
- An ability to set goals based on their view of the environment.

3.1 Agent model

The key to the agent model is the concept of agent role as introduced by Werner [31]. A social group, or group of cooperating agents, is defined by

$$\Sigma\Gamma = \ < L,\ G,\ \Sigma T,\ \textbf{Roles},\ \Omega >$$

where

- L a language.

- G a group of agents satisfying the social structure. Note that each agent $A \in G$ may have a number of roles given by $\mathbf{RoleCapacity}(A)$.
- ΣT is a social structure, or set of roles $\Sigma T = \ < \mathrm{rol}_1, \ldots, \mathrm{rol}_n >$, i.e., the objective.
- **Roles**, a function specifying the role of agents, $A \in G$, with the restriction: if $\mathrm{rol}_i \in \mathbf{Role}(A)$ then $\mathrm{rol}_i \in \mathbf{RoleCapacity}(A) \cap \Sigma T$
- Ω the environment.

In terms of problem solving a role specifies a partial strategy and by combining roles a complete strategy can be formulated. Notice that by specifying different social structures (ΣT) different groups of cooperation can be formed. We use this characteristic in section 3.2 to create a self-structuring multi-agent system. A natural result of the use of role is that the greater the role capacity of each agent, the greater the flexibility of the group. Finally note that roles themselves may have preconditions that determine when the role is active and conditions to determine when the role is inactive, i.e., role can be specified as a rule set.

Role is not sufficient to define an agent model, in addition an agent requires

$$A = \ < \mathbf{RoleCapacity}(A),\ \mathbf{Behaviour}(A),\ \mathbf{Belief}(A) >$$

- $\mathbf{RoleCapacity}(A)$, the roles that the agent can take on.
- $\mathbf{Behaviour}(A)$, determined by its roles, includes functionality and, input/ output requirements
- $\mathbf{Belief}(A)$, what the agent knows about the environment and other agents, this may include knowledge of the roles of other agents

Practically this is interpreted as requiring an agent to include

- Functionality
- Data structure
- Interaction rules.

where the interaction rules are used to specify the input/output requirements of the agent and the communication protocols to use on each input/output port. Finally the interaction rules are used to represent and manipulate agent roles.

Once specified the agent functionality, data structure and interaction rules define an agent template, i.e., a blueprint from which agents can be created as needed. Agent templates form the building blocks of the multi-agent system.

3.2 Multi-Agent model

With the above agent model in mind a multi-agent system can easily be defined as follows.

The multi-agent system consists of a network of processors which can support (execute) one or more agents. The network includes a mechanism for facilitating communication between agents.

The multi-agent system is started by creating an instance of one of the defined agent templates. Once created, an agent interacts with the environment. Through an understanding of its own role and the possible roles of other agents the agent establishes cooperation groups, possibly by creating an instance of another agent template or by communicating with an existing agent. The group of agents then cooperate to manipulate the environment to achieve the desired behaviour.

3.3 Implementation

Two system agents are required to develop the multi-agent system described above. The first is an agent which creates instances of agent templates, a copy of this launch agent is executed on each processing node. Existing agents request the local launch agent to create a new agent, the launch agent then randomly chooses to either launch the agent locally or request a remote launch agent to launch the agent on that node. The launch agent informs the requesting agent where the new agent was launched.

The other system agent is a "switchboard" agent that is very similar to a conventional blackboard [7, 10]. Inter-agent communication is via message passing, a unique message identifier is created for each communication variable. Each processing node has a switchboard agent which stores the name and address of local communication variables and the address of remote switchboards. Communication between agents thus requires the initiating agent to always specify a communication variable and, when required, a remote node name (hence the launch agent informs the requesting agent where the new agent was launched). The switchboards (local and remote) then route the message to the destination agent and communication is established.

Practically, we have implemented a multi-agent system under windows NT. The launch and switchboard agents are implemented using a combination of `DDE` and `spawn` function calls.

4 Multi-agent fault-tolerance characteristics

As mentioned in the introduction the multi-agent system resulting from the above implementation is capable of dynamic self structuring, it is this self structuring that creates fault-tolerant characteristics. This section describes these fault-tolerant characteristics and their mechanisms. The characteristics are demonstrated experimentally in section 5.

4.1 Autonomy

A multi-agent control system is designed with agent autonomy as one of the design requirements. Autonomy is possible by programming the agent to act rationally when communication with other agents is lost, i.e., the agents' *role* may change when communication is lost. Rational behaviour is the focus of much research effort, but it has been shown that under various assumptions of rational behaviour, tacit cooperation results and possibly sub-optimal, but stable/safe configurations are reached [23]. At its most simple level a rational action for an agent that has lost communication may be to move into a safe shutdown state. However, more importantly, using rational definitions such as Highest Cumulative Reward [25] (HCR) or Maximise Expected Utility [24] (MEU) it is possible for the agent to actively influence the environment without seriously degrading performance.

Practically, autonomy and rationality mean that should two cooperating groups of agents lose contact with each other, global performance is not seriously compromised provided the assumptions under which rational behaviour was defined are present. See for example the power generation grid in section 5.

4.2 Dynamic structuring

Throughout the development of our experimental multi-agent system an emphasis was placed on the ability to dynamically structure the multi-agent system. The use of role enables this dynamic structuring. If, for some reason, an agent identifies that it is not capable of fulfilling its present *role* (due to loss of communication or active fault identification), the agent can

- Look for an alternative method of fulfilling its *role* by attempting to communicate with another agent (possibly launching the agent dynamically).
- Request the destruction (or exclusion from a cooperating group) of an agent that is prohibiting its *role* fulfillment.
- Change its *role* to either work around the fault or attempt to repair the fault.

In each case, the agent's interaction rules must be designed to achieve the required behaviour. The PID example in section 5 demonstrates this behaviour.

4.3 Solution construction and fallback solutions

The dynamic nature of the multi-agent system means that solutions to the global problem are constructed as interaction rules relating to *role* are fired. Firing interaction rules results in communication between agents and possibly the creation of new agents, which, in turn, fire interaction rules. If carefully designed, this characteristic can be used to develop a solution which is initially coarse (say one agent), but is refined as more agents are brought in by interaction rules. This provides an obvious solution fallback facility.

The refrigeration example in section 5 uses this technique.

5 Experimental system(s)

Three experimental multi-agent systems are used to describe the fault-tolerant characteristics outlined in the previous section. Each system has been implemented using the models described in section 3. Unfortunately, space restricts discussion of all implementation details. More detail is given in the references.

The power grid system [27, 17] consists of multiple generator agents (30+ agents) cooperating to supply electricity to a single grid agent. The system demonstrates how autonomy and rational behaviour can result in cooperation and fault tolerance characteristics. The system operates by the grid agent broadcasting a required electricity load and estimated cost/MW price. The generator agents then bid to supply a fixed load at a fixed cost/MW rate, this bid is determined from a cost curve and calculated so as to maximise generator profit (MEU). The grid agent then selects bids to satisfy the load and minimise cost (MEU), the resulting grid agent selections are broadcast with the actual cost/MW and the generator agents can resubmit bids. Throughout this process the generator agents act independently of each other, yet through rational actions based on the MEU principle, the process converges to both a Nash equilibrium and Pareto Optimum [28], i.e., the individual generators cannot improve their profit without adversely affecting other agents. Also, the grid agent is guaranteed to pay a minimum to the generators to supply the load. From a fault-tolerance perspective, the multi-agent system does not depend on the number of generator agents (a fault could result in some generators not being able to bid, i.e., a portion of the grid becomes disconnected) yet coherent global performance is still possible. Figure 1 shows how a required load is satisfied simply through application of the MEU principle.

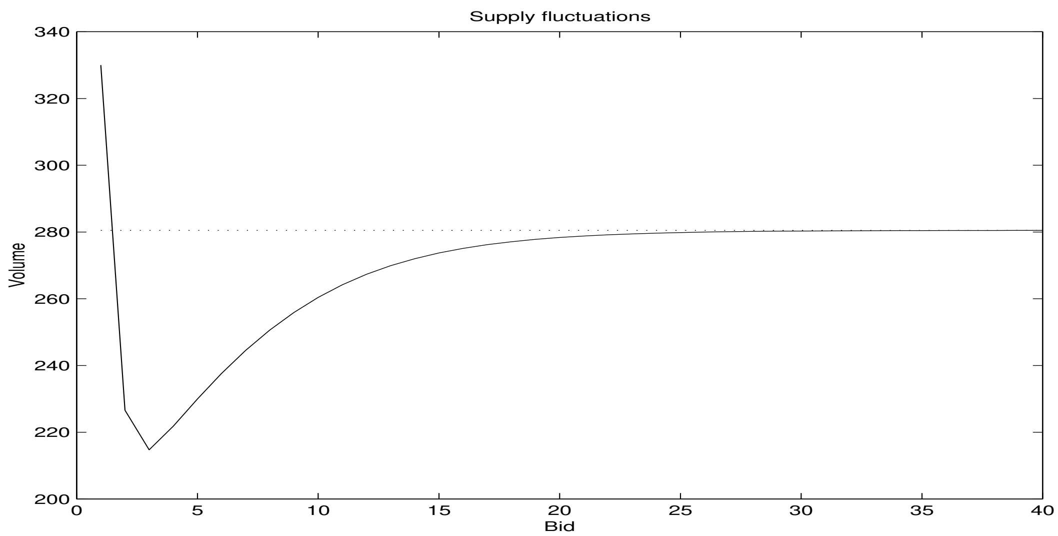

Figure 1: Cooperation via autonomous rational agents.

A multi-agent proportional-integral-derivative (PID) controller [29] demonstrates how a change in agent role can be used to adapt a multi-agent system. The PID controller consists of a sensor agent, a control agent, a configuration agent, a derivative agent and an integral agent. The system operates by initially creating a sensor agent which then creates a control agent. The primary role of the control agent is to provide an actuation signal to the plant under control this involves coordinating information from derivative and integral agents if necessary. In order to determine whether either a derivative or integral agent is required the control agent uses a configuration agent. During operation if an error causes the required configuration to change derivative and/ or integral agents are either created or destroyed. Figure 2 shows this behaviour, the plots indicate processor loading when new agents are created and destroyed — at around 40 seconds a new agent configuration was created.

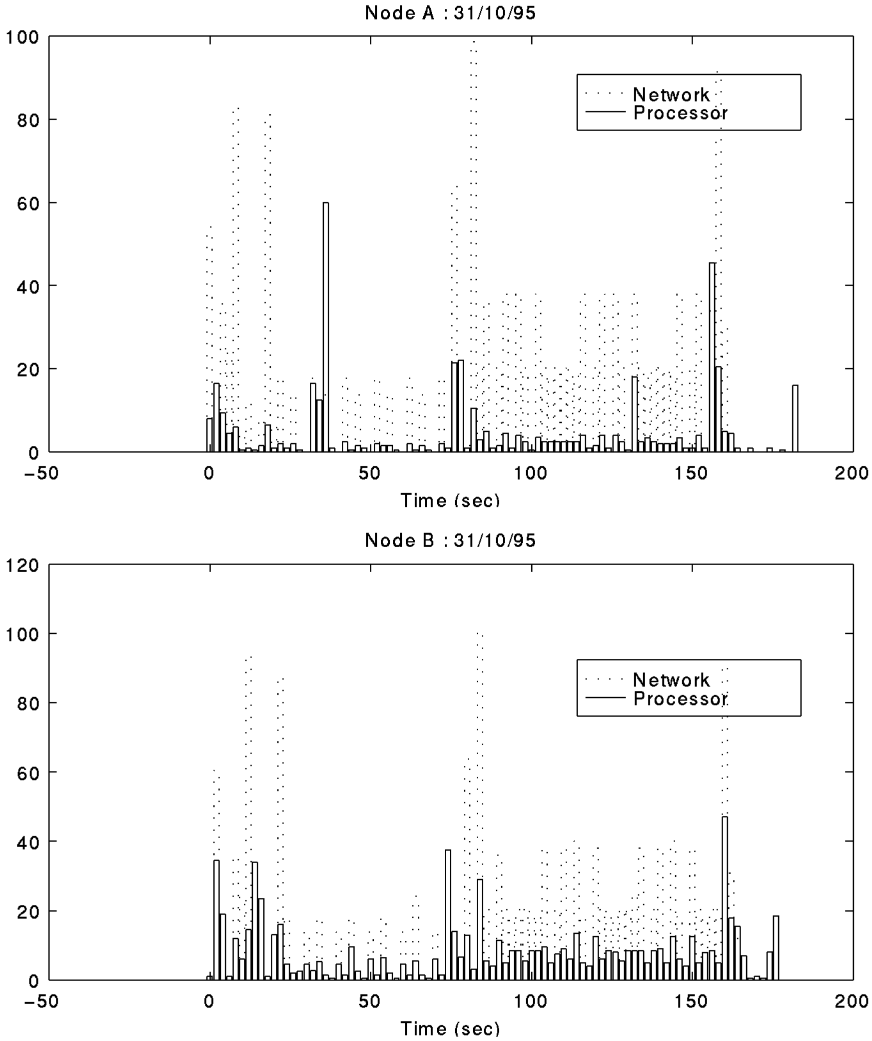

Figure 2: Simulated dynamic operation for measurement intervals of 2 seconds. Network load is measured in units of 10 Bytes per second and processor loading as % time active in interval.

A multi-agent system for control of an underground water reticulation system [18] demonstrates how the multi-agent system constructs solutions. Briefly the system under control consists of a sequence of reservoirs and heaters, the control objective is to maintain a supply of heated water at the outlet reservoir. Agents are defined for each heater and reservoir. The multi-agent system is used to develop a plan to control the heaters and reservoirs to meet the supply. Figure 3 shows comments generated by the multi-agent system when particular heaters are unavailable. Figure 3 does not completely demonstrate the process of constructing a solution, however the system operates by agents proposing plans, only to have them modified by other agents due to environmental constraints. Each plan proposed consists of a "like to have" component and "at worst" component so that at all times there is a workable solution. Once again the plan is constructed by the agents by considering their role and the role of other agents.

```
Experiment 2:
Requirements

  • Flow = 5 l/min.

  • Temperature = 40°C.

  • Only heaters 0110 available (constraint)

Results
Planned Heater configuration = 0110 with the surface
"heater" plant outlet being heated to 29.3°C and
underground outlet set to 40°C.
Plan Comments

  • No series heater branches available -- limited maximum
    Temp.

  • Inlet being heated.

  • Plan finalised.

Prior to heating the surface "heater" plant outlet an
underground outlet temperature of 34.2°C was proposed by
the underground agent.
```

Figure 3: Refrigeration Experiment: Plan 2 Results.

6 Conclusion

This paper reports on a multi-agent framework that has been used for control of various physical systems. The framework is capable of complex behaviour that can be used to construct controllers with fault-tolerant characteristics. In particular, the following characteristics have been identified

- Agent autonomy
- Dynamic structuring
- Solution construction and fallback solutions

While the framework is capable of implementing these mechanisms, it is necessary for the system designer to specify how these behaviours are to be achieved (implementation details specific to each problem). Three implemented multi-agent control systems, each of which uses one of the fault-tolerance mechanisms, have been discussed.

To conclude, by borrowing ideas from artificial intelligence, multi-agent systems provide a novel, powerful, framework for sophisticated high-level fault-tolerance in control systems. This multi-agent approach is demonstrated by the discussed examples and is worthy of further investigation, particularly with respect to formal proof and modelling methods.

References

[1] A. Agha Gul. *Actors: A Model of Concurrent Computation in Distributed Systems.* MIT Press, 1986.

[2] P. Antsaklis and K. Passino. *An Introduction to Intelligent and Autonomous Control.* Kluwer Academic Publishers, 1993.

[3] S. Bird. Toward a taxonomy of multi-agent systems. *International Journal of Man Machine Studies*, 39(4):689–704, October 1993.

[4] B. Chandrasekaran. Natural and social system metaphors for distributed problem solving: Introduction to the issue. *IEEE Transactions on Systems Man and Cybernetics*, 11(1):1–5, January 1981.

[5] R. Davis and R. G. Smith. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, 20:63–109, January 1983.

[6] E. Durfee. The distributed artificial intelligence melting pot. *Transactions on Systems Man and Cybernetics*, 21(6):1301–1306, November 1991.

[7] R. Englemore and T. Morgan. *Blackboard Systems.* Addison-Wesley, 1988.

[8] L. Gasser and M. Huhns, editors. *Distributed Artificial Intelligence Volume II.* Morgan Kaufmann, 1989.

[9] C. Harris. Guest editorial. *International Journal of Control*, 56(2):259–261, August 1992.

[10] B. Hayes-Roth. A blackboard architecture for control. *Artificial Intelligence*, 26(3):251–321, July 1985.

[11] J. Hoffmeyer The swarming body. Accessed at *http://www.molbio.ku.dk/MolBioPages/abk/PersonalPages/Jesper/Swarm.html* on 6 March 1997.

[12] N. Jennings, L. Varga, R. Aarnts, J. Fuchs, and P. Skarek. Transforming standalone expert systems into a community of cooperating agents. *Engineering Applications of Artificial Intelligence*, 6(4):317–331, August 1993.

[13] W. Kornfeld and C. Hewitt. The scientific community metaphor. *IEEE Transactions on Systems Man and Cybernetics*, 11(1):24–33, January 1981.

[14] S. Kraus. Negotiation and cooperation in multi-agent environments. *Artificial Intelligence*, 94(1-2):79–97, July 1997.

[15] G. Lekkas, N. Avouris, and G. Papakonstantinou. Development of distributed problem solving systems for dynamic environments. *IEEE Transactions on Systems, Man, and Cybernetics*, 25(3):400–414, March 1995.

[16] V. Lesser and D. Corkill. Functionall acurate, co-operative distributed systems. *IEEE Transactions on System Man and Cybernetics*, 11(1):81–96, January 1981.

[17] I. MacLeod and A. Stothert. Implementation of a multi-agent system for power pool modelling. Technical Report TRR/R/97/R97032, ESKOM, Technology Group TRI, July 1997.

[18] I. MacLeod and A. Stothert. Distributed intelligent control for a mine refrigeration system. *IEEE Control Systems Magazine*, 1998. In print.

[19] G. Morgan. *Images of Organization.* Sage, 1986.

[20] G. O'Hare and N. Jennings. *Foundations of Distributed Artificial Intelligence.* John Wiley and Sons, 1996.

[21] D. K. Pradhan. *Fault-Tolerant Computer System Design*, chapter 3, pages 150–181. Prentice Hall, 1996.

[22] M. Rao. Frontiers and challenges of intelligent process control. *Engineering Applications of Artificial Intelligence*, 5(6):475–481, November 1992.

[23] S. Russell. Rationality and intelligence. *Artificial Intelligence*, 94(1-2):57–77, July 1997.

[24] R. Sainsbury. *Paradoxes.* Cambridge University Press, 1988.

[25] Y. Shoham and M. Tennenholtz. On the emergence of social conventions: modelling, analysis, and simulations. *Artificial Intelligence*, 94(1-2):139–166, July 1997.

[26] A. Stothert. *Coordinating Agent Interaction in Distributed Intelligent Control Systems.* PhD thesis, University of the Witwatersrand, Johannesburg, March 1996.

[27] A. Stothert and I. MacLeod. Feasibility study on the application of multi-agent techniques for power generation grid modelling and control. Technical Report TRR/E/96/EL194, ESKOM, Technology Group, October 1996.

[28] A. Stothert and I. MacLeod. Competitive bidding as a control problem. *IEEE Transactions on Power Systems*, 1997. In preparation.

[29] A. Stothert and I. MacLeod. Using intelligent agent templates for dynamic structuring of distributed computer control systems. *Engineering Applications of Artificial Intelligence*, 10(4):335–343, August 1997.

[30] M. Torngren. On the modelling of distributed real-time control applications. In *13th IFAC Workshop on Distributed Computer Control Systems*, pages 13–18, Toulouse-Blagnac (France), September 1995. IFAC.

[31] E. Werner. Cooperating agents: A unified theory of communication and social structure. In L. Gasser and M. Huhns, editors, *Distributed Artificial Intelligence Volume II*, chapter 1, pages 3–36. Morgan Kaufmann, 1989.

[32] E. Werner and Y. Demazeau, editors. *Decentralized A.I. 3: Proceedings of the Third Eurpoean Workshop on Modelling Autonomous Agents in a Multi-Agent World.* North-Holland, 1992.

[33] D. White and D. Sofge. *Handbook of Intelligent Control.* Van Nostrand Reinhold, 1992.