# A Hierarchical Blackboard Architecture for Distributed AI Systems

Michael Weiss* and Franz Stetter
Lehrstuhl für Praktische Informatik I
Universität Mannheim
6800 Mannheim 1, Germany

## Abstract

*A framework for distributed problem solving should support the incremental development and integration of problem solvers into the system in agreement with the principles of stepwise refinement and functional decomposition. The organizational structure of the framework should guide the mapping of each of the functional areas onto agents and its communication structure should reflect this decomposition.*

*We describe a hierarchical multi-blackboard architecture, called Hierarchical Blackboard System (*HBBS*) which satisfies these requirements. It serves as an integration architecture for heterogeneous expert systems. Important features of the system, such as the hierarchy concept, communication strategy and configuration management of the system at run-time are discussed. A working prototype* HBBS *for distributed CAD/CAE of communication systems is described.*

## 1 Introduction

Problem solving for complex, real-world problems requires the combined effort of several problem solvers. Each of these problem solvers provides expertise in a particular area. In our notion problem solvers are not limited to rule-based expert systems, but also include number-crunching programs, simulators and intelligent user interfaces.

In the approach of distributed artificial intelligence, problem solvers are decentralized and loosley coupled [11]. They can work concurrently, so that overall exextion time is greatly reduced. They may be distributed to several different machines and use multiple representation paradigms and programming languages.

*E-mail: *weiss@pi1.informatik.uni-mannheim.dbp.de*

Frameworks for distributed problem solving should provide support for an incremental development and integration of problem solvers into the system in agreement with the principles of stepwise refinement and functional decomposition; an organizational structure that guides the mapping of each of the functional areas onto agents; and a communication structure that reflects this decomposition.

This paper describes a blackboard-based framework HBBS (Hierarchical Blackboard System) which satisfies these requirements. We will first give a discusson of previous approaches in distributed AI with particular emphasis on hierarchy-based approaches to motivate our idea of hierarchy. Important features of the system, such as communication strategy and configuration management of the system at run-time are subsequently discussed. Finally, a working prototype HBBS for distributed CAD/CAE of communication systems is presented in some detail.

## 2 Previous Work

### 2.1 Models of Distributed AI

There are three models for DAI prominent from the literature: the blackboard model, the actor model and the contract net approach.

The blackboard model introduced in the Hearsay II system [5] is a model of inter-agent communication that uses a global data structure, referred to as the 'blackboard', which can be accessed by a number of knowledge sources (KS). These knowledge sources communicate by manipulating the contents of the blackboard.

Knowledge sources and the blackboard interact according to the following cycle: poll, hypothesize, then test. Initially, some tentative hypothesis is put on the blackboard. Then the knowledge sources are polled to see if they can offer a refinement to the current



0-8186-2830-8/92 $03.00 © 1992 IEEE

hypothesis. Each knowledge source that believes it can contribute to the refinement indicates its ability to do so, in turn. Subsequently, one of the knowledge sources is selected by the scheduler, and the refinement is applied. After that, a test is performed involving the other agents to evaluate this refinement, which is then either rejected or adopted.

The original Hearsay-II approach, however, suffered from the problem that soon the blackboard became a bottleneck, and from reliability problems. A distributed version of Hearsay-II was then considered to alleviate these problems [12]. It consisted of six geographically distributed nodes with separate blackboards in each node. Access to remote blackboards was by means of messages sent to guard processes in each node.

The actor model [9] is built around the concept of 'actor'. Actors communicate with each other in a direct manner by messages which are actors themselves. The behavior of an actor is defined by a script which prescribes the actions for actor to take upon receipt of any particular message. Actors communicate only with a subset of actors contained in its acquaintance list. This model alleviates the congestion problems associated with the Hearsay-II approach.

Both the blackboard model, as described above, and the actor model do not address the problem of distributing tasks onto agents. This problem is more commonly referred to as task decomposition.

The contract net model [20, 18] provides a solution to the decomposition problem which is based on the metaphor of negotiation among autonomous intelligent beings. It has been applied to vehicle tracking in a distributed sensor system and in task allocation manong the members of a distributed system CPUs [18]. The contract net protocol proposes that an agent (the manager) which identifies a task to be done, issues a task announcement by way of a broadcast to all agents available (the potential contractors). These examine the task and place a bid for the task if they are willing and able to perform it. If several managers compete for bids, the task for which the agent is most suited is selected for the bid. The manager then compares all the bids and awards the contract to the agent it thinks most suitable. A communication channel between manager and contractor is then established.

## 2.2 Distributed AI Frameworks

Recently, systems have been developed with the intention of providing a generic framework for distributed AI systems.

A number of different approaches exist. In [10] a hierarchically distributed system of agents is proposed, in which communication is only permitted between direct descendants of specific agents. Also models for heterogeneous agents employing different representations have been proposed, most notably [16].

Most of these approaches use a single paradigm only. These include the Distributed Vehicle Monitoring Testbed (DVMT) [3, 13], the blackboard systems BB1 [8], GBB [6] and the systems of the AGE family which include AGE, CAGE, POLIGON [15], and TRICERO [24].

The DVMT is a testbed for distributed problem-solving research in a distributed sensor domain. It consists of a number of cooperating blackboard nodes with. Each node is responsible for part of the sensed area, nodes develop partial maps in parallel and exchange their results to construct a complete map of vehicles moving through the area. Otherwise they are identical. A simulation system has been built for DVMT which incorporates the language EFFIGE for describing the organizational structure of a distributed processing network [19]. Such a description specifies the system's functional components, their responsibilities and resource requirements, and the relations among them.

Another approach to distribute a blackboard system is to keep the blackboard data in shared memory and to distribute the knowledge sources on processing nodes, which can execute in parallel. This approach is illustrated by systems like POLIGON [15] and BLONDIE-III [14].

A more direct form of distribution is to partition the problem into independent subproblems, where each subproblem is assigned to a separate processing node. Different blackboard systems are assigned to each node. TRICERO [24] is an example of this kind of system. TRICERO consists of three autonomous expert systems for signal understanding in the area of air defense. The two lower-level systems, ELINT and COMINT, interpret radar and voice data, respectively. CORRELATION integrates reports it receives from ELINT and COMINT to analyze global aircraft activity. It also provides feedback to ELINT and COMINT.

Each of TRICERO's expert systems is implemented as a blackboard using the AGE framework. Since three separate machines were not available for running each of the three expert systems of TRICERO in its own machine, multiple blackboard systems were emulated on a single machine.

To conclude with, we will consider two more recent developments, a real-time multi-blackboard sys-

tem, HOPES [2], and the distibuted diagnosis expert system TOAST, as one of the few distributed AI systems actually implemented and in use.

HOPES isa hierarchically organized parallel expert system. It provides a framework for real-time parallel expert systems. Its basic structure is illustrated by figure 1.

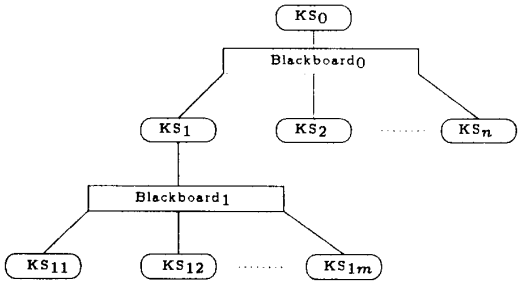



Figure 1: System Structure of HOPES

The tree structure of HOPES was adopted for several reasons. First, the organization of the KSs naturally corresponds to the multi-stage analysis or pipelining found in real-time processing. In signal understanding application, for example, there are several stages from raw sensor data to final interpretation to each task.

Second, a tree provides a convenient way for task decompostion. Should a task be too complex, it can be further divided into subtasks. Finally, all KSs are capable of running concurrently. Each KS, execpt for the root KS, is used to perform a sub-task in parallel. It must be noted, however, that information flow in HOPES is bottom-up only. Feedback and tasking information as in the TRICERO system are not transferred between blackboard nodes.

The hierarchical structure of HOPES is motivated as follows. First, a single blackboard may become a bottleneck when the number of KSs increases (as has been observed with the Hearsay-II architecture). Second, the use of a multi-blackboard architecture will be able to reduce information complexity for complex tasks. Third, since information is distributed over several blackboards, each blackboard contains relatively small amount of information, which increases system efficiency. Finally, managing the multi-blackboard architecture is easier than managing a big, single blackboard, because management is spread over several KSs.

Communication and cooperation of knowledge sources in HOPES is achieved by meta-knowledge. Besides domain knowledge, each KS is instrumented with knowledge about information exchange and about data transformation. The information exchange knowledge is used with a similar purpose as acquaintance lists in the actor model: If a KS 'thinks' that an intermediate result it has obtained may be useful to other KS it puts this result into a globally accessible section of its blackboard. The announcment of the result consists of an 'abstract' that states purpose and possible contribution of the result, the name of the announcer, and a list of names of possible 'benificiaries', i.e. KSs that may find the information useful.

Whenever a KS needs the help of other KSs, it checks the announcements on the blackboard for potential benificiaries listed there. Then it considers the abstract of the announcement to find out whether the information there is really useful. Benificiaries listed in each announcement are functionally equivalent to triggers in the Hearsay model. In the same way, abstracts serve as preconditions. The name of the announcer may then be used as an index to obtain the information. If necessary, the data transformation knowledge is used to interpret the information. After reading an announcement, a KS deletes its name from the list of benificiaries of that announcement.

Simulation results have shown the feasibility of the hierarchical approach of HOPES. The speed-up factors computed demonstrated that a multi-blackboard architecture approach is capable of substantially reducing the possibilities of contention in communication channels, i.e. blackboards.

The diagnosis system TOAST [21] is a development which is largely independent of other distributed AI systems. It is based on an extension of OPS5 for inter-system communication called Cops (concurrent production system). This extension allows modifying access to the inference machine of other OPS5 programs by asynchronous messages.

A blackboard system is implemented in TOAST as a collection of processes. One process is assigned to handle blackboard functions. Its working memory constitutes the actual blackboard. Other processes create elements on this blackboard directly by remote write access, which is provided by the Cops extension to OPS5. For reading from the blackboard TOAST introduces the concept of 'ambassadors'. An ambassador is a set of rules in the production memory of the blackboard process that describes how working memory elements are to be read and manipulated. The production memory also contains a monitor which does housekeeping on the blackboard and oversees the ambassadors. At present, ambassadors cannot be created

dynamically but must be loaded at initialization of the blackboard process.

Toast consists of a diagnosis subsystem, a user interface expert and a component for concurrent simulation in a separate blackboard.

Multi-paradigm problem-solving frameworks have also been built. Of particular interest is the MACE (multi-agent computing environment) [7]. It is based on the blackboard, actor and contract net concepts described above. Experimental systems have been built to validate MACE, including a re-construction of the contract net system and an implementation of a simple distributed blackboard system.

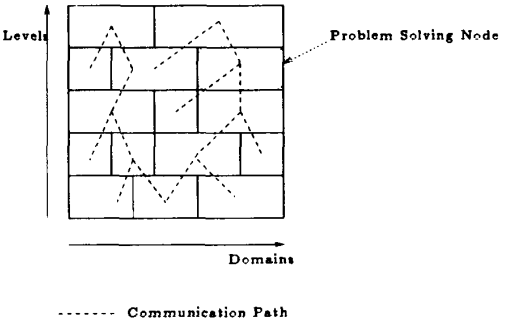



Figure 2: Assignment of Problem Solvers

# 3 A Hierarchical Blackboard System

## 3.1 Hierarchy Concept

As the TRICERO and the HOPES examples show, subsystems should be logically independent in order to reduce the need for coordination between subsystems. The analysis of abstract information is assigned to higher-level nodes and the analysis closer to input data to lower-levels. It becomes evident that such an approach satisfies our goal that the organizational structure of the framework guide the mapping of the functional areas onto problem solving nodes.

Results are transferred to higher levels in very much the same way. There is always at least one KS attached to the blackboard, which has also access to a higher level blackboard. From this it can be seen that all intermediate results from KSs at a certain layer cannot appear on higher levels, if they are not relevant to those higher levels. In this way the communication structure effectively limits the access of a problem solving node to interfaces defined by functional decomposition. An example of how the communication of multiple blackboards in a hierarchy of blackboards can be achieved in a distributed environment is given by the TOAST system.

The need for abstraction and logical independence of problem solving tasks motivates a hierarchical approach. The hierarchy concept is implemented by assigning problem solvers to a single level of abstraction, and restriction of a problem solver's access to immediately adjacent levels in the hierarchy. Each problem solver is further assigned to a domain on its level of the hierarchy (see figure 2).

This organization avoids the bottleneck, which occurs, when problem solvers of different levels of the hierarchy and of different relevance concerning their contribution to the problem solution can access the same blackboard.

## 3.2 Inter-KS Communication

In this section we show how the hierarchical structure is introduced as a natural extension to the shared memory approach of communication among KSs in the original blackboard concept.

The basic structure of a problem solving node is shown in figure 3. Messages can only be passed between immediately adjacent levels of the hierarchy.

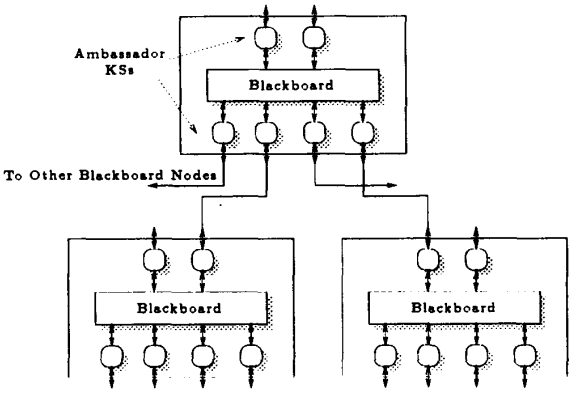



Figure 3: Inter-KS Communication in HBBS

A subsystem must be able to understand what lower hierarchies can contribute to the current task. This understanding is provided by meta-knowledge which is contained in 'ambassador' knowledge sources (adopted from [21]). An ambassador knowledge source acts as a representative of another subsystem, which is capable of selecting information relevant to this

subsystem from the blackboard. The ambassador approach avoids the specification of 'information exchange' knowledge as in the HOPES system. The decomposition into subsystems assumes that blackboard information is principally interesting to all systems directly attached to the blackboard. Note that there may be more than one ambassadors for communication with higher level nodes in the hierarchy.

Subsystems can make changes on adjacent subsystems or subsystems on the same level at will by sending a message to that blackboard node or putting a message on an agenda, respectively. They can be sent asynchronously and are processed in the subsystem when all previous messages have been acted on. Message passing is implemented on top of the process intercommunication primitives available in Berkley UNIX. KSs are defined using the syntax given below.

```
(define-ks '(<ks-name>
  (<trigger-condition>)
  (<pre-condition>)
  (<obviation-condition>)
  ([(make ...) ...]
   [(send <subsystem-name>  ...)])
))
```

Each KS is uniquely identified by name. When a KS is executed by the control mechanism, first a trigger condition is tested, whether the blackboard information produced by the change is useful to the subsystem represented by the KS. Then, a pre-condition is checked in order to set some variables whose values are not set in the trigger. The obviation condition serves a similar purpose, except that it executes a Lisp expression on the variables set, which must evaluate true. The action part of the KS can contain any number of calls to *make* and *send*. *Make* is used to put information on the local blackboard, while *send* creates a message to the subsystem specified which must be an adjacent node.

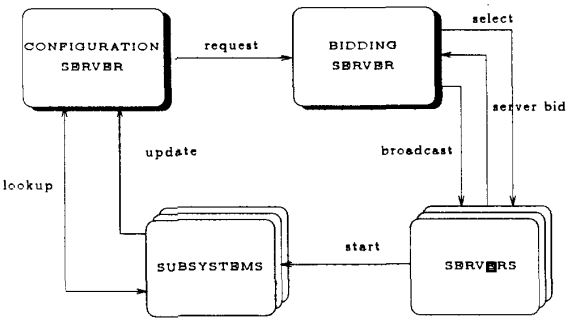
The control cycle may be summarized as follows:

1. Take the message from the top of the agenda; this may be either a new blackboard object or a hypothesis.
2. Try all knowledge sources in turn to check whether its trigger condition is satisfied.
3. If so, match its pre-condition with the objects on the blackboard.
4. If the pre-condition is satisfied, test for the obviation, condition, which is a lisp expression that must be true for the knowledge source to execute.
5. If so, execute the action part; execution of the action part produces new blackboard objects and hypotheses.
6. Increment the current cycle count.
7. Test the agenda if it is empty. Then wait for the arrival of new messages from adjacent subsystems.

Currently the cycle count is used as priority for entries on the agenda. This implements a simple recency-based execution strategy.

## 3.3 Run-Time Configuration

The architecture used for maintaining subsystem locations and initially starting up HBBS is shown in figure 4.




The configuration server maintains the information about subsystem locations (host info, etc.) in a central database, whose location is fixed and globally known. Whenever a subsystem wants to "hook up" to some other subsystem, it first queries the database for the location of that tool. In this way the database documents the current configuration of the whole system in terms of subsystem locations.

When a subsystem started up and another subsystem the subsystem wants to connect to is not already running, i.e. on the database, this subsystem will be started up by the bidding server. Any such subsystem be started up in the sequence of start-up requests and its location updated in the configuration server's database.

Interaction with the bidding server users a bidding protocol similar in spirit to the contract net model: *broadcast* a subsystem request to a number of servers (each on a different host) which can execute the requested system, then *gather* server replies (bids) and finally *select* from those offering to run subsystem (based on the load on that machine etc.) and post an execute request to it. More details on the bidding protocol can be found in [22].

## 4 An Application of HBBS

We currently study the application of the hierarchical blackboard paradigm to the construction of distributed CAD/CAE systems. This application was selected intentionally to provide a real measure to our proposed architecture. Our presentation of a working HBBS prototype will proceed in the order of the incremental design used in building it.

An initial system consisted of the editor for local area networks hlaned and the Petri net editor xsim, which each interface to an interaction subsystem IA via then HLANED and XSIM subsystems (figure 5). The inclusion of the latter is motivated by the possibility of extending the architecture to the use of multiple editors of each type, while preserving the abstraction of a single editor to the rest of the system.

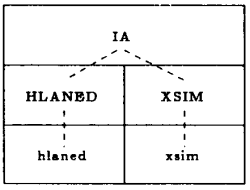



Figure 5: The IA Subsystem

Whenever the design of the local network is modified, the information on the HLANED blackboard is also modified, which in turn notifies the IA subsystem via the IA ambassador knowledge source information flow). XSIM will react to the new information on the IA and in turn inform xsim of the change. In this way hlaned and xsim can keep a consistent view on naming and toplogy.

This architecure is now extended to include a simulation subsystem. This system is first defined separately (figure 6).

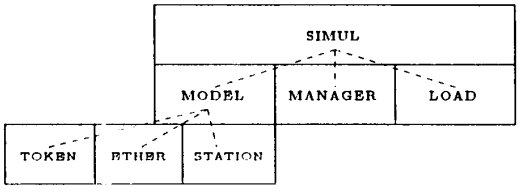



Figure 6: The SIMUL Subsystem

It includes a system for distributed simulation for local area networks with subsystems for model management, simulation process management and load balancing. The basic architecute is completed by the timing analysis subsystem XPAN, which does a Petri net analysis of the nets in xsim with the transisition times from the simulation subsystem. Also, the database PROTOTYPE which contains prototype definitions for the model components has been added (figure 7).

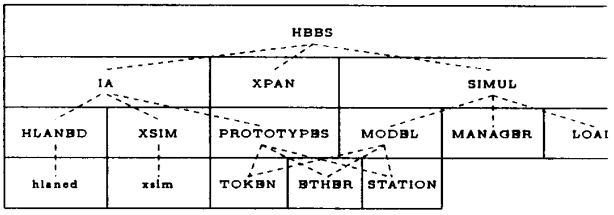



Figure 7: The Structure of the HBBS prototype

As we can see, HBBS uses both bottom-up and top-down information flow. This makes it a mixed-initiative approach. More details about the modeling and simulation approach used in the prototype system can be found in [23].

HBBS has been implemented in the programming language Scheme, which is a Lisp dialect [1] with a static closure mechanism. With static closure an object-oriented style can be easily emulated. The blackboard, knowledge sources and the control mechanism of a subsystem are defined as objects.

## 5 Conclusion

We describe a new blackboard concept for hierarchical multi-expert architectures, which naturally extends existing tendencies to introduce hierarchy and concurrency into blackboard systems, and attempts to make them more generally applicable. The new concept is demonstrated with a working protoype in a system for distributed CAD/CAE of communication systems. It is our conviction that the usefulness of distributed AI methods can not be evaluated 'in the test tube', i.e. using a simulation approach. Future research will focus on a design methodolgy for distributed AI systems based on the hierarchical blackboard system model.

## References

[1] Abelson, H.; Sussman, G.; Sussman, J. 1985.

*Structure and Interpretation of Computer Programs*, MIT Press.

[2] Dai, H.; Anderson, T.J.; Monds, F.S. 1990. "A framework for real-time problem solving," *9th European Conference on Artificial Intelligence ECAI '90*, Stockholm, Aug: 179–185.

[3] Durfee, E.H.; Lesser, V.R.; Corkill, D.D. 1987. "Cooperation through communication in a distributed problem solving network," in: Huhns, M.N. (ed.), *Distributed Artificial Intelligence*, Pitman, London, pp. 29–58.

[4] Durfee, E.H.; Lesser, V.R.; Corkill, D.D. 1989. "Cooperative distributed problem-solving," in: Barr, A.; Cohen, P.R.; Feigenbaum, E.A., *The Handbook of Artificial Intelligence*, Addison-Wesley, Reading.

[5] Erman, L.D.; Hayes-Roth, F.; Lesser, V.R.; Reddy, D.R. 1980. "The Hearsay-II speech-understanding system: Integrating knowledge to resolve uncertainty," *ACM Computing Surveys* 12: 213–353.

[6] Gallagher, K.Q.; Corkill, D.D.; Johnson, P.M. 1988. *GBB Reference Manual: GBB version 1.2*, COINS technical report 88-86, Department of Computer and Information Science, University of Massachusetts, Amherst, July.

[7] Gasser, L.; Braganza, C.; Herman, N. 1987. "MACE: A flexible testbed for distributed AI research," in: Huhns, M.N. (ed.), *Distributed Artificial Intelligence*, Pitman, London, pp. 119–152.

[8] Hayes-Roth, B. 1985. "A Blackboard Architecture for Control," *Artificial Intelligence* 26: 251–321.

[9] Hewitt, C. 1977. "Viewing control structures as patterns of passing messages," *Artificial Intelligence* 20(3) June: 323–364.

[10] Huhns, M.N.; Stephans, L.M.; Bonnel, R.D. 1983. "Control and co-operation in distributed expert systems," *Proceedings of the IEEE Southeast Conference*, Orlando, Florida, April: 241–245.

[11] Huhns, M.N. 1987. "Foreword," in: Huhns, M.N. (ed.), *Distributed Artificial Intelligence*, Pitman, London, pp. v–ix.

[12] Lesser, V.R.; Erman, L.D. 1980. "An experiment in distributed interpretation," *IEEE Transactions of Computers* 29(12) Dec.: 1144–1163.

[13] Lesser, V.R.; Corkill, D.D. 1988. "The distributed vehicle monitoring testbed: A tool for investigating distributed problem solving networks," in Englemore, R.; Morgan, T. (eds.): *Blackboard Systems*, Addison-Wesley, Reading, pp. 353–386.

[14] Liempd, G. van; Velthuijsen, H.; Florescu, A. 1990. "Blondie-III," *IEEE Expert*, Aug: 48-55.

[15] Nii, H.P.; Aiello, N.; Rice, J. 1988. "Frameworks for concurrent problem solving: A report on CAGE and POLIGON," in Englemore, R.; Morgan, T. (eds.): *Blackboard Systems*, Addison-Wesley, Reading, pp. 475–502.

[16] O'Hare, G.M.P. 1986. "New directions in distributed artificial intelligence", *Proceedings of the 2nd International Expert Systems Conference*, London, Oct.: 137–148.

[17] O'Hare, G.M.P. 1989. "Designing intelligent manufacturing systems: A distributed artificial intelligence approach," *Computers in Industry*, Elsevier.

[18] Parunak, H.V.D. 1987. "Manufacturing experience with the contract net," in: Huhns, M.N. (ed.), *Distributed Artificial Intelligence*, Pitman, London, pp. 285–310.

[19] Pattison, H.E.; Corkill, D.D.; Lesser, V.R. 1987. "Instantiating descriptions of organizational structures," in: Huhns, M.N. (ed.), *Distributed Artificial Intelligence*, Pitman, London, pp. 59–96.

[20] Smith, R.G. 1980. "The contract net protocol: High-level communication and control in a distributed problem solver," *IEEE Transactions of Computers* 29(12) Dec.: 1104–1113.

[21] Talukdar, S.N.; Cardozo, E.; Leão, L.V. 1986. "Toast: The Power System Operator's Assistant," *IEEE Expert*, July: 53–60.

[22] Weiss, M.; Stetter, F. 1992. "The bidding model of configuration control," to appear in: *Fachtagung CAD '92*, Informatik Fachberichte, Springer, Berlin.

[23] Weiss, M.; 1992. "Multi-level simulation of real-time LANs," to appear in: *Proceedings of the European Simulation Multiconference 92*, SCS.

[24] Williams, M.A. 1988. "Hierarchical multi-expert signal understanding," in Englemore, R.; Morgan, T. (eds.): *Blackboard Systems*, Addison-Wesley, Reading, pp. 475–502.