MCC Technical Report Number ACT-RA-215-90

ROI for DAI

Michael N. Huhns and David Murray Bridgeland

June 1990

MCC Nonconfidential

Microelectronics and Computer Technology Corporation
Artificial Intelligence Laboratory
3500 West Balcones Center Drive
Austin, TX 78759-6509
(512) 338-3651
huhns@MCC.COM

ROI for DAI

Michael N. Huhns and David Murray Bridgeland
Reasoning Architectures Group
MCC

Abstract

The Reasoning Architectures group at MCC has been conducting research in the area of distributed artificial intelligence (DAI). This executive summary presents the potential benefits ("return on investment") to be gained from this research. In particular, we describe several applications, ranging from process control to office automation, that can be solved with *RAD*, a distributed expert system shell that we have developed. *RAD* enables human and computational agents to collaborate in solving problems.

1 Uses and Benefits

At a large chemical plant in rural Oklahoma, dozens of expert systems are used to control the processing of petroleum-based chemicals. As with any plant of this type, the processing is highly interconnected, with chemicals made in one part of the plant used in producing the chemicals made in another part of the plant. Recently, a boiler failure caused an expert system to shut down the production of a solvent that was needed in another process producing latex paint. Unfortunately, the expert system controlling the paint process did not find out about the shut down until the solvent in the input pipe to the column dried up. After the dried paint was cleaned from the column six months and $2 million later, the paint process was back on line.

The problem was that the processes in this plant were connected at the physical level, but the expert systems were not connected at the knowledge level, even though they were written in the same language, ran on the same hardware, and were connected by ethernet. They were not designed to communicate! They were unaware of the decisions being made by the other

expert systems, so they were unable to take corrective action until, as in this case, it was too late.

So, why not centralize the control of the plant by using just one large expert system? Then there would be no need for communications. Unfortunately, this makes control of the plant extremely complex and very unreliable. Also, the development effort required for such a large expert system would be prohibitive.

RAD has been developed to provide a better solution to this kind of problem, and it can solve it in either of two possible ways. First, any expert system constructed using *RAD* automatically has the ability to communicate and interact with any other expert system constructed using *RAD*. Second, any *existing* expert system constructed using another language can communicate with another through the *RAD* framework, without being rewritten.

> An automotive parts manufacturer encountered a similar problem when it began to automate its factory. It installed small expert systems at each machining operation to monitor the parts produced. When too many parts were produced out-of-tolerance, the machine would be taken down for maintenance, such as to adjust or replace the tool. The length of down-time was dependent on the amount of maintenance required.
>
> Unfortunately, subsequent machining operations could not be informed about the estimated down-time, and when their stock failed to arrive they didn't know what to do. If the down-time was short, they could simply wait, but if the down-time was long, they would have to seek an alternate source of stock, such as inventory in a warehouse. If properly informed, upstream machining operations might be able to produce alternative parts, so that the failed machine would cease to be a bottleneck.

RAD provides the same solution here as for the process-control problem: it enables the expert systems to interact intelligently. The down-time can be managed in the most productive way for the factory.

> A manufacturer of electronic test equipment uses several expert systems to trouble-shoot the oscilloscopes it produces—one expert system for the power supply, one for the waveform generator, one for the event counter, etc. However, when the filter

> capacitor in the power supply fails, it causes a diode in the waveform generator to fail also. None of the expert systems is able to diagnose this fault correctly, because the local problems they are designed to analyze do not match the global behavior they observe and are asked to explain.

Clearly, the trouble-shooting would be more successful if the expert systems could reason jointly and cooperate. The alternative, constructing one large expert system to perform the entire diagnosis, loses the benefits of modularity inherent in the use of multiple expert systems. Furthermore, the individual expert systems can be reused to trouble-shoot other pieces of equipment, such as function generators and logic analyzers.

> When customers of a major computer company call an 800-number to report their computer system has crashed, the field service division has a number of expert systems available to help diagnose the reason for the crash. However, field service first has to decide which of the expert systems to run. They do this by asking the customer several questions about his system and the circumstances of the crash, and by transferring his call, often several times, to the appropriate field service engineer who knows how to run the right expert system. Unfortunately, each engineer who is contacted asks many of the same questions. Worse, multiple faults or problems spanning the specialties of several engineers are often misdiagnosed.

This is a problem in locating the proper expertise to apply and in coordinating its application when several experts are involved. Imagine the following improvement to this scenario: when the call is received, the receptionist asks a few key questions and enters the answers into a database of error reports. An expert system, with expertise in overall system diagnosis *and* in the capabilities of specific field service engineers, analyzes the error report and recommends a particular engineer. The call is transferred to that engineer, along with the error report. By asking additional questions, this engineer refines the error report and either runs his more specific expert system on the customer's problem or invokes the high-level expert system to transfer the problem to a more suitable engineer. As a result, the customer is asked a

minimum number of questions, a complete record is obtained, and the appropriate diagnostic systems are invoked to determine the reason for the crash. The keys to this scenario are having an expert system assisting each field service person and *an ability for these expert systems to communicate and to reason collectively.*

> The claims department of a Northeast insurance company is now using a database of document images, instead of paper files, to manage the paperwork associated with the processing of a claim. Rather than physically carrying files from one desk to another, the workers access images of a client's documents electronically. They would like to automate the workflows of claim processing within the company, so that, depending on its features, a particular claim is automatically routed to the right people. For example, an automobile claim for more than the blue book value of the car needs to be routed to someone who will investigate, and probably lower the claim amount to the blue book value. However, if the car is a 1965 Mustang, the blue book does not reflect its value to collectors, and the claim needs to be routed to someone with expertise in classic cars.

Other researchers have investigated the routing of work through an office environment of distributed workstations [Singh and Forman 1988]. Our research contributes to this effort by supporting the distributed intelligence that is needed to handle the exceptions that inevitably arise. This intelligence is best expressed in rules. *RAD* is specialized for encoding these rules and then applying them at the appropriate times. In addition, an important feature of *RAD* is that it can be distributed among the workstations; the knowledge and reasoning needed for handling exceptions is then available at the workstations where the exceptions occur.

The interconnection of workstations through communication networks has led to new problems in software maintenance. Software maintenance is difficult, and is especially problematic when the software is meant to be executed in a distributed environment. In this case, local modifications might be made to the software that result in global incompatibilities and the failure of the software to perform as expected. Consider the following scenario:

A software package for formatting documents (LaTeX) is distributed to all branch offices in an organization, allowing documents to be exchanged efficiently among the offices in raw form, but then reformatted locally for use within each office. LaTeX is customized at each office to reflect preferences in the appearance of documents and to include such things as a local address. One office decides to include an abstract on the cover page of its documents, using space reserved for a footer (which was not being used anyhow). Unfortunately, a second office decides to include a copyright notice in the footer to its documents. The result? Documents can no longer be exchanged between these two offices.

Now suppose that the formatting software is described by a set of beliefs, and that changes to the software are justified by these beliefs. For example, the justification for adding an abstract is the belief that the footer is not being used. After their respective software modifications, the two offices do not have consistent beliefs about the use of a footer. The distributed truth-maintenance system in *RAD* maintains the consistency of such beliefs and, in this case, would cause one of the offices to invalidate and remove its modification, thereby restoring the compatibility of the software.

The above examples are within the reach of current RAD technology. The following example articulates a future possibility for distributed artificial intelligence: an automated electronic market!

Rosalind Shea, the senior partner of a Los Angeles legal firm, decides she wants to upgrade the personal computers in her office. She realizes that this is a sizable order—at least $300,000—and hence she has significant market leverage. Rosalind starts up her purchasing agent, an expert system for purchasing. The agent downloads the latest product information from the electronic yellow pages of Pacific Telesis, and Rosalind browses through the information. Finally she decides that she wants thirty 586-class PCs, each with a 20-megabyte main memory, a 500-megabyte hard disk, ethernet connections, and a CD ROM drive. She is not brand sensitive, but she wants all products to rate at least three stars in *Info Review*, and she wants a three-year full war-

ranty on everything. Finally, she would like everything delivered and installed within a month, for the lowest price possible.

The agent contacts vendors of systems and components over the next several hours. It negotiates back and forth with sales agents (also expert systems) representing the vendors. These sales agents have up-to-date information about products, prices, and terms of sales. Like the purchasing agent, they exhibit sophisticated negotiating skills.

Finally, the purchasing agent settles on a package, choosing one vendor to supply the PCs with the main memory and the ethernet connections, a second to supply the hard drives, a third to supply the CD ROM drives, and a local firm to do the installation. The purchasing agent has managed to cut deals for extended warranties with the vendors of all components except the hard drives, so it has purchased an extended warranty for the hard drive from yet another vendor, a mainstream insurance company that has begun to write warranties electronically. The next day, Ms. Shea approves the package and finalizes the contract.

An automated electronic market such as this requires a communication network, a DAI communication protocol, negotiation abilities for agents with different goals, and specific domain knowledge about sales, marketing, and finance. We are developing the fundamental technologies, such as our proprietary distributed truth-maintenance system, that permit this vision to become a reality.

2 Background

Knowledge-based systems have become an important part of computing. There are estimates of over 100,000 fielded systems to date. These systems are mostly small, independent, and developed for specific applications using off-the-shelf expert system shells. These shells, including OPS5, Knowledge Craft, M.1, KEE, ART, Goldworks, Nexpert Object, and Proteus, are most suitable for monolithic applications involving the knowledge of a single human expert. But two trends have recently become apparent: 1) systems are being developed for larger and more complicated domains, and 2) there are

attempts to use several small systems in concert when their application domains overlap. Both of these trends argue for knowledge-based systems to be developed in a modular and distributed fashion, where the modules are constructed to interact productively. It is convenient to treat these modules as intelligent agents.

Distributed artificial intelligence is concerned with how a decentralized group of intelligent computational agents should coordinate their activities to achieve their goals. When pursuing common or overlapping goals, they should act cooperatively so as to accomplish more as a group than individually; when pursuing conflicting goals, they should compete intelligently. Interconnected agents can cooperate in solving problems, share expertise, work in parallel on common problems, be developed, implemented, and maintained modularly, be fault tolerant through redundancy, represent multiple viewpoints and the knowledge of multiple human experts, and be reusable. Additional reasons and motivations are presented in [Huhns 1987], [Bond and Gasser 1988], and [Gasser and Huhns 1989]. DAI is the appropriate technology for applications where

- expertise is distributed, as in design;
- information is distributed, as in office automation;
- data are distributed, as in distributed sensing;
- rewards are distributed, as in automated markets;
- decisions are distributed, as in manufacturing control; and
- knowledge bases are developed independently but must be interconnected or reused, as in next-generation knowledge engineering.

DAI has been gathering an increasing amount of attention lately. There have been several successful implementations of DAI systems, notably the Hearsay II system for speech understanding [Erman *et al.* 1980], the DVMT for distributed sensing [Durfee *et al.* 1987], the Pilot's Associate for control of jet fighters [Smith and Broadwell 1988], and the MINDS system for information retrieval [Huhns *et al.* 1987]. These systems were each developed for a specific application. *RAD* is the first general-purpose platform for multiagent system development [Arni *et al.* 1990].

There have been many other attempts to develop systems of cooperating agents. Early attempts, based on the blackboard model, involved agents with independent knowledge bases [Jagannathan *et al.* 1989]. The independence was achieved by restricting agent interactions to modifications of a global data structure—a blackboard—and by minimizing overlap in the agents' knowledge. Later systems allowed richer agent interactions and overlapping knowledge, but the agents were required to have consistent knowledge and to reason monotonically. This led to representational problems: different experts in the same domain often have different perspectives and *conflicting* knowledge, making it difficult to construct a coherent problem-solving system for that domain. One solution was to allow inconsistent knowledge bases; this enabled the conflicting knowledge to be represented, but it did not confront the problem of how to resolve the conflicts.

A few researchers have explored negotiation as a means to mediate among conflicting agents. The systems they developed involved either monotonic reasoners, or nonmonotonic, but memoryless, reasoners, i.e., reasoners that simply discard old solutions and re-solve in the face of conflicts. We agree that negotiation is the correct approach, but that the agents must be able to revise their plans incrementally as they interact. They must be able to communicate directly, with each other and with human agents, and they must be able to assess and maintain the integrity of both the communicated information and their own knowledge. Then they can successfully coordinate their activities and cooperate in solving mutual problems.

3 *RAD* Technology

The Reasoning Architecture group at MCC is addressing the above problems through the development of *RAD*. *RAD* enables a set of knowledge-based systems, constructed quasi-independently, to act as a set of cooperating agents, working together to solve a problem. Developers of distributed reasoning systems can exploit a divide-and-conquer approach to development; they can build smaller, more manageable knowledge-based agents. These smaller agents might represent alternative points of view on a problem; there is no longer a need for global consistency across an entire large system. These smaller agents can also be *reused* in different combinations for solving additional problems as they arise. A further advantage of this approach is

that it enables systems to be physically distributed in the world, just as the problems that they address are. The intelligence needed for such problems can be embedded throughout a system of distributed workstations and made available where appropriate.

RAD is an extension of an MCC-proprietary expert system shell called Proteus, which provides high-performance forward and backward reasoning using Warren Abstract Machine technology, a frame system integrated into a typed unification algorithm, a justification-based truth maintenance system, and a contradiction resolution mechanism. To support cooperative distributed problem solving, *RAD* incorporates a communication channel among the agents, a communication protocol for exchanging goals and solutions on this channel, a representation for agents and their capabilities, a Contract Net mechanism for control of multiple computational agents, and a proprietary distributed truth-maintenance system (DTMS) that enables globally-coherent solutions to be achieved [Bridgeland and Huhns 1990]. *RAD* supports interaction among both human and computational agents. Also, it maintains the integrity of the knowledge bases of its computational agents. These features distinguish *RAD* from all other expert system shells.

The DTMS allows each agent to rely on the results of another's reasoning without having to keep track of the details of that reasoning. However, there is no requirement for two agents to agree completely. The DTMS enforces local consistency within each agent, while enabling negotiation about inconsistencies among agents. When two or more agents disagree about belief in a datum *and when this disagreement is encountered during problem solving*, then negotiation among the agents will ensue to resolve the disagreement. The negotiation procedure involves an exchange of justifications among the agents. The negotiation is necessary to ensure that the global solutions to the problems posed to the agents are coherent.

The *RAD* framework and its corresponding collection of agents execute on a network of computer workstations. The agents operate within this framework asynchronously and, in general, autonomously. *RAD* permits the collection of agents to be dynamic, allowing agents to come and go. It accomplishes this by requiring all agents to register with a *nameserver* that maintains a directory of agent addresses. The agents use the nameserver to locate and then communicate with each other. Communication involves an exchange of messages, specifically, queries and assertions. The agents can be either reactive or ingenuous, i.e., they can either respond to questions and

commands from another agent or initiate dialogs with another agent.

The following is a typical scenario for the use of *RAD*: there is a loosely-coupled network of experts, each with expertise in a particular area, and there is a problem that they must solve that is beyond the capabilities of any one of them. The experts can together solve the problem, but they must cooperate to do so. Furthermore, some of these experts may be knowledge-based systems, i.e., computational agents, some may be humans, and others may be lower-level computational entities, such as databases, software simulators, and neural nets. *RAD* provides specific assistance for the development of computational agents and neural nets, and provides the overall framework within which all of these kinds of agents can operate and interact. Their interaction enables their cooperation and, ultimately, the solution of their problem.

Future versions of *RAD* will include communication protocols that will enable agents constructed in other rule-based languages, such as CYC and OPS5, to interact with *RAD* agents. Future versions also will increase the effectiveness and efficiency of the *RAD* agents by providing a common knowledge base of problem solving methods. This knowledge base will support models for the beliefs, goals, and intentions of each agent. Agents will then have an understanding of each other and the roles that they play in an overall application. Their actions will then be flexible, but robust, and applicable to a wider range of problems.

RAD is only a first step toward cooperative distributed problem solving among a heterogeneous group of agents: it does not guarantee successful and efficient cooperation, but it provides the facilities that make cooperation possible. The next steps will require increased intelligence and capabilities for each agent, resulting in more sophisticated interactions occurring at a higher level. We are providing these capabilities through our research.

4 The Competition

We are not the only ones to recognize the benefits of DAI. Besides research efforts underway at many universities, there are several commercial tools available for developing DAI applications. These tools include G2 from Gensym Corporation, Nexpert Object from Neuron Data, Inc., GBB from GBB Inc., and GEST from Georgia Tech Research Corporation. G2 provides a

communication mechanism, more primitive than the one in *RAD*, for two knowledge-based systems *written in G2* to interact. Nexpert Object can interconnect existing databases (not knowledge bases), if there is an object in Nexpert Object that is programmed to interpret and translate between the schemas of the databases. GBB and GEST are blackboard tools that allow two knowledge sources to communicate *indirectly* through the blackboard. The use of a blackboard for communication is the weakest form of interaction possible among knowledge sources, and is a subset of the interaction facility provided in *RAD*. There are no tools other than *RAD* that provide facilities for distributed reasoning, a Contract Net mechanism for control of distributed experts, or a communication mechanism for the *post facto* integration of expert systems.

5 The Opportunity

The Reasoning Architectures group is uniquely positioned to exploit the potential of distributed artificial intelligence. We are hosting in Austin the next International Workshop on DAI. One of us, Michael Huhns, is an editor of two of the three books available on DAI and a contributor to the third. Our development and analysis of the DTMS is being presented at AAAI-90. And most importantly, the *RAD* platform for the development of general-purpose multiagent systems is incomparable: there simply are no competing tools available. (But this may not be true for long—both DFKI with ESPRIT funding in Europe and NTT in Japan have large efforts underway in DAI.) *RAD* can support a wide range of shareholder applications, and enables our further development of new DAI technologies. To capitalize on our advantages, our current shareholders can collaborate with us in constructing new applications of DAI, and potential shareholders are sought who can apply and market our technology.

References

[Arni *et al.* 1989] Natraj Arni, et al., "Proteus 3: A System Description," MCC Technical Report No. ACT-AI-226-89-Q, Microelectronics and Computer Technology Corporation, Austin, TX, June 1989.

[Arni *et al.* 1990] Natraj Arni, et al., "Overview of RAD: A Hybrid and Distributed Reasoning Tool," MCC Technical Report No. ACT-RA-098-90, Microelectronics and Computer Technology Corporation, Austin, TX, March 1990.

[Bond and Gasser 1988] Alan H. Bond and Les Gasser, *Readings in Distributed Artificial Intelligence*, Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1988.

[Bridgeland and Huhns 1990] David M. Bridgeland and Michael N. Huhns, "Distributed Truth Maintenance," *Proceedings AAAI-90*, Boston, MA, July 1990.

[Durfee *et al.* 1987] Edmund H. Durfee, Victor R. Lesser, and Daniel D. Corkill, "Coherent Cooperation among Communicating Problem Solvers," *IEEE Transactions on Computers*, vol. C-36, 1987, pp. 1275–1291.

[Erman *et al.* 1980] Lee D. Erman, F. Hayes-Roth, Victor R. Lesser, and D. R. Reddy, "The Hearsay-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty," *Computing Surveys*, vol. 12, no. 2, June 1980, pp. 213–253.

[Gasser and Huhns 1989] Les Gasser and Michael N. Huhns, eds., *Distributed Artificial Intelligence, Volume II*, Pitman Publishing, London, 1989.

[Huhns 1987] Michael N. Huhns, ed., *Distributed Artificial Intelligence*, Pitman Publishing, London, 1987.

[Huhns *et al.* 1987] Michael N. Huhns, Uttam Mukhopadhyay, Larry M. Stephens, and Ronald D. Bonnell, "DAI for Document Retrieval: The MINDS Project," in [Huhns 1987], pp. 249–284.

[Jagannathan *et al.* 1989] V. Jagannathan, Rajendra Dodhiawala, and Lawrence S. Baum, eds., *Blackboard Architectures and Applications*, Academic Press, San Diego, CA, 1989.

[Petrie 1989] Charles J. Petrie, "Reason Maintenance in Expert Systems," MCC Technical Report No. ACA-AI-021-89, Microelectronics and Computer Technology Corporation, Austin, TX, February 1989.

[Singh and Forman 1988] Baldev Singh and Ira R. Forman, "Coordination Systems: Manual Interactions," MCC Technical Report No. STP-229-88(P), Microelectronics and Computer Technology Corporation, Austin, TX, September 6, 1988.

[Smith and Broadwell 1988] D. Smith and M. Broadwell, "The Pilot's Associate—An Overview," *SAE Aerotech Conference*, Los Angeles, CA, May 1988.