Enhancing Performance of Cooperating Agents in Real-thm Diagnostic Systems

U. M. Schwuttke and A. G. Quan
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109

818-354-1414
ums@puente.jpl.nasa.gov
alan@puente.jpl.nasa.gov

ABSTRACT

We present a data-driven protocol and a supporting architecture for communication among cooperating intelligent agents in real-time diagnostic systems. The system architecture and the exchange of information among agents are based on simplicity of agents, hierarchical organization of agents, and modular non-overlapping division of the problem domain. These features combine to enable efficient diagnosis of complex system failures in real-time environments with high data volumes and moderate failure rates. Preliminary results of the real-world application of this work to the monitoring and diagnosis of complex systems are discussed in the context of NASA's interplanetary mission operations,

To appear in the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93)

1.0 INTRODUCTION

The interaction and coordination of multiple agents in distributed problem-solving systems has been of interest for a variety of domains whose complexity exceeds the practical capability of monolithic solutions. Examples of domains in which distributed systems have been explored include monitoring [Lesser 1988], planning [Howe 1990], [Bratman 1988], and diagnosis [D'Ambrosio 1990]. These and other approaches [Durfee 1987], [Gasser 1988], [Hayes-Roth 1988] emphasize some form of iterative exchange of partial information among nodes for the purpose of eventual convergence on complete solutions.

Recently, the need for mechanisms of cooperation that are sufficiently robust for real-world applications has been addressed [Jennings 1992] as part of GRATE*, an implementation effort targeted at monitoring. GRATE* makes contributions toward a clearer and more easily implementable interaction of agents during collaborative problem solving. GRATE* addresses a problem domain in which events occur unpredictably and decisions may be based on incomplete or imprecise data. Towards this end, the notion of joint responsibility is proposed as an alternative to the more conventional notion of agents acting in self-interest. The potential for large communication overhead is a possible disadvantage of the GRATE* system, particularly for applications with time critical analysis. The protocol and architecture, described in this paper builds on the, notion of joint responsibility, and uses modular problem decomposition and data-driven reasoning in order to minimize communication between agents. This approach has been applied to the MARVEL system [Schwuttke 1992] for automated monitoring and diagnosis of Voyager spacecraft telemetry and has been shown to achieve robust and coherent behavior for complex, real-time diagnostic agents embedded in a Conventional (algorithmic) monitoring system.

2.0 THE CHARACTERISTICS OF AGENTS IN MARVEL.

Agents are embedded diagnosticians. Diagnostic modules are embedded in more efficient algorithmic code. The algorithmic code performs all functions that do not explicitly require reasoning capability, so that the use of the less efficient reasoning modules is limited to those functions for which it is essential.

Diagnosis is data-driven. Forward-chaining demons are used to represent domain knowledge. Reasoning is activated by the appearance of data that requires diagnosis. The initial determination that diagnosis is required is made by algorithmic monitoring code, which detects potential anomalies algorithmically and passes the. anomalous data to an appropriate diagnostician, In the absence of anomalous data within its domain, an agent is idle.

The domain of individual agents is constrained. An agent is responsible for a small, clearly partitionable domain of expertise. Partitioning is governed by the natural decomposition of the system being diagnosed. This helps overcome disadvantages associated with rule-based systems for which, typically, implementation can be intractable, execution is non-deterministic and relatively slow, and verification can be difficult. Small, modular knowledge-bases enable developers to handle more easily definable subproblems. Smaller knowledge bases execute more efficiently, because less time is spent in search. Finally, smaller knowledge-bases are easier to verify.

The domain of individual agents is non-overlapping. A particular domain of expertise is assigned only to one agent to avoid redundant reasoning.

Agents carry individual responsibility for problems entirely within their domain. Agents have sufficient knowledge to be fully accountable for diagnoses within their areas and have no knowledge of other domains. This requires that accountability for locally detectable failures must be local.

Failure domains may not map directly to agent domains. Diagnosis requires more than one agent when the symptoms manifest themselves in more than domain.

Mets-knowlmlgc enables agents to instigate cooperation for diagnoses beyond their domain. Agents have meta-knowledge to identify symptoms of failures that could possibly extend beyond their domain. Meta-knowledge is contained in a set of rules in each knowledge base, and is associated with the occurrence of events for which the cooperat ion of ot her agents might be requi red,

Agents report all problems that extend beyond their domain. Meta-knowledge enables an agent to determine which symptoms from its domain may portend problems beyond its domain. The mcta-knowledge also includes the specific agent(s) to which the information should be forwarded,

A hierarchy of agents provides coordination. An agent forwards all known information pertaining to failures beyond its domain to another agent at the next higher level in the hierarchy. The underlying assumption on forwarded messages is "better safe than sorry"; it is up to the agent receiving the information to determine whether a fault requiring a diagnostic message and an alarm has occurred or whether the anomalous data has some other explanation. This agent may also receive messages from other lower level agents. Agents at the higher level are also implemented according to the principles outlined here; thus reasoning at the higher level of the hierarchy is also data driven. The agents at the higher level arc activated by messages from lower level agents, just as the lowest level agents were activated by messages of symptoms detected by algorithmic code. Communication is one way, in most cases, and messages are directed with mcta-knowledge to the relevant agent(s) in order 10 complete the final analysis of the anomalous data and provide diagnosis of any associated failures.

Agents share responsibility for diagnosis of problems that overlap domains. J oint responsibilit y exi sts in that the lower-level agents are responsible for report i ng appropriate symptoms upwards in the hierarchy and the higher-level agent(s) are responsible for correct l y determining whether failures have occurred and providing appropriate diagnoses. This differs from the "self interest" model of communication [Durfee 1988] and is similar to the joint responsibilit y model [Jennings 1992,] in which agents must temper their self-in[c.rest with consideration to a group. These models have parallels in social organizations, with the first being more representative of an unstructured society and the second paralleling the actions of individuals who arc dedicated (perhaps for reasons of self-intcmt) to fulfilling a successful role in a structured organization such as a corporation. in the latter case, independent agents work together with appropri atc (and hierarchical) division of responsibility towards fulfilling a common goal. Real-wor]d applications can bc sufficiently complex that only this second type of organization may enable timely, robust, and coherent behavior.

3.0 COOPERATING AGENTS IN A DISTRIBUTED ARCHITECTURE

The distributed architecture shown in Figure 1 is based on a central message routing scheme that is not shown in the figure. The various agents arc allocated among a configuration of UNIX workstations. The data management module receives data from a source (in the case of our current application, the data is spacecraft telemetry received from JPL's ground data system) and allocates it to the appropriate subsystem monitor based on identification of data type. (Our system is partitioned according to the partitioning of the spacecraft itself, with onc subsystem monitor for every spacecraft subsystem covered by MARVEL. Spacecraft subsystems include command and data, attitude and articulation control,propulsion, telecommunications, thermal, and power. Such a partitioning reflects the natural partitioning of the system being monitored, which is desirable for rc.al-time diagnostic architectures.) Each of the subsystem monitors provides algorithmic functions such as validation of telemetry, detection of anomalies, trend analysis and automatic reporting. These functions, while not in themselves of interest in AI or computer science research arc vital components of a real-world diagnostic system. They arc implemented here in conventional C code for performance reasons. In addition, each subsystem process provides diagnosis of failures based on anomalous data, and recommendation of corrective actions. The latter two functions are provided by knowledge-based agents that are embedded within each of the individual subsystem monitors. The remaining modules include the graphical user interface and display processes for each of the subsystem monitors, and the system-level diagnostic agent for handling failures that manifest themselves across multiple subsystems (and therefore cannot be completely analyzed by any one subsystem alone),

The interconnectivity of the distributed system is provided by a TCP/IP central router program and a set of messaging routines that are linked into the subsystem processes. All processes are connected to the central router by UNIX sockets. The

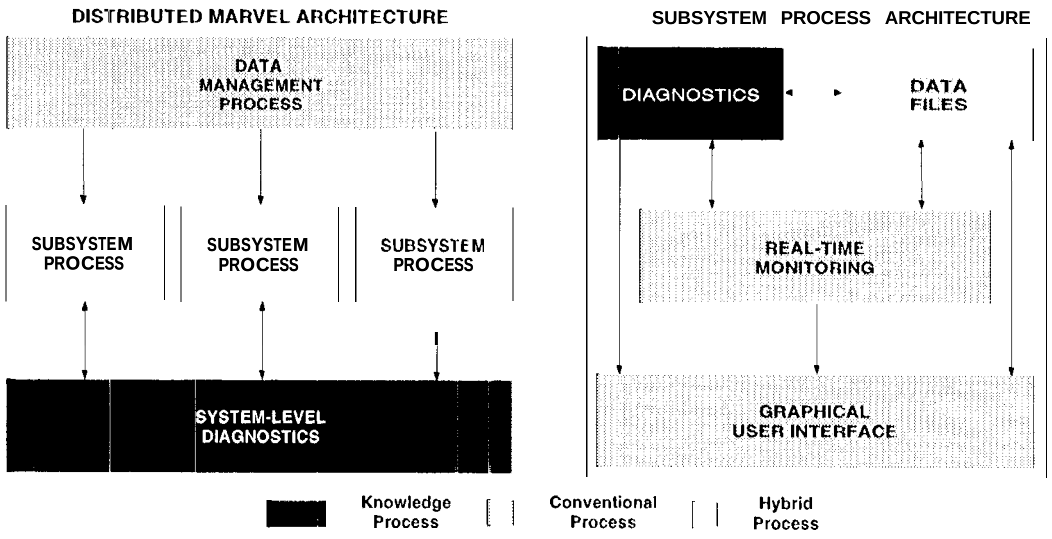

Figure 1. The distributed architecture on the left can currently be configured to run on one to four UNIX workstations, with the most common operational configuration involving two workstations (for compatibility with analyst responsibilities). The hybrid subsystem processes on the left are composed of conventional and knowledge processes, as shown in the figure on the right. Knowledge, processes are used only when a reasoning capability is explicitly required.

basic measurement of performance for the distributed system is the speed-up S(N), defined as the sequential execution time divided by the execution time on N processors. It has not been possible to measure a unique value S(N) because of the heterogeneous nature of the agents. This heterogeneity arises because the processing loads of the six basic agents (the data management module, the four subsystem modules, and the system-level diagnostician) are not identical. Our alternative to this measurement is the lowest speedup of the individual modules. With a four-processor implementation, a speedup of 3.6, or 0.9N was observed. This result indicates that MARVEL is a highly efficient distributed system. Two factors contribute to these results. The first is the modularity inherent in the application (and in many other complex applications). The second factor is a distributed design that effectively minimizes the need for interprocess communication.

4.0 APPLICATION TO MONITORING AND DIAGNOSIS OF A REAL-TIME SYSTEM

in this section we provide an example of cooperation between multiple hierarchical agents in an actual real-time. system, as shown in Figure 2. This figure depicts four knowledge-based agents (shown in black and), each of which has expertise in a different domain of the engineering subsystems of the Voyager spacecraft. Two of these agents are responsible for diagnosing anomalies in specific spacecraft subsystems: Computer Command Subsystem (CCS), and Attitude and Articulation Control Subsystem (AACS). A third agent, the System Level knowledge agent, is at a higher level in the agent hierarchy and is responsible for diagnosing anomalies that cannot be fully analyzed in any single subsystem domain. A fourth agent provides data quality information to the other agents based on data from the Telecommunications Subsystem (Telecom), so that when data quality is unreliable, alarms resulting from the diagnostic activity of the other agents can be suppressed.

All diagnostic communication bet ween agents is coordinated by the System Level agent. All data quality messages are handled by an algorithmic software module that also communicates with the graphical user interface. There is no direct com-munication between subsystem agents. As explained in the previous section, each agent has an algorithmic telemetry monitor process associated with it.

'J'he Telecom agent differs from the other two subsystem agents in that its purpose is to determine the quality of the telemetry data being received from the spacecraft, rather than diagnose subsystem anomalies that occur on the spacecraft. The data quality level is passed to the Data Quality Management process, which in turn sends this information to the various telemetry monitor processes. If the data qual-ity is determined to be very poor, the reporting of anomalies is partially suppressed

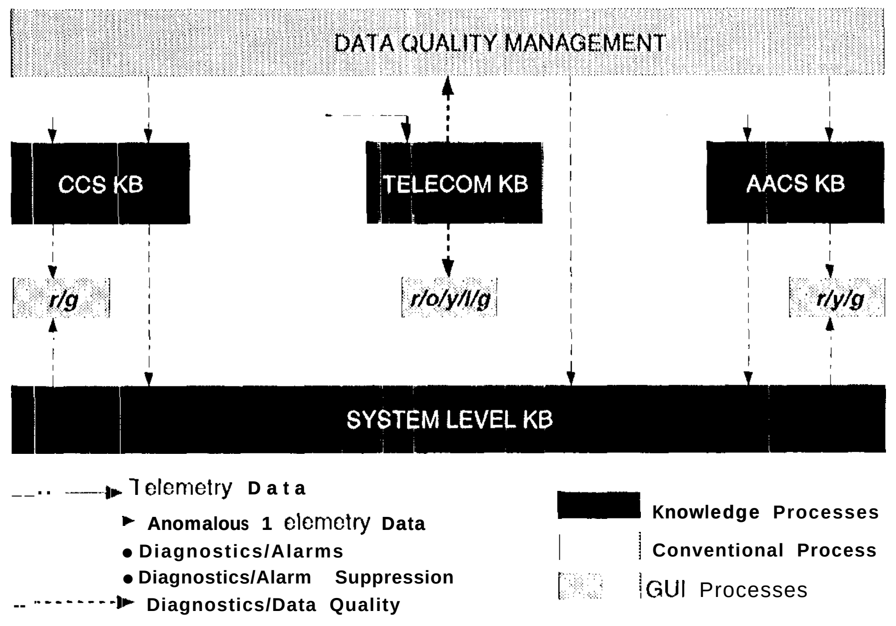

Figure 2. Communication among four intelligent agents in a real-titnc monitoring and diagnosis system. '1 'he system also cent sins algorithmic and graphcial user interface (Gill) processes (which provide r-c(i/green or red/orange/yel low/l in]c/green alarm notification as determined by the knowledge processes).

(as explained below) since the telemetry that led to the anomaly diagnosis is probably not reliable.

Our example begins with the arrival of telemetry from the CCS subsystem which indicates an abnormally high computer event count. The CCS event count is incremented each time that an event is initiated by the spacecraft computer, One type of event is fault-protection, which attempts to automatically correct a fault that has been detected or protect against harmful consequences of such a fault. 'Thus, an abnormally high event count could indicate entry into fault protection sequences. The CCS telemetry monitor compares the telemetry event count to the predicted event count and finds that they are not equal. Since this is an anomaly, the monitor passes the event count to the CCS knowledge agent for further analysis. The CCS agent finds that the telemetry event count exceeds the expected event count by ***56.*** A difference of 56 in the event count may indicate that a "heartbeat failure" has occurred on the spacecraft. The CCS "heartbeat" is a signal (called a "power code") sent every ten seconds from the AACS subsystem to the CCS subsystem on board the spacecraft. If the signal is received at the end of the expected time interval, the CCS spacecraft subsystem assumes that the AACS subsystem is functioning, normally. If on the other hand the CCS fails to receive the heartbeat signal twice in a single hour ("heartbeat failure"), the CCS assumes that the AACS has failed in some wa:y, and it issues a series of commands to switch to redundant back-up components in the AACS, in an effort to correct the problem.

However, a difference of 56 between the actual event count and expected event count is not enough evidence in itself to conclude that a heartbeat failure has occurred,, "There may have been other events not related to the heartbeat that happened to increase the event count by 56. Furthermore, there is no way to confirm the occurrence of a heartbeat failure from any of the CCS telemetry. The CCS agent knows that a complete diagnosis of the. problem is beyond its domain and so in this case it passes on the heartbeat failure evidence to the higher level System Level agent for further aria] ysis and possible confirmation by other agents,

Like the subsystem agents, the System Level agent is data-driven. Upon receipt of the message from the CCS agent, the System 1 .evel agent asserts a fact into its local knowledge base indicating that a possible heartbeat failure. was detected by the CCS. This fact matches half of the antecedent of a data-driven rule in the System Level KB, but this is not sufficient to fire the rule. The heartbeat failure anomaly can be confirmed by diagnostic rules in the AACS agent, but at this point no other messages have been received at the S ystem Level, so not hi ng is reported to the user. The System Level returns control to the telemetry monitor process.

The next telemetry to arrive is a status word from the AACS subsystem. The AACS telemetry monitor compares the telemetry status word to the expected status word value and finds that they arc not equal. It then passes the status word to the AACS knowledge agent. "Jim agent analyzes the status word bits and determines that several A ACS components have been swapped off-] i nc and their redundant back-up units have been activate.d. Based on ibis pattern of events it concludes that a possible heartbeat failure has occurred. But this information by itself is not enough to be certain that a heartbeat failure has actually taken place. The AACS agent knows that a complete diagnosis of the problem is beyond its domain and will require information from onc or more other agents. So it sends a message to the System Level agent notifying it of the possible heartbeat failure.

When the System Level agent receives the heartbeat failure message from the AACS agent, it asserts a fact into its local knowledge base indicating that a possible heartbeat failure was detected by the AACS. At this point the previously asserted fact from the CCS agent combined with the. ncw fact from the AACS agent match the complete antecedent of a data-driven rule in the System Level knowledge base, and the rule fires. The consequent of the rule causes an anomaly message to be sent to the System Level monitor for display to the user. However, before this message is displayed, the monitor checks the current data quality as determined by the Telecom agent. If the data quality is in the range of marginal to error free, the monitor displays both the anomaly message and the data quality level in the output window, sounds an audible alarm, and turns the output window color to red. If on the other hand the data quality is poor, meaning excessively noisy, then the telemetry that led to the anomaly diagnosis was probably corrupted during transmission, and the resulting conclusion is probably incorrect. in the latter case the anomaly message is still output to the user, but the alarm is not sounded and the output window color is not changed. In addition, the data quality is displayed along with the anomaly message so that the user is informed that the anomaly diagnosis was probably due to data corruption.

This example illustrates the cooperation and communication between four different knowledge agents in a hierarchical organization. Information from all the agents is required in order to provide a complete diagnosis of the anomaly condition. These agents illustrate the principles outlined in section 2. Each agent is a data-driven diagnostician responsible for a constrained non-overlapping domain. Each of the subsystem agents has meta-knowledge that allows it to identify symptoms that may indicate problems beyond its domain, and it reports these symptoms to a higher level agent for cooperative multi-agent analysis,

5. PRELIMINARY RESULTS

The distributed architecture described in this paper has been applied to the MARVEL system for real-time spacecraft diagnostics. It has been recently developed, as a follow-on to a uniprocessor version that could accommodate only one of the three subsystem agents cm any one installation. Preliminary tests have demonstrated that the distributed system can process up to 1500 telemetry values per minute. Individual subsystem agents can successfully diagnose anywhere from 2 to 220 anomalies per minute, depending on the complexity of reasoning that is required. The System Level agent can process up to 300 anomalies per minute. For anomalies that require analysis from multiple agents (e.g., heartbeat failure), the maximum number of anomal ics that can be processed in a given period of' t i me is equal to the speed of the slowest agent involved in the analysis (assuming all agents execute concurrently), plus approximately 1/5 second for System 1 .evel inferencing. This is well within acceptable limits for real life mission operations demands,

6. CONC1 .USIONS

The MARV10. distributed architect ure demonstrates the successful implementation of multiple cooperating agents in a complex real-time diagnostic system. We have designed an architect ure that facilitates concurrent and cooperat ive processing by multiple agents in a hierarchical organization. These agents adhere to the concepts of clata-driven embedded diagnosis, const rained but complete non-overlapping domains, meta-knowledge of global consequences of anomalous data, hierarchical reporting of problems that extend beyond an agent's domain, and shared responsibilit y for problems that over] ap domains.

The MARVEL architecture is simple and wc]] suited for real-time telemetry analysis. Conventional processing is used wherever possible in order to facilitate performance. The kJlow]cdgc-based agents are embedded within the algorithmic code, and arc invoked only when necessary for diagnostic reasoning. Distribution of telemetry monitoring and agent processes across workstations provides significant improvement in performance. These qualities allow for efficient real-time. diagnosis of anomalies occurring in a complex system.

7. REFERENCES

[Bratman 1988] M. E. Bratman, D. J. Israel, M. L. Pollack. Plans and Resource-Bounded Practical Reasoning. Computer Intelligence 4,349-355.1988.

[D'Ambrosio 1990] B. D'Ambrosio. Constrained Rational Agency. IEEE Transactions on Systems, Man, and Cybernetics. 1990.

[Durfee 1987] E. H. Durfee, V. R. Lesser, 11.11. Corkill. Coherent Cooperation Among Communicating Problem Solvers. IEEE Transact ions on Computers, C-36:1275-1291, 1987

[Durfee 1988] Cooperation through Communication in a Distributed Problem Solving Network, in Distributed Artificial Intelligence, Vol. 2. Pitman Publishing, 1988.

[Gasser 1987] L. Gasser, C. Braganza, and N. Herman. MACE: A Flexible Test-bed for Distributed Al Research. Distributed Artificial Intelligence, M. N. Huhns, Ed., Pitman/Morgan Kaufman, 1987.

[Hayes-Roth 1988] F. A. Hayes-Roth, L. Erman, S. Fouse, J. Lark and J. Davidson. ABE: A Cooperative Operat ing S ystem And Development Environment. Al Tools & Techniques, Ed M, Richer, Ablex. 1988.

[Howe 1990] A. E. Howe, D. M. Hart, and P. R. Cohen. Addressing Real-time. Constraints in the Design of Autonomous Agents. COINS Technical Report 90-06. Universit y of Massachusetts at Amherst. 1990.

[Jennings 1992] N. R. Jennings and E. H. Mamdani. Using Joint Responsibility to Coordinate Collaborative Problem Solving in Dynamic Environments. in Proceed - ings of the Tenth National Conference on Artificial Intelligence, San Jose, California. 269-275.

[1 ,esser 1988] V. R. Lesser and D. D. Corkill. The Distributed Vehicle Monitoring Testbed: A Tool for Investigating Distributed Problem Solving Networks. Al Magazine 4(3):1 5-33.

|Schwuttke 1992] U. M, Schwuttke, A. G. Quan, R. Angelino, C. L. Childs, J. R. Veregge, R. Yeung, and M. B. Rivera. "MARVEL: A Distributed Real-time Monitoring and Analysis Application" Innovative Applications of Artificial Intelligence., Vol. 4., MIT Pros, 1992.