Performance Results of Cooperating Expert Systems in a Distributed Real-time Monitoring System

U. M. Schwuttke, J. R. Veregge, 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

ABSTRACT

A distributed monitoring and diagnosis system has been developed and successfully applied to real-time monitoring of interplanetary spacecraft at NASA's Jet Propulsion Laboratory. This system uses a combination of conventional processing and artificial intelligence. Four knowledge-based diagnosis modules are embedded within a monitoring system that detects on-board spacecraft anomalies. Each of the four knowledge-based systems is unique with respect to the others in its implementation or use, resulting in an interesting set of performance results that have been used as guidelines for the design of our next generation real-time diagnostic systems. Details of the distributed architecture, and the general characteristics of the embedded diagnostic systems are also provided.

1.0 INTRODUCTION

There are numerous definitions for real-time systems, the most stringent of which involve guaranteeing correct system response within a domain-dependent or situationally-defined period of time. For applications such as diagnosis, in which the time required to produce a solution can be non-deterministic, this requirement poses a unique set of challenges in dynamic modification of solution strategy that conforms with maximum possible latencies. However, another definition of real-time is relevant in the case of monitoring systems where failure to supply a response in the proper (and often infinitesimal) amount of time allowed does *not* make the solution less useful (or, in the extreme example of a monitoring system responsible for detecting and deflecting enemy missiles, completely irrelevant). This more casual definition involves responding to data at the same rate at which it is produced, and is more appropriate for monitoring applications with softer real-time constraints, such as interplanetary exploration, which results in massive quantities of data transmitted at the speed of light for a number of hours before they even reach the monitoring system.

The latter definition of real-time has been applied to the MARVEL system [Schwuttke et al. 1992] for automated monitoring and diagnosis of spacecraft telemetry. An early version of this system has been in continuous operational use since it was first deployed in 1989 for the Voyager encounter with Neptune. This system remained under incremental development until 1991 and has been under routine maintenance in operations since then, while continuing to serve as an AI testbed in the laboratory. A second generation Galileo application has been on-line for only one year and is still under active development. The second generation system builds on experience gained with the earlier embedded diagnosis systems to achieve an order of magnitude increase in processing capability.

The system architecture has been designed to facilitate concurrent and cooperative processing by multiple diagnostic expert systems in a hierarchical organization. The diagnostic modules adhere to concepts of data-driven reasoning, constrained but complete nonoverlapping domains, meta-knowledge of global consequences of anomalous data, hierarchical reporting of problems that extend beyond a single domain, and shared responsibility for problems that overlap domains. The system enables efficient diagnosis of complex system failures in real-time environments with high data volumes and moderate failure rates, as indicated by extensive performance measurements.

2.0 COOPERATING DIAGNOSIS SYSTEMS IN A DISTRIBUTED ARCHITECTURE

The need for robust mechanisms of cooperation among real-time diagnostic modules has been an important driver of the system architecture. The notion of joint responsibility [Jennings and Mamdani, 1992] as an alternative to the more conventional notion of agents acting in self-interest [Durfee 1988, Cohen 1990] has been amended with modular problem decomposition and data-driven reasoning in order to minimize the need for communication between agents. The various modules in the distributed architecture of Figure 1 are 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 te-

The research described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology under a contract with the National Aeronautics and Space Administration. The authors wish to acknowledge strong support from JPL's Voyager and Galileo Projects, Multimission Operations Support Office and Director's Discretionary Fund.

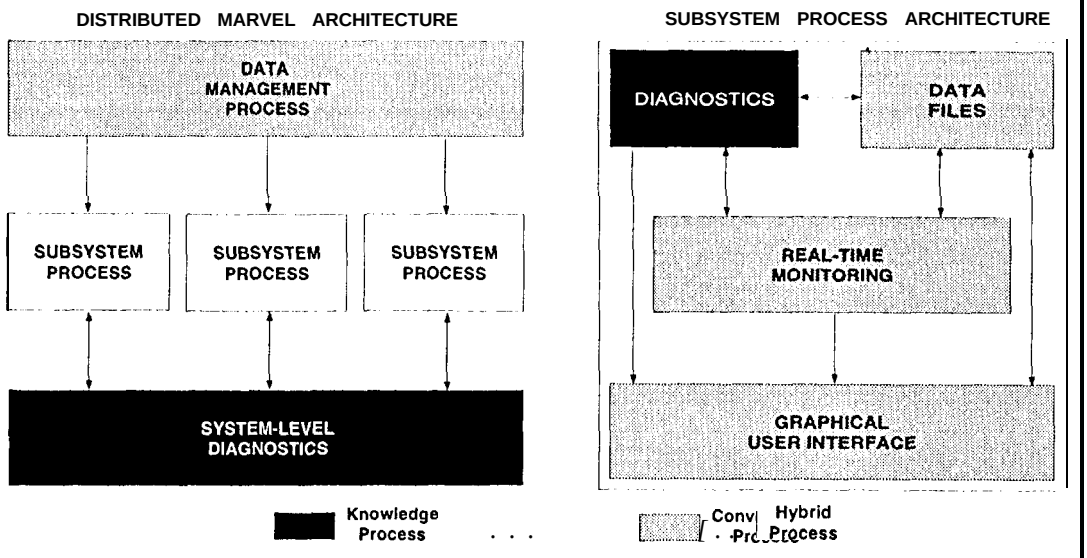

Figure 1, The distributed architecture on the left can currently be configured to run on one to four UNIX workstations. The hybrid subsystem processes on the left are composed of conventional and knowledge processes, as shown in the figure on the right. Knowledge processes are used only when a reasoning capability is explicitly required,

lemetry 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 structure of the spacecraft, with one subsystem monitor for every spacecraft subsystem monitored by MARVEL, including command, flight data, attitude and articulation control, and telecommunications; propulsion, thermal, and power have not been addressed.)

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, are vital components of a real-world diagnostic system. In addition, each subsystem process can provide diagnosis of failures based on anomalous data and recommend corrective actions. The latter two functions are provided by knowledge-based modules 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). Detailed reasoning examples that illustrate cooperation among diagnosis modules are presented elsewhere [Schwuttke and Quan 1993].

3.0 EXPERT SYSTEM CHARACTERISTICS

Rule-based diagnostic modules are embedded in 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.

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, a diagnostic system is idle.

Each diagnostic system 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 nondeterministic 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.

Each diagnostician has sufficient knowledge to be fully accountable for diagnoses within its area and has no knowledge of other domains. This requires that accountability for locally detectable failures must be local. However, the participation of more than one diagnostic system is required when symptoms manifest themselves in more than one domain. Each diagnostic system has the necessary meta-knowledge to identify symptoms of failures that could possibly extend beyond its domain. Metaknowledge is contained in a set of rules in each knowledge-base, and is associated with the occurrence of events whose analysis may require the cooperation of other agents

An expert forwards all known information pertaining to failures beyond its domain to another agent at the next higher level in the hierarchy. The underlying approach on forward-

ed messages is conservative; 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. When necessary, metaknowledge is used to direct messages to the relevant agent(s) in order to complete the final analysis of the anomalous data and provide diagnosis of any associated failures.

4.0 EXPERIMENTAL RESULTS

The distributed architecture described in this paper has been applied to two generations of real-time monitoring systems, The Galileo system, currently under development, does not yet include on-line modules for diagnosis. The Voyager system, completed in1991, contains four diagnostic expert systems (developed using a commercial shell) in a two-level hierarchy.

Conventional monitoring modules for four of the spacecraft subsystems were completed: the flight data subsystem, the computer command subsystem, the attitude and articulation control subsystem, and the telecom subsystem. Three of our expert systems are embedded in conventional modules that provide data access/manipulation and monitoring in addition to providing graphical user interfaces and other subsystem specific automation. The system-level diagnostician is not embedded within another module.

The computer command subsystem (CCS) expert contains on the order of 150 rules, focuses on a relatively broad domain analysis, and is invoked very frequently (for almost every parameter). The attitude and articulation control subsystem (AACS) expert contains approximately 100 rules, and focuses on a more narrow domain of analysis. It is invoked infrequently. The telecom expert system contains on the order of twenty-five rules and is invoked continuously (for every parameter). The flight data subsystem (FDS) module does not contain an expert system.

Experimental evaluation on a network of workstations (Sun Microsystems Spare LXS running Solaris 2.2) involved a series of tests to determine the maximum number of data parameters that could be processed per module per second (a subsystem module includes both the conventional and knowledge-based components as shown in Figure 1). The primary purpose of this evaluation was to learn about the performance of the expert systems and apply our insights to future development on the Galileo application. This evaluation was not motivated by a need to improve the performance of the Voyager system, as current data rates are considerably slower than during the planetary encounters and are easily handled by the existing software configuration.

The results are shown in Figure 2. The baseline performance was below expectation, with FDS, CCS, AACS and Telecom processing 26,3,24, and 428 parameters per second respective] y,or481 total parameters per second processed by the entire system. Performance profiling revealed that file 1/0 and the graphical user interfaces (GUIs) rather than the diagnostic modules were primary performance bottlenecks.

With regard to these bottlenecks, the four modules can be categorized as follows. FDS and CDS have moderately complex GUIs, and perform significant file 1/0. AACS has the most complex GUI and performs very little file 1/0, because the input files read by this subsystem are sufficiently small that they are read entirely into memory upon system initialization. Telecom has a simple GUI and performs no file 1/0.

Optimizing file 1/0 where possible improved performance to 53, 16, 81, and 428 parameters per second. (This is the only improvement discussed in this section that was carried forward to the operational system.) Simplifying the graphical user interface by eliminating real-time scrolling windows (known to be computationally inefficient in MOTIF user interfaces; considered desirable by end-users and thus included in the FDS, CCS, and AACS modules of the operational system) further improved performance to 53, 35, 172, and 428 parameters per second. Eliminating the graphical user interface entirely resulted in further performance increases to 67, 35, 646, and 570 parameters per second. Finally, eliminating the expert systems yielded performance of 67, 273, 668, and 570 parameters per second.

These results made it possible to gain a number of new insights with regard to our system. The biggest surprise was the high performance of the telecom module. The combination of the small knowledge base and the simple user interface enables processing of 428 parameters per second. Elimination of both the GUI and the expert system only results in a further performance improvement on the order of 25 percent, indicating that no substantial penalty is associated with the significant enhancement to functionality provided by these two components of the module. The next generation system will benefit from this result, in that frequently performed analysis that requires the use of an expert system will be implemented with a number of small, cooperating modules rather than one larger module. This in itself is not unexpected; it is the magnitude of the benefit that was surprising. Further performance improvement could likely be gained with a more efficient expert system shell. This will be investigated although we do not currently expect more than an additional order of magnitude improvement.

The AACS expert system is larger by a factor of four, and slower, in the worst case, by over two orders of magnitude. This can be explained by a significantly larger search space and greater depth in each search. Performance could likely be improved with a faster reasoning shell and by modularization of the knowledge base. However, the diagnostic component of this module is invoked sufficiently rarely (often less than once per hour) that this is not an important bottleneck. In the case of this type of module, it is preferable to simplify the GUI, which continues to impose considerable resource overhead.

The CCS expert system is large and is invoked regularly as part of ongoing trend analysis in that subsystem module. Elimination of the expert system results in an additional order of magnitude increase in performance, providing further indication that a large knowledge base is inappropriate for frequently invoked real-time diagnosis. The CCS knowledge base is characterized by breadth rather than depth. As a result, it would both beneficial (and straightforward) to reduce it to three or more component modules without imposing significant overhead from resulting interprocess communication. (If this were implemented, the CCS module would still be 1/0 bound, as it reads from a number of very large files.)

As a result of these insights, the Galileo implementation takes a more efficient approach to file 1/0. It also tends to be more efficient in its graphical user interface, in that it does

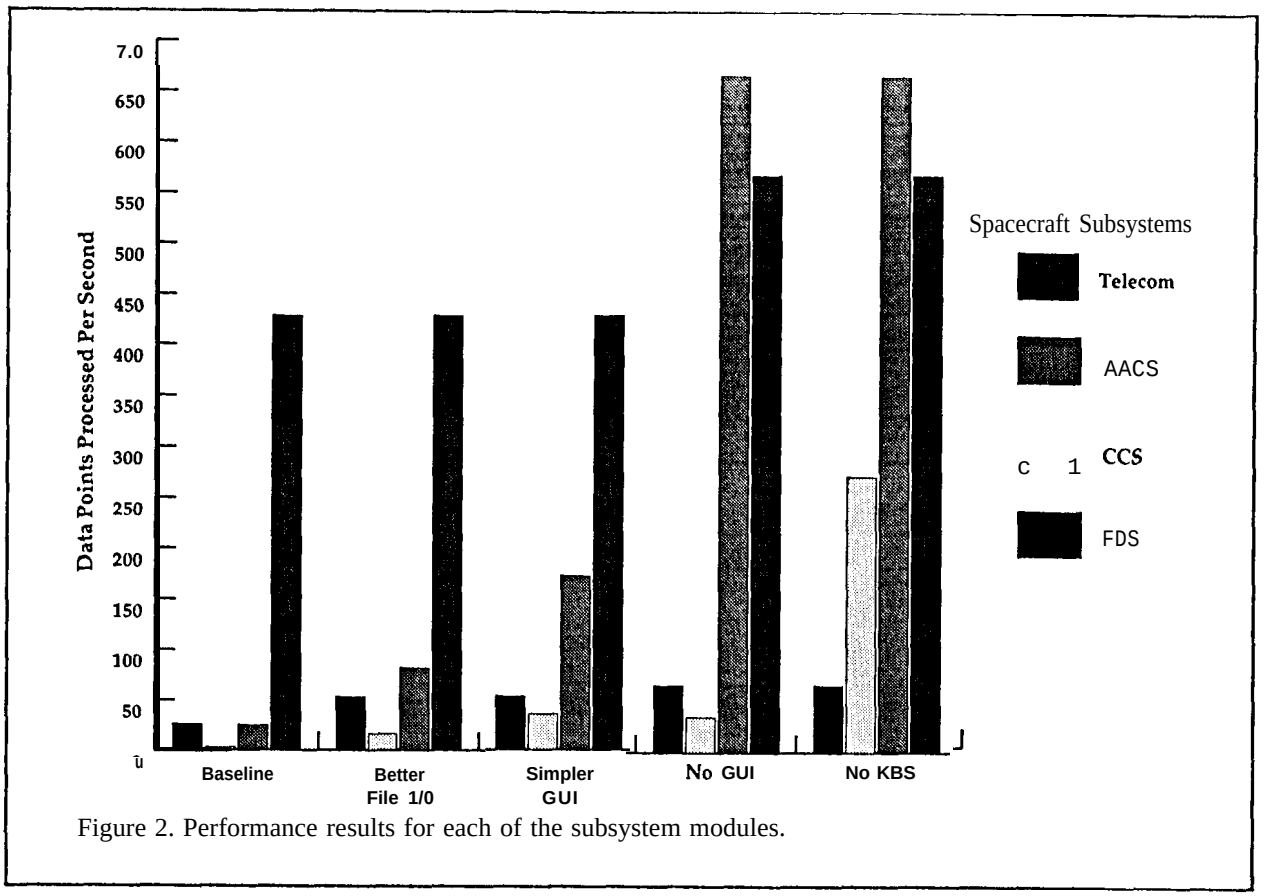

Figure 2. Performance results for each of the subsystem modules.

not include some of the higher-overhead user interface widgets. Such changes impact functionality, requiring a certain amount of negotiation with end-users (who are typically willing to compromise in favor of performance). In addition, the Galileo system makes greater use of the distributed architecture with more than one module per subsystem, and more than one diagnostic component per module.

5.0 CONCLUSION

The MARVEL distributed architecture demonstrates the successful implementation of multiple cooperating agents in a complex real-time diagnostic system, We have designed an architecture that facilitates concurrent and cooperative processing by multiple agents in a hierarchical organization. These agents adhere to the concepts of data-driven embedded diagnosis, constrained but complete nonoverlapping domains, metaknowledge of global consequences of anomalous data, hierarchical reporting of problems that extend beyond an agent's domain, and shared responsibility for problems that overlap domains.

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

Maximum modularization of frequently invoked reasoning modules will enable significant performance improvements in the next generation system.

6.0 REFERENCES

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

Durfee, E. H. 1988. "Cooperation through Communication in a Distributed Problem Solving Network. " In *Distributed Artificial Intelligence,* Vol. 2. Pitman Publishing, 1988.

Jennings, N. R.; and Mamdani, E. H. 1992. "Using Joint Responsibility to Coordinate Collaborative Problem Solving in Dynamic Environments." In *Proceedings of the Tenth National Conference on Artificial Intelligence,* San Jose, California, 269-275.

Schwuttke, U. M.; Quan, A. G.; Angelino, R.; Childs, C. L.; Veregge, J. R.; Yeung, R.; and Rivera, M. B. 1992. "MARVEL: A Distributed Real-time Monitoring and Analysis Application." In *Innovative Applications of Artificial intelligence 4,* MIT Press.

Schwuttke, U. M.; and Quan, A. G. 1993. "Enhancing Performance of Cooperating Agents in Real-time Diagnostic systems." In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence,* Chambery, France. 332-337.