N95-19647
34082
P-10

The Meteorological Monitoring System for the Kennedy Space Center/Cape Canaveral Air Station

Allan V. Dianic
(alland@fisher.css.gov)
ENSCO, Inc. Applied Research and Systems Division
Melbourne, Florida

Abstract

The Kennedy Space Center (KSC) and Cape Canaveral Air Station (CCAS) are involved in many weather-sensitive operations. Manned and unmanned vehicle launches, which occur several times each year, are obvious examples of operations whose success and safety are dependent upon favorable meteorological conditions. Other operations involving NASA, Air Force and contractor personnel -- including daily operations to maintain facilities, refurbish launch structures, prepare vehicles for launch and handle hazardous materials -- are less publicized but are no less weather-sensitive.

The Meteorological Monitoring System (MMS) is a computer network which acquires, processes, disseminates and monitors near real-time and forecast meteorological information to assist operational personnel and weather forecasters with the task of minimizing the risk to personnel, materials and the surrounding population. CLIPS has been integrated into the MMS to provide quality control analysis and data monitoring. This paper describes aspects of the MMS relevant to CLIPS including requirements, actual implementation details and results of performance testing.

1.0 Introduction

The Meteorological Monitoring System (MMS) is designed to operate in a networked environment for the support of meteorological and operational personnel (see Figure A.1 on page 7). The MMS consists of a Preprocessor (PPR) and one or more Monitoring and Display Station (MDS). The MMS Preprocessor acquires data from sensors located in and around the KSC/CCAS area, and places them into one of the MMS data classes. These classes categorize data into common types for uniform processing (see Figure A.2 on page 7). The data are then subjected to a quality-control check by an embedded CLIPS implementation and disseminated to the network of MMS Monitoring and Display Stations (MDS). The MDS, using a second MMS implementation of CLIPS, provides the end user with a tool to monitor weather and generate warnings and alerts when weather conditions violate published criteria. The system maintains a database of operations and associated weather criteria for the user to select. Once activated, the meteorological constraints for an operation are transformed into a series of CLIPS rules which, along with a current stream of near real-time and forecast data, are used to trigger alarms notifying the user of a potentially hazardous weather condition. Several user-defined functions have been added to CLIPS, giving it the ability to access MMS resources and alarms directly.

The use of CLIPS for this effort was in large part a research effort: we were interested in making use of a tool that could add a great deal of flexibility and power to the QC and monitoring tasks. However, we were concerned about its implementation and the extent to which performance would have to be sacrificed in exchange for that flexibility. This paper describes aspects of the MMS relevant to CLIPS including requirements, actual implementation details and results of performance testing. The conclusion will discuss the overall success of the effort and future expan-

sion possibilities.

The MMS was tested on a DECstation 5000/125 uses a MIPS R3000A processor running at 25 MHz with 32 megabytes of memory. This was a capable mid-level workstation in 1991, but is slower than comparably priced systems currently available. The system was developed under Ultrix using C, CLIPS version 5.0, X windows and SL-GMS, a tool for dynamic graphical screen management.

2.0 Quality Control

Quality control is a function that attempts to identify and tag erroneous weather measurements and observations. Missing or invalid data may be introduced by hardware or software failures and must be identified so that they do not negatively affect the operation of the MMS.

The MMS Preprocessor is responsible for acquisition, quality control analysis and dissemination of meteorological data. CLIPS has been embedded in the quality control (Data_QC) module, and analyzes each data point processed by the system. The Preprocessor currently ingests a minimum of 1000 weather measurements each minute; currently these data are contained in approximately 11 kilobytes of ASCII text.

2.1 Implementation

Data_QC runs as a background process monitored by the system health facility. During initialization, the data templates representing MMS data classes are loaded into the shell. Data_QC then waits for a message from the data ingest routine indicating that a set of data is ready to be analyzed. The process executes a Unix 'fork' after identifying the data class of the incoming data. While the parent process waits for the child to complete, the child loads raw data into the shell using the proper data template. QC rules are loaded next, and the shell decision process is initiated.

Quality control rules for each data class are structured to operate in two layers. The first layer of rules performs the actual analysis of the data while the second provides the mechanism for saving the data following the operation. The order of execution is controlled by setting the salience of the rules such that the second layer follows the first. For example, 'windspeedqc' is a first level rule which evaluates wind facts for a specific condition (a value outside of a range). When the condition is satisfied (wind speed in error), the fact being examined is retracted and reasserted with the appropriate QC tag. The second level rule 'windsavefacts' exists only to save each fact using the user defined function 'record_met_fact.' After the shell concludes execution, the child process saves the list of recorded facts to a disk file and sends a control message to trigger distribution of the data to the MMS network. The child process then terminates, and the parent waits for the next message.

The current set of rules implements simple range checking. A more sophisticated set of analysis rules could be easily developed and implemented. Quality control rule files are stored in a standard location (a library directory) and are reloaded for each execution, thus allowing for changes to be made while the Preprocessor continues to operate. Each data class has its own rule file, and modified rules will take effect during the next execution cycle for the affected data class.

2.2 Performance

The use of CLIPS for the QC function adds to the power and flexibility of this facility; however, the capabilities of the shell must also be considered in terms of its ability to complete all tasks within specified time requirements. If it cannot, then an alternate method would have to be implemented.

The CLIPS/QC processing times were obtained by inserting time checkpoints at different locations in the code and performing a series of test executions. The data obtained from these tests were used to calculate the maximum, minimum and mean performance times for each data class currently processed. The data, summarized in Table 2.1, expresses both a base processing estimate and three total processing estimates. The base estimate expresses the maximum, minimum and mean time for the quantity of data expected to arrive each minute. The total processing estimates are based on the average performance of the shell's processing of lightning data. The minimum, maximum and average times are multiplied by the number of lightning strikes (20, 60 or 100) and added to the base processing estimate listed in the first section of the table.

Table 2.1 CLIPS/QC Performance (all times are in seconds)

Class	*Iterations*	*Data Pts per Iteration*	*Maximum*	*Minimum*	*Average*
Wind	37	62	6.870	2.150	3.936
Temperature	37	69	4.850	1.470	2.378
Elec. Potential	31	625	22.430	7.540	11.457
General Met	37	1	0.340	0.180	0.241
			---	---	---
Base Processing Estimate			34.490	11.340	18.012
Single Lightning Strike	220	1	0.380*	0.180	0.240
Total processing estimates including lightning:					
Total/20 strikes	42.090	14.940	22.815		
Total/60 strikes	57.290	22.140	32.420		
Total/100 strikes	72.490	29.340	42.026		

* The maximum value within 3-standard deviations was used, above which there were only four points: 0.410, 0.430, 0.560, 0.590. The sample standard deviation was 0.053.

The performance of the shell in this implementation is well within the time requirement for handling expected data loads. The minimum one-minute data load had an average processing time of 18.012 seconds, which is only 30 percent of the maximum available time. In the MMS's current environment (Florida), lightning activity is a daily issue during the summer months. Data loads in excess of one strike per second are experienced frequently and such loads increased the average processing time to 32.420 seconds (54 percent of maximum); 100 strikes per minute required 42.026 seconds (70%). These results confirm that CLIPS can handle such a task in a near real-time environment. Admittedly the current rules are very simple, but this was necessary to validate a minimum ability to operate in this environment. The unanswered question remaining to be explored concerns enhancements to the complexity of QC rules and the effect such enhancements would have on processing times. While such a discussion is not within the scope of this paper, it

seems that enhancements are possible. Average idle time for the QC function was 18 seconds (30%) out of each minute, which would indicate available capacity for more sophisticated analysis techniques. Additional consideration must be given to the fact that newer, faster hardware would certainly provide faster processing of shell tasks.

3.0 Monitoring

The MMS Monitoring and Display Station (MDS) is responsible for receiving, monitoring and displaying meteorological and forecast data. CLIPS has been embedded in the monitoring (Data_Monitor) module and controls the activation and deactivation of alarm conditions based upon data and constraints. The MDS provides the user with all the necessary user interfaces for controlling and observing Data_Monitor activity.

3.1 Implementation

Data_Monitor consists of two separate processes; the Monitoring_Controller and Monitoring_CLIPS. Monitoring_Controller runs as a background process monitored by the system health facility. Monitoring_CLIPS is a module containing both CLIPS and user defined functions.

Monitoring_Controller waits for a message from the another MDS process which would indicate a change in status (i.e. new meteorological data received, operation activation, monitoring pause or resume). The controller uses a simple scheduling control mechanism to record such events and trigger shell executions. When a shell is to be executed, the controller retrieves all active meteorological constraints and builds CLIPS rules to search the fact base. Rules for each data class are written into separate disk files. Monitoring_CLIPS is started by Monitoring_Controller through a Unix 'fork' to process rules and data for a specific data class. One Monitoring_CLIPS is started for each data class scheduled to be processed. The Monitoring_Controller then waits for the children to complete their executions.

Monitoring_CLIPS initializes the shell, loads the data templates representing MMS data classes and then loads data for the class specified. Rules built by the controller are loaded last, and then the shell is executed through a 'RunCLIPS' function call. The monitoring rules are built differently based upon the current violation state of each constraint. An unviolated constraint is translated into a 'normal' rule set consisting of two rules. A constraint currently in violation is translated into a 'deactivation' rule set consisting of three rules.

The first rule of the 'normal' set verifies the existence of at least one applicable data point and asserts an enabling fact if such a point is found. The left-hand side (LHS) of the main constraint rule checks the existence of the enabling fact and examines the appropriate meteorological facts and QC tags. If the LHS is satisfied, the RHS will trigger an alarm signal using the user defined functions 'mtu_set_violation' and 'send_activation_message' so that the user interface will activate a visual and audible alarm.

The first rule of the 'deactivation' set is the same as that in the 'normal' set, as is the LHS of the main rule; however, the RHS differs in its structure and function. Since the goal of the 'deactivation' set is the opposite of the 'normal' set, its function is to search for a counterexample to the premise that no violation exists. If the counterexample exists in the fact base, the RHS will retract the enabling fact asserted in the first rule and undefine the third rule of the 'deactivation' set. No additional alarm is generated, and the screen icon in the user interface will continue to indicate a

violation. Without the existence of a counter example, the RHS of the third rule will clear the violation by using the user defined functions 'mtu_set_violation' and 'send_deactivation_message.'

A practical example of the monitoring function may be found by considering the following scenario: a payload canister is to be hoisted at KSC launch complex 39A. The meteorological constraints for that operation include a steady state wind limit of 17.2 knots and the absence of actual or forecast lightning activity within a 5 nautical mile radius. An MMS user could activate the 'Hoisting' operation from the list of those available, which would retrieve the associated constraints, load them into the monitoring table and notify the Monitoring_Controller process. Rules for wind and lightning strike data will be built from the constraints using the 'normal' form, and the two shells will be scheduled to execute. For wind class data two rule sets would be built: Hoist_1, which will alert when wind reported by sensor 9 exceeds of 17.2 knots, and Hoist_2, which does the same using sensor 10. The execution sequence for the wind class rules are depicted in Figure B.1 on page 8. Notice that the Hoist_1 rule violates after finding the sensor 9 wind speed of 19 knots at time=0. The violation continues as the measurement for time=1 still exceeds the threshold. Finally at time=2 the sensor 9 wind speed has passed below 17.2 knots. Notice that from time=0 through time=2 that sensor 10 was reporting invalid wind speed datum, and the Hoist_2_enable rule didn't fire; only when a valid measurement arrived at time=3 did Hoist_2_enable find a valid wind speed, and therefore assert the enabling fact (enable_Hoist_2).

3.2 Performance

The monitoring of meteorological measurements and observations can be greatly enhanced by the use of an expert system shell. In addition to finding violations of individual meteorological parameters, special sets of rules could search the fact base for a number of different conditions which may signal the onset of severe or threatening weather. The concern in using CLIPS, or any other shell, is one of performance. While the monitoring task has a less stringent time requirement than that of CLIPS/QC, it is difficult to express that value as a specific number. Processing certainly must be completed within the time period of the data, which is generally one minute. Occasional spikes above that level are acceptable so long as the overall performance does not exceed that level.

The CLIPS/Monitoring processing times were obtained by inserting time checkpoints at different locations in the code and performing a series of test executions. Tests were performed using the full MDS software with a test driver providing a full set of meteorological data at normal intervals. The number of constraints and facts were varied to test the performance characteristics of this implementation. Time samples were taken for constraint levels of 1, 10, 20, 30,...,100 and for fact levels of 0, 100, 200, 300,...,1000.

The performance of the shell was well within the time requirements stated above. Results from the test sample data set, consisting of 1038 time samples and summarized in Table 3.1, indicated that average performance for a heavy load will require about 15 seconds for setup (i.e. fact assertion, rule construction and loading) and about 30 seconds for the shell to complete processing. The resulting total of approximately 45 seconds is only 75% of the one minute target. Such results bode well not only for the ability of CLIPS to handle the current workload, but its ability to handle more complex tasks involving more a greater number of facts and more complex rules.

Additional performance analysis has been provided in Appendix C on page 10. Included in this appendix are three graphs that describe the manner in which actual shell processing time was

observed to vary in response to different combinations of values for the number of constraints and the number of facts. (Note: These times exclude the amount of time required to build rules and assert meteorological facts.)

Table 3.1 Monitoring Performance Tables (all times in seconds)

Setup Times			Facts			
Constraints	100		500		1000	
	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
1	2.03	0.03	7.92	0.48	15.40	0.95
10	2.27	0.27	7.70	0.39	15.16	1.04
30	2.18	0.18	7.83	0.02	15.00	0.55
50	2.17	0.15	7.96	0.48	15.44	1.52
Processing Times			Facts			
Constraints	100		500		1000	
	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
1	0.22	0.04	1.04	0.38	1.84	0.32
10	1.52	0.03	7.04	0.70	13.99	1.14
30	3.59	0.01	16.58	0.13	28.01	0.86
50	4.13	0.23	15.85	0.20	29.82	0.49

An interesting CLIPS performance characteristic is the apparent "levelling-off" of the processing time around 30 constraints. This can be seen in the bottom figure of Appendix B, which shows "crunch time" (i.e. shell processing time) as a function of the number of constraints for selected levels of the number of facts. In each of the curves in this figure, the observed processing time increases in a roughly linear fashion when the number of constraints was less than about 30, and the slopes of these curves increase as the number of facts increases. After 30 constraints, however, these curves level-off to be nearly horizontal. At this point, the effect of increases in the number of constraints on processing time is quite minimal. This is significant when we consider that the MMS monitoring table is currently capable of holding 1000 constraints. Since the performance times seem to level-off above a relatively modest number of constraints, activation of a much larger group should not result in excessively large execution times.

4.0 Conclusion

The CLIPS expert system shell performs well in both MMS implementations. Although neither the quality control nor the monitoring implementation constitutes a complex use of expert systems technology, this application of CLIPS is useful in several ways. First, it is currently functional and useful, and data are processed and monitored in an effective manner within all time requirements. Second, enhancements are possible that would more fully exploit CLIPS capabilities. Third, the system has potential for growth through the utilization of a more powerful set of rules and the use of another CLIPS tool known as COOL (CLIPS Object Oriented Language). The primary objective of the MMS project was to create a system that would be useful to weather forecasters and operational personnel. We believed that CLIPS could greatly enhance the system and help us to reach our objective, but we also had reservations concerning implementation and performance. It is now clear that the inclusion of CLIPS was a proper decision based upon its performance and capabilities.

Appendix A MMS Architecture

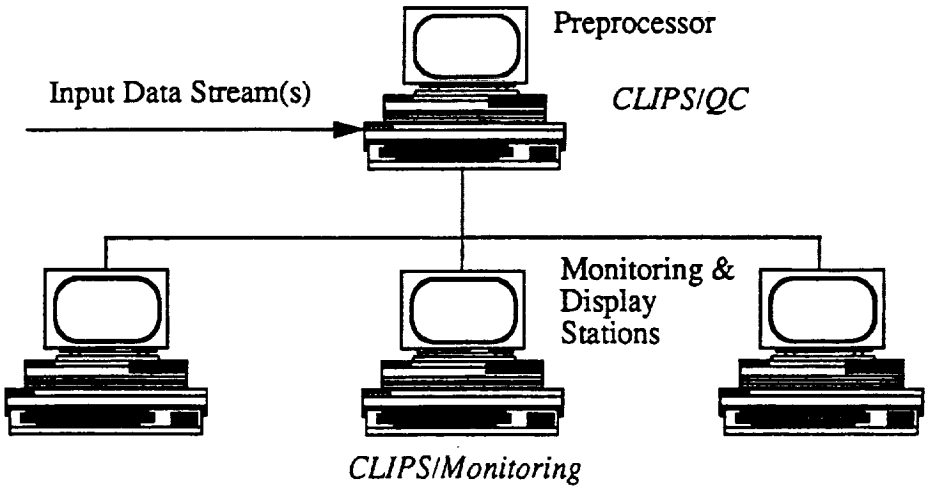


Figure A.1 MMS network overview

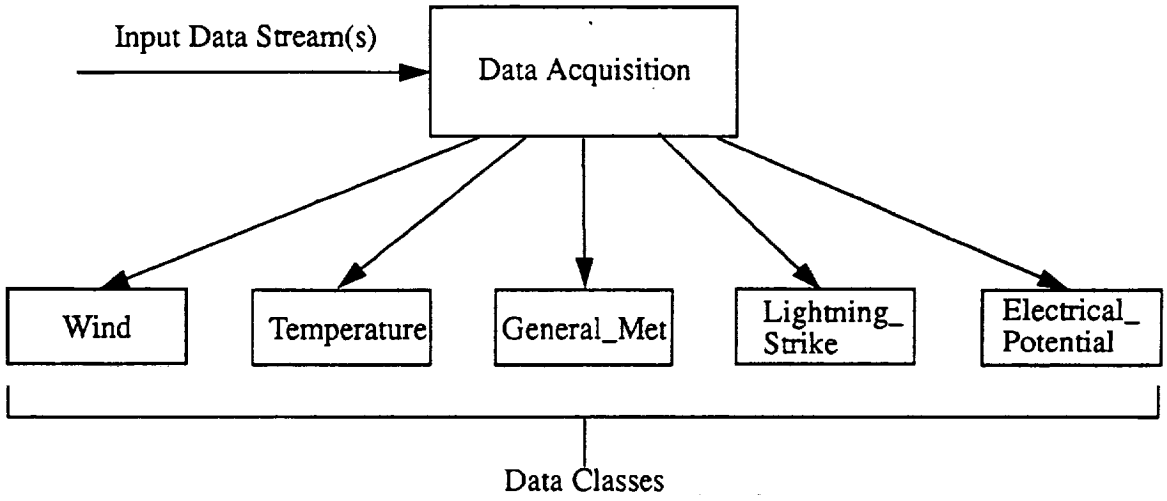


Figure A.2 MMS data classes

Appendix B Sample Monitoring Scenario

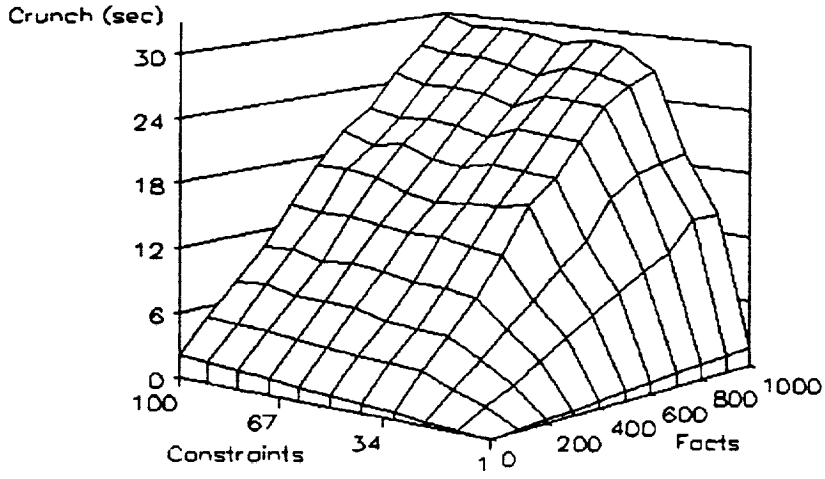
Fact Base	Rules	Alarms	Time
Sen 9 Spd 19 QC-Ok Sen 10 Spd 99 QC-Bad Sen 11 Spd 16 QC-Ok Sen 12 Spd 15 QC-Ok Sen 13 Spd 16 QC-Ok	Hoist_1_enable Hoist_2_enable Hoist_1_main Hoist_2_main *New data received, rules built and loaded.*	None	0
Sen 9 Spd 19 QC-Ok Sen 10 Spd 99 QC-Bad Sen 11 Spd 16 QC-Ok Sen 12 Spd 15 QC-Ok Sen 13 Spd 16 QC-Ok enable_Hoist_1	Hoist_2_enable Hoist_1_main Hoist_2_main *Hoist_1_enable finds data, asserts enabling fact, and undefines itself. Hoist_2_enable finds no data (sensor 10 QC is bad).*	None	0
Sen 9 Spd 19 QC-Ok Sen 10 Spd 99 QC-Bad Sen 11 Spd 16 QC-Ok Sen 12 Spd 15 QC-Ok Sen 13 Spd 16 QC-Ok	Hoist_2_enable Hoist_1_main Hoist_2_main *Hoist_1_main finds wind violation, triggers alarm and retracts enabling fact.*	Hoist WIND	0
Sen 9 Spd 18 QC-Ok Sen 10 Spd 99 QC-Bad Sen 11 Spd 15 QC-Ok Sen 12 Spd 13 QC-Ok Sen 13 Spd 16 QC-Ok	Hoist_1_enable Hoist_2_enable Hoist_1_main Hoist_2_main Hoist_1_Deactivate *New data received, rules built and loaded.*	Hoist WIND	1
Sen 9 Spd 18 QC-Ok Sen 10 Spd 99 QC-Bad Sen 11 Spd 15 QC-Ok Sen 12 Spd 13 QC-Ok Sen 13 Spd 16 QC-Ok enable_Hoist_1	Hoist_2_enable Hoist_1_main Hoist_2_main Hoist_1_Deactivate *Hoist_1_enable finds data, asserts enabling fact, and undefines itself. Hoist_2_enable finds no data (sensor 10 QC is bad).*	Hoist WIND	1
Sen 9 Spd 18 QC-Ok Sen 10 Spd 99 QC-Bad Sen 11 Spd 15 QC-Ok Sen 12 Spd 13 QC-Ok Sen 13 Spd 16 QC-Ok	Hoist_2_enable Hoist_1_main Hoist_2_main *Hoist_1_main finds wind violation, undefines deactivation rule and retracts enabling fact.*	Hoist WIND	1

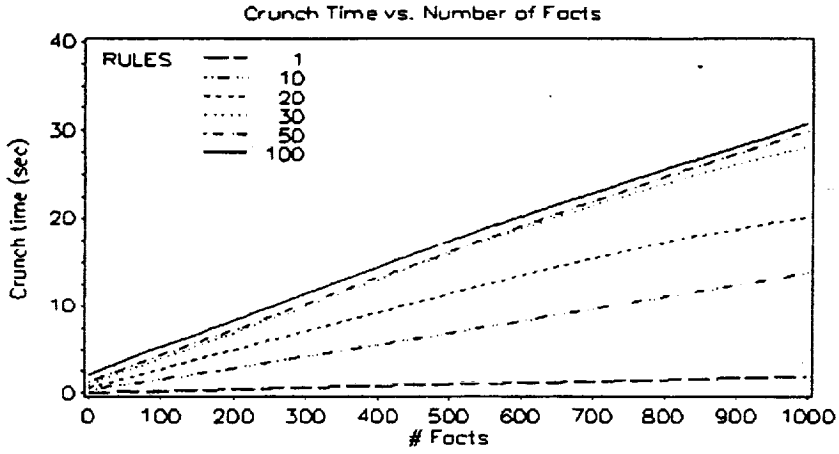
Figure B.1 Monitoring Scenario

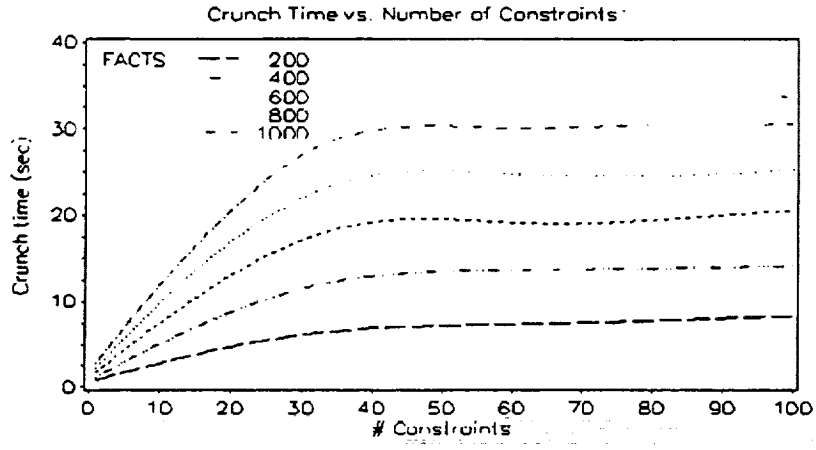
Fact Base	Rules	Alarms	Time
Sen 9 Spd 17 QC-Ok Sen 10 Spd 99 QC-Bad Sen 11 Spd 13 QC-Ok Sen 12 Spd 12 QC-Ok Sen 13 Spd 13 QC-Ok	Hoist_1_enable Hoist_2_enable Hoist_1_main Hoist_2_main Hoist_1_Deactivate *New data received, rules built and loaded.*	Hoist WIND	2
Sen 9 Spd 17 QC-Ok Sen 10 Spd 99 QC-Bad Sen 11 Spd 13 QC-Ok Sen 12 Spd 12 QC-Ok Sen 13 Spd 13 QC-Ok enable_Hoist_1	Hoist_2_enable Hoist_1_main Hoist_2_main Hoist_1_Deactivate *Hoist_1_enable finds data, asserts enabling fact and undefines itself. Hoist_2_enable finds no data (sensor 10 QC is bad).*	Hoist WIND	2
Sen 9 Spd 17 QC-Ok Sen 10 Spd 99 QC-Bad Sen 11 Spd 13 QC-Ok Sen 12 Spd 12 QC-Ok Sen 13 Spd 13 QC-Ok	Hoist_2_enable Hoist_1_main Hoist_2_main Hoist_1_Deactivate *Hoist_1_main finds no violation. Deactivation rule clears indicator, retracts enabling fact.*	None	2
Sen 9 Spd 17 QC-Ok Sen 10 Spd 19 QC-Ok Sen 11 Spd 15 QC-Ok Sen 12 Spd 14 QC-Ok Sen 13 Spd 16 QC-Ok	Hoist_1_enable Hoist_2_enable Hoist_1_main Hoist_2_main *New data received, rules built and loaded.*	None	3
Sen 9 Spd 17 QC-Ok Sen 10 Spd 19 QC-Ok Sen 11 Spd 15 QC-Ok Sen 12 Spd 14 QC-Ok Sen 13 Spd 16 QC-Ok enable_Hoist_1 enable_Hoist_2	Hoist_1_main Hoist_2_main *Hoist_1_enable finds data, asserts enabling fact and undefines itself. Hoist_2_enable finds data, asserts enabling fact and undefines itself.*	None	3
Sen 9 Spd 17 QC-Ok Sen 10 Spd 19 QC-Ok Sen 11 Spd 15 QC-Ok Sen 12 Spd 14 QC-Ok Sen 13 Spd 16 QC-Ok	Hoist_1_main Hoist_2_main *Hoist_1_main finds no violation. Hoist_2_main finds wind violation, triggers alarm and retracts enabling fact.*	Hoist WIND	3

Figure B.2 Monitoring Scenario (continued)

Appendix C CLIPS/Monitoring Performance Graphs







REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	Nov/94	Conference Proceedings, September 1994

4. TITLE AND SUBTITLE

Third CLIPS Conference Proceedings - Volumes I and II

5. FUNDING NUMBERS

6. AUTHOR(S)

Gary Riley, editor

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Lyndon B. Johnson Space Center
Houston, Texas 77058
I-NET, Inc.
Houston, Texas 77058

8. PERFORMING ORGANIZATION REPORT NUMBERS

S-785

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA CP-10162

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified/Unlimited
Available from the NASA Center for AeroSpace Information
800 Elkridge Landing Road
Linthicum Heights, MD 21090-2934
(301) 621-0390 Subject Category: 61

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Expert systems are computer programs which emulate human expertise in well defined problem domains. The potential payoff from expert systems is high: valuable expertise can be captured and preserved, repetitive and/or mundane tasks requiring human expertise can be automated, and uniformity can be applied in decision making processes. The C Language Integrated Production System (CLIPS) is an expert system building tool, developed at the Johnson Space Center, which provides a complete environment for the development and delivery of rule and/or object based expert systems. CLIPS was specifically designed to provide a low cost option for developing and deploying expert system applications across a wide range of hardware platforms. The development of CLIPS has helped to improve the ability to deliver expert system technology throughout the public and private sectors for a wide range of applications and diverse computing environments. The Third Conference on CLIPS provided a forum for CLIPS users to present and discuss papers relating to CLIPS applications, uses, and extensions.

14. SUBJECT TERMS

Expert Systems, Programming Languages, Computer Techniques

15. NUMBER OF PAGES

401

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Unlimited