JPL Publication 88-14

Diagnosing Faults in Autonomous Robot Plan Execution

Raymond K. Lam
Rajkumar S. Doshi
David J. Atkinson
Denise M. Lawson

March 1, 1988

National Aeronautics and Space Administration
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

JPL Publication 88-14

Diagnosing Faults in Autonomous Robot Plan Execution

Raymond K. Lam
Rajkumar S. Doshi
David J. Atkinson
Denise M. Lawson

March 1, 1988

National Aeronautics and
Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

The research described in this publication was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

ABSTRACT

A major requirement for an autonomous robot is the capability to diagnose faults during plan execution in an uncertain environment. Many diagnostic researches concentrate only on hardware failures within an autonomous robot. Taking a different approach, this publication describes the implementation of a Telerobot Diagnostic System that addresses, in addition to hardware failures, failures caused by unexpected event changes in the environment or failures due to plan errors. One unique feature of the system is the utilization of task-plan knowledge and context information to deduce fault symptoms. This forward deduction provides valuable information on past activities and the current expectations of a robotic event, both of which can guide the plan-execution inference process. The inference process adopts a model-based technique to recreate the plan-execution process and to confirm fault-source hypotheses. This technique allows the system to diagnose multiple faults due to either unexpected plan failures or hardware errors. This research initiates a major effort to investigate relationships between hardware faults and plan errors, relationships that have not been addressed in the past. The results of this research will provide a clear understanding of how to generate a better task planner for an autonomous robot and how to recover the robot from faults in a critical environment.

CONTENTS

Figures

SECTION 1

INTRODUCTION

An autonomous robot is a vehicle that has some degree of intelligence to plan and navigate in an uncertain environment. It must be capable of performing some useful task without the intervention of human operators. There is a wide range of applications suitable for autonomous operation due to either hazardous environments or human inaccessibility. These areas include mining operation, space exploration, space operation, underwater exploration, nuclear plant servicing, and military surveillance.

The operational environment of an autonomous robot differs from the manufacturing environment of an assembly robot in several ways. One example that demonstrates these differences is the autonomous robot in the space environment:

1. Human assistance in maintaining, diagnosing, and repairing the robot is very difficult, and sometimes impossible, to achieve in space. Thus, the robot must operate independently and autonomously to perform very complex tasks.

2. A space robot performs many unique tasks in different scenarios, as opposed to solely performing repetitive routines. For instance, a space robot's tasks include the acquisition of satellites, the repair of damaged instruments, and the assembly of large space structures. The space robot must actively utilize special expert knowledge to generate and execute specific task plans for these unique scenarios.

3. There is a communication delay between a space robot and the ground station, a situation that limits the availability of sensor information and past-activity information. This limitation reduces the effectiveness of performing real-time diagnosis on earth.

4. When a failure occurs on a space robot, it is critical to discover the origin of the failure as quickly as possible and to understand how and why the failure occurred. This must be done in order to recover from failure gracefully and prevent future occurrences of the same failure.

5. A space robot has minimal self-checking and sensing capabilities due to its physical size limitation.

One such example of a space robot is the research robot called Telerobot, which is being developed at the Jet Propulsion Laboratory (JPL) to perform satellite servicing. The Telerobot will be controlled by human operators on earth. Having been given a specific task by the operators, the Telerobot should be able to generate a plan, carry out the operations, and complete the task. A major concern is that the Telerobot have the capability to diagnose faults during plan execution in the space environment. This fault diagnosis requires the interpretation of multiple faults in the environment,

the handling of plan faults and hardware faults, and the prevention of any further errors that could cause irreversible damage to the Telerobot itself. Therefore, the Telerobot should have a diagnostic capability to interpret the environment at failure, reason about the fault symptoms, classify the faults, generate hypotheses about abnormalities, and search for an appropriate solution.

The Telerobot Diagnostic System (TDS), a NASA-sponsored research project at JPL, addresses the issues mentioned above. In this publication, we will discuss approaches to fault diagnosis, factors that influenced the determination of the TDS approach, and the TDS architecture in a space environment. The space environment is similar to other critical environments that demand the diagnostic capabilities of an autonomous robot. Hence, the TDS approach can be applied to different areas in the autonomous robot domain.

SECTION 2

APPROACHES TO FAULT DIAGNOSIS

In recent years, a great deal of research has been devoted to knowledge-based diagnostic systems in several domains.[1-5] Few of these systems, however, address the problems inherent in the Telerobot domain.

Collectively, knowledge-based fault diagnostic systems can be classified as one of two approaches.[6] The first approach involves reasoning from device-specific symptom/cause relationships, and the second involves reasoning from the basic principles of connectivities and functionality.

In an example of the first approach, Gini et al.[7-9] present a symbolic reasoning technique to automate fault recovery in manufacturing robots. Implementation by these authors classifies operations based on the semantic and qualitative meaning of actions, and maps each operation into a predetermined relationship table to identify a set of possible faults. The technique uses this set of data to infer from the existing fault symptoms and sensor history in an attempt to deduce the source of the fault. By taking advantage of heuristic knowledge in the robotic domain, this classification and mapping technique reasons about the relationships between operations and the possible faults. A major drawback to the symptom/cause reasoning approach is its inability to explicitly know the underlying mechanisms of operations. Also, it requires much effort to generate an adequate set of symptom/cause relationships.

In an example of the second approach, Krishnamurthi et al.[10] developed an assembly robot fault-diagnosis expert system using a combination of deep and shallow modeling approaches. The system diagnoses electromechanical problems by combining deep reasoning based on the design description of robot components and shallow reasoning based on simple decision rules, which emulate the reasoning process of a human robot-diagnosis expert. To diagnose a fault, the system reasons about physical and electrical interconnections, the expert's knowledge and experience, and test data. Since the system is used as an advisor to assist the technician in diagnosing hardware faults, and the robot is simply performing repetitive routines in a static manufacturing environment, the system does not infer from past operation history or plan knowledge to identify fault sources. Rather, it reasons about the current situation and relies on test data reported by the technician to guide the diagnostic process.

Neither of these systems addresses all of the environmental and operational requirements in the Telerobot domain. The following sections will discuss Telerobot system requirements and associated constraints leading to a fault diagnostic technique that attempts to address these issues.

SECTION 3

FACTORS IN DETERMINING THE APPROPRIATE TELEROBOT DIAGNOSTIC SYSTEM APPROACH

3.1 TELEROBOT DOMAIN CONSTRAINTS

One consideration in determining an effective Telerobot diagnostic approach is the factor of Telerobot domain constraints, which can be classified as either environmental or operational.

3.1.1 Environmental Constraints

An important characteristic for the Telerobot system to have is the ability to perform complex tasks in many unique scenarios. This variety of scenarios means that the TDS must be able to understand the context of the environment and the implication of the environmental constraints in order to diagnose faults effectively.

In addition, changes in the system configuration made by each specific task require that the TDS be able to recognize each new configuration of components. These components include the manipulators, instruments, sensors, and objects in the environment. Different configurations can lead to different interpretations of the same set of sensor data. For example, while alarmed force/torque sensor data may indicate that a moving arm has hit an obstacle, it could also indicate that a stationary arm is trying to pick up an unmovable object. Because of the limited availability or inaccuracy of sensor data, a factor of uncertainty is added to the understanding of the environment. Therefore, the TDS must be able to reason in this uncertain environment in order to predict the consequences of an operation and to resolve conflicts among sensor data. In cases where faults are propagated from previous operations, the TDS must be able to reason backward to determine where the faults occurred and why they were not detected when they occurred.

Finally, because space operation demands a highly reliable autonomous robot to perform critical tasks, it is essential that the TDS provide a complete picture on how and why faults occurred in order for fault recovery to regain control of the Telerobot and to prevent future fault occurrences.

3.1.2 Operational Constraints

The TDS must have a clear interpretation of the task plan in order to effectively handle the complexity of Telerobot functions. Multiple mechanical maneuvers, such as dual arm operations, are sometimes required, and such maneuvers can often lead to hardware faults caused by malfunctions of the manipulators. These maneuvers can also lead to the discovery of plan faults caused by coordination problems in the task plan. The TDS must be able to interpret the task plan in order to differentiate between these types of faults. Also, a complex task normally involves a large number of actions, and it is difficult to generate a complete set of fault symptom/cause relationships

for all actions. An understanding of the task plan can compensate for this incomplete set of relationships.

Another operational constraint involves the taxonomy of Telerobot control. Telerobot control is structured hierarchically, separating specific knowledge about an operation into several levels. The TDS must be capable of organizing and inferring knowledge from multiple levels in order to deduce the sources of faults. Active testing on the manipulators with incomplete knowledge about the environment at each level could lead to secondary effects and could ultimately cause catastrophic failure of the system.

Because of the uncertainty of the environment, it is almost a necessity to interpret the environment first in order to capture the Telerobot status and effectively identify the faults. Also, the complexity of the Telerobot operation and the inability to actively test for problem identification implies that the task plan becomes the only mechanism for capturing the Telerobot configuration, as well as the point of failure of the Telerobot. An event-driven approach that uses events to drive the interpretation also takes advantage of constraint knowledge.

3.2 TELEROBOT KNOWLEDGE ENVIRONMENT

In addition to Telerobot constraints, another important issue to consider in determining the appropriate TDS approach is the knowledge environment. The Telerobot knowledge environment is characterized by the hierarchical structure of the Telerobot control environment as shown in Figure 1. The Telerobot control architecture generates plan knowledge at three levels: Task Planning knowledge at the top of the hierarchy, followed by Process Command knowledge at the intermediate level, and Machine Command knowledge at the bottom of the hierarchy.[11] Task Planning knowledge resides in a Telerobot subsystem called the Task Planner, and is concerned with the overall context of a task. (See Section 4.2, "Telerobot System Overview," for a brief description of this and the following subsystems.) The Task Planner generates a logical global plan to achieve the objective of a given task. Process Command knowledge resides in the Run Time Controller subsystem, and involves the detailed specification of the subtasks. The Run Time Controller generates physical and geometrical requirements to specify the precise manipulations of Telerobot components. Machine Command knowledge is used by the Manipulation and Control Mechanization subsystem and the Sensing and Perception subsystem to control the basic manipulation and sensor capability of the Telerobot.

The Telerobot knowledge environment determines what and how information is available to the TDS. Since there is more information available in the Telerobot knowledge environment than in possible Telerobot problems and solutions at the time of failure, it makes sense to concentrate on the event to drive the diagnosis, rather than concentrating directly on the problem. Also, because of the hierarchical nature of the Telerobot knowledge environment and control structure, it is advantageous to adopt the event-driven approach to understand the knowledge environment at different levels and to exploit plan knowledge for interpreting symptoms.

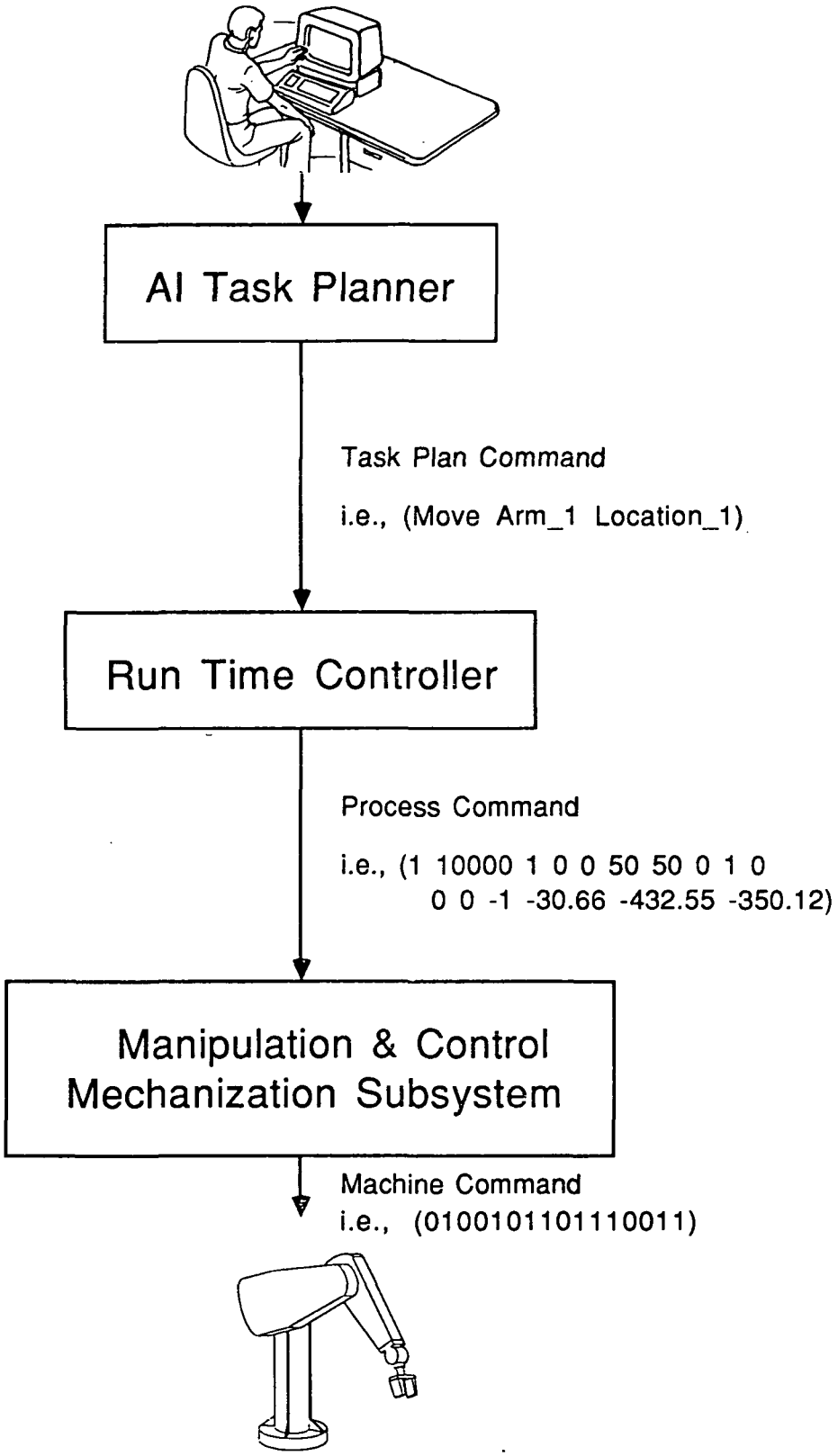

Figure 1. Example of the Telerobot Control Environment

SECTION 4

THE TELEROBOT DIAGNOSTIC SYSTEM TECHNIQUE

4.1 TDS APPROACH

Considering the nature of Telerobot constraints and the nature of the Telerobot knowledge environment, an event-driven approach was chosen for the Telerobot fault-diagnosis system. By determining the context of the environment first, the TDS can better interpret component relationships and reason forward to identify the specific step of an action that caused a fault. Also, since a plan can contain many sequential actions and thus generate a large search space, by interpreting the Telerobot task objective and the environment first, the TDS can reduce the search space. It does so by enhancing the coordination and interpretation of knowledge at different levels and identifying which actions demand detailed analysis and which actions require only high-level attention. Using the event-driven approach also leads to a better understanding of the environment. Since the interpretation of the environment is more complete, there is an increased capability to understand the problems and correctly deduce the fault sources.

The event-driven approach addresses diagnosis in two phases. The first phase interprets the existing Telerobot environment at failure in order to deduce fault symptoms, and the second phase recreates the task-plan execution process in an attempt to discover the sources of the faults.

4.1.1 The Robotic Environment Interpretation Phase

Because of the complexity of tasks and the existence of unique scenarios, it is important for the TDS to exploit context knowledge of the environment and plan knowledge of the task plan in order to determine the configurations of the Telerobot at failure. Context knowledge is concerned with the representation of component state information in symbolic form and the interrelationship of components in the Telerobot environment. Context knowledge is generated via the interpretation of characteristics of components, initial state information, sensory data, and state information at failure. The availability of context knowledge enables the TDS to resolve conflicts about sensory data, to identify symptoms at different control levels, to resolve ambiguous state information, and to produce unique interpretations of sensor data of the Telerobot environment.

Plan knowledge is concerned with the operation of actions that modify the component state of the environment. The interpretation of these operations provides the possible changes to the environment and the accessibility of components at each state. The availability of plan knowledge enables the TDS to determine the effects of the operation, review mechanical problems in the manipulators, discover plan errors, confirm the consistency of actions at different control levels, and isolate the components involved in the abnormal operation.

The availability of both context knowledge and plan knowledge enables the TDS to interpret the environment and thus determine the

Telerobot status at failure. It combines symptoms from both types of knowledge to resolve conflicts and deduce a list of suspected failed components. The suspect component list is used to limit the number of hypotheses during plan execution simulation in the second phase and to determine the appropriate control level for the task-plan execution simulation process.

4.1.2 The Plan Execution Inference Phase

The Plan Execution Inference phase attempts to discover the source of a fault by recreating the task plan execution process. Each action is performed in a simulated environment so the TDS can observe the consequences of that action. The TDS adopts a model-based inference technique[12-14] using behavioral models of components and functional models of operations to infer the fault sources. This technique uses deep-level models that provide a capability to predict consequences of an operation, to recreate the plan execution process, and to confirm fault-source hypotheses.

The flexibility of the model-based technique allows the TDS to simulate plan execution with any combination of actions or manipulator configurations. This reduces dependency on imperfect sensory data while still taking advantage of the data's potential when available. The model-based technique allows the reasoning process to switch control levels at will to concentrate on the suspected components. More importantly, reasoning with the behavioral and functional models of the components provides the capability to hypothesize multiple faults that sometimes may occur. When multiple faults occur, the TDS can simulate the abnormal behaviors of each individual component in sequence to hypothesize the collective consequence of the faults.

4.2 TELEROBOT SYSTEM OVERVIEW

The Telerobot system at JPL consists of three PUMA 560 manipulators and four cameras as shown in Figure 2. Two manipulators are used to perform tasks, and the third manipulator is installed with a two-camera stereo vision system to monitor manipulations in the testbed. In addition, two wing cameras are used to track the illustrated satellite. The manipulators are controlled hierarchically by three subsystems: the Task Planner, the Run Time Controller, and the Manipulation and Control Mechanization subsystem. These subsystems contain specialized knowledge on task planning, process commands, and machine commands, respectively. In addition to these subsystems are the Execution Monitor,[15] which observes the Telerobot status, the Telerobot Diagnostic .System, which detects faults, and the Sensing and Perception subsystem, which provides visual information. A diagram of the Telerobot system is shown in Figure 3.

4.3 TDS ARCHITECTURE

The Telerobot Diagnostic System was developed on a Symbolics 3640 LISP Machine using the ZetaLisp language and a JPL-developed expert

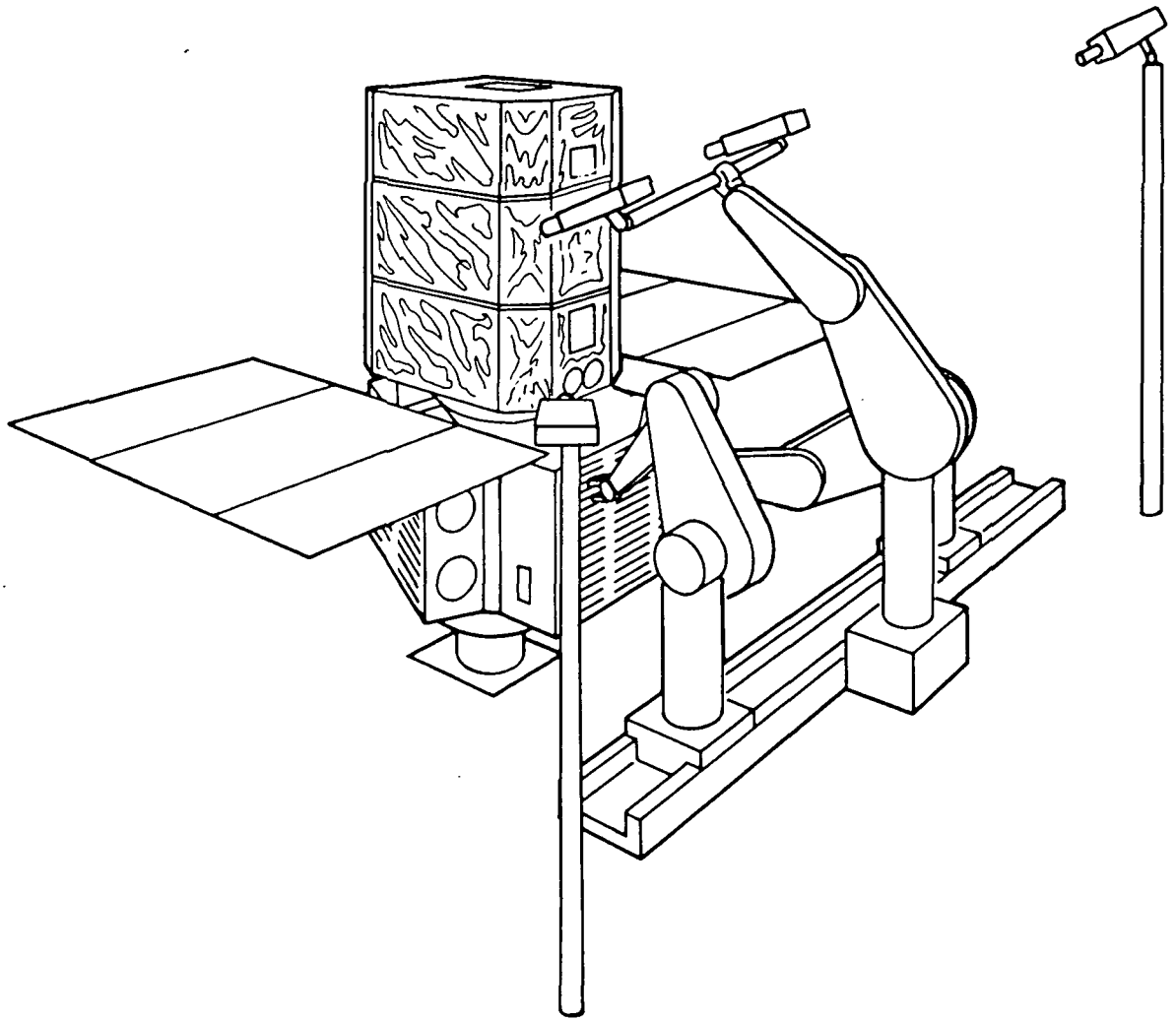

Figure 2. Telerobot Operation Configuration

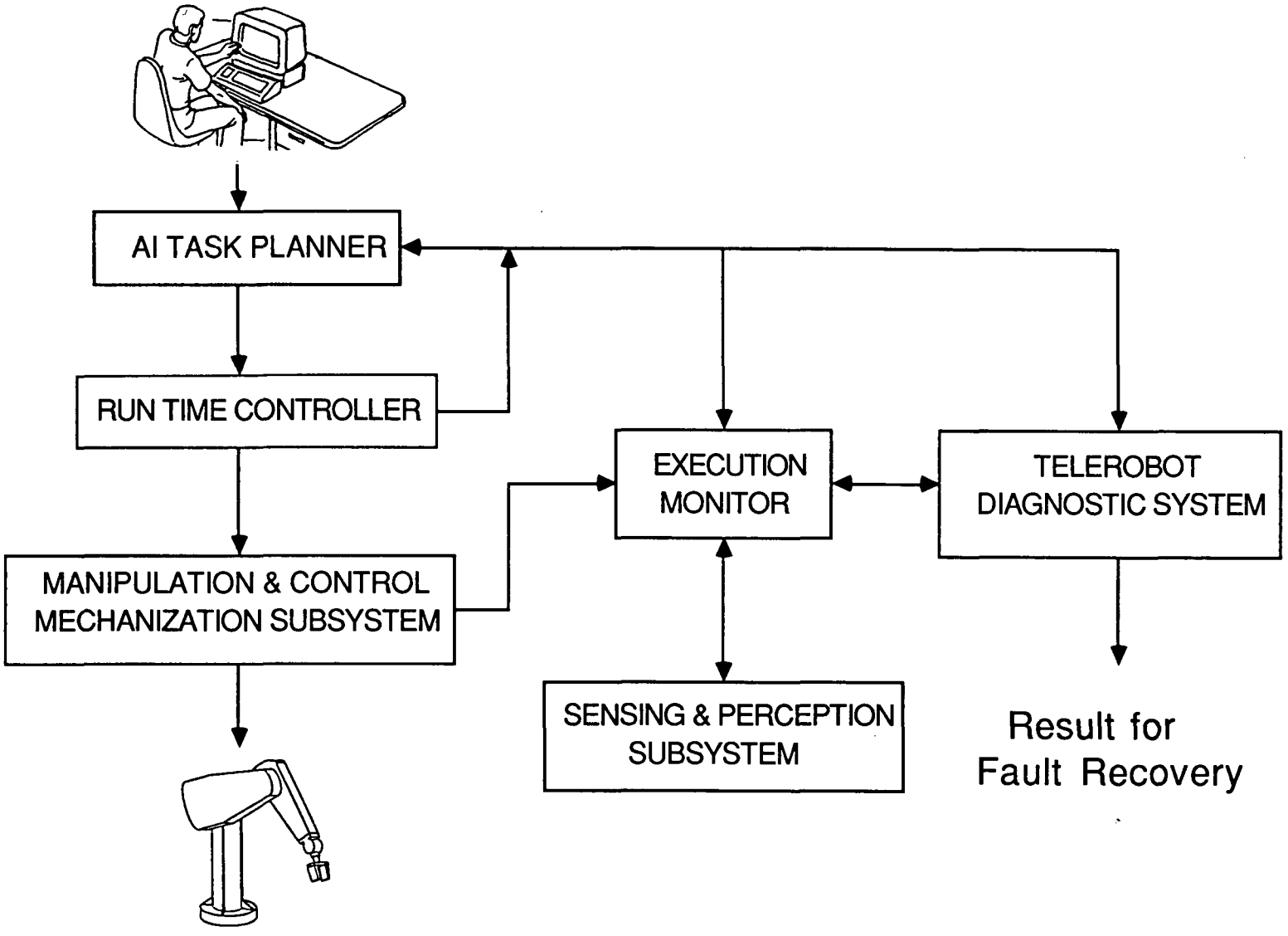

Figure 3. Telerobot System Diagram

system tool called TELESIS.[16-17] Most of the knowledge is implemented as production rules, and the modules are implemented as procedural functions.

4.3.1 TDS System Overview

The inference structure of the TDS is composed of six modules: the Alarm Handler, Sensor Interpreter, Context & Plan Interpreter, Symptom Identifier, Fault Region Identifier, and Plan Execution Inference Module. During the interpretation phase, the TDS uses the first five modules to reason about the existing environment and identify possible sources of errors. The sixth module, Plan Execution Inference, is used during the second phase to simulate the task-plan execution process, to hypothesize past activities, and to confirm the sources of the faults. A diagram of the Telerobot Diagnostic System is shown in Figure 4. The following sections will describe the operations of each module.

4.3.1.1 Alarm Handler. Many sensors, such as grip sensors and force/torque sensors, exist in the Telerobot system. During the plan execution phase, the Execution Monitor examines those sensors that are directly relevant to the action currently executing. Sensors not relevant to the current action are stored in an Event Trace, which is located within the Execution Monitor. When a sensor value exceeds its thresholds of validity, the Execution Monitor will activate the TDS Alarm Handler and report the abnormal data.[18] The Alarm Handler translates the alarm values into a symbolic form that represents the alarmed state of the sensors. For example, if the Gripper should be in a closed state *(value = 0)* and, for some reason, it is open *(value = 1)*, the Alarm Handler will translate this numerical information into the symbolic form *(Status Gripper Open)*.

A sensor in a state of alarm can affect an action in one of two ways. If the sensor does not affect the continuation of the current plan step within the action, then the plan step will be completed before the Alarm Handler halts the action. For instance, in the ACQUIRE action illustrated in Figure 5, suppose that the *Contact_Sensor* indicates that grip status goes into alarm during Primitive Plan Step 2 *(Close Arm_1)*. This is not critical to the completion of the *CLOSE* command, therefore the Alarm Handler allows the *CLOSE* command to finish before halting the *ACQUIRE* action at Primitive Plan Step 2.

On the other hand, if the sensor data is critical to the continuation of the current plan step, then the Alarm Handler will cause the action to be halted immediately. For example, if force/torque sensors used in a compliance motion detect excessive force, then the action will be halted immediately to avoid possible component damage.

4.3.1.2 Sensor Interpreter. During the Interpretation phase, the Execution Monitor passes the valid sensor data to the TDS Sensor Interpreter, which translates the sensors' numerical values into symbolic forms in a similar manner as the Alarm Handler. The Sensor Interpreter compares this information with the expected conditions of the current action. The physical

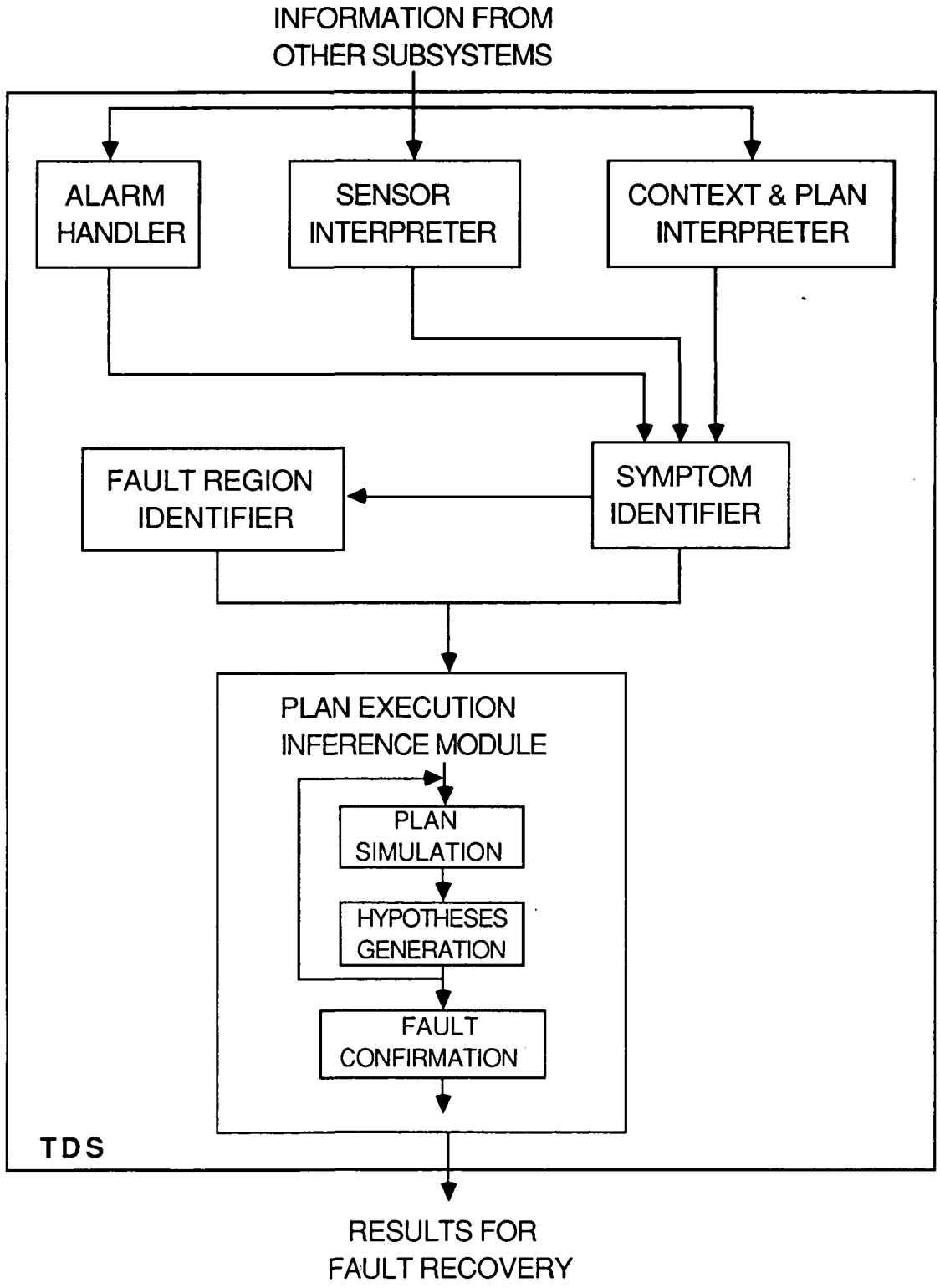

Figure 4. Telerobot Diagnostic System Diagram

ACTION: Acquire *Block_A* **With** *Arm_1*
From *Location_1*
To *Location_2*

Preconditions:

(At *Block_A Location_1)*
(Contains *Arm_1* Nil)
(Status *Arm_1* Open)
(At Nil *Location_2)*

Primitive Plan Steps:

1. **Move** *Arm_1* **To** *Location_1*
2. **Close** *Arm_1*
3. **Move** *Arm_1* **To** *Location_2*

Expected State:

(Contains *Arm_1 Block_A*)
(Status *Arm_1* Close)
(At Nil *Location_1*)
(At Arm_1 *Location_2)*

Figure 5. Example of the Acquire Action

components that successfully generate the expected conditions of the current environment will be considered as valid components. These components will be eliminated from fault consideration during the second phase, Plan Execution Inference.

The Sensor Interpreter is also responsible for translating the sensor data in the Execution Monitor's Event Trace. It does this to provide historical information to the Plan Execution Inference process.

4.3.1.3 Context & Plan Interpreter. One objective of the Context & Plan Interpreter module is to abstract information from the Task Planner and the Run Time Controller. Another objective is to interpret for each action the two types of knowledge generated from this module—context knowledge and plan knowledge.

Context knowledge comprises information about stationary objects and instruments in the environment. This includes the location, orientation, and characteristics of each object involved in a given action. The spatial relationship of objects and the constraints of the objects on each other are represented explicitly. However, plan errors or propagated faults are difficult to diagnose by inferring only sensor data, so context knowledge is essential in order for the Symptom Identifier to isolate the faults. For example, suppose *Block_A* and *Block_B* lie against each other, and there is an operation to move *Block_A* to a new location. This operation can accidentally displace *Block_B* and still can carry out the execution successfully. However, if at a later time the manipulator wants to grasp *Block_B*, the previous displacement will cause failure on that operation since *Block_B* is not at its anticipated position. By reasoning with context knowledge about the relationship between *Block_A* and *Block_B*, and with plan knowledge of the manipulator's trajectory path, the Symptom Identifier will be able to deduce that *Block_B* was possibly displaced during the *MOVE* operation for *Block_A*.

Plan knowledge is subdivided into task-plan knowledge provided by the Task Planner, and manipulator knowledge generated by the Run Time Controller. Task-plan knowledge provides high-level information on how to achieve an objective, and includes information on the intentions of the objective, the actions needed to achieve the task, the expected state of the environment after the operation, and the relationship between objects and actions before and after the action (i.e., the effect of a *MOVE* operation on *Block_A*). Task-plan knowledge also includes constraint information on each action. For example, if a *MOVE* operation cannot be completed, the task-plan knowledge and the location of the manipulator can be used to deduce why the operation failed. If the manipulator failed outside of the proposed *MOVE* path, then the Symptom Identifier module will assume a mechanical failure of the manipulator. If, however, the operation failed while the manipulator was on course, and all environmental and operational constraints were satisfied, then the Symptom Identifier module will suspect a coordination problem in the task plan.

Manipulator knowledge is generated from the Run Time Controller and concerns the kinematics and dynamics of the manipulator. The manipulator configurations along the trajectory path can provide information

such as potential spatial problems, joint limitations, and critical paths or configurations that should be avoided. For example, suppose the gripper attempts to pick up a slippery object. The grasping and transporting postures of the manipulator during the *GRASP* operation can predict what may happen to that object. If the object has a degree of freedom toward the ground, then it will most likely slip from the gripper, and the Symptom Identifier will assume that the object was dropped somewhere along the trajectory path.

In summary, the Context & Plan Interpreter uses knowledge about the environment and knowledge about the task plan and manipulators to deduce the Telerobot status at failure.

4.3.1.4 Symptom Identifier. The Symptom Identifier accepts alarm conditions from the Alarm Handler, the status of valid components from the Sensor Interpreter, and state information from the Context & Plan Interpreter. Initially, it checks for consistency of information from these modules. When conflicts arise, the Symptom Identifier will reason about the contradictions and consult with the vision system and other sensing systems to identify the errors. It then attempts to understand the implications of the existing environment, deduce symptoms, and generate a list of suspected components that may have caused the faults.

Shown in Figure 6 is an example of the Symptom Identifier process. Assume that *Arm_1* is supposed to pick up *Block_A* from a table. *Block_A* is very slippery, so the manipulator should pick it up in a posture that allows it no degree of freedom toward the ground. However, in this case, the Task Planner did not consider the slippery characteristic of the block. Therefore, during the operation, the Alarm Handler will detect that the *Contact_Sensor* suddenly opens, and it will interrupt the system from further operation. The Symptom Identifier accepts state information from the Alarm Handler, the Sensor Interpreter, and the Context & Plan Interpreter, then checks this data for consistency. It rules out the assumption that the *Contact_Sensor* is broken since *Force_Sensor* information and vision data contradict the assumption. The other assumption, that the object is missing, remains consistent with the state information. Therefore, components that could have caused the object to disappear—arm trajectory and object characteristics—are placed on the suspect component list. The Symptom Identifier reasoning process uses all of the information from the three modules to deduce this suspect component list, and fault inferencing will be confined to just these components.

Another responsibility of the Symptom Identifier module is to classify fault type. Because of the complex nature of the task plan, the TDS must avoid a complete search of the task plan for suspected components to prevent a combinatorial explosion of the search space. One method to limit this problem is to isolate the faults within a specific region. The Symptom Identifier classifies faults as either local faults or propagated faults. When a fault occurs during execution of an action or during the verification of an expected state of an action, the fault is classified as a local fault. On the other hand, if the fault happens during verification of preconditions of an action, it will be classified as a propagated fault. Propagated faults are normally caused by the side effect of previous actions.

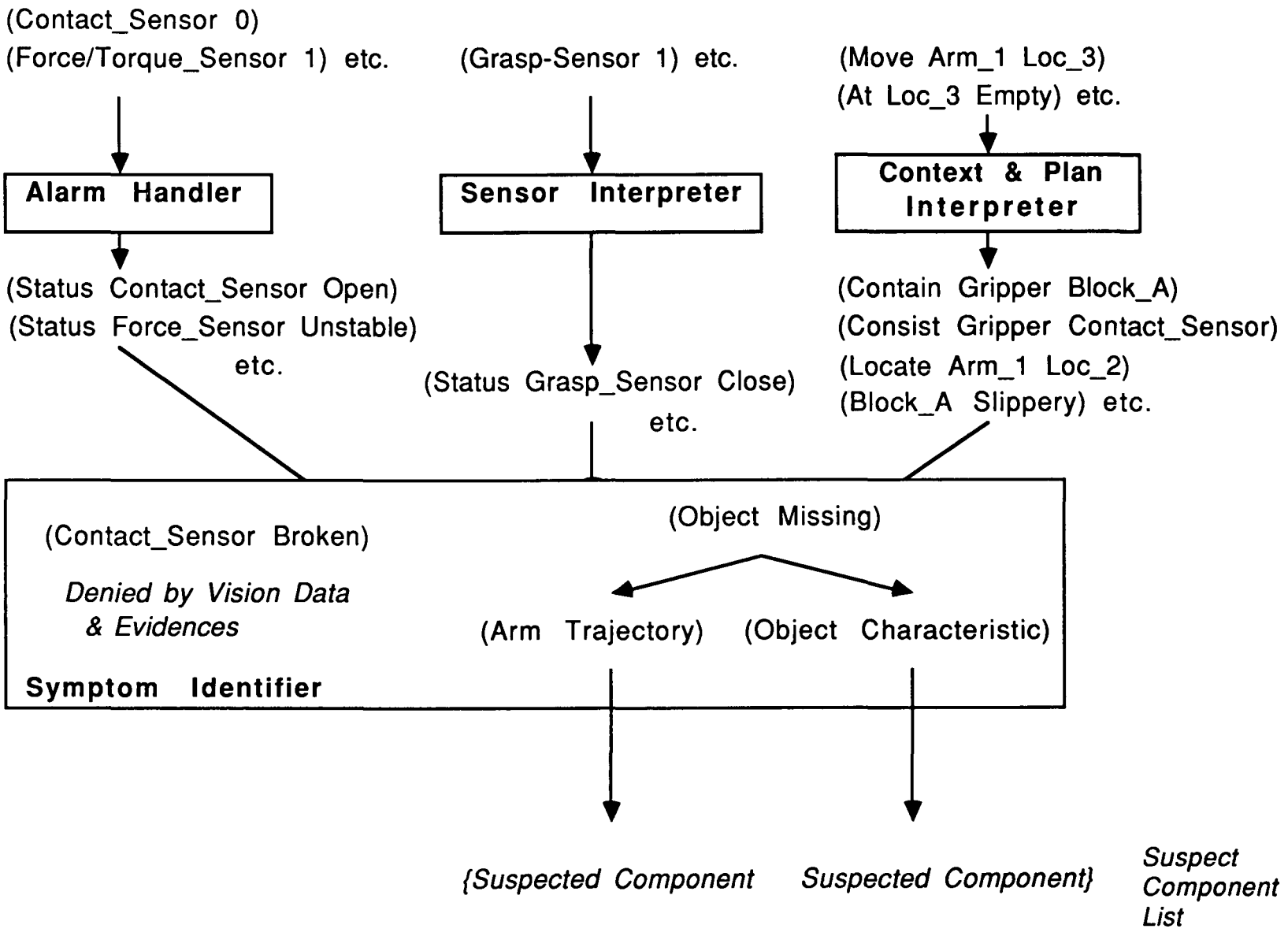

Figure 6. Symptom Identification Process

If the fault is propagated, the suspect list will be given to the Fault Region Identifier to determine the suspected fault region. If the fault is local, the suspect list will be used directly by the Plan Execution Inference Module to diagnose faults on the current action.

4.3.1.5 Fault Region Identifier. The Fault Region Identifier isolates a suspected region where propagated faults in the task plan may have originated. It uses the suspect component list generated by the Symptom Identifier to reason backward from the current action until it identifies the prior action that generated the violated precondition of the current action. Since the validity of this precondition is established at the end of the prior action, and the violation occurred subsequent to the prior action, the prior action becomes the beginning of the suspected region that contains the violation. After the region is isolated, the Fault Region Identifier examines the relationship between the suspected components, the violated precondition, and each action in the suspected region in order to identify actions that are irrelevant to the violation. These irrelevant actions will be eliminated from consideration during fault diagnosis. The Plan Execution Inference Module will examine all of the relevant suspected actions in the region.

4.3.1.6 Plan Execution Inference Module. The objective of the Plan Execution Inference Module is to infer the sources of faults from the fault symptoms and the component model. The module generates hypotheses from the suspect component list and utilizes behavior models to analyze the characteristics of components. It also employs functional models to predict the consequence of actions and compares hypothesized results with alarm conditions to confirm faults.

The inference process is represented as a context-tree data structure. The nodes in the context tree represent state information and the arcs represent actions that caused the state to change. The root node contains the initial state of the environment. The various levels of the tree represent alterations of the environment, and nodes on the same level correspond to multiple hypotheses for consequences of that action. Hence, any vertical path from the root node to an end node represents a series of environmental changes that correspond to state change hypotheses. Figure 7 illustrates an example of the context-tree structure generated by the Plan Execution Inference process.

For local fault diagnosis, the Plan Execution Inference Module generates a root node that contains a set of states representing the initial environment of the current action. The current action normally consists of several plan steps. Each plan step causes a state change to the environment, which is represented as a new level of nodes on the context tree. The Plan Execution Inference Module reasons from the suspect component list and the state information to determine the state change hypotheses on that level (the arcs), and reasons from the behavior and functional models to determine the consequence of the hypotheses (the new level of nodes). When any consequence conflicts with the alarm environment, that hypothesis will be eliminated from further consideration. Hypothesis generation and plan simulation for each plan step proceed in a breadth-first manner. However, when the state information in any hypothesized path matches the alarm

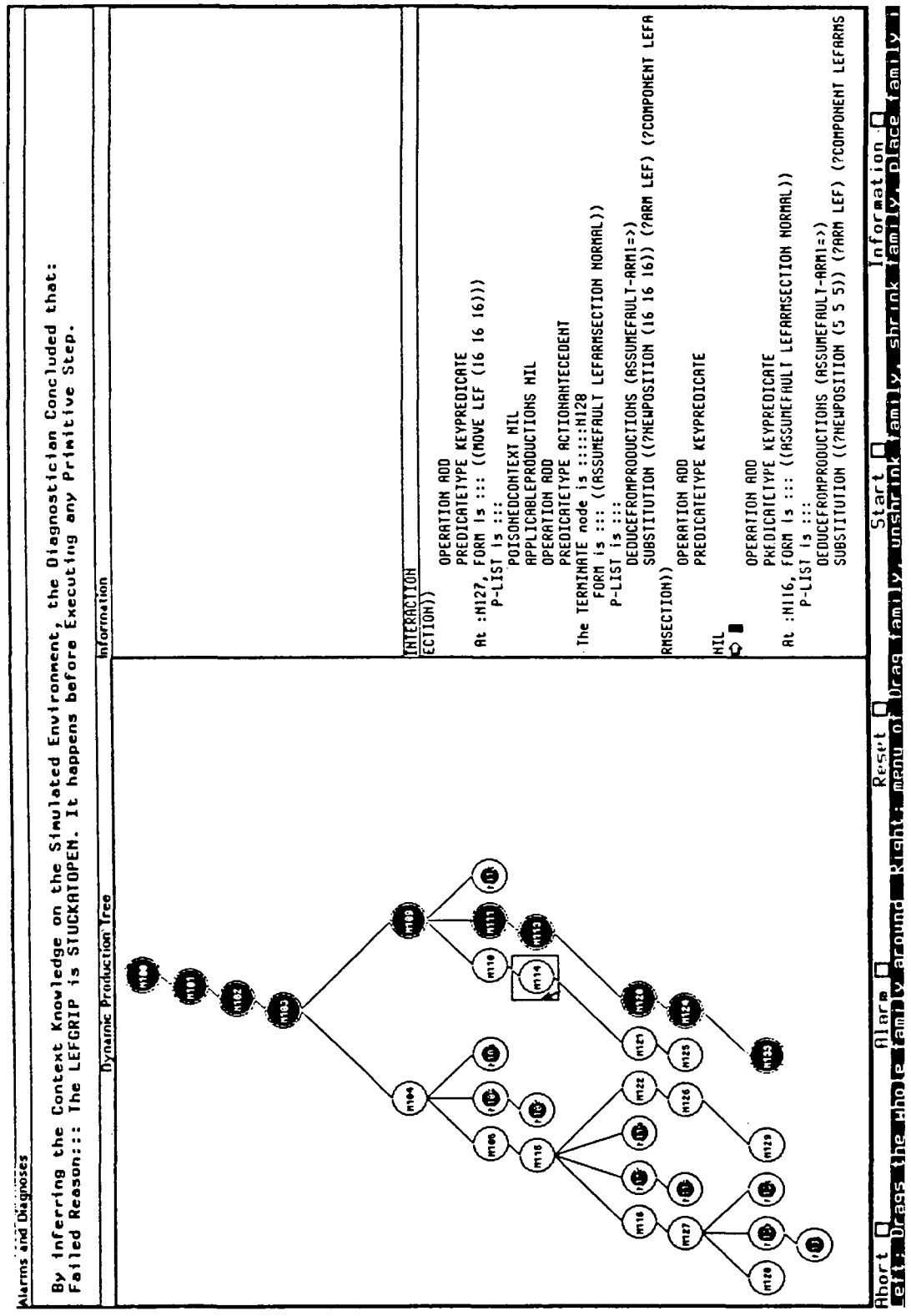

Figure 7. Example of the Plan Execution Inference Process

environment, the Inference module will begin a depth-first diagnosis along that hypothesized path. If the environment in the hypothesized path is consistent with the alarm environment at the point where the malfunction occurred, the TDS will report the abnormal hypotheses along that path as the sources of the various faults, and provide this information for replanning purposes. Otherwise, if the hypothesized path environment does not match the alarm environment where the malfunction occurred, the TDS will backtrack and diagnose other hypothesized paths.

For propagated fault diagnosis, the Plan Execution Inference Module evaluates each action in the suspected region. Relevant actions are analyzed in detail similar to the local fault diagnostic process. Irrelevant actions are not examined at the plan step level (they are assumed to be normal), but exist in the context tree to preserve the coherency of plan execution simulation. This process will continue on all actions until the TDS simulates an environment that is identical to the alarm environment and identifies the propagated faults.

SECTION 5

CONCLUSION

The Jet Propulsion Laboratory's Telerobot plan-execution Diagnostic System demonstrates an effective event-driven approach to the problem of fault diagnosis in uncertain domains. The use of environmental knowledge and plan knowledge, with respect to a given action, supplements the TDS model-based inferencing technique, thus enhancing diagnostic capability for plan and hardware faults in various scenarios. The plan-execution inference process allows the system to diagnose multiple faults at multiple levels of abstraction. Also, because of the generality of the Telerobot Diagnostic System, it can be applied to other autonomous robot domains as well. However, the system has the following drawbacks, which need to be investigated in the ongoing research. First, the TDS may fail in the absence of a complete model of the behavior and function of the actions. Also, inherent uncertainties in the environment that are unknown to the TDS may cause the system to fail.

A preliminary version of the Telerobot Diagnostic System has been successfully tested in simple scenarios. Efforts are continuing on further refinement of the system for diagnosis in more complex environments. Future research will investigate the feasibility of using a similar approach to perform task planning in addition to fault diagnosis. The process of combining knowledge, extracted from multiple control levels, to increase the effectiveness of the system will be studied, along with an investigation of how knowledge used during the planning phase can be exploited by the diagnostic process to eliminate redundant reasoning on irrelevant issues.

SECTION 6

ACKNOWLEDGMENTS

The authors wish to thank Mark James for his support on the use of the "TELESIS" expert system tool. Additionally, the authors are grateful to Jim White and Rajiv Desai for their criticisms and comments. We are very grateful to Henry Stone, Anatoly Lokshin, Bob Balaram, John Beahan, Samad Hayati and Kam-Sing Tso for providing operational information and demonstrations on the Telerobot.

REFERENCES

1. Davis, R., "Diagnosing via Causal Reasoning: Paths of Interaction and the Locality Principle," *Proceedings of the National Conference on Artificial Intelligence*, August 1983, Washington, D.C., pp. 88-94.

2. Genesereth, M.R., "Diagnosis Using Hierarchical Design Models," *Proceedings of the National Conference on Artificial Intelligence*, August 1982, Pittsburgh, PA, pp. 278-283.

3. Genesereth, M.R., "The Use of Design Descriptions in Automated Diagnosis," *Artificial Intelligence*, 24, 1984, pp. 411-436.

4. Georgeff, M.P., and Lansky, A.L., "A System for Reasoning in Dynamic Domains Fault Diagnosis on the Space Shuttle," *SRI International Technical Note 375*, January 1986, Artificial Intelligence Center, SRI International.

5. Kuipers, B., "Commonsense Reasoning About Causality: Deriving Behavior from Structure," *Artificial Intelligence* 24, 1984, pp. 169-203.

6. Larner, D.L., "A Recursive Expert Troubleshooting System Utilizing General and Specific Knowledge," Second Conference on Artificial Intelligence Applications, December 1985, Miami Beach, FL, pp. 34-41.

7. Gini, M., and Smith, R., "Monitoring Robot Actions for Error Detection and Recovery," *Proceedings of the Workshop on Space Telerobotics*, July 1987, Pasadena, CA, Volume III, pp. 67.

8. Gini, M., Doshi, S., Garber, M., Smith, R., and Zualkernan, I., *Symbolic Reasoning as a Basis for Automatic Error Recovery in Robotics*, Technical Report TR 85-24, Computer Science Department, University of Minnesota, August 1985.

9. Gini, M., and Gini, G., "Towards Automatic Error Recovery in Robot Programs," *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, August 1983, Washington, D.C., pp. 821-823.

10. Krishnamurthi, M., Mayer, R.J., and Friel, P.G., "Robot Fault Diagnosis Using Deep and Shallow Modeling Approaches," *Robotics and Expert Systems*, June 1986, NASA/Johnson Space Center, pp. 269-278.

11. Balaram, B., "A Run-Time Architecture for the JPL Telerobot," *Proceedings of the IEEE International Conference on Robotics and Automation*, 1987, Raleigh, NC, p. 310.

12. Adams, T.L., Orr, G.L., and Tollander, C.J., "An Artificial Intelligence Approach to Coordinated Fault Diagnosis, Control and Planning for the Space Station Electrical Power System," *20th Intersociety Energy Conversion Engineering Conference*, August 1985, Miami Beach, FL, pp. 472-478.

13. Adams, T.L., "Model-Based Reasoning for Automated Fault Diagnosis and Recovery Planning in Space Power Systems," *Proceedings of the AIAA Space System Technology Conference*, June 9-12, 1986, San Diego, CA, pp. 69-78.

14. Pan, Y.C., "Qualitative Reasoning with Deep Level Mechanism Models For Diagnoses of Mechanism Failures," *First International Conference on the Applications of Artificial Intelligence,* December 1984, Denver, CO., pp. 295-301.

15. Doyle, R., Atkinson, D.J., and Doshi, R.S., "Generating Perception Requests and Expectations to Verify the Execution of Plans," *Proceedings of the Fifth National Conference on Artificial Intelligence,* August 1986, Philadelphia, PA, pp. 81-88.

16. James, M., "TELESIS: Tool Environment and Language for Expert System Implementation and Synthesis," unpublished document.

17. James, M., "TELESIS: User's Manual," unpublished document.

18. Doyle, R., Sellers, S.M., and Atkinson, D.J., "Predictive Monitoring Based on Causal Simulation," unpublished document.

TECHNICAL REPORT STANDARD TITLE PAGE

1. Report No. JPL 88-14	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle Diagnosing Faults in Autonomous Robot Plan Execution		5. Report Date March 1, 1988 6. Performing Organization Code
7. Author(s) R. Lam, R. Doshi, D. Atkinson, D. Lawson		8. Performing Organization Report No.
9. Performing Organization Name and Address JET PROPULSION LABORATORY California Institute of Technology 4800 Oak Grove Drive Pasadena, California 91109		10. Work Unit No. 11. Contract or Grant No. NAS7-918 13. Type of Report and Period Covered JPL Publication
12. Sponsoring Agency Name and Address NATIONAL AERONAUTICS AND SPACE ADMINISTRATION Washington, D.C. 20546		14. Sponsoring Agency Code RE 159 BK-549-02-31-02-00

15. Supplementary Notes

16. Abstract

A major requirement for an autonomous robot is the capability to diagnose faults during plan execution in an uncertain environment. Many diagnostic researches concentrate only on hardware failures within an autonomous robot. Taking a different approach, this publication describes the implementation of a Telerobot Diagnostic System that addresses, in addition to hardware failures, failures caused by unexpected event changes in the environment or failures due to plan errors. One unique feature of the system is the utilization of task-plan knowledge and context information to deduce fault symptoms. This forward deduction provides valuable information on past activities and the current expectations of a robotic event, both of which can guide the plan-execution inference process. The inference process adopts a model-based technique to recreate the plan-execution process and to confirm fault-source hypotheses. This technique allows the system to diagnose multiple faults due to either unexpected plan failures or hardware errors. This research initiates a major effort to investigate relationships between hardware faults and plan errors, relationships that have not been addressed in the past. The results of this research will provide a clear understanding of how to generate a better task planner for an autonomous robot and how to recover the robot from faults in a critical environment.

17. Key Words (Selected by Author(s))	18. Distribution Statement
Computer Programming and Software Cybernetics Systems Analysis	Unclassified - Unlimited

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	vi + 28	

HOW TO FILL OUT THE TECHNICAL REPORT STANDARD TITLE PAGE

Make items 1, 4, 5, 9, 12, and 13 agree with the corresponding information on the report cover. Use all capital letters for title (item 4). Leave items 2, 6, and 14 blank. Complete the remaining items as follows:

3. Recipient's Catalog No. Reserved for use by report recipients.

7. Author(s). Include corresponding information from the report cover. In addition, list the affiliation of an author if it differs from that of the performing organization.

8. Performing Organization Report No. Insert if performing organization wishes to assign this number.

10. Work Unit No. Use the agency-wide code (for example, 923-50-10-06-72), which uniquely identifies the work unit under which the work was authorized. Non-NASA performing organizations will leave this blank.

11. Insert the number of the contract or grant under which the report was prepared.

15. Supplementary Notes. Enter information not included elsewhere but useful, such as: Prepared in cooperation with... Translation of (or by)... Presented at conference of... To be published in...

16. Abstract. Include a brief (not to exceed 200 words) factual summary of the most significant information contained in the report. If possible, the abstract of a classified report should be unclassified. If the report contains a significant bibliography or literature survey, mention it here.

17. Key Words. Insert terms or short phrases selected by the author that identify the principal subjects covered in the report, and that are sufficiently specific and precise to be used for cataloging.

18. Distribution Statement. Enter one of the authorized statements used to denote releasability to the public or a limitation on dissemination for reasons other than security of defense information. Authorized statements are "Unclassified—Unlimited," "U. S. Government and Contractors only," "U. S. Government Agencies only," and "NASA and NASA Contractors only."

19. Security Classification (of report). NOTE: Reports carrying a security classification will require additional markings giving security and downgrading information as specified by the Security Requirements Checklist and the DoD Industrial Security Manual (DoD 5220.22-M).

20. Security Classification (of this page). NOTE: Because this page may be used in preparing announcements, bibliographies, and data banks, it should be unclassified if possible. If a classification is required, indicate separately the classification of the title and the abstract by following these items with either "(U)" for unclassified, or "(C)" or "(S)" as applicable for classified items.

21. No. of Pages. Insert the number of pages.

22. Price. Insert the price set by the Clearinghouse for Federal Scientific and Technical Information or the Government Printing Office, if known.