KNOWLEDGE REPRESENTATION AND REASONING FOR FAULT IDENTIFICATION IN A SPACE ROBOT ARM

Luigi Portinale
Dipartimento di Scienze e Tecnologie Avanzate, Università del Piemonte Orientale,
C.so Borsalino 54, 15100 Alessandria (ITALY)
phone: +39 0131-283815, fax: +39 0131-254410, e-mail: portinal@al.unipmn.it

Pietro Torasso, Gianluca Correndo
Dipartimento di Informatica, Università di Torino
C.so Svizzera 185, 10149 Torino (ITALY)
phone: +39 011-6706711, fax: +39 011-751603, e-mail: torasso@di.unito.it

ABSTRACT

The construction of diagnostic systems able to manage tasks like fault detection, fault localization or fault identification in autonomous spacecraft is currently considered a big challenge for Artificial Intelligence techniques. In the present paper we report on the work done inside a project sponsored by ASI (the Italian Space Agency) aimed at building an intelligent multi-agent system for the control and supervision of the SPIDER Manipulation System with some form of interaction with the human operator. In particular, we will discuss knowledge representation and reasoning issues related to the construction of a model-based diagnostic component which has to co-operate with other modules of the system. An in-depth analysis of FMECA documents has guided the modeling of the domain knowledge on the faulty behavior of SPIDER. In this paper, problems related to the choice of the suitable modeling formalism involving abstractions and interaction among components are formally addressed, as well as the definition of innovative diagnostic strategies able to deal with the huge number of possible diagnoses that may arise during the diagnostic activity. The paper reports some preliminary results of the prototypical version of the diagnostic module on simulated data.

1 INTRODUCTION

In recent years, a lot of attention has been paid to investigate perspectives and technical problems involved in the supervision of autonomous spacecraft [10]. In particular, the construction of diagnostic systems able to manage tasks like fault detection, fault localization or fault identification in such spacecraft is currently considered a big challenge for Artificial Intelligence techniques [17]. Indeed, within the mission of Deep Space 1 experiments are scheduled for testing the functionality of Remote Agent which include planning and scheduling of mission activities as well as fault detection and reconfiguration [4]. More information about the actual experiments is reported at the web site http://rax.arc.nasa.gov.

Of course, providing such an autonomy is consequent to an activity aimed at studying and proposing the most suitable formalisms and techniques for solving the above problems. These problems remain very difficult, even when we take into consideration "interactive" autonomy, where some form of interaction with human operator (either on ground or on board) is required.

In the last decade several approaches based on Model-Based Reasoning techniques have been proposed for diagnostic problem solving [11]: many approaches exploit some form of behavioral models of the system under examination (see for example [7]) for detecting and identifying faults. A typical problem in such a case involves how to identify relevant components of the system and their behavior (correct and/or faulty) both in terms of behavioral modes (diagnostic hypotheses) and their observable consequences (symptoms). Another relevant problem is the development of appropriate diagnostic strategies, since it is well known that in the worst case, model-based diagnosis from a computational point of view [1].

In the present paper we report on the work done inside the project *An Intelligent System for Supervising Autonomous Space Robots* sponsored by ASI (the Italian Space Agency), aimed at building an intelligent system for the control and supervision of a spacecraft. The chosen testbed of the project is the robot arm of the SPIDER Manipulation System (SMS) developed by ASI and TecnoSpazio [12]. While other partners of the project are responsible for planning and scheduling [2], image and sensory interpretation [3], interaction with human operator and supervision [9], our group is responsible for developing a diagnostic component able to identify failures and malfunctions of the SPIDER arm. While the diagnostic agent should be autonomous in deriving possible diagnoses given a set of observations about the behavior of the

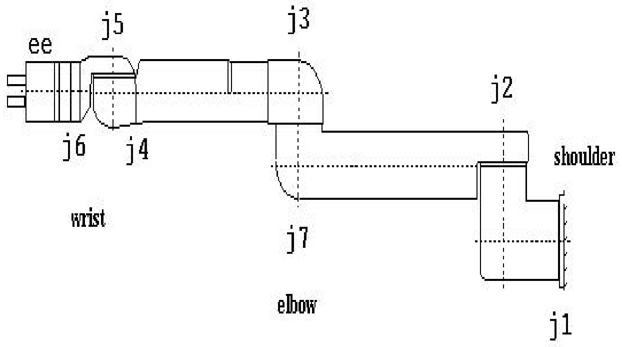


Figure 1: The Spider Arm

robotic arm, it has also to interact with the human operator, by providing him/her with the most plausible diagnoses explaining the observations.

It is worth noting that the diagnostic agent has the goal of detecting, localizing and identifying faults concerning the arm and not of dealing with failure of the plan or activity it currently performs (i.e. plan failures). For this reason, the diagnostic component we have devised is based on the information extracted from system level FMECA (Failure Mode Effects and Criticality Analysis) documents [5], where possible fault modes of SPIDER and their effects are considered and analyzed.

The paper is organized as follows: in section 2 the features of the SPIDER arm are briefly outlined, in section 3 the model of the arm is discussed and in section 4 the diagnostic strategies based on such a model are defined, while in section 5 the notion of abstract model for diagnosing particular faults is introduced; finally section 6 briefly reports about system implementation.

2 OVERVIEW OF THE SPIDER MANIPULATION SYSTEM

The SPIDER Manipulation System (SMS) is a space robot system whose main component is the SPIDER robot arm, a 7 d.o.f (degrees of freedom) robot arm developed by ASI and TecnoSpazio and designed for external space station environment, equipped with a Force/Torque sensor and two sensorized fingers on the end effector. The SMS has been completed and tested in 1998 and its use is expected in some planned missions. It can perform the following tasks: installation and removal of small payload containers on exposure attachment ports, handling of small payloads for scientific and technological investigations, close-up visual inspection of payload units through a camera installed on the arm. All the above tasks can be monitored from ground and no crew intervention is required during nominal operations.

A schematic view of the SPIDER arm is reported in figure 1; the 7 d.o.f. of the arm are obtained by means of 7 joints (j_1 through j_7 in fig. 1), each one equipped with a position and a thermal sensor. The same kind of sensors are also present at the end effector (*ee*); moreover, a Force/Torque sensor is positioned in correspondence of the *ee* to monitor the force applied by the jaws (fingers) on the possible payload. From FMECA documents, the following components (system level items) can be identified; for each joint: the *mechanical part of the joint*, the *electrical part of the joint*, the *harness and electrical connectors for the joint* and the *joint engine*; in addition, the SMS consists of the following components: the *end effector*, the *end effector engine*, the *power supply* and several electronic components, namely the *drive electronics*, the *control electronics* and the *acquisition electronics*. Sensors are not included in the system level items; from the diagnostic point of view this means that we do not model sensor failures (i.e. sensors are reliable components).

Each one of the above components may exhibit, in addition to the normal or nominal behavior, different potential faults whose consequences in terms of observable parameters (usually sensor reports) are described in the FMECA sheets [5]. We explicitly distinguished between observables coming from sensors (*sensorial manifestations*) and other observables that may require more complex operations for getting their value. In the next section we will discuss how the nominal and faulty behavior has been modeled in the diagnostic system we have developed.

3 MODELING THE SPIDER BEHAVIOR

Following the tradition of the model-based approach to diagnosis, we have devised a component oriented model for SPIDER, where each identified component (i.e. the system level items identified from FMECA documents and described in section 2) can assume different behavioral model, one *normal* mode and several *fault* modes [8]. In particular, we adopted a logical approach where each component is identified with a particular predicate, whose admissible values are the behavioral modes of the represented component. For example, the fact that the end effector is in the nominal behavior is modeled by means of the ground atom $ee(normal)$. A set of predicates has also been devised for representing *observable parameters* as well as *contextual information*. Indeed, in the SPIDER domain, the observed behavior of the arm is strictly related to contextual information, usually represented by the particular type of command the arm is executing. For this reason, in the ontology of our model we explicitly consider the presence of *context* predicates. Relationships between components, contextual situations and observable parameters are modeled by means of *definite clauses*[1]. The choice of definite clauses allows us to resort to a class of models deeply investigated and widely adopted, which are not too complex from

[1]Actually, we model observable parameters in a slightly more complex way, by associating with them internal system states having observable manifestations. For the sake of simplicity, in the present paper we will make the simplifying assumption that components and contexts are directly associated with observable parameters. This just simplifies the ontology and it is no restrictive at all.

a computational point of view, while preserving a significant modeling power.

The development of a model containing knowledge on both the nominal and the faulty behavior of the SPIDER arm gives us the possibility of solving three different kinds of problems: *fault detection*, *fault localization*, *fault identification*. It is worth noting that the above problems can be solved as far as two assumptions hold: the model is reasonably complete and the discretization of observable measurements into qualitative values captures interesting behavior.

As concerns the completeness of the model, we are confident that the set of behavioral modes for each component is almost complete, since FMECA documents contain a detailed list of faults for each component. More critical is the assumption that the relations between behavioral modes of the components and the observables are accurate. In fact, FMECA does not provide a complete list of the interactions between different faults affecting the same observable parameter, so we had to make some extra-assumptions to model these interactions.

Example 1. Two components of the SPIDER manipulator system are the *control electronics* (ce) and the *drive electronics* (de); from FMECA documents it is possible to determine that both components influence the *current level* ($curl$) at the drive electronics level. In particular, there are two faults of the above components (fault identified as $sp21060$ for de and fault $sp23060$ for ce respectively) that determine an overcrossing of the current limit. Since no other information about $curl$ is present in the FMECA sheets, a first choice we have made has been to assume two possible values for the parameter $curl : \{normal, high\}$ to model the fact that an overcrossing of the current limit will result in a high current level. Since both ce and de components have several behavioral modes (one normal mode where no fault is exhibited, 6 fault modes for de and 5 fault modes for ce), a complete model relating them to the parameter $curl$ should take into account all the possible interactions between behavioral modes (in this case 42 interactions).

This kind of problem is not peculiar of this part of the model, but it arises every time that more than one component influences a given parameter. To overcome this model complication, we have chosen to adopt a modeling assumption borrowed from *Bayesian Network* theory: the *noisy-max interaction* [15]. Using the noisy-max interaction, only the influence of each single component (and not of every combination of components) on the given parameter has to be specified: the assumption needed in order to apply noisy-max is that the admissible values of the involved parameter have to be ordered. Given a particular instantiation of the set of components influencing the parameter, the value assumed by the latter is the maximum among the values that are determined by each single component. Returning to our example, we can order the values of the parameters $curl$ as $normal < high$. Let now $\{normal, sp21020, sp21030, sp21120, sp21140,$ $sp21160, sp21060\}$ be the behavioral modes (one normal and 6 faulty) of the de component and $\{normal, sp23010, sp23020, sp23140, sp23160, sp23060\}$ be those of the ce component; we can model single-component interactions as follows: for the drive electronics

$de(normal) \rightarrow curl(normal)$
$de(sp21020) \rightarrow curl(normal)$
$de(sp21030) \rightarrow curl(normal)$
$de(sp21120) \rightarrow curl(normal)$
$de(sp21140) \rightarrow curl(normal)$
$de(sp21160) \rightarrow curl(normal)$
$de(sp21060) \rightarrow curl(high)$

and for the control electronics

$ce(normal) \rightarrow curl(normal)$
$ce(sp23010) \rightarrow curl(normal)$
$ce(sp23020) \rightarrow curl(normal)$
$ce(sp23140) \rightarrow curl(normal)$
$ce(sp23160) \rightarrow curl(normal)$
$ce(sp23060) \rightarrow curl(high)$

By adopting noisy-max, we implicitly model all the interactions between de and ce on $curl$; for instance the combination $\{de(sp21120), ce(sp23160)\}$ will cause $curl(normal)$ since both $de(sp21120)$ and $ce(sp23160)$ determine this value, while the combination $\{de(sp21120), ce(sp23060)\}$ will cause $curl(high)$ since $de(sp21120)$ would determine $curl(normal)$, $ce(sp23060)$ would determine $curl(high)$ and $normal < high$.

As concerns the discretization of observable parameters, it is common for most artifacts to have a range of nominal values and to have range of slight and large deviations. When a parameter exceeds the range of nominal values, some form of alarm arises and the human (or the software) agent has to start some activity, in order to figure out whether something unexpected is actually occurring in the system. In case of SPIDER, fault detection is more complex since it does not depend only on the fact that one (or more than one) parameter has a value outside the nominal range. In particular we recognize that a fault exists (fault detection) if there is discrepancy between what we expect to observe in case all components are OK (that is each component has the normal mode) and what we actually observed. Such expectations depend not only on the mode of the components, but also on the values of the contextual information.

4 DIAGNOSTIC STRATEGIES

As said above, the diagnostic agent has to be able to perform both fault detection and fault identification. By taking into consideration that the model of SPIDER is (almost) complete and the domain theory contains also rules for describing the correct behavior, the first inference step performed by our diagnostic system is a prediction step under the assumption that all components are OK. If there is at least one discrepancy between the predictions and the observations, other reasoning steps have to be activated. It is worth noting that this prediction step is computationally cheap, since it involves deductive closure on a definite clause theory (see [13]).

If a discrepancy exists, the fault identification step has to be activated and the reasoning mechanisms involved in such a step are by far more expensive from a computational point of view. In component-oriented model-based diagnosis, the notion of partial diagnosis (and of kernel diagnosis) has received significant attention [7] in order to concisely characterize a (potentially large) set of diagnoses. The basic idea behind kernel diagnosis is to include in the diagnosis just the assignment to the components for which the observation imposes some constraints; in other words, a component is not mentioned in the diagnosis if all the behavioral modes of the component are consistent with the observations.

The use of kernel diagnosis does not guarantee at all that the number of kernel diagnoses is small for a given diagnostic problem. In fact, this may happen in the SPIDER domain where for some diagnostic problems, hundred of kernel diagnoses may be generated. This has also the side effect that the time for generating kernel diagnoses can be too high. We noticed this problem by performing some experiments with a first prototype of the diagnostic system; as a consequence we had to move to alternative forms for representing diagnoses in a concise way and for generating them by taking into consideration computation time.

We generalized the notion of abductive diagnosis presented in [6], by introducing the notion of scenario. Let c be a predicate representing the component c having D_c possible behavioral mode (thus the ground instance $c(m)$ represent the component c being in mode m); technically, a *scenario* is represented as a particular kind of *conjunctive normal form* (CNF) formula, where each conjunct is a disjunction of (at most D_c) ground instances of the same predicate c (see [14] for more details).

The basic idea consists in building a representation centered around components, able to capture a number of different diagnoses that involve the same set of components, but which assign different behavioral mode to each mentioned component.

Example 2. Consider the model of example 1; let us suppose that an overcrossing of the current limit at de level is reported (i.e. we observe $curl(high)$). By considering ce and de components, the two following scenarios can represent a set of 14 diagnoses:

$de(sp21060) \wedge (ce(normal) \vee ce(sp23010) \vee ce(sp23020) \vee ce(sp23140) \vee ce(sp23160) \vee ce(sp23060))$

$ce(sp23060) \wedge (de(normal) \vee de(sp21020) \vee de(sp21030) \vee de(sp21120) \vee de(sp21140) \vee de(sp21160)) \vee de(sp21060))$

Indeed, it is easy to verify that there are 14 different conjunctions of ground atoms of the type $ce(a) \wedge de(b)$ represented by the two above scenarios.

The introduction of a representation based on the notion of scenario has two advantages:

- it reduces the number of diagnoses to be presented to the human operator, so that we reduce the information overflow when the diagnostic problem under examination has a very large space of solutions;
- it provides information about fault localization, since the results are centered around components so that the different faulty modes assigned to a component are represented in just one structure; this indeterminacy in assigning a unique faulty behavior to that component is then made explicit.

The introduction of the notion of scenario does not mean that the solution of a diagnostic problem is unique, as it was apparent from the example above. Since the notion of scenario is more general than the one of diagnosis, we had to define preference criteria for ranking different scenarios. In model-based diagnosis, a number of criteria have been used for ranking solutions: minimal cardinality, minimality, kernel diagnosis, probabilistic measures. There is a large variety of different preference criteria that can be defined in order to rank scenarios: in particular, in some situations one could prefer quite specific scenarios (for most of the components the scenario indicates just a single behavioral mode), in order to reduce the effort for further discrimination; in other cases one could prefer just to look at faulty components without paying too much attention to the set of specific faults possible for that particular component.

In [14] we have defined a set of preference criteria on scenarios, based on the notion of *minimum description length* or MDL [16], a criterion widely used in Machine Learning for ranking alternative descriptions of a learned concept. The basic idea is to consider a suitable encoding of a scenario and to prefer scenarios having minimum coding length. In particular, useful results have been obtained by considering an encoding where unconstrained components (i.e. components for which every behavioral mode is still possible within the given scenario) are not weighted, and constrained components are weighted proportionally to the number of possible behavioral modes within the scenario and to the prior probability of such modes (see [14] for a more detailed discussion). The adoption of a preference criterion based on MDL is quite relevant, because it can be used not only for ranking scenarios at the end of the fault identification step (that is, for deciding which are the best ones to be presented to the human operator), but also to guide the search process to generate just the most preferred ones.

Since the diagnostic process is in general quite expensive from a computational point of view, the ability of reducing the search space is very important from a practical point of view, even if it cannot guarantee the tractability of all the diagnostic problems.

Let OBS be the set of observable parameters (manifestations or symptoms) to be explained in the current case; the diagnostic search strategy is outlined in figure 2. Some comments are worthwhile:

- the diagnostic strategy is activated by invoking `cover(initial_scenario,OBS)` where `initial_scenario` is the trivial scenario where for each components all the behavioral modes (the normal one as well the faulty ones) are considered admissible and OBS represents the set of manifestations to be explained in the specific case under examination.
- in order to solve a diagnostic problem the inference mechanism considers one observation at

```
cover(current, to_be_expl)
IF to_be_expl = empty_set
  THEN
   BEGIN
    print("found solution:", current);
    IF no_more_solution_needed THEN EXIT
   END
  ELSE
   BEGIN
   O := first(to_be_expl);
   to_be_expl := to_be_expl - {O};
   expl(O):= explanations of O;
   new_scen := empty_set;
   FOREACH S in expl(O)
     BEGIN
      expl(O) := expl(O) - {S};
      new_scen := union(new_scen,merge(S,current))
     END
   new_scen := heuristic_sort(new_scen);
   FOREACH(S in new_scen) cover(S, to_be_expl)
   END
```

Figure 2: Skecth of the Diagnostic Strategy

each step and for the chosen observation the inference mechanism determines all the possible ways such an observation can be explained (in terms of abductive reasoning). Instead of representing the alternative explanations in terms of partial diagnoses (usually a very large set of diagnoses) these explanations are summarized in a (relatively) small number of alternative scenarios.

- The resulting scenarios (i.e.`expl(O)`) are merged with the scenario under examination: the `merge` operation combines the restrictions of the possible behavioral modes for a component determined so far (represented by `current`) with the restriction derived by explaining the current manifestation `O` to be processed (represented by scenario `S`). It is possible that an inconsistency arises, if the behavioral modes of a given component consistent with the manifestations considered so far are not within the set of possible assignments of behavioral modes necessary for explaining manifestation `O`. When an inconsistency arises the scenario is disregarded (i.e. it is not included into `new_scen`).
- the set of scenarios generated by considering manifestations up to `O` are sorted according to the chosen preference criterion (a suitable adaptation of the MDL principle) and the diagnostic process continues by considering remaining observations (the ones not yet considered). As soon as all the manifestations have been considered, the resulting scenario is a solution to the diagnostic problem. The search continues is one is interested to consider alternative solutions to the specific diagnostic problem.

It is clear that the search strategy is essentially a hill-climbing technique and therefore it does not guarantee that solutions are generated in order of preference criterion. However, in the specific case of SPIDER, the adoption of the above strategy resulted to be very satisfactory as shown by results summarized on table 1. In this table we report the average results concerning two batches of experiments consisting of 250 simulated cases each. Since at this stage of the project we do not have access to real data, cases have been generated by means of a *simulator* we have developed on the behavioral model of the SPIDER arm. Each case is obtained by injecting a particular set of faults and by setting some suitable parameters like for instance the probability of non sensorial predicted manifestations to be part of the actual symptoms of the case. The first line of table 1 concerns a batch with 1 injected fault, while the second line concerns a batch with 2 injected faults. We tested the diagnostic algorithm by setting a time-out of 30 seconds on CPU time (on a Pentium II) and by measuring the following parameters reported in the table: the average expansion factor (EF) representing the percentage of the whole search space (in terms of expanded nodes) that has been visited to find the optimum, the number of times where optimum is the first solution (O1), the number of times where optimum is in the first 4 solutions (O4), the average distance of the coding length of the first solution with respect to the optimum (DO) normalized in $[0, 1]$ with respect to the maximum value, the percentage of time-outs (TO) occurred in the batch. As we can notice the performance of the algorithm appears to be very good, both in quantitative (e.g. EF) and in qualitative terms (e.g. O1, O4 and DO). In particular, it is worth noting that very often the algorithm is able to get the optimum as a first solution (or at least in the first 4); moreover even when the optimum is not obtained as a first solution, the quality of such a first solution is very high as suggested by reported values on DO. Results reported in table 1 refer to just one particular coding function for scenarios, where the contribution of components that, in the given scenario may assume all admissible modes is not weighted; work in [14] reports similar results also for alternative codings.

EF	**O1**	**O4**	**DO**	**TO**
35.81%	247	249	0.003	0
21.06%	237	247	0.027	0

Table 1: Diagnostic Algorithm: Experimental Results

5 DIAGNOSIS WITH ABSTRACT MODELS

Despite the interesting results obtained by adopting the diagnostic strategy based on the notion of scenario and discussed above, the SPIDER domain has some peculiarities that require the introduction of other reasoning and representation mechanisms in order to supplement the basic mechanisms described above. In order to gave a flavor of the problems to be faced, let us consider the case where all the observable parameters related to the joint positions have a qualitative value indicating a large deviation from the expected one. Each single observation can be explained by the mechanical and/or electrical faults of the joint which the parameter sensor is associated to. Since the abnormal manifestations are related to all joints, all joints have to be assumed faulty (more precisely a huge number of scenarios have to be generated to take into account the possible combination of mechanical and electrical faults of every joint). However, according to FMECA sheets, a fault in the control electronics *ce* may cause deviations from the expected positions for all the arm joints. It is clear that, in order to explain the above manifestations, it is much more preferable to assume a fault in the control electronics rather that to assume that there are many simultaneous concurrent faults, each one related to a single joint.

Even if in principle such a situation can be dealt with just using a preference criterion, we have preferred to approach the problem by explicitly representing the phenomenon. In particular, we have made use of a notion of *abstraction* both at the level of manifestations that at the level of domain knowledge. As concerns manifestations, we have introduced rules for the synthesis of *abstract manifestations* which summarize the behavior of a number of observed manifestations. For example, to deal with the problem introduced above, concerning an abnormal deviation of every joint position, an abstract manifestation $all\_joint\_pos(abnormal)$ has been introduced, with the meaning that the position of every joint of the arm is deviating from its nominal value.

As concerns domain knowledge we have derived an abstract model relating the behavioral modes of the components with abstract manifestations. In this way the abstract model shares some portion of the detailed domain knowledge, but it includes clauses specific for the abstract model (e.g. clauses relating faults of the control electronics *ce* with the abstract manifestation all_joints_pos).

In case of SPIDER, the abstract model is significantly more concise (and simpler) than the detailed model. The diagnostic system is able to work with both the abstract model and the detailed one. The control strategy first tries to activate rules for inferring abstract manifestations. If this inference step succeeds (i.e. at least one abstract manifestation is inferred), the set of observations to be explained is modified by adding the abstract manifestations and by deleting the detailed manifestations subsumed by the abstract one. The fault identification process is activated and the abstract model is used for finding the explanations of the observations. If at least a solution exists (represented by one or more scenarios explaining the manifestations) the process can be considered completed and there is no need of invoking again the fault identification process on the detailed domain theory (unless the user explicitly requires this step). On the contrary, a failure in producing a solution by using the abstract model does not mean a failure in the overall diagnostic process. The diagnostic process is indeed activated for an attempt to explain the set of detailed observations, by using the detailed domain theory.

Example 3. Let us consider again the control electronics component *ce*; among its faults there are 4 faults (namely $sp23010, sp23020, sp23140, sp23160$) that, when present, imply a deviation of each joint position. Let j_pos_i be the predicate representing the position of joint i, with $1 \leq i \leq 7$; the detailed model concerning *ce* and the joint positions will have the following clauses for each one of the 7 joints:

$ce(normal) \rightarrow j\_pos_i(normal)$
$ce(sp23010) \rightarrow j\_pos_i(abnormal)$
$ce(sp23020) \rightarrow j\_pos_i(abnormal)$
$ce(sp23140) \rightarrow j\_pos_i(abnormal)$
$ce(sp23160) \rightarrow j\_pos_i(abnormal)$

Moreover, it follows from FMECA that abnormal positions can result from specific faults (namely $sp11170, sp11150, sp11030$) of the mechanical part of a joint; let j_i be the predicate modeling this component (i.e. the mechanical part of joint i), then the detailed model will also include the following clauses for each joint[2]:

$j_i(normal) \rightarrow j\_pos_i(normal)$
$j_i(sp11170) \rightarrow j\_pos_i(abnormal)$
$j_i(sp11150) \rightarrow j\_pos_i(abnormal)$
$j_i(sp11030) \rightarrow j\_pos_i(abnormal)$

Concerning this part of model, the detailed model will result in a total of 63 clauses.

If we consider now the abstract model, while no difference arises with respect to components j_i, the part relating *ce* and the joint positions can be abstracted by using the abstract manifestations all_joint_pos in the following way:

$ce(normal) \rightarrow all\_joint\_pos(normal)$
$ce(sp23010) \rightarrow all\_joint\_pos(abnormal)$
$ce(sp23020) \rightarrow all\_joint\_pos(abnormal)$
$ce(sp23140) \rightarrow all\_joint\_pos(abnormal)$
$ce(sp23160) \rightarrow all\_joint\_pos(abnormal)$

resulting only in 33 clauses (28 relating j_i with j_pos_i and the 5 above).

Moreover, in case we observe for each joint i the manifestation $j\_pos_i(abnormal)$, we can substitute this set of manifestations by synthesizing it into the abstract manifestation $all\_joint\_pos(abnormal)$ and, by using the abstract model, we will avoid at this level of abstraction the generation of diagnoses involving components j_i. In fact, the diagnostic process will result in the generation of just one scenario:

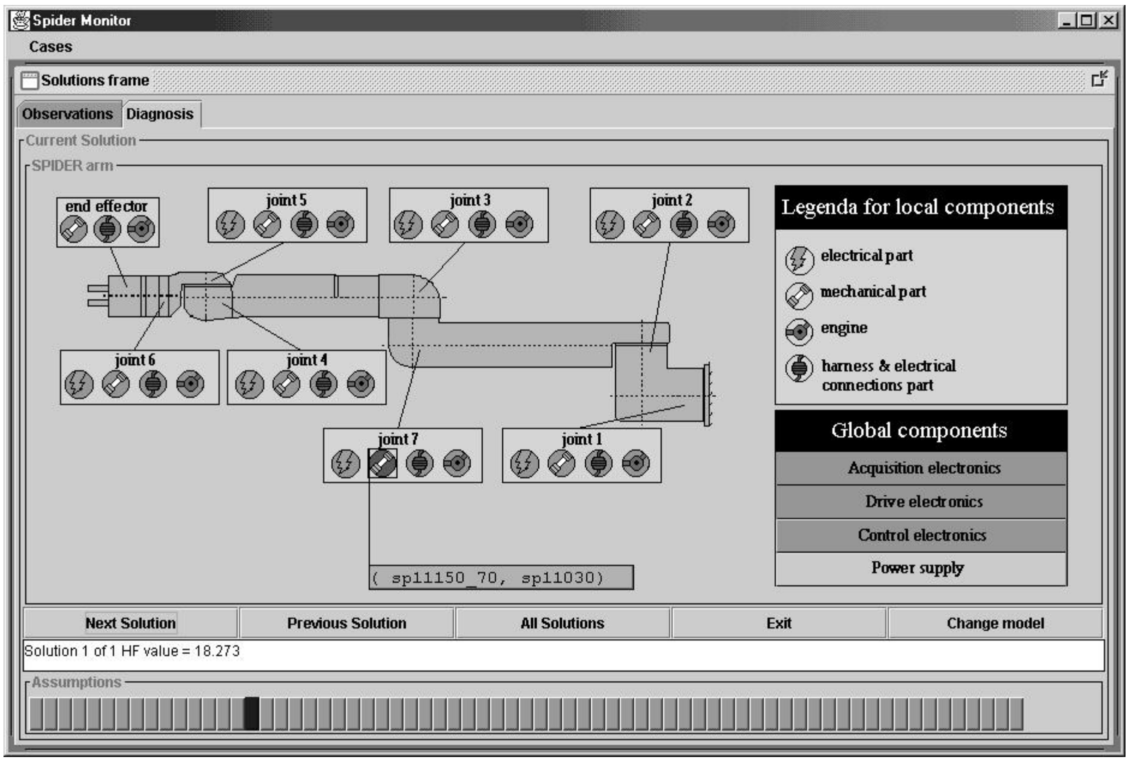
$ce(sp23010) \lor ce(sp23020) \lor ce(sp23140) \lor ce(sp23160)$

where only the control electronics is involved. In this way, working on the abstract model, the set of diagnoses involving the fault of just one component (*ce*) is preferred over diagnoses that, in order to account for the observations, have to hypothesize a fault on each of the 7 arm joints.

6 IMPLEMENTATION

The diagnostic system described in this paper has been implemented as a prototypical system in Java (`jdk1.2`) on a Pentium II architecture running the Windows98 operating system. The system integrates

[2]Actually the situation is even more complex, since the other parts of a joint (for instance the electrical part) exhibit the same behavior and in the current version of the model, the abnormality of a joint position is actually modeled with two different values, representing a small and a large deviation from the nominal value respectively.



```
( sp11150_70, sp11030)
Solution 1 of 1 HF value = 18.273
```

Figure 3: Diagnostic System Interface

the simulator for case generation with the diagnostic problem solver. Cases can be saved into a case library and then loaded for resolution. After a case has been loaded, a window reporting the state of the observed parameters and of contextual information can be displayed; after resolution, results of diagnostic reasoning can also be displayed on a separate window (see figure 3). The system interface allows the user to have a schematic view of the SPIDER arm, where for each joint, all the components (mechanical, electrical, connectors and engine) can be separately considered and the behavioral modes assigned by a given scenario can be displayed. In addition, the status of global components (like electronics units) can be examined by means of a separate set of buttons. The user may then control the generation of diagnoses, by requiring the computation of the next scenario, the computation of all possible scenarios or by changing the model from the abstract to the detailed one.

ACKNOWLEDGMENTS

The research described in this paper has been funded by ASI under grant n. 155/97 and is part of the project "An Intelligent System for Supervising Autonoumous Space Robots". Partners in the project are IP-CNR, IRST-ITC, Universities of Genova, Palermo, Parma, Roma and Torino.

References

[1] T. Bylander, D. Allemang, M. Tanner, and J. Josephson. The computational complexity of abduction. *Artificial Intelligence*, 49(1-3):25–60, 1991.

[2] A. Cesta, P. Riccucci, M. Daniele, P. Traverso, E. Giunchiglia, and M. Piaggio. JERRY: A system for the automatic generation and execution of plans for robotic devices - the case study of the SPIDER arm. In *Proc. 5th Int. Symp. on Artificial Intelligence, Robotics and Automation in Space, iSAIRAS99*, Noordwijk, NL, 1999.

[3] A. Chella, S. Gaglio, D. Guarino, and I. Infantino. An artificial high-level vision agent for the interpretation of the operations of a robotic arm. In *Proc. 5th Int. Symp. on Artificial Intelligence, Robotics and Automation in Space, iSAIRAS99*, Noordwijk, NL, 1999.

[4] S. Chien, D. DeCoste, R. Doyle, and P. Stolorz. Making an impact: Artificial intelligence at the jet propulsion laboratory. *AI Magazine*, 18(1):103–122, 1997.

[5] M. Colomba, P.G. Magnani, P. Rigatelli, E. Re, and V. Venturini. Project P0051: System level FMECA. Technical Report BMAS-SM-TS-038, Tecnospazio SpA, 1997.

[6] L. Console and P. Torasso. A spectrum of logical definitions of model-based diagnosis. *Computational Intelligence*, 7(3):133–141, 1991.

[7] J. de Kleer, A. Mackworth, and R. Reiter. Characterizing diagnoses and systems. *Artificial Intelligence*, 56(2–3):197–222, 1992.

[8] J. de Kleer and B.C. Williams. Diagnosis with behavioral modes. In *Proc. 11th IJCAI*, pages 1324–1330, Detroit, 1989.

[9] A. Dell'Arciprete, A. Finzi, F. Pirri, G. Rossi, and M. Schaerf. A system integrating high and

low level planning of complex tasks with a three-dimensional visualiser. In *Proc. 5th Int. Symp. on Artificial Intelligence, Robotics and Automation in Space, iSAIRAS99*, Noordwijk, NL, 1999.

[10] R. Doyle. The emergence of spacecraft autonomy. In *Proc. AAAI 97*, pages 756–761, Providence, RI, 1997.

[11] W. Hamscher, L. Console, and J. de Kleer. *Readings in Model-Based Diagnosis.* Morgan Kaufmann, 1992.

[12] R. Mugnuolo, S. Di Pippo, P.G. Magnani, and E. Re. The SPIDER manipulation system (SMS). *Robotics and Autonomous Systems*, 23(1-2):79–88, 1998.

[13] L. Portinale and P. Torasso. On the usefulness of re-using diagnostic solutions. In *Proc. 12th European Conf. on AI - ECAI 96*, pages 137–141, Budapest, 1996.

[14] L. Portinale and P. Torasso. Diagnosis as a variable assignment problem: a case study in space robot fault diagnosis. In *Proc. IJCAI 99*, Stockholm, 1999.

[15] M. Pradhan, G. Provan, B. Middleton, and M. Henrion. Knowledge engineering for large belief networks. In *Proc. 10th Conf. on Uncertainty in Artificial Intelligence*, pages 484–490, Seattle, WA, 1994.

[16] J. Rissanen. A universal prior for integers and estimation by minimum description length. *Annals of Statistics*, 11(2):416–431, 1983.

[17] B.C. Williams and P.P. Nayak. A model-based approach to reactive self-configuring systems. In *Proc. AAAI 96*, pages 971–978, Portland, 1996.