Livingstone and the Remote Agent Experiment
Richard Dearden, RIACS/NASA Ames Research Center
James Kurien, Palo Alto Research Center
Peter Robinson, QSS/NASA Ames Research Center

Abstract

Diagnosis is becoming an increasingly important component of NASA missions. In this article we present the Livingstone diagnosis system, which flew on-board the Deep Space One mission. Livingstone is a discrete model-based system that does fault detection, identification, and recovery. We discuss the strengths and weaknesses of this approach, and describe two future applications of diagnosis within NASA, one that is well suited for Livingstone, and another where new techniques must be developed.

Keywords: Diagnosis, model-based systems, hybrid systems.

Introduction

A number of recent NASA space missions, as well as the plans for future missions, have highlighted the importance of on-board autonomy. For example, the Deep Space One spacecraft had an on-board planner and diagnosis system that controlled the spacecraft for part of the mission, accepting high-level goals from ground control, and planning actions to accomplish these goals based on information about the spacecraft’s current state. Even more ambitious autonomy will be required for future planned missions such as the 2003 and 2009 rover missions to Mars, as direct control of spacecraft from Earth is made impractical by distance and the fact that a rover can only communicate with Earth when Earth is above the horizon from its location on Mars, or when a satellite is overhead. One very important aspect of on-board autonomy is fault detection and recovery, and more generally, tracking the true state of a spacecraft. The ability to do this is critical since any plan needs to know what state it is starting in if it wants to be able to ensure its success. If components have failed, the planner needs that information in order to find ways to work around the failure, and the planner may also need to know details of the spacecraft’s current situation to determine the effects of its actions.

This article describes a number of projects carried out at NASA Ames Research Center aimed at providing model-based diagnosis technology for a variety of NASA missions. In the next section we will describe Deep Space One, in which a model-based diagnosis system was used in space for the first time. Deep Space One was an experimental spacecraft designed to demonstrate a number of technologies in space. The performance of the on-board diagnosis system, known as Livingstone, has generated a lot of interest from other missions, and it seems certain that similar systems will be a feature of many future NASA spacecraft. Following our brief description of Deep Space One, we present the design and capabilities of Livingstone, and the lessons we have learned from its deployment on Deep Space One and elsewhere. To finish the paper we look at two future applications of diagnosis in NASA, one for which we expect Livingstone to be well
suited, and another that exposes some of the weaknesses we have found with Livingstone, and will require other techniques to be developed.

**Deep Space One and the Remote Agent Experiment**
Remote Agent is a reusable artificial intelligence (AI) software system designed to allow spacecraft, life support systems, chemical plants, or other complex systems to operate robustly with minimal human supervision, even in the face of hardware failures or unexpected events. In a set of experiments onboard Deep Space One in May 1999, Remote Agent became the first AI system to control a spacecraft. It planned mission activities based on goals provided by human operators and sent commands to various spacecraft subsystems to carry out its plan. This was the first time a spacecraft had diagnosed an onboard failure and re-planned parts of its mission to ensure mission goals were achieved.

The essential characteristics of Remote Agent are that it is *model-based* and that it is *goal directed*. In traditional software programs and expert systems, the programmer decides what the result of a program should be and writes down instructions or rules that attempt to achieve those results. The computer executes the instructions or fires the rules with no knowledge of what the intended result was or how it is achieving it. The term *model-based* refers to the fact that Remote Agent is given a model, or general description, of the behavior and structure of the spacecraft it is controlling. *Goal-directed* refers to the fact that the operator tells the Remote Agent the desired goal. Remote Agent uses its models and reasoning algorithms to find actions directed toward the goal, even if failures or anomalies occur.

To illustrate the impact of these characteristics, we present a few highlights of a Remote Agent (RA) application. In its alpha test, RA was given models of a spacecraft and asked to achieve the goal of maneuvering it into orbit around Saturn, in simulation. Several hours before the maneuver, RA used its models of the spacecraft to develop a plan of action. This plan involved changing direction by firing the engine, but there was insufficient power to fire the engine and operate the science instruments simultaneously. Since obtaining science data is a primary mission goal, RA planned to turn the science instruments back on once the engine had finished firing. At the appropriate time, RA executed its plan. From its models, RA knew that the engine should fire if it had fuel and oxidizer. It opened the valves to start fuel flow and observed that there was zero acceleration. From this, it reasoned that either the engine was not firing or the accelerometer was malfunctioning. Fuel pressure readings indicated that a valve supplying the main engine had not opened, which suggested that the accelerometer was correct. Using its models again, RA found a second path for fuel through a redundant valve and issued commands to redirect the fuel around the stuck valve. It then determined that the engine was firing and the plan completed. If it had not, RA would have looked for other repairs to the system, or would have created a new plan to achieve its goals even with the failures. Though many details have been omitted for clarity, this example shows how Remote Agent uses its model to reason about the effects of a plan and the consequences of any faults.
As shown in Figure 1, Remote Agent integrates three separate technologies: an on-board planner-scheduler, a robust plan execution system, and a model-based fault diagnosis and recovery system called *Livingstone*, which we describe in detail below.

For its first flight test, the Remote Agent team developed models of the New Millennium Program’s Deep Space One spacecraft and integrated the Remote Agent software into Deep Space One’s flight software. Remote Agent and the models completed extensive pre-flight testing and two operational readiness tests, and were then uploaded to the spacecraft. In mid-May 1999, Remote Agent was turned on onboard the spacecraft for a series of flight tests. During these tests, control of the spacecraft was turned over to Remote Agent, which generated plans that achieve high-level mission goals such as accumulation of thrust with the ion propulsion system and imaging of asteroids. It executed its plans by sending commands to the spacecraft, tracking the state of the spacecraft as each command was given. Since a spacecraft anomaly could not be relied upon to occur during the experiments, false sensor data that simulated failures were injected. Remote Agent was required to diagnose the failures and take the correct recovery actions to continue execution of its plan. In one instance, Remote Agent’s ability to command part of the spacecraft was intentionally disabled, forcing Remote Agent to replan its mission without that capability. Remote Agent was also responsible for responding to real anomalies in the parts of the spacecraft it commanded, though none occurred.

**Livingstone**

Livingstone is a model-based discrete controller that infers the current state (mode) of each device making up the system being controlled and recommends actions to reconfigure the system so that it achieves the currently desired configuration goals, if possible. In practice, these configuration goals could be provided by a human or by some automated system such as the Smart Executive (Exec) in Figure 1, which decomposes a high level plan into a series of configuration goals to be achieved. For the sake of the discussion below, we will assume the Exec is providing the configuration goals.

To track the modes of system devices, Livingstone eavesdrops on commands that are sent to the spacecraft hardware by the Exec. As each command is executed, Livingstone receives observations from the spacecraft’s sensors, abstracted by monitors that produce discrete qualitative values from the continuous sensor readings. Livingstone combines these commands and observations with declarative models of the spacecraft components...
to determine the current state of the system, and reports it to the Exec. Figure 2 shows the operation of Livingstone on a simple example in which a switch fails to operate correctly. In the nominal case, Livingstone merely confirms that the commands had the expected effect on spacecraft state. In case of failure, Livingstone diagnoses the failure and the current state of the spacecraft and provides a recovery recommendation. A single set of models and algorithms are exploited for command confirmation, diagnosis and recovery.

**Figure 2. Information Flow in Livingstone**

The capabilities of the Livingstone inference engine can be divided into two parts: mode identification (MI) and mode reconfiguration (MR). MI is responsible for identifying the current operating or failure mode of each component in the spacecraft. Following a component failure, MR is responsible for suggesting reconfiguration actions that restore the spacecraft to a good configuration. MI’s mode inference allows the Exec to reason about the state of the spacecraft in terms of component modes or even high-level capabilities such as “able to produce thrust” rather than in terms of low-level sensor values. MR supports the run-time generation of novel reconfiguration actions to return components to the desired mode or to re-enable these high level capabilities.

Livingstone uses algorithms adapted from model-based diagnosis and concurrent transition systems [11, 12, 22] to provide the above functions. The idea underlying model-based diagnosis is that a combination of component modes is a possible description of the current state of the spacecraft only if the set of models associated with these modes is consistent with the observed sensor values. Following de Kleer and Williams [13], MI uses a conflict directed best-first search to find the most likely combination of component modes consistent with the observations. MR uses the same search to find the least-cost combination of commands that achieve the desired goals. Furthermore, MI and MR use the same system model to perform their function. The combination of a single search algorithm with a single model, and the process of exercising these through multiple uses, contributes significantly to the robustness of the complete system. This methodology is independent of the actual set of available sensors and commands, and does not require that all aspects of the spacecraft state are directly observable, providing an elegant solution to the problem of limited observability.
Model-based diagnosis algorithms have a number of other advantages. First, the search algorithms are sound and complete, providing a guarantee of coverage with respect to the models used, as long as there are no feedback loops in the models. If there are, then additional clauses need be added [23]. Second, the model building methodology is modular, which simplifies model construction and maintenance, and supports reuse. Third, the algorithms extend smoothly to handling multiple faults and recoveries that involve multiple commands. Fourth, the algorithms do not require explicit fault models for each component, although they will exploit them if models are available.

Livingstone extends the basic ideas of model-based diagnosis by modeling each component as a finite state machine, and the whole spacecraft as a set of concurrent, synchronous state machines. Modeling the spacecraft as a concurrent machine allows Livingstone to effectively track concurrent state changes caused either by commands or by component failures. The behavior of each component state or mode is captured using abstract, or qualitative, models [14]. These models describe qualities of the spacecraft’s structure or behavior without the detail needed for precise numerical prediction, making abstract models much easier to acquire and verify than quantitative engineering models. Examples of qualities captured are the power, data and hydraulic connectivity of spacecraft components and the directions in which each thruster provides torque. While such models cannot quantify how the spacecraft would perform with a failed thruster for example, they can be used to infer which thrusters are failed given only the signs of the errors in spacecraft orientation. Such inferences are robust since small changes in the underlying parameters do not affect the abstract behavior of the spacecraft. In addition, abstract models can be reduced to a set of clauses in propositional logic. This form allows behavior prediction to take place via unit propagation, a very efficient inference procedure.

It is important to note that the Livingstone models are not required to be explicit or complete with respect to the actual physical components. Often models do not explicitly represent the cause of a particular behavior in terms of a component’s physical structure. For example, there are numerous causes of a stuck switch: the driver has failed; excessive current has welded it shut, and so on. If the observable behavior and recovery for all causes of a stuck switch are the same, Livingstone need not model the physical structure responsible for these fine distinctions. Models are always incomplete in that they have an explicit unknown failure mode. Any component behavior that is inconsistent with all known nominal and failure modes is consistent with the unknown mode. In this way, Livingstone can infer that a component has failed, even though the failure was not foreseen or was left unmodeled because no recovery is possible. By modeling only to the level of detail required to make relevant distinctions in diagnosis (distinctions that prescribe different recoveries or different operation of the system) we can describe a system with qualitative "common-sense" models which are compact and quite easily written.

**Livingstone: The Future**

The successful deployment of Livingstone as part of Remote Agent demonstrated the effectiveness of discrete model-based diagnosis, and is helping to convince NASA
mission managers that on-board diagnosis is a technology they want on their missions. However, Remote Agent, along with more recent attempts to deploy Livingstone [27], have also pointed out some of the weaknesses of Livingstone and allowed us to draw some conclusions about its usefulness.

The major limitation of Livingstone and related approaches is the use of discrete models with monitors to discretize continuous sensor values. For some domains, discrete models are very attractive, as they are intuitive, simple to encode, and computationally tractable. A discrete abstraction is best where the abstraction from a continuous to a discrete variable can be performed independent of other variables and the particular configuration of the system. An example of such a domain is digital circuit diagnosis, where the sensor inputs are discrete to begin with. In this class of domains, Livingstone can be used with relative ease. In contrast, there is a large class of systems where Livingstone can be applied, but with considerable effort. Deep Space One probably falls into this class; Livingstone was successfully applied, but to make it work, a lot of effort went into modeling and monitor design. Finally, there is a class of problems where new techniques must be developed. An example of this class is planetary rover diagnosis. Here, there is so much interaction between the rover and its environment that the monitors would need to look at a large number of sensors in order to find a good discretization for a single sensor value. For example, consider a sensor that measures the current drawn by one wheel of a rover. An unusually high current may be the result of a number of things, including faults in the wheel such as damaged bearings, but also normal conditions such as driving fast, or driving up a hill. To successfully determine whether the observed current is nominal or high, we need to examine the speed the wheel is rotating, the attitude of the rover, and potentially the behavior of the other wheels as well as the current itself.

Below, we describe two current research projects, one where Livingstone seems very suitable, and another (planetary rovers, as mentioned above) where new approaches to diagnosis have been needed.

**Space Station Command and Control**

Livingstone is being applied to the Command and Data Handling (C&DH) system of the International Space Station (ISS) [24]. The C&DH system is the nervous system of ISS and is composed of a three-tiered network of computers networked through redundant 1553 buses. The complexity of this network is illustrated in Figure 3. The Tier 1 computers control ISS system level functions, Tier 2 computers are responsible for ISS subsystems including electrical, guidance, navigation and control and life support, and Tier 3 computers perform the actual sensing and commanding of ISS. Since all other ISS subsystems depend upon the C&DH system for operation, this system can serve as the foundation of a model-based IVHM vision for ISS.
When faults occur in the C&DH system, the safety of the crew and ISS itself can be at stake. Over the past few years, the C&DH system has experienced a number of failures including a high profile triple failure, which, if the shuttle had not been docked with ISS, would have severed communication between ISS and mission control and resulted in the loss of attitude control. Due to the high probability of additional failures and the costs of around-the-clock monitoring of ISS, NASA is very interested in automated methods that can identify failures.

The C&DH domain is ideal for modeling in Livingstone due to the close analog between its hardware topology and component-connection models (components are defined for each computer and connections between components are defined for each bus). Similar arguments can be made concerning C&DH Command and Control Software (CCS). In addition, there are several properties of the bus communication messages that make them attractive to model in Livingstone. First, the propagation of digital bus messages is easier to model than the propagation of analog quantities such as liquids or gases because there is little or no lag and noise, which make correlating predicted and observed system behavior much simpler. Second, the 1553 bus protocol is a synchronous protocol with well-defined command/response handshaking. Finally, most commands provide status talkback information that can be exploited to determine when C&DH events occurred. When serious faults occur in the C&DH system, mission control dumps a set of BIT (Built-In Tests) results to the ground. These are the results of a predefined set of hardware and software tests for each computer and bus. Taken together they represent the current state of the C&DH system. The C&DH Livingstone models perform sensor fusion over these BIT results by making hardware and software dependencies between the BIT test results explicit, and provide a single model to enforce consistency over these results. This model can be used to play back the sequence of commands that precipitated an error state, or to examine what-if scenarios.

Many challenges remain before the utility of the Livingstone C&DH system can be determined, but by making explicit the hardware and software dependencies between ISS C&DH components, by providing real-time methods to search these dependencies for
fault consistency and by providing methods to track multiple fault candidates, we expect that Livingstone will provide ground and space-based tools that will significantly increase the state estimation capabilities of ISS operations.

**Diagnosis for Planetary Rovers**

For the 1997 Mars Sojourner mission, all planning for the rover was done on Earth. Every 24 hours, the rover reported back its current position and status, a plan for the day’s activities was made, taking into account the current performance of the rover, and this plan was sent back to Sojourner to be executed. If there was a problem with the plan, for example if an instrument failed to operate, or if the rover hit a rock, it would wait until the next communications pass for new instructions. It is estimated that the rover spent as much as 70 percent of its time waiting for new instructions after its plan had failed. This approach was adequate for the Sojourner mission, but future missions are much more ambitious and will require the rover to achieve more over its lifetime, to drive to places it couldn’t see at the beginning of the day, and to autonomously carry out tasks such as placing a scientific instrument against a rock. This degree of autonomy cannot be accomplished without on-board diagnosis, both of the internal state of the rover, and of its interaction with the environment. Unfortunately, as we argued above, planetary rovers are an example of a system for which discrete diagnosis techniques such as Livingstone are inappropriate, so we need to develop new approaches to diagnosis.

To perform diagnosis on the rover, we need to explicitly reason about the continuous behavior of the system. To do this we need a hybrid system model of the rover. A hybrid system is one that displays a mixture of discrete and continuous behavior. For example, consider again a single wheel of a rover. The wheel has a number of discrete modes of behavior, including driving forward, being idle, and various fault modes, such as having a burned-out motor. In each of these modes, the relationships between continuous variables such as the current drawn by the wheel and its speed can be described by a set of equations, but these equations are different for each mode of the system. To successfully diagnose the rover we need to build hybrid models of each of its subsystems. Given such a model, diagnosis becomes the problem of determining the current discrete mode of the system, plus the values of the continuous variables.

We can think of diagnosis as a form of belief update. At any point in time, we have some beliefs about the state of the system, we get some new observations about the system from our sensors, and we update our beliefs to reflect this new information. In Livingstone, our beliefs were in the form of a single hypothesis about the system and update consisted of checking to see if the observations were consistent with the hypothesis, and if not, finding a new hypothesis that was consistent. For the rover, we keep a probability distribution over the states of the system, and increase or decrease the probability of each state depending on how well it explains the observations. Having a probability distribution rather than a single diagnosis allows an on-board planner to consider not only the best action for most likely state of the rover, but also the consequences of the action in other states that are less likely but still possible. Unfortunately, updating a full probability distribution over a continuous state space is computationally very expensive. There are a number of possible solutions to this...
problem, and we are currently investigating the use of particle filters [25], a Monte Carlo approximation technique in which this distribution is approximated with a set of samples.

In a particle filter, each sample represents a possible state the rover might be in. When we perform diagnosis, we predict a possible future state for each sample, and look at how well the predicted state matches the observations. Samples that do not explain the observations well are removed and replaced by copies of samples that explain the observations better. At any time, the set of samples is an approximation to the true distribution over the rover’s state, and properties such as the most probable system mode can easily be computed from the samples.

This hybrid approach is a big step from Livingstone in terms of the types of systems it can be applied to, but also in terms of its computational requirements. Although preliminary results are promising [26], there are many challenges to be overcome before the system will be ready for deployment in an actual NASA mission. Our current aim is to have it ready for the 2009 rover mission to Mars, but in the mean time, there are many applications, such as the ISS C&DH domain we described above, for which existing approaches are well suited. We expect to see diagnosis systems becoming increasingly common on future NASA spacecraft as autonomy becomes a necessary part of missions, and demonstrations such as Deep Space One continue to showcase the valuable role of diagnosis.

Many of the following papers may be found on the World Wide Web at http://ic-www.arc.nasa.gov/ic/projects/mba/


