Costs and Benefits of Model-based Diagnosis

James Kurien
NASA Ames Research Center
MS 269-2, Moffett Field, CA 94035
e-mail: James.A.Kurien@nasa.gov

María Dolores R-Moreno
European Space Research and Technology Centre (ESTEC)
European Space Agency
and
Departamento de Automática.
Universidad de Alcalá.
Ctra. Madrid-Barcelona, km 33,6.
28805 Alcalá de Henares (Madrid), Spain.
e-mail: mdolores@aut.uah.es

Abstract—Over the past 20 years, there has been much work in the area of model-based diagnosis (MBD). By this we mean diagnosis systems arising from Computer Science or Artificial Intelligence approaches where a generic software engine is developed to address a large class of diagnosis problems [1], [2]. Later models are created to apply the engine to a specific problem. These techniques are very attractive, suggesting a vision of machines that repair themselves, reduced costs for all kinds of endeavors, spacecraft that continue their missions even when failing, and so on. This promise inspired a broad range of activity, including our involvement over several years in flying the Livingstone and Livingstone 2 on-board model-based diagnosis and recovery systems as experiments on two spacecraft [3], [4], [5], [6], [7].

While a great deal was learned through a variety of applications to simulators, testbeds and flight experiments, no project adopted the technology as an operation tool and the expected benefits have not yet come to fruition. This led us to ask what are the costs of using MBD for the operational scenarios we encountered, what are the benefits, and how do we approach the question of whether the benefits outweigh the costs? How are missions today approaching fault diagnosis and recovery during operations? If we characterize the cost and benefits of using MBD, how would it compare with traditional ways of making a system more robust? How did expectations for MBD compare to benefits seen in the field and why?

The literature does provide existing cost models for related endeavors such as integrated vehicle health management [8], [9], [10]. It also provides excellent narratives of why projects chose not to use MBD after considering it [11]. However, we believe that this paper is the first to unpack and discuss the cost, benefit and risk factors that impact the net value of model-based diagnosis and recovery. We use experience with systems such as Livingstone as an example, so our focus is on-board model-based diagnosis and recovery, but we believe many of the insights and remaining questions on the costs and benefits are applicable to other diagnosis applications.

While the analysis is not yet mature enough to provide a quantitative model of when on-board model-based diagnosis would be an effective choice, it lays out the cost/benefit proposition and identifies several disconnects that we believe prevent adoption as an operational tool. While we do not suggest metrics for every cost, benefit and risk factor we identify, we do discuss where each factor arises in development or operations and how model-based diagnosis and recovery tends to leverage or exacerbate each. As such we believe the analysis is of use to those developing MBD or related techniques and those who may employ them. It also serves as one example of how final impact on the customer's process may come to differ from expectations based on technical capability.

In this paper we present a cost/benefit analysis for MBD, using expectations and experiences with Livingstone as an example. We provide an overview of common techniques for making spacecraft robust, citing fault protection schemes from recent missions [12], [13]. We lay out the cost, benefit and risk advantages associated with on-board MBD, and use the examples to probe each expected advantage in turn. We conclude our analysis with a summary of our method for analyzing the costs and benefits in a particular domain, and encourage others to come forward with analyses of costs and benefits for fielded systems. Finally, we discuss related work both in terms of similar analyses and fielded systems.

1-4244-1488-1/08/$25.00 ©2008 IEEE
IEEEAC Paper #1280

TABLE OF CONTENTS

1. INTRODUCTION

Our experience and observations upon developing and deploying model-based diagnosis (MBD) and other model-based technologies to a variety of testbeds, flight experiments and spacecraft operations led us to explore why our expectations for the benefits and impact of MBD upon spacecraft operations have not yet been matched by the effective benefits seen in the field. This in turn led to a series of questions. What at are the costs of using MBD, what are the benefits, and how can we approach the question of whether the benefits outwight the costs? How are missions today approaching fault diagnosis and recovery during operations? If we characterize the cost of using MBD, how would MBD compare in a cost/benefit tradeoff with traditional ways of making a system more robust? Is there something about planning, in many ways a similar technology, that has made it more successful, and what does that tell us?

The paper is organized as follows. In the next section, we present a brief description of the spacecraft fault protection problem and common practice for addressing it, followed by a very brief overview of the Livingstone model-based diagnosis and recovery system. We then consider Livingstone's impact on spacecraft fault protection practice, both in terms of the impact we expected based on its capabilities and the impact we have observed in the subsequent years. To understand this discrepancy, we introduce a set of factors we believe influence the adoption rate of this technology and perhaps similar on-board technologies. We express net impact as a product of expected value, cost and risk rather than simply capabilities. For each of these, we identify aspects of model-based diagnosis that contribute, and how we would evaluate them using the Livingstone experience as an example. In some cases we identify outstanding problems or offer brief suggestions about what could be done to change the equation.

In the Conclusion we end our discussion of net impact by summarizing our own experience with the cost, risk and value tradeoff of for model based diagnosis, though we encourage the reader to consider their own analysis of this or other technologies using a similar breakdown of net impact. In Related Work, we discuss a small set of systems that are model-based, or pertain to diagnosis and recovery, or have been fielded in operational use, our ultimate interest being a system that is all three. We briefly discuss our understanding of why model-based diagnosis technology has not had the success of related model-based technologies in terms of our net impact factors.

2. SPACECRAFT FAULT PROTECTION

Before discussing model-based diagnosis we briefly discuss spacecraft fault protection for the purpose of comparison. The primary purpose of fault protection is to ensure that anomalies or operational problems encountered during operation of the spacecraft do not result in permanent reduction in the spacecraft's capabilities or loss of the mission itself. As Neilson explains in an excellent overview of the fault protection system for the Mars Exploration Rovers (MER), fault protection is an engineering process that incorporates robustness to faults into spacecraft hardware, software, systems engineering and operations [12]. All of these systems are engineered to work together to reduce the likelihood that any reasonably plausible contingency will result in permanent loss of mission capabilities. This paper is largely concerned with on-board software for active fault detection, isolation and recovery (FDIR). We note though that the on-board system is just one aspect of the overall fault protection engineering process, and that the on-board system for protecting the spacecraft is typically a mix of hardware and software. Traditionally, the fault protection engineering process is driven by fault modes, effects and criticality analysis (FMECA). This process typically determines the possible faults of a system or subsystem, some notion of their likelihoods, and an analysis of the impact of each. If the likelihood of a fault is deemed sufficiently high and the impact sufficiently negative, the analysis would also include how the fault would be detected and what the appropriate response would be.

The appropriate response to a fault may depend upon the phase of the mission. We use the term *critical phase* of a mission to mean periods of a mission where the spacecraft must take specific actions (a *critical sequence*) or loss or serious degradation of the mission will result. For example during the entry, descent and landing (EDL) sequence each MER rover entered the Martian atmosphere at over 10,000 miles per hour. At specific times or altitudes it jettisoned a protective shell, deployed airbags, and the like. Failure to execute a step in this six minute sequence would end the mission. We say the system must *fail operational* in that any failure that occurs must be taken into account by the fault protection approach, for example by switching between redundant subsystems, to allow execution to continue. Accordingly, during development of the spacecraft an enormous amount of attention is given to the precise critical sequence the spacecraft will execute, which failures are likely, how they will be detected, and how they will be immediately mitigated. Since response must be timely, it must be available on-board. This may involve something as simple as a table mapping observed sensor values to commands that should be issued in response, for example to switch to a redundant backup. For very complex critical sequences on large spacecraft, a much more elaborate method of determining responses may be developed, such as for the Cassini orbital insertion at Saturn [14].

In a *non-critical phase*, it is still possible to damage or lose the mission due to a fault, but there isn't the added constraint of having to execute a critical sequence. Thus typical fault protection systems tend to focus on mitigating or postponing the impact of the fault by changing the spacecraft's state or behavior. Often if a serious problem is detected, the spacecraft is placed into a *safe mode* where the only actions taken are those that maintain the spacecraft in a quiescent state. For example on the MER rovers, draining the battery risks not being able to heat the rover during the cold Martian night and

not being able to communicate with Earth when expected. The system level MER fault protection includes a hardware battery controller that disconnects the batteries should a serious hardware or software failure begin to drain them. In this case, the system's heaters, solar panels, and flight software work together to ensure the rover does nothing but stay warm and wake once during the day to contact Earth using very few hardware and software subsystems and minimal power. If the situation appears to be less grievous or more localized, then a less drastic response may be taken at first. On the MER rovers, subsystem behaviors incorporate subsystem level fault protection [15]. If the rover's arm draws more than the allowed current during use, it is marked failed. The arm behavior ignores any subsequent requests to use the arm, and the rover driving behavior is disabled if the arm is not stowed away. Routine communication with Earth and other tasks that do not involve the arm continue as normal. During the cycle of downlinking telemetry and uplinking sequences (typically the next day), ground operators can inspect the telemetry and debug the arm before re-enabling it. This system has protected the MER spacecraft/rovers for approaching five years. A great summary of the anomalies and faults encountered by the rovers in the first 780 sols (Martian days) of operation is available [16].

The approach of safing the entire spacecraft or select subsystems when faults are suspected has the advantage that one need not know exactly which fault is causing the operational problem. If the rover arm exceeds an operational current limit, use of the arm ceases, which applies for a limitless number of reasons the arm might be malfunctioning from a motor short to a rock stuck in an actuator. Similarly, if the battery is being dangerously drained, the flight software may shut down it's activities, but at some point if the situation persists the hardware will effectively turn off the rover and restart it in a fresh, quiescent state when solar power is available.

This fault protection approach is to carefully engineer the robust but minimal fail operational capabilities needed for critical periods, and otherwise engineer hardware, software and operations so the spacecraft can be put in a safe state in response to plausible anomalies. This characterizes a wide range of fault protection systems that have been flown. The remainder of this paper is about experience gained attempting to improve upon this approach using model-based diagnosis technology. The intent was to provide greater spacecraft autonomy and robustness, greater science return from missions, reduced operations costs, reduced analysis cost for a mission, and reduced flight software development cost.

The next section describes Livingstone and Livingstone 2, two model based diagnosis systems that have been flown as experiments on-board NASA spacecraft as well as demonstrated on a number of testbeds. The following section describes Livingstone was expected to provide the benefits described above. Later, we describe why we believe these benefits were not realized, and provide some analysis of where we believe the problem lies.

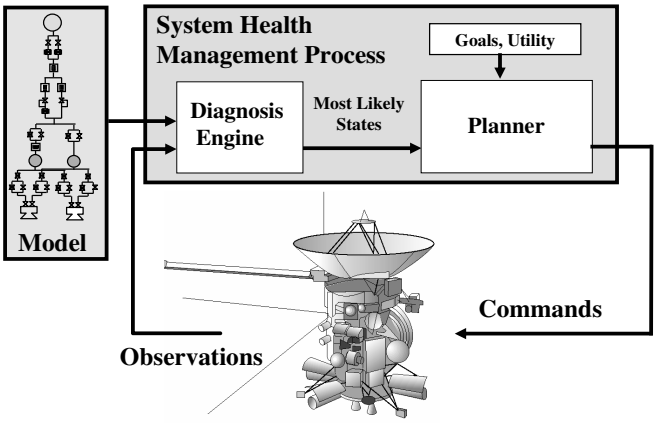

Figure 1. Model-based Diagnosis and Recovery

3. THE LIVINGSTONE SYSTEM

Over the past 20 years, there has been much work in the area of model-based diagnosis (MBD). By diagnosis, we mean the problem of observing a mechanical, software or other system and determining what failures, if any, its internal components have suffered. For example, if our car won't start but we observe that the lights work, we might infer that the problem is the starter and not the battery. By model-based diagnosis, we mean diagnostic systems arising from Computer Science or Artificial Intelligence approaches where a generic software engine or set of principles is developed in the hope of addressing a large class of diagnosis problems [1], [2]. Later, models that adapt the generic engine to a specific diagnosis problem (e.g., the automotive starting system) are created. The diagnostic engine is given the model and fed observations from sensors on the mechanical system being diagnosed, and is intended to automatically infer which components of the system are failed, and perhaps in what manner. The particular representation and algorithms used are beyond the scope of this paper, but an excellent survey of the seminal work in this field can be found in [17].

The Livingstone system [3], [4] builds on this work, and is meant to monitor execution of commands, diagnose failures and provide recovery actions for complex systems such as spacecraft. Figure 1 illustrates its operation at an idealized, schematic level. We use the Cassini spacecraft as a benchmark example, as was the norm in the Livingstone papers. To avoid a persistent confusion, we note Livingstone was run against a simulation of the Cassini spacecraft and not as a part of the Cassini mission. Livingstone requires a high level model of the components of the spacecraft and their operation. During operation, Livingstone is fed the stream of commands that are being given to the spacecraft and the readings from sensors that relay the internal state of the spacecraft (e.g. switch status bits, temperature sensors, pressure transducers, etc.). Livingstone uses the model of the spacecraft's components and the command stream to predict the

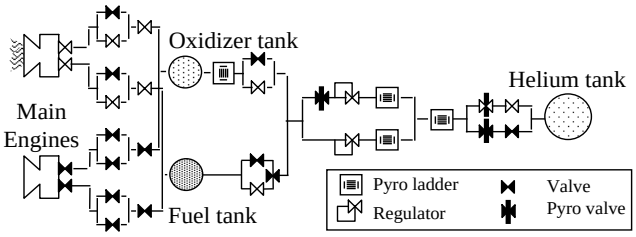

Figure 2. Cassini Main Propulsion System

values of the sensors that should result from the commands assuming no components are failed. If there is a discrepancy between the predicted and observed sensors, then a failure is assumed. Livingstone uses the model of the components to simulate different combinations of failures. It uses a diagnosis algorithm to quickly focus on the combination of failures that would predict the sensor values that are being observed. This combination of component failures is the diagnosis. In case of failure, Livingstone is able to use the same predictive model to suggest commands that achieve a desired property (e.g. engine is receiving fuel) and thus mitigate the failure. It can for example infer that one should switch between the failed component and a similar set of components that performs the same function. For example, consider the Cassini main propulsion system in Figure 2. The purpose of the system is for the helium tank to pressurize the oxidizer tank and fuel tank in order to push fuel and oxidizer to exactly one of the engines, where it is ignited to create thrust. Owing to Cassini's decade long mission, the system has many redundant paths between the various tanks and a backup engine. Valves in parallel allow a second path in case a valve sticks shut. Valves in series allow a path to be turned off if a valve sticks open. When run against a simulation of this propulsion system, Livingstone's task was to monitor the commands given to the valves and infer if the pressure readings were consistent with the expected state of the valves. If not, Livingstone was able to determine the smallest number of valves to open or close to ensure oxidizer and fuel could reach exactly one engine. Thus Livingstone could effectively manage the configuration of the system so that it always produced thrust when desired even in the face of failures.

4. Expected Benefits

The promise of a system like Livingstone flows from the concept that we describe the nominal and (perhaps multiple) failure behaviors of each type of component, and how the components are interconnected. Livingstone then infers the system-wide behavior given any combination of nominal and failed components and any sequence of commands. Thus the user models that a valve can be open and allow fluid flow, or closed and stop fluid flow, or stuck open or stuck closed, and can arrange valve combinations far more complex than the Cassini model shown. Livingstone in turn performs the system-wide reasoning during operations to start and stop the engines (or achieve any other state described by the model) in the face of multiple valves sticking closed or open. If we could use MBD to replace manual reasoning through system wide interactions under a multitude of scenarios with describing the local behavior of components, we envisioned a wide range of benefits.

- **Significantly lower costs for critical phases**

The fail-operational response needed during a critical phases would be inferred from the model on-line during execution of the critical phase. Thus, we would reduce manual analysis of how spacecraft subsystems would interact during an anomaly that was needed to engineer static fail-operational responses. Effort could be shifted to testing the responses of the inference system.

- **Higher mission return**

By inferring fail-operational responses for any desired state of the spacecraft, rather than manually engineering them for specific critical sequences, we could fail operational even in non-critical periods. During a fault, rather than safing and waiting for intervention, the fault protection system would diagnose which component was failed. It would return the spacecraft to an operational state by reconfiguring redundant systems, resetting components, or cancelling only those activities that depended upon failed components. Routine operations would continue without intervention from the ground, increasing science return and decreasing operations costs.

- **Greater robustness than traditional fault protection**

The on-board diagnoser could potentially infer diagnoses of double and triple failures and suggest workarounds, where a traditional fault protection might typically have pre-computed responses for single failures or certain likely and critical anomalies involving multiple failures.

- **Reduce fault protection implementation costs**

Rather than use a custom written fault protection and safing system, the spacecraft fault protection system would employ a re-usable diagnosis engine that inferred diagnostic and recovery responses on the fly. Thus a decrease in flight software costs was projected.

5. Livingstone Flight Experience

A large number of talented people developed Livingstone and L2 applications for a wide variety of systems, which allowed us to gain the experience upon which this paper is based. Table 1 lists applications developed for simulators, testbeds and two flight experiments, along with the year, and the number of different failures modeled. Where available, the number of person months spent developing the Livingstone model and person months developing the entire application (models, signal conditioning code, integration, etc) is shown.

Livingstone was chosen in 1996 to be a part of the Remote Agent spacecraft autonomy architecture [18]. In May 1999 it was flown on the Deep Space 1 spacecraft as a technology experiment [5]. The first author gained direct experience with model-based diagnosis technology by doing Livingstone development, modeling of the Deep Space 1 (DS-1) spacecraft, and participating in mission operations during the experiment. Livingstone was activated on DS-1 for a 20

Type	Domain	Subystem		Year	System Effort Months	Model Effort Months	Types of Components
Simulator	Cassini	Propulsion	Liv.	1996			
Flight Exp	Deep Space 1	Attitude control, switches	Liv.	1998	96	36	12
Testbed	ISRU	Chemical reactor	Liv.	1998	96	32	
Testbed	Interferometry	Optical bench	Liv.	1999			
Simulator	micro spacecraft		Liv.	2001			
Simulator	rover	Drive systrem	Liv.	2001			
Simulator	X-34	Propulsion	L2	2002			26
Simulator	X-37	Electronics	L2	2002	9	5	18
Flight Exp	Earth Orbiter 1		L2	2003	12	2.8	

Table 1. Livingstone and Livingstone 2 Applications

hour test and a 6 hour test. During these tests, the Remote Agent was run on top of the DS-1 flight software, which included a fault protection system. During the test Livingstone was fed simulated sensor readings consistent with a set of four pre-determined failure scenarios: switch position indicator failed, camera power switch stuck on, science instrument not responding and thruster stuck closed. In the first case, Livingstone correctly ignored the sensor, and in the remaining cases recommended recoveries of re-trying the command, power-cycling the instrument, and switching thruster control modes, respectively.

Subsequently, we developed the follow-on Livingstone 2 system (L2), with several technical improvements [4]. With L2 and the associated modeling tools, our experience applying model-based diagnosis and recovery systems to spacecraft greatly expanded. L2 was flown as a technology experiment on the Earth Observer 1 spacecraft (EO-1) spacecraft, again flying on top of the existing fault protection system. L2 was activated on EO-1 for a total of 143 days in 2004 and 2005 and diagnosed 13 simulated failures [7]. Flight experiments for the X-34 and X-37 spacecraft also were developed, though those vehicles were never flown [6].

In addition to these flight experiments, Livingstone or L2 was applied to a Mars in-situ propellant production testbed (ISRU) at Kennedy Space Center, a testbed for the Space Inteferometry Mission at the Jet Propulsion Laboratory, the Bio-Plex Mars habitat testbed at Johnson Space Center, a testbed for the PSA micro spacecraft [19], and a rover testbed [20]. All of these efforts generated a great deal of knowledge about the needs and requirements for advanced on-board fault protection.

6. LIVINGSTONE'S IMPACT

A great deal was learned from the efforts of the many talented teams that used Livingstone. In retrospect though, it's fair to say that MBD did not transform, or even impact how fault protection is done on NASA missions. To our knowledge, no NASA spacecraft has used Livingstone or any system like it as a part of its fault protection system, and after the DS-1 and EO-1 experiments, there have yet to be additional research-funded flight experiments. This begs the question of why we did not observe the wide range of benefits we expected from applying model-based diagnosis and recovery to spacecraft operations, and why there was no 'mission pull' and adoption of the technology.

It is tempting to explain the lack of penetration by imagining missions are averse to incorporating new technologies. Certainly revolutionary approaches are taken all the time. Consider the airbag landing system used on PATHFINDER and MER. Perhaps there is some hesitance to use unfamiliar or model-based software. Consider the case of planning and scheduling technology. Coincidentally, Livingstone flew with the HSTS planner during the Remote Agent experiment, and L2 flew with the CASPER planner on EO-1. At a high level, planners are similar to Livingstone in that they comprise a generic inference engine and a model used to adapt it to a problem. A planner chooses actions to take to achieve a goal, rather than failures to explain a symptom, but the algorithms and models are similar in the grand scheme of things. As described in Related Work, HSTS evolved into ground-based tools that have generated thousands of plans for the MER rovers and are scheduled for follow on missions. CASPER became an operational tool on EO-1, continuing to run on-board for years, plan over 100,000 goals to date, and save millions of dollars in operating costs [21]. Thus missions are willing to adopt new, model-based software technologies.

The decision by a mission to adopt a technology as a baseline tool is a combination of the value, cost and risk given the characteristics of that particular mission so we began to consider those terms. The three are intertwined, as a mission may make a significant investment to buy down risk, as illustrated by the inclusion of a backup engine on Cassini, and conversely a useful mission feature that provides value may be abandoned if the cost cannot be kept under control. In these terms, we believe it's relatively easy to understand the

lack of penetration of model-based diagnosis and recovery. We provide an overview before a detailed analysis.

Consider the critical phases of a mission and the proposal to use model-based diagnosis and recovery to provide the required fail operational capability. Added value and reduced cost from employing MBD have not be clearly shown, but they would have to be incredibly high to offset the risk of not being able to verify in advance exactly how the spacecraft was going to respond when an incorrect response will result in mission loss. Thus we will argue the risk of MBD during critical phases is high, and the value is unclear.

Consider the non-critical phases of a mission and the proposal to add fail operational capability. Typically, spacecraft go into safe mode very rarely. When they do, operators typically would like to understand what has occurred before continuing operations. Surveying real mission anomalies also reveals that few are failures that a priori we would have modeled and been able to recover. Thus we will argue the value provided by MBD during non-critical phases is not high, especially when considered against the cost and risk involved.

In the next sections, we present a method for organizing the factors we identified to determine whether the value of a sophisticated on-board diagnosis and recovery system (model-based or not) is worth the cost and risk involved, depending on the particular characteristics of the application. We also support our analysis of the costs and benefits of model-based diagnosis using the MER and DS-1 missions as examples.

7. THE COST/BENEFIT TRADEOFF

As technology developers, there is a natural tendency to focus on the capabilities of a technology, such as MBD's ability to find diagnoses and make recoveries. In retrospect, the post hoc operational value of a technology to a mission is more complex. The value of a diagnosis captured by our models, for example, is only realized if that particular failure occurs and it is correctly diagnosed. Similarly, the cost of model development doesn't reflect cost increases or savings during mission analysis, testing or operations. The benefit provided by continuing to operate must be balanced with the risk of incurring further damage to the spacecraft. We therefore began to enumerate these factors and tried clarify our own thinking about them, hoping in the best case of giving potential customers a way of "unpacking" the problem of estimating the value of MBD in their circumstances. This raised the issue of how to have an intelligent discussion about this estimate when many of the factors needed to make the estimate are unknown *a priori* (e.g. whether a failure will eventually occur) or may be contentious.

Here, we were directly inspired by the Drake Equation [22], [23]. In the 1950's, Frank Drake began estimating the number of intelligent civilizations in the galaxy. He cleverly formulated his estimate so the terms mirror the steps required for a planet to exist with a civilization on it: the number of stars in the galaxy, times the percentage that have planets, times the percentage in which life arises at all, and so on. The beauty and perhaps the point is one can agree with or discuss the structure of the Drake equation while completely disagreeing with the estimate Drake makes. The equation itself is a tool for thinking about the factors impacting the problem, focusing discussion, and reasoning about how changes in various assumptions influence the outcome. Thus, with tongue-in-cheek apologies to Frank Drake, may introduce the Kurien-Moreno equation of the net value of on-board model-based diagnosis, and perhaps other technologies, as shown in Figure 3.

Equation 1 of Figure 3 expresses the simple notion that the expected net value of using MBD over the life of a mission depends upon the expected value that the technology provides, minus the additional risks of using it and the cost of deploying it. By expected value V_{mbd}, we mean the product of how much value MBD provides in a given situation, and the likelihood of that situation arising in practice. By cost due to risk, C_{risk}, we mean the loss of mission or capability that could occur, and the increase or decrease in likelihood that MBD causes. Finally, by cost of deployment, C_{dep} we mean the cost increase or decrease resulting from use of MBD in the on-board fault protection system.

To promote discussion, Equation 2 unrolls each of these simple terms in a way that mirrors the events that will unfold to determine the value of an MBD application. For example, one term in expected value, V_{mbd}, is the value, V_r to the mission of having the immediate diagnosis and recovery provided by MBD. The other terms are the likelihood that a particular failure occurs, the likelihood of the diagnoser having a model of the failure and making the correct diagnosis, and the likelihood that the diagnoser is able to find a recovery that improves the state of the spacecraft. Each of these is a term we can discuss when evaluating whether MBD is a good fit for a particular mission, or when thinking about what we can do to make MBD more attractive to missions.

In the subsequent sections, we expand the expected value, risk and cost terms as shown in Equation 2. We introduce the factors we have identified as contributing to each of the terms, and describe how our experience with model-based diagnosis would lead us to estimate those factors.

8. EXPECTED VALUE

Whatever value we argue MBD provides, before it can provide that value a failure must occur, the MBD system must correctly diagnose the failure and it must suggest a useful response. Thus, we can decompose the likelihood of providing value into the likelihood of each event in this sequence. The *expected* value of doing diagnosis is therefore:

$V_{mbd} = P_f * P_d * P_R * V_r *$

where

$$NetValue = V_{mbd} - C_{risk} - C_{dep} \quad (1)$$

$$NetValue = \overbrace{V_r * P_f * P_d * P_R}^{V_{mbd}} - \overbrace{P_{cf} * C_{cf}}^{C_{risk}} - \overbrace{C_m + C_a + C_{test} + C_r}^{C_{dep}} \quad (2)$$

Figure 3. The (tongue-in-cheek) Kurien-Moreno Equation

- P_f = likelihood of a failure actually occurring (≤ 1)
- V_r = value having a diagnosis and response that's automatic (timeliness, reduced operations cost)
- P_d = likelihood of correctly diagnosing the failure that actually occurs (≤ 1)
- P_r = likelihood of providing proper response (≤ 1)

Let us consider the value provided during non-critical operations. In this case, we consider the value of having on-board diagnosis and recovery, V_r, to be equivalent to the number of days of operations that would be lost waiting in safe mode if we did not have on-board recovery. We would expect P_f, the rate of occurrence of failures within on-board spacecraft systems to be quite low in general. This is important because the product $P_f * V_r$, the expected amount of time the spacecraft spends in safe mode, is an upper bound on the value of on-board recovery, representing the ideal the case where our MBD system is infallible and completely trusted. This is simply the nature of attempting to provide value with diagnosis systems.

To provide a concrete example, Table 2 summarizes Nelson's excellent overview of the anomalies in MER operations that occurred in the first two years of commanding two rovers on Mars. The first column is a short name we have given to the anomaly. The second denotes whether the problem was due to software, hardware or interactions with the environment. We presume that most MBD applications apply to hardware and software failures, not dynamic interactions with the environment, though this is incidental to our analysis. The third column denotes how many days passed between when the anomaly occurred and the subsystem involved in the anomaly was put back in to routine operations. Note in many cases other non-effected subsystems were put back in to operation sooner. The final column describes what response was developed by the MER anomaly analysis team.

This table captures the anomalies that caused the two rovers to enter a fault response during approximately 1550 days worth of rover operations. The small number of hardware and software related anomalies suggests that P_f is quite small. The amount of time spent on anomaly recovery was about 2% of the operational time, including a significant amount of time responding to environmental problems such as being stuck in soft sand. Thus for MER we know the product $P_f * V_r$ turned out to be 2% of the operational time of the rover. This represents the ideal upper bound of the value of using MBD to achieve fail operational during routine operations if every anomaly could be resolved autonomously on-board.

The remaining terms, P_d and P_r, will dictate how far we are from that ideal return, and represent the likelihood that a MBD system will correctly diagnose and recover from the failures that occur during operations. Looking at the failures in Table 2 the first question to consider is out of the enormous space of component failure modes, what is the likelihood that *a priori* our models would have covered diagnosis and recoveries for these failures? The second question is, what is the likelihood that our automated software would have produced recoveries that were both correct and more valuable than simply shutting down subsystems and waiting for ground analysis as the existing MER system does? Since MER did not use an on-board MBD system, discussion of P_d and P_r may be considered conjecture. We believe one can get a sense of the (small) magnitude of P_d and P_r by looking at the amount of time that the engineers who built the rover, flight software and operations system spent carefully probing the rovers with small experiments and deriving a recovery and new operational policy that would accommodate the failure and the rover's interaction with the environment. We believe that the MER pattern of infrequent, short bursts of extremely careful analysis indicates V_{mbd} would be significantly less than adding 2% to the rovers operational time.

We briefly consider the much shorter Livingstone flight experiments. No real spacecraft failures were diagnosed during two day Livingstone flight experiment or the 143 day L2 flight experiment, so it's difficult to measure the value. Two failures in experiment-related software did occur during and interfere with the flight experiment of Livingstone on DS1. One failure we had not even considered modeling, and the other was explicitly declared out of scope during the modeling effort. Therefore the only two actual failures known to have occurred during a Livingstone flight experiment were not recovered from [5]. This again suggests P_f, P_d and P_r are not terribly high.

This leaves V_r, the value that using Livingstone could provide if failures did occur and Livingstone covered them properly. It is interesting to note that the engineering of the DS1 Livingstone model was carried out by understanding what the appropriate recovery was, then developing a model that captured it. For example, if the spacecraft flight software reported the pointing of the spacecraft to be inaccurate, Livingstone would diagnose which thruster was stuck, and simply change to a second control mode that used different thrusters. Of course, the flight software was computing the error and presumably could switch control modes when the error in the current control mode was significant. Ironically, during the

experiment the error tolerance used to trigger Livingstone's response had to be tuned lower than that used by the DS1 flight software, to ensure that if a real failure occurred, the flight software wouldn't correct for it before Livingstone had a chance to do the component-level diagnosis and offer a recovery. Thus we fear V_r for the Livingstone application on DS-1 would offer little above the existing flight software even if failures had occurred and had been diagnosed. So we again we find it difficult to argue how for the Livingstone flight experiments, V_{mdb} was significant.

In general, we believe P_f is low for operational spacecraft, meaning even the potential benefit for MBD is limited. In addition, for the types of potential applications we are familiar with, failures that do cause loss of operational time are typically complex and unexpected. Consider failure of the Galileo antenna to deploy or the MER rover rebooting due to flash problems. They typically require detailed analysis, creativity and validation before returning the spacecraft to an operational state, further lowering the expected value of MBD. Situations where combinatorics rather than deep knowledge is the issue, as was initially encountered in Cassini's 27 valve propulsion system and its 2^{27} configurations, do not appear to be the driving problem in spacecraft fault protection. Thus for the missions we've considered in detail (all unmanned, deep space missions) we would assign a low expected value to the use of MBD. We next consider risk and cost.

9. RISK

Initially, we imagined that MBD would reduce risk of mission loss by generating diagnoses and recoveries on the fly and increasing the range of situations over which specialized fault responses were available. In retrospect customers were concerned with increased risks, some of which are described below, that could arise from the use of MBD. We can think of the expected cost due to additional risk of using an autonomous MBD system as

$$C_{risk} = P_{cf} * C_{cf}$$

where

- P_{cf} = likelihood that more complex response of the MBD system will itself cause an anomaly or compound the failure
- C_{cf} = cost of resulting anomaly in terms of lost science, extra operator time, loss of mission capabilities

In order for MBD systems to have an impact, one key would be to reduce C_{risk} to a level the mission customer is comfortable with. We can divide sources of perceived risk into three categories: increase in P_{cf} due to the fact that the MBD system is capable of a wider range of responses and generates them on the fly, increase in P_{cf} from continuing to operate after an anomaly when not strictly necessary, and increase in C_{cf} if the actions of an autonomous on-board recovery system must be mitigated when the spacecraft has ceased operating according to design and the ground team must intervene.

Missions considered that MBD would increase P_{cf} because of the increased complexity of the software's response and our inability to concisely characterize, enumerate and validate the range of diagnoses and responses the system might undertake. Rather than engineer a small number of safing responses that are as broadly applicable as possible, MBD seeks to generate recovery reponses that are as specialized as possible on the fly. It's interesting to note that this does not necessarily imply that an MBD system responds to a broader range of anomalies, simply that it responds in a more specialized fashion to each. This means there are far more variations in spacecraft response based on its state, and the full set of conditions and responses could not be enumerated and tested. In addition, the purpose of MBD is to propagate information across the modeled system to allow variations in response. Thus it can be difficult to even concisely describe how small, non-local variations in the space of inputs will impact the response, and difficult to argue that a specific set of test cases provides good coverage for validation. Combating this perceived increase in P_{cf} is a challenge since the ability of MBD to respond with a far wider range of behaviors than traditional fault protection is both its selling point and the source of concern that the system will do something unpredictable. Some work has been done to apply model checking approaches [24], but this remains a significant issue for adoption.

An additional increase to P_{cf} comes from the desire to continue to operate the spacecraft via a recovery generated onboard when it is possible to safe the spacecraft and await expert analysis. Consider the three wheel problems of Table 2, *Wheel drive actuator*, *Rock stuck in wheel*, and *Stuck in sand*. One can imagine mis-diagnosing which failure was occurring and applying the recovery meant for another (*e.g.*, attempting to drive in circles when stuck in sand rather than when a rock is in the wheel) could permanently trap or damage the rover. We don't have examples of Livingstone misdiagnosing a spacecraft failure as no failures occurred during the flight experiments, but we do have examples of false positives (indicating a failure when none exists) during the EO-1 experiment. Thus it's important to keep in mind that MBD may cause us to execute actions that are inappropriate for the true state of the spacecraft. This increase in P_{cf} though the possibility of exacerbating a failure through continued operation rather than safing is one of the items we are asking missions to trade against V_{mbd} estimated in the previous section in order to justify MBD.

Note the cost C_{cf} is not just potential loss of mission. Having specialized, generated responses to anomalies may make it harder and more costly to determine what exactly the spacecraft believes it is doing should something go wrong and mission controllers need to intervene. This is hard to characterize exactly, as we don't have good examples of a model-based fault protection system resolving an anomaly in operations, or of it needing assistance or intervention from the ground. We experienced a little of this in the Remote Agent

Brief Description	Type	Days of Analysis	Recovery process
Wheel drive actuator	HW	4	Experiment to characterize capability of wheel Warm actuator before use Drive backward dragging wheel
Steering current	HW	4	Experiment to characterize source of current increase Use 'K' turns to avoid steering failed wheel
Shoulder actuator current	HW	17	Experiment to characterize shoulder motor degradation. Characterization of future arm failure on driving Change stowage policy to minimize thermal cycling & forestall failure
Heater stuck on	HW	*	Determine that survival heater was stuck on * Operations continued while problem was addressed Implement policy to remove batteries from power bus at night Rely on solar power to wake rover at dawn Trade off power savings vs. degradation of instrument due to cold
Late wakeup/dust storm	Env	1	Solar panels woke up rover slightly late due to dust storm Rover missed time to start sequence, waited in standby mode Plan future sequences to start at least 1 hour after expected wakeup
Rock stuck in wheel	Env	7	Current spike explained by seeing rock in the wheel in imagery Several days of careful driving to dislodge rock
Stuck in sand	Env	40*	Imagery suggests rover is not moving, wheels 70% buried in sand * Continue all non-drive activities on Mars during analysis Set up testbed with similar consistency soil, practice escape strategies Carefully drive out using escape strategy Augment driving policy to avoid wheel embedding on future drives
Flash file system anomaly	SW	14	Overloaded file system table prevents creation of new files. Rover continously reboots. Understand what on-board fault protection system is doing, what is causing reboots Send command to rover to start up without file system, gain control of rover Determine issue with file system. Clear and rebuild file system Carefully manage production of files. Return to nominal ops. Later upload patch.
Race condition during startup	SW	2?	Lose comm window every few hundred sols Added short activity keep out period after startup
Imaging race condition	SW	2	Imaging HW shut down while sequence still reading data from HW Shut down sequence fixed to halt imager sequence before HW shut down
Corrupt command	SW	6*	Solar conjunction test of corrupt commands overloads command handler *Normal commanding resumed after solar conjunction over
Variable evaluation exception	SW	4*	Same global defined in two sequences running in parallel, result in fault * Includes idle weekend. Do not run two scripts that define same global
Upload fault	SW	2	Initial uplink through orbiter experiment overloaded CPU Pad uplink file, limit size

Table 2. MER Anomalies

Experiment, due to actual, non-diagnosed anomalies during the experiment [25]. It may be better to consider how difficult it is to debug anomalies from millions of miles away with (comparatively) simple safing and recovery actions, as in the FLASH anomaly in the Spirit Mars rover [13].

For the flight experiments of Livingstone and L2 on DS-1 and EO-1, the flight software of each spacecraft included a complete, separate fault protection system which protected the spacecraft. On DS-1, for example, the spacecraft's flight software included a fault protection system that monitored the spacecraft for indications that any occurrence, including commands given by Livingstone, were putting the spacecraft in a risky situation. The fault protection system would then stop all commanding of the spacecraft and put the spacecraft into a safe mode to await contact from mission controllers. In addition, Livingstone's communication with the spacecraft was done through a filter which ensured only specific commands which had been analyzed for safety could be sent from Livingstone to the spacecraft. If the underlying fault protection system were activated, that filter would be completely turned off and Livingstone would be terminated.

In an unforgiving environment such as space, Livingstone's ability to provide novel diagnoses and recoveries to failure combinations we had not explicitly considered was far less important than being able to verify exactly how it was going to respond in the most likely and most critical anomaly situations. Guarding an MBD system with a traditional fault protection system and restricting the commands it can give is one approach to bringing it to the level of predictability needed to convince mission stakeholders the spacecraft will not go in to an unsafe state. This strategy was appropriate for experiments whose purpose was to show the technology could be flown. In routine operations, it would tend to undermine the cost and value arguments of using MBD. If MBD technology, or to an extent, any autonomous on-board technology, were to make an impact on operations, we believe this open question of predictability, validation and risk must be addressed regardless of what the proposed value is.

10. COST

In initially thinking about the cost of model-based diagnosis systems during DS-1 development, the focus was largely on

the cost of developing a model of the spacecraft for the MBD system. A more complete picture of the deployment costs, C_{dep} can be thought of as follows:

$$C_{dep} = C_a + C_m + C_{test} + C_r$$

where

- C_a = cost of analysis needed to develop the MBD system
- C_m is the cost of developing models for the failures and recoveries the MBD system will cover and integrating the model
- C_{test} = added or reduced cost of verification and testing
- C_r = amortized cost of research and development of these systems

Roughly speaking, by C_a we mean the cost to determine what should be modeled and how, where by C_m we mean the actual effort required to implement the models and integrate them with the spacecraft. Here we expected a substantial advantage over versus the cost of writing a fault protection system and doing all of analysis to determine the correct response to critical, mission ending anomalies. We would thus expect substantial pull from missions, which are always under cost pressures, to use MBD technologies. In this section, we consider why that was not the case.

We first consider C_a and why we believe use of MBD has not produced a drop in analysis costs. We imagined future missions might use the ability of model-based diagnosis to propagate behavior of individual components across a system model to generate recovery responses on the fly or before flight, meaning the analysis costs of a mission using MBD would drop. Two issues are where the models come from and what analysis do they eliminate? First model-based diagnsosis requires a diagnostic model which is somewhat different than the simulation models used in routine spacecraft development. The DS-1 experience was that it's necessary to know the aspects of each component relevant to failure, the plausible failures to be modeled, how they manifest themselves locally, and so on in order to scope and write a model. Thus we expect FMECA analysis would continue to drive modeling, rather than a model existing through some other process and then being used to replace analysis.

Second, MBD is not equivalent to fault protection, so we don't expect the model for the former to eliminate the analysis for the latter. Fault protection is a system engineering process that impacts the design of hardware, software and operational procedures. It must ensure, for example, if any hardware, software, environmental or operational problem is draining the spacecraft's batteries, the combined hardware, software, and operations system has the maximum likelihood of stabilizing the situation before the vehicle is lost. This is a much broader problem than that of on-board component-level diagnosis and reconfiguration of the spacecraft. In addition, the need for component-level diagnosis typically not a driver. For example, the fault protection design might place all non-essential devices on a separate power bus, which is simply turned off if there is any power-related anomaly. Thus in the flight experiments, it was still necessary to employ a fault protection system which did not need to do detailed diagnosis, while running Livingstone, which did not perform system fault protection. Since the Livingstone models do not address system fault protection, we don't expect the models to provide a significant drop in analysis costs for developing the basic fault protection capability missions require.

We also believe that use of MBD intrinsicaly drives analysis costs up. First, the main value we ascribed to MBD was that it would do detailed diagnosis and recovery autonomously. This means the analysis and modeling needed to diagnose and recover failures, and the non-trivial task of encoding those capabilities into software, must be done *a priori*, before we know which failures will occur. Thus we may perform the detailed analysis and modeling needed to automatically recover for many faliures that never occur. In contrast, the traditional fault protection strategy only performs this kind of detailed, *a priori* analysis for critical sequences and the process for safing the spacecraft. The majority of possible faults simply trigger the safing system without being diagnosed to the component level. The analysis is then done *post hoc* for only those failures that actually occur, and without the need to codify the diagnosis into a model. Second, during non-critical periods the fault protection scheme typically identifies only faults (e.g. the battery voltage is too low) which can be used to find a pre-planned response that is meant to cover a huge space of problems induced by the hardware, software or environment. An MBD system requires a more broader model in able to perform component level diagnofsis. Finally, to generate recoveries, the MBD system must model some of the nominal behavior of the system as well.

With respect to C_m, once the analysis is done to determine what level of detail must be modeled and what failures should be covered, modeling is often relatively easy. There may be cases where representing a particular failure or getting the appropriate recovery to be inferred may be tricky for a given modeling language and diagnosis approach. A second source of cost is integrating the model with the spacecraft. Signals generated by the spacecraft's internal sensors may need to be conditioned so that transient disturbances do not cause false positives in the diagnosis system, or they may be abstracted from real values to trends or qualitative ranges to match the diagnostic algorithm. The Livingstone experience is this can represent at least as much work as modeling. Livingstone is attempting to autonomously infer the failure within individual components and do so preferrably before they cause a system level fault such as a battery undervoltage. Thus we believe signal conditioning and integration costs are higher than for fault protection systems that typically look at a smaller number of system-level measures and are inferring a less detailed estimation of the spacecraft state.

With respect to C_{test}, one of the main characteristics of Livingstone is the ability to generate combinations of diagnoses and recoveries from the possible diagnoses and recoveries for individual components. However, it was still necessary to work through the possible failures, how the failure would propagate through the system, and how Livingstone would respond, then validate the expected behavior through testing. Given the space of possible responses Livingstone could generate, the issue of how to validate it at a reasonable cost if it were to be run as an operational system were an issue during flight and remain an issue.

As a point of interest, Table 1 lists approximate development costs for Livingstone applications where it was possible to make an estimate. On the Deep Space 1 experiment, three people worked part time for a total of approximately thirty six person-months developing a model of 5 subsystems of the spacecraft. Approximately 96 person months were spent including the model and all of the integration and sensor signal conditioning necessary to run on-board the spacecraft. For the Earth Observer 1 experiment, a total of 2.8 person-months were spent modeling and approximately 12 person months was invested in integration and signal conditioning. It's important to note that these figures are not the cost of the fault protection system, as Livingstone alone was unable to provide fault protection for the spacecraft. These are the costs to develop a diagnosis and recovery system for combinations of failures in specific subsystems.

In summary, we believe use of Livingstone and similar technologies does not eliminate or appear to reduce the need to perform analysis for fault protection. The need to develop a separate set of component-level diagnostic models adds cost, and does not significantly offset fault protection analysis costs. The desire to have, *a priori*, a system that can autonomously diagnose and recover a spacecraft appears to introduce additional detailed analysis, model encoding, signal conditioning and testing that is not necessary if the spacecraft is simply put in a safe mode when possible. We believe the approach is also at a cost disadvantage due to the more complex testing and verification requirements necessary to ensure any of the additional diagnoses and novel recoveries Livingstone might come up with during autonomous operations would not endanger the spacecraft.

11. CONCLUSIONS

In this paper we have discussed the expectations for model-based diagnosis and recovery systems such as Livingstone and why we believe not all of those expectations were met. We attempted to lay out the basic cost/benefit drivers in a domain of interest (unmanned spacecraft) and our understanding of why model-based diagnosis as well as recovery have found relatively little traction.

We can grossly characterize the common practice in fault protection to be identifying those contingencies where an active, specific response to anomalies must be made (e.g., loss of a motor during an orbital insertion) and providing identification and response to those states. In other anomalous conditions, the spacecraft is safed and engineers diagnose the problem *post hoc*. For the missions we've considered, this approach seems to provide lower risk and more than adequate value in terms of anomaly response when compared to MBD. In addition we have not yet developed or seen an argument or demonstration that the total analysis, development and testing cost for the common practice is higher than an alternative based upon MBD.

During non-critical mission phases, the net value of having on-board diagnosis and recovery is low since we are free to simply put the spacecraft into a safe mode and only then invest resources attempting to find a diagnosis or response. During critical phases (as well as non-critical) the real need to circumscribe and validate the responses of the fault protection system decreases the proposed value of MBD's ability to generate novel responses, while increasing its testing and analysis costs in an attempt to contain risk. We also believe the key questions of how on-board, component-level diagnosis fits in with and adds value to the broader task of fault protection engineering, and how MBD technologies would reduce fault protection costs remain open. Thus at least for the type of missions with which we are most familiar we believe it is difficult to justify the use of on-board, generative diagnosis and recovery systems like those we have been involved with based on cost, risk or value.

This is not to say there aren't ways to make MBD more attractive to missions. One theme is to use MBD to assist in ground-based diagnosis and recovery. As with the Eureka system described in Related Work, we do not have a satisfactory argument of how a model-based diagnosis system would assist domain experts in the kind of unanticipated anomalies that they find challenging. This may change however if domain experts are not available, or if the system has become so complex a realistically sized group of experts cannot diagnose it without system-level inference tools. Alternatively, it may be possible that focusing model-based diagnosis technology on both the design process and on-line diagnsosis could lead to greater success, as suggested by the TEAMS and TEAMS-RT systems in Related Work. We believe a similar analyis of the expected and actual impact of these or any other applications of model-based diangosis technology would be of interest to potential customers, advocates and developers.

For potential customers of model-based diagnosis or other similar technologies, we hope this paper might inspire ways of thinking about the effective value to a project or mission. For advocates, we hope this paper might gather common criticisms of model based diagnosis technology into specific categories which can be addressed through research and development or rebutted through counter-examples or demonstration.

12. RELATED WORK

In this section we discuss a small set of systems that are model-based, or pertain to diagnosis and recovery, or have been fielded in operational use, our ultimate interest being a system that is all three.

There are many commercially successful diagnosis applications. For example, every car sold in the United States since 1996 is required to comply with the On-Board Diagnostics II (OBDII) standard, which specifies a set of onboard tests and makes diagnostic information available to off-board diagnostic systems. Some vehicles are even able to perform some amount of active testing, and have a "limp home" mode when sensors or engine control actuators are suspect [26]. However, what we are specifically interested in is systems that generate diagnoses on-line, based on some generic engine or set of principles plus a domain description of some sort, rather than systems where engineering analysis is encoded into static code or a table representing a recovery policy.

Researchers at Xerox PARC developed a model-based system for planning and scheduling print jobs within reconfigurable high Xerox end printers [27] that has been used in Xerox products since 1995. Engineers developing new variations of the machines write constraints between the sheets of paper and components in the printing process, and do not explicitly write a schedule or scheduling software for the machine. For each print job, a model-driven scheduler within the printer develops a schedule that is optimal for the job characteristics and the constraints imposed by the components available in the machine. This represents a commercially successful deployment of model-based reasoning, but does not appear to include any diagnosis, recovery or handling of anomalies operation of the machine.

Researchers at PARC did develop a model-based diagnosis system for copiers[28] with the intention of deploying it on laptops to assist field technicians responsible for diagnosing and repairing copiers. After being presented to technicians, the system was not deployed and a community knowledge sharing system, Eureka was deployed instead [11]. To paraphrase the Eureka paper, the model-based diagnoser was not deployed because technicians knew how to identify and correct common faults, and small optimizations in that process were not of high value. The real issue was unexpected issues that were not forseen during the design of the machine or development of the diagnostic models. These might arise from operating the machine in extreme environments, unintentional interactions in components in a newly released design, unanticipated failure modes as the machines age, and so on. Thus, after a study of the technicians' work process Eureka did not attempt to augment or replace the expert technicians' ability to perform diagnostic reasoning. Eureka is a knowledge management system that allows technicians to exchange tips on new faults, diagnoses and responses as they are created in the field, the success of which can be judged by its 20,000 users.

Researchers have developed a real-time model-based reasoning system for the RASCAL flight research aircraft (the UH-60 Helicopter) [29] based on the TEAMS and TEAMS-RT systems [30]. Given a model of signal propagation between components and the placement of sensors within a system, TEAMS performs off-line testability analysis to analyze and quantify which component failures can be detected and which can be further isolated, as well as making recommendations to improve testability. TEAMS has been used for testability analysis of several large aerospace systems. TEAMS-RT uses the same model to determine which components may be failed from pass/fail outcome of sensor tests. We were unable to find references indicating whether TEAMS-RT has been used thus far in a fault protection system or ground-based diagnostic aid during baseline aerospace operations. Our conjecture is that TEAMS-RT would enjoy a reduction in modeling cost relative to Livingstone-style inference, especially if TEAMS were used for design-time testability analysis. However, we suspect the question would remain about what value model-based, component-level diagnosis provides if the domain has infrequent, unanticipated failure scenarios, the need for carefully crafted recoveries, and experts available to perform diagnosis.

The very successful Cassini mission, whose main propulsion system was later used as a benchmark problem in development of Livingstone, made use of a very capable rule-based fault diagnosis and recovery system in operations [31], [14]. That is, through a FMECA process, the set of critical failures, the symptoms or monitored sensor values that would result, and the appropriate responses were derived. These mappings from monitored values to diagnosed states were encoded in rules. The appropriate commands to respond to each state were similarly encoded. For situations that could not be mapped directly from the sensor states to a diagnosis, the spacecraft would temporarily be set to a simple, safe state then the spacecraft would execute a sequence of commands designed to reveal the problem or move the spacecraft from the safe state to an operational state.

The EO1 [21], 2003 Mars Exploration Rovers[32], 2007 Phoenix Mars Lander [33] and 2009 Mars Exploration Rover [33] all used or are preparing to use model-based planning and scheduling software in routine operations. Thousands of daily operational plans have been generated for the Mars Exploration Rovers, and Casper planner was the basis of a low cost mission extension for the EO-1 spacecraft. We believe some of the differences in impact between the conceptually similar technologies of model-based planning and model-based diagnosis can be put into the context of our analysis. First, the likelihood of the system being called into use, P_f in our analysis, is 1.0 in the planning case, as the planner is typically used to generate plans for routine operations. Contrast this with the diagnosis system, where P_f is the product of the likelihood of each failure and the likelihood that the failure was *a priori* covered by the diagnostic model. Second, based on our experience with diagnosis and planning, the cost

of modeling, C_m, is lower for planning. The planner model concerns only nominal operations of the spacecraft, which is typically well understood, often well documented, and in some cases can be translated into a model from existing mission artifacts [33]. Compare this with model-based diagnosis, where the modeling task is to write a set of models that capture, again *a priori*, a set of relevant failures and how the failure signals are propagated across the system once it stops behaving according to its nominal model.

A number of analytic approaches exist for examining the cost/benefit tradeoffs of Integrated Vehicle Health Management (IVHM) or Integrated System Health Management (ISHM) technologies [8], [9], [10]. IVHM is broader and somewhat orthogonal to model-based diagnosis and recovery as we have considered it, in that IVHM is typically concerned with all aspects of supporting operations of one or a fleet of systems. The focus of IVHM is typically on increasing operational availability and reducing maintenance and support costs, with somwhat less emphasis on on-line diagnosis of an operational system to allow it to continue a specific sortie or mission. Williams for example describes a discrete event simulation that can compute different quantitative measures of effectiveness such as missions completed or number of vehicles in maintenance in a given period [8]. The tool is aimed at analyzing the sensitivity of various operational concepts to given assumptions about the performance of various IVHM capabilities. This paper suggests a very similar, less mature but potentially more specialized method for for evaluating model-based diagnosis specifically. It also focuses on how assumptions about performance and the customer's operational model (the input to these analysis methods) led to over estimating the impact of model-based diagnosis and recovery, and what factors in hindsight might have allowed a more accurate cost/benefit analysis.

ACKNOWLEDGMENTS

The inspiration for Livingstone and the original design and implementation are due to Brian Williams and Pandu Nayak. Pandu Nayak and the first author developed L2. L2 is now maintained by Lee Brownstone at NASA Ames Research Center. We would like to thank Greg Dorais, Scott Poll, Peter Robinson, Adam Sweet, Sandra Hayden, Lorraine Fesque and Thomas Starbird for discussing early versions of the analysis in this paper. Three anonymous reviewers also provided valuable feedback to improve this paper. The Livingstone and L2 applications mentioned in this paper were created through the efforts of a great many talented people including Lee Brownstone, Scott Christa, Charlie Goodrich, Sandra Hayden, William Larson, Scott Poll, Adam Sweet, Bill Millar, Mark Schwabacher, Rich Washington, and others.

REFERENCES

[1] R. Reiter, "A theory of diagnosis from first principles," *Artificial Intelligence*, vol. 32, no. 1, pp. 57–96, 1987, reprinted in [17].

[2] J. de Kleer and B. C. Williams, "Diagnosis with behavioral modes," in *Proceedings of IJCAI-89*, 1989, pp. 1324–1330, reprinted in [17].

[3] B. C. Williams and P. P. Nayak, "A model-based approach to reactive self-configuring systems," in *Procs. AAAI-96*, 1996, pp. 971–978.

[4] J. Kurien and P. P. Nayak, "Back to the future with consistency based trajectory tracking," in *Proceedings of AAAI-00*, 2000.

[5] D. E. Bernard, G. A. Dorais, C. Fry, E. B. G. Jr., B. Kanefsky, J. Kurien, W. Millar, N. Muscettola, P. P. Nayak, B. Pell, K. Rajan, N. Rouquette, B. Smith, and B. C. Williams, "Design of the remote agent experiment for spacecraft autonomy," in *Procs. IEEE Aerospace*, 1998.

[6] W. Maul, A. Chicatelli, C. Fulton, Balaban, A. Sweet, and S. Hayden, "Addressing the Real-World Challenges in the Development of Propulsion IVHM Technology Experiment (PITEX)," in *AIAA 1st Intelligent Systems Technical Conference*. AIAA, Sep. 2004.

[7] S. Hayden and A. Sweet, "Livingstone Model-Based Diagnosis of Earth Observing One," in *AIAA 1st Intelligent Systems Technical Conference*. AIAA, Sep. 2004.

[8] Z. Williams, "Benefits of ivhm: An analytical approach," in *2006 IEEE Aerospace Conference Proceedings*, 2006.

[9] G. J. Kacprzynski, M. J. Roemer, and A. J. Hess, "Health management system design: Development, simulation and cost/benefit optimization," in *2002 IEEE Aerospace Conference Proceedings*, 2002.

[10] C. Hoyle, A. Mehr, I. Y. Tumer, and W. Chen, "Cost benefit quantification of ishm in aerospace systems," in *2007 ASME International Design Engineering Technical Conference*, 2007.

[11] D. G. Bobrow and J. Whalen, "Community knowledge sharing in practice: The eureka story," *Reflections, the Journal of the Society for Organizational Learning*, vol. 4, 2002.

[12] T. C. Neilson, "MER on-board surface fault protection," in *Proceedings of IEEE SMC 2005*. IEEE, October 2005. [Online]. Available: http://hdl.handle.net/2014/38695

[13] G. E. Reeves and T. C. Neilson, "The Mars Rover Spirit FLASH anomaly," in *Proceedings of IEEE Aerospace Conference*. IEEE, 2005. [Online]. Available: http://hdl.handle.net/2014/39361

[14] G. Brown, D. E. Bernard, and R. D. Rasmussen, "Attitude and articulation control for the cassini spacecraft: A fault tolerance overview." in *AIAA/IEEE Digital Avionics Systems Conference*, 1995.

[15] G. E. Reeves, "An overview of the Mars Exploration Rovers flight software," in *IEEE Systems, Man and Cybernetics International Confer-*

ence. IEEE, October 2006. [Online]. Available: http://hdl.handle.net/2014/37499

[16] J. Matijevic and E. Dewell, "Anomaly recovery and the mars exploration rovers," in *SpaceOps 2006*. AIAA, 2006.

[17] W. Hamscher, L. Console, and J. de Kleer, *Readings in Model-Based Diagnosis*. San Mateo, CA: Morgan Kaufmann, 1992.

[18] N. Muscettola, P. P. Nayak, B. Pell, and B. C. Williams, "Remote Agent: To boldly go where no AI system has gone before," *Artificial Intelligence*, vol. 103, pp. 5–47, 1998.

[19] G. A. Dorais, S. D. Desiano, Y. Gawdiak, and K. Nicewarner, "An autonomous control system for an intravehicular spacecraft mobile monitor prototype," in *Proceedings of the 7th International Symposium on Artificial Intelligence, Robotics,and Automation in Space*, 2003.

[20] J. L. Bresina, K. Golden, D. E. Smith, and R. Washington, "Increased flexibility and robustness for mars rovers," in *Fifth International Symposium on Artificial Intelligence, Robotics, and Automation in Space*, 1999.

[21] S. Chien, R. Sherwood, D. Tran, R. Castano, B. Cichy, A. Davies, G. Rabideau, N. Tang, M. Burl, D. Mandl, S. Frye, J. Hengemihle, J. DAgostino, R. Bote, B. Trout, S. Shulman, S. Ungar, J. VanGaasbeck, D. Boyer, M. Griffin, H. Burke, R. Greeley, T. Doggett, K. Williams, V. Baker, and J. Dohm, "Autonomous Science on the EO-1 Mission." in *Procs. of the 7th International Symposium on AI, Robotics and Automation in Space (i-SAIRAS),Nara, Japan*, 2003.

[22] F. Drake, "The e.t. equation, recalculated," *Wired*, December 2004.

[23] F. Drake and S. Dava, *Is Anyone Out There? The Scientific Search for Extraterrestrial Intelligence*. Delacorte Press, 1992.

[24] A. Cimatti, C. Pecheur, and R. Cavada, "Formal verification of diagnosability via symbolic model checking," in *International Joint Conferences on Artificial Intelligence*, 2003.

[25] P. Nayak, D. Bernard, G. Dorais, E. Jr, B. Kanefsky, J. Kurien, W. Millar, N. Muscettola, K. Rajan, N. Rouqette, B. Smith, W. Taylor, and Y. Tung, "Validating the ds1 remote agent experiment," in *Fifth International Symposium on Artificial Intelligence, Robotics and Automation in Space*, 1999.

[26] C. O. Probst, *How to Understand, Service, and Modify Ford Fuel Injection Electronic Engine Control*. Robert Bentley, 1993.

[27] M. P. J. Fromherz, D. G. Bobrow, and J. de Kleer, "Model-based computing for design and control of reconfigurable systems," *AI Magazine*, 2003.

[28] D. G. Bell, D. G. Bobrow, B. Falkenhainer, M. Fromherz, V. Saraswat, and M. Shirley, "Rapper: the copier modeling project," in *International Logic Programming Symposium*, 1991.

[29] A. Patterson-Hine, W. Hindson, D. Sanderfer, S. Deb, and C. Domagala, "A model-based health monitoring and diagnostic system for the uh-60 helicopter," in *Americal Helicopter Society 57th Annual Forum*, 2001.

[30] C. Deb, K. R. Pattipati, V. Raghavan, M. Shakeri, and R. Shrestha, "Multi-signal flow graphs: A novel approach for system testability analysis and fault diagnosis," in *IEEE AES Systems Magazine*, 1995.

[31] J. Hackney, D. E. Bernard, and R. D. Rasmussen, "The cassini spacecraft:object oriented flight control software," in *Guidance and Control Conference*, 1993.

[32] J. Bresina, A. Jnsson, P. Morris, and K. Rajan, "Activity Planning for the Mars Exploration Rovers," in *Fourteenth International Conference on Automated Planning and Scheduling*, 2005.

[33] A. Aghevli, A. Bachmann, J. Bresina, K. Greene, B. Kanefsky, J. Kurien, M. McCurdy, P. Morris, G. Pyrzak, C. Ratterman, A. Vera, and S. Wragg, "Planning Applications for Three Mars Missions with Ensemble," in *International Workshop on Planning and Scheduling for Space*, 2006.

James Kurien received his B.S. and M.S. degrees in Computer Science from Rensselaer and M.S. and Ph.D. from Brown University. From 1996 to 2001 he contributed to the Livingstone and L2 systems at NASA Ames Research Center. He served on the development and flight operations teams for the Remote Agent experiment which tested Livingstone, a planner and an executive as an experiment on the Deep Space 1 spacecraft. From 2001 to 2003, he was a member of the research staff at Xerox PARC. In 2003, he returned to NASA Ames Research Center to lead development of a tactical planning system to be employed on the 2007 Phoenix Mars Lander and 2009 Mars Science Laboratory. Aspects of the of the system are in daily use by the extended mission of the Mars Exploration Rovers.

Maria Dolores R-Moreno received an M.S. degree in Physics from the Universidad Complutense de Madrid and her Ph.D. in Computer Science from Universidad de Alcalá. She subsequently spent one year at NASA Ames Research Center as a postdoc, and nine weeks at ESA's European Space Research and Technology Centre (ESTEC). She is currently a professor in the Computer Science Department at the Universidad de Alcalá.