Applying Model-Based Reasoning to the FDIR of the Command & Data Handling Subsystem of the International Space Station

Peter Robinson[1] Mark Shirley[2] Daryl Fletcher[3] Rick Alena[2] Dan Duncavage[4] Charles Lee[3]

NASA Ames Research Center[2]/Johnson Space Center[4]

QSS Group Inc[1], SAIC[3]

Moffett Field, CA. 94035

probinson@mail.arc.nasa.gov

Keywords model-based reasoning, testability, FDIR, fault detection, hardware and software faults, International Space Station, command and data handling, caution and warning, rate monotonic scheduling

Abstract

All of the International Space Station (ISS) systems which require computer control depend upon the hardware and software of the Command and Data Handling System (C&DH) system, currently a network of over 30 386-class computers called Multiplexor/Dimultiplexors (MDMs)[18]. The Caution and Warning System (C&W)[7], a set of software tasks that runs on the MDMs, is responsible for detecting, classifying, and reporting errors in all ISS subsystems including the C&DH. Fault Detection, Isolation and Recovery (FDIR) of these errors is typically handled with a combination of automatic and human effort.

We are developing an Advanced Diagnostic System (ADS) to augment the C&W system with decision support tools to aid in root cause analysis as well as resolve differing human and machine C&DH state estimates. These tools which draw from sources in model-based reasoning[16,29], will improve the speed and accuracy of flight controllers by reducing the uncertainty in C&DH state estimation, allowing for a more complete assessment of risk. We have run tests with ISS telemetry and focus on those C&W events which relate to the C&DH system itself. This paper describes our initial results and subsequent plans.

1. Introduction

The Aerospace Safety Advisory Panel (AVSP)[3] identified the C&DH system as a critical system that needs continued attention. Even as the ISS complexity increases, budget pressures may require scaling back of the ISS ground support team and the ISS crew[2]. There is a strong need for easy-to-configure automation tools which assist the ground and flight teams to assess complex situations, suggest recovery options[9] and track the response of automated C&DH FDIR.

The paper begins by presenting an overview of the hardware and software of the C&DH system. At ISS assembly complete, the C&DH system will be made up of over 60 MDMs executing 1 million lines of Ada flight software tasks. Each MDM executes the Ada tasks according to a rate monotonic schedule (RMS)[5]. We highlight the challenging nature of C&DH system FDIR, including high rates of false-positive C&W events[3] and unexplained hardware and software anomalies referred to in Problem Report and Corrective Action (PRACA) documents[1,8]. (Figs. 1,2,3,4)

We use structural dependency models of the C&DH system made up of a network of hardware/software paths of components which govern both nominal and off-nominal modes of behavior. The components consist of MDMs, their internal boards, buses and software structure. This information is derived from hardware schematics[18], standard out (STDOUT)[25], C&W fault trees[19], as well as software source code.

The models are utilized by both symptom- and simulation-based tools. The symptom-based tool TEAMS[17,29], from the testability community, over-

lays functional signal/test component relationships over structural-dependency models. The test points are evaluated from observations. Diagnosis is determined by tracing back from all active test points to those components which contain the test point signals. The simulation-based tool L2[15,16], from the model-based diagnosis community, propagates "pieces of stuff" (material, electricity, heat, information) through the structural dependency networks in order to compare predictions of C&DH behavior against ISS C&DH observations. Diagnosis is determined by tracing back to components whose predictions lead to inconsistencies with observations (conflict generation). Through a process called candidate generation, L2 derives a diagnosis by toggling the nominal and off-nominal modes of the failed components in order to achieve a consistent set of mode assignments to all components which accounts for all observations (and commands).

By integrating TEAMS and L2, we can close the FDIR loop. TEAMS and L2 provide state estimation capabilities (at different levels of abstraction), and have the ability to continue to perform state estimation in the face of partial data dropouts. L2 also provides regulation capabilities. TEAMS addresses issues of scale-up and speed of diagnosis for the whole C&DH system. L2 can automatically select the active paths of the structural dependency models either through commands or inferring reconfiguration. L2 provides capabilities to explain its reasoning processes.

We present two scenarios derived from Guidance Navigation and Control (GNC) C&DH events which occurred on days 88-90 2002. The first scenario demonstrates a model-based method of determining whether two C&W events have a common root cause. We map each C&W event to the subset of the ISS hardware components for which it is responsible. We employ set covering methods over these subsets to determine potential root causes for C&W events. If the data from two C&W events can be accounted for by a single component of the model, we assume the C&W events have a common root cause. (Figs 5,6,7,8).

A second scenario addresses the issue that C&W events for MDM failure can be ascribed to both hardware and software causes. We provide a method to trace information dependencies over time between Ada tasks and the global shared memory called the Current Value Table (CVT) (Fig 10.). We accomplish this by extending the methods of model-based diagnosis (MBD)[15,16] to model software components [11,12,13] as well as hardware components. We model the timing of Ada tasks and their I/O with the CVT. This allows us to shadow the execution of the Ada tasks in order to track software errors. When Ada task exceptions occur, we will provide software dependency information to determine which portions of the CVT to focus in order to discriminate between competing root causes of exceptions.(Figs. 10,11,12)

We conclude by comparing our approach to other tools (CRANS[14], PEM cells[21]) and address issues of future work.

2.Command & Data Handling System (C&DH)

The function of the C&DH system is to provide hardware and software to support command and control of the ISS, services for flight and ground operations, and science payloads. This is achieved through a three tiered network of computers(MDMs) all running Ada tasks and interconnected by MIL-STD 1553B data buses (Fig. 1). The Tier 1 MDMs run Command and Control Software (CCS) that controls system-wide functions such as ISS mode. The Tier 2 MDMs are responsible for subsystem level functions for Electrical Power Systems (EPS), Guidance Navigation and Control (GNC), Environmental Control and Life Support Systems (ECLSS), Thermal Control System (TCS) as well as others. Tier 3 MDMs interact with the multitude of sensors and effectors onboard the ISS.

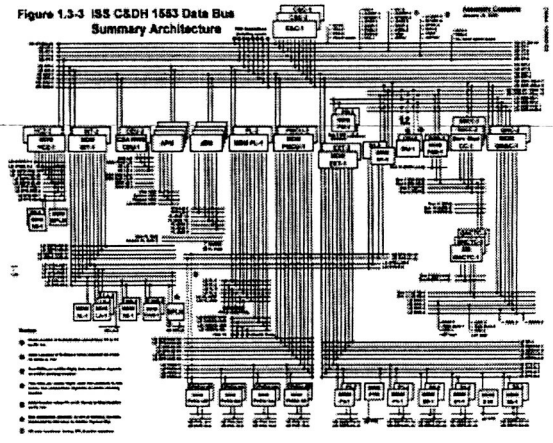

Figure 1. The C&DH system consists of a three-tiered hierarchy of networked computers and buses.[18]

Each MDM consists of a power supply and an IOCU (Input/Output Control Unit) card that contains the 386 SX processor and the 1553 Bus Interface Adaptor (BIA) to connect the MDM to upper tier MDMs. The MDM can also be configured with up to five I/O cards and five 1553 Serial-Parallel-Digital (SPD) network cards. Each SPD card connects the MDM to lower tier MDMs via the SPD card. For example, a GNC MDM (Fig. 2) has an IOCU with a BIA connected to upper tier bus CB_GNC_1 as well as five SPD 1553 cards. Each SPD card is attached to two buses. For example, SPD1 card attaches to buses LB_GNC_2 and LB_TS_1.

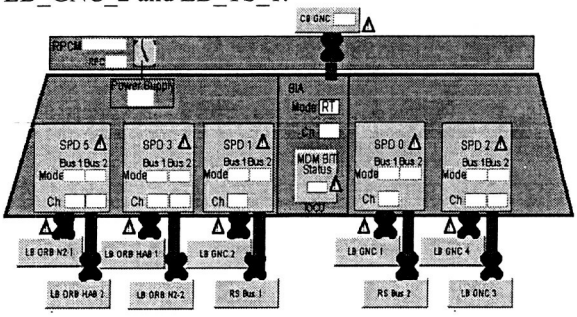

Figure 2. The GNC MDM model of an MDM from a PCS display[28]. The MDM has a power supply card, an IOCU card, and 5 SPD network cards.

When an MDM is booted-up and its software is loaded, a cyclic scheduler is invoked. The cyclic scheduler is responsible for executing Ada tasks according to a rate monotonic schedule (RMS)[5]. The C&DH system has an internal rate of 80 Hz, with all software running at 10Hz, 1Hz and 0.1 Hz rates. If the task rates are multiples of each other a feasible RMS schedule can more easily be achieved. The Ada tasks communicate via a global shared memory called the current value table (CVT) (Fig. 3). The propagation of commands and data through the network is accomplished through a set of realtime Ada routines for I/O and 1553 communication where the commands and data cyclically read from and write to the CVT, in service of MDM specific User Application Software (UAS).

When Ada tasks fail, exceptions are thrown by tasks and caught by exception handlers within the dynamic scope of the task (Fig. 4). These exception handlers respond to a variety of known error conditions including invalid data for sensors, bus and command data parameters, task overruns and watchdog timer timeouts. When failures occur, the MDMs usually transition to the diagnostic state and are taken offline. Frequently occurring Ada task exceptions are addressed with counters for classes of exceptions. For those classes, transition to diagnostic mode occurs only when an accumulated threshold is crossed

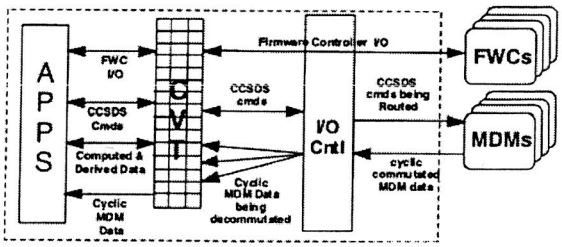

Figure 3. Ada tasks communicate via a global shared memory called the current value table (CVT)[27].

Ada tasks can fail for several reasons. The loss of hardware components that software routines depend upon can cause an Ada task to wait until it times out. Overruns can cause it to experience constraint violation and divide-by-zero errors. Violating any of the assumptions of rate monotonic scheduling (RMS) can also cause exceptions. Liu and Leyland[5] identify the assumptions: (1) the requests for all tasks are periodic, (2) tasks are independent and non-interacting, (3) execution time for each task is constant, (4) each task must be complete before the next request for it occurs and (5) task switching is instantaneous. Finally, the sheer complexity of the C&DH software is at the root for additional Ada task failures. According to [6], "It is not possible to achieve 100% test coverage [of the software] due to the enormous number of permutations of states in a computer program execution, versus the time it would take to exercise all those possible states. Also there is often a large indeterminate number of environmental variables, too many to completely simulate".

We will focus on classes of **unexplained C&DH MDM failures** (PRACAs[8] #2593, #3031, #3019[1]). These problem reports state: "There is insufficient data to determine the root cause"[1], due to unknown hardware/software interactions. The integration of model-based diagnosis methods with techniques from program slicing[11,12,13] provides an approach to combining hardware and software FDIR within a sin-

gle approach. We plan for our system to shadow the execution of the Ada tasks to extend software V &V to the runtime, operational environment (through the use of framecount and checkpoint data). When an exception handler is invoked, we can provide software dependency information to determine the root cause and determine which portions of the CVT to focus on to discriminate between competing fault hypotheses (e.g. Fig. 10). We will rely upon the use of the MDM Application Development Environment (MADE)[20] to the emulate of suspect Ada tasks and provided insight into their anomalous behavior.

```
Cyclic_Task
  begin
    task_processing;
    exception when
              constraint_error | numeric_error |
               program_error| storage_error |
               task_error =>
               Call Ada Exception Handler;
              when others =>
               Call Default Exception Handler;
   end;
   ... Call task_overrun_handler
end Cyclic_Task
```

Figure 4. Template of realtime cyclic Ada task [27]

3. Caution and Warning (C&W) System

The primary fault detection system of the C&DH system is the C&W system[7]. This system executes at a 10 Hz rate, cyclically evaluating each of 10,000 C&W events which characterize the ISS state. For the C&DH system two important classes of C&W events are MDM failure and bus failure events (Figs. 7,8). Each C&W event is defined by a fault tree whose logic reflects a complex and/or description of low-level ISS telemetry parameters as well as the results of intermediate trees. The fault trees of the C&W events have a limited notion of context, which has caused the C&W system to experience an unacceptably high **false-positive rate.** C&W events are false positive for several reasons including when: 1) sensors bin in/out of nominal due to incorrect limits, or 2) operations are performed such as depressurization which cause sensors to leaves nominal range, or 3) cascading failures occur such as power supply failure leading to a string of dependent failures. Often, string of dependent failures. Often, mission controllers turn off false-positive C&Ws; however, the Aerospace Safety Advisory Panel has stated[3]: "Avoid the need to inhibit C&W alerts by countering the root causes of false alarms."

4. Approach

We seek to address these two classes of problem: **unexplained C&DH PRACAs** and **unacceptable false-positive C&W rates** by the use of dependency tracking tools which can trace C&W events to root causes through the hardware/software paths governing their operation. We base our approach on the C&DH ground handbook which instructs flight controllers to respond to MDM failures by dumping a set of onboard logs and buffers to augment the standard cyclical telemetry [10]. We rely on the Diagnostic Data Server (DDS) [4] to parse these logs and temporally organize the ISS events from cyclic telemetry, logs files and 1553 bus messages. We also base our approach on the C&DH flight handbook[31], which instructs the astronauts to respond to a partial loss of an MDM by addressing each C&W in the order it was received. Our tools will help organize C&W events by root cause.

5. Are Two C&W Events Related?

To determine whether any set of C&W events have a common root cause, we derive a diagnosis with all the data related to the set of C&W events of interest. If the minimal diagnosis is a single fault, we assume the set of C&W events has a single root cause. This accomplished by first pre-computing for each C&W event the subset of components of the C&DH components for which it is responsible. Once the components sets are defined for all C&W events of interest, we utilize set covering methods over all the C&W event component sets. A non-nil intersection is a necessary but insufficient condition for determining if C&W events are related. This due to the fact that the intersection of the component sets for each C&W, is a static analysis of the topologies, without the introduction of command and sensor information from parameters in the C&W fault trees as well as logs dumped to MCC for analysis which could further reduce intersection.

We explore this process with two C&W events related to the GNC MDM system which took place on days 88 and 90 in 2002. As it turns out, C&Ws 5392

and 5014 are **not** related; i.e. the minimal diagnosis is double fault. We briefly introduce the portions of the C&DH/GNC topology necessary for this example and then step through the process of determining dependencies between these C&W events.

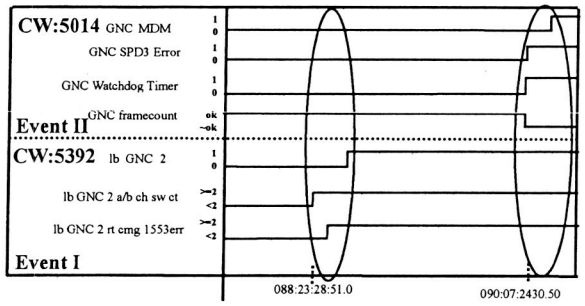

Figure 5. Telemetry parameters used by Event I (CW# 5392) and Event II (CW# 5014).

The GNC MDM has five network cards (Figs. 2,6 (Event II)). One of these network cards, SPD1, is connected to the LB_GNC_2 bus as the bus controller (BC) (Fig 6 (Event I)). The BC can send/receive information via A/B redundant channels to a set of three remote terminals (RTs) connected to GNC devices called the reaction gimbal (RGS), the control momentum gyro (CMG) and the global position system (GPS).

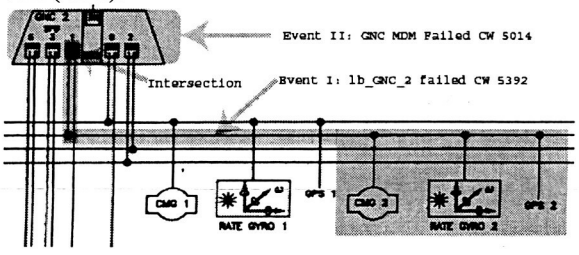

Figure 6. The components involved in Events I and EII have an intersection at network card SPD1.

At 11:00 pm on day 88 of 2002, the 1553 error count for RT CMG on bus LB_GNC_2 exceeded its limits. (Fig. 5, Event I). Automatic bus FDIR took over and activated the A/B channel switch to determine if switching channels would stop the RT CMG 1553 error messages. It did not, and soon the A/B channel switch counter exceeded its limits continually trying to address RT CMG. This caused Event I: C&W event 5392 (LB_GNC_2 failed) to be raised. Over a day passed, then the network card SPD3 fails at the same time as a watchdog timer (WDT) error occurs. These events cause the GNC MDM to go into diagnostic mode, which causes the framecount to stop, which causes Event II: C&W event 5014 (GNC MDM failed) to be raised.

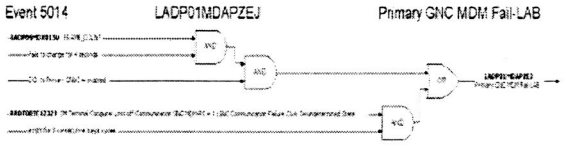

Figure 7. C&W Event 5014: GNC MDM Fail: "**If** the frame count has not changed for four seconds and I/O is ok **then** GNC MDM is failed". [19]

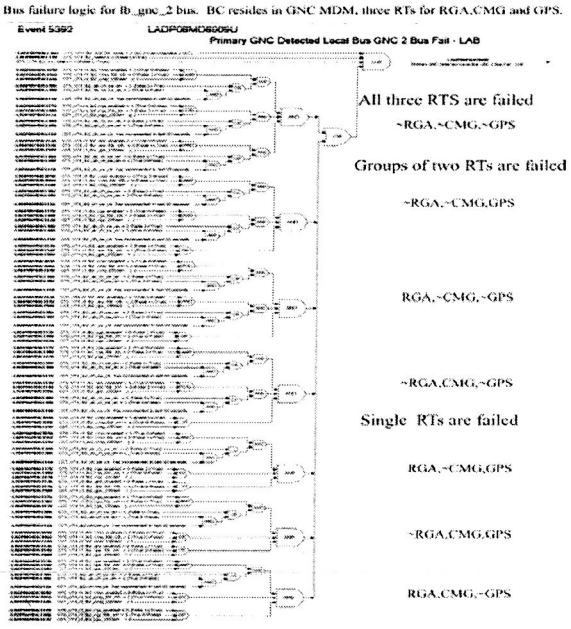

Figure 8. C&W Event 5392: LB GNC2 Bus Fail: "**If** any of the three bus RTs are failed **then** LB_GNC_2 bus is failed" [19]

We develop models of an MDM and bus required to build the structural dependency models. The MDM model is developed from the connectivity information in the GNC MDM definition (Fig. 2,6 (Event II)) as well as the fault tree for C&W event 5014(Fig. 7). We represent both *working* and *~working* models of the MDM. The *working* model states that the MDM is working provided that the framecount is changing (dframecount) and that all of its internal components

are nominal: ((dframecount <> 0) ^ SPD1 ^ SPD3) . The *~working* model is derived using De Morgan's laws from the working model: (dframecount = 0) v ~SPD1 v ~SPD3). For the sake of this example, we have only included in the GNC the components needed for our scenario (Fig 9).

The bus model is developed from the bus connectivity information of the LB_GNC_2 bus (Fig. 6, Event I) and the fault tree for C&W event 5392 (Fig. 8). The fault tree in Fig. 8 states that the bus is failed if any of its RTs (CMG, RGA and GPS) or the BC is failed. (see large text in Fig. 8). The ~working model of the bus components is developed by reducing the fault tree propositions to prime implicant form through the use of a 4-variable Karnaugh map: *~working*: (~SPD1 v ~RGA v ~CMG v ~GPS). The *working* model is derived by use of De Morgan's laws: *working*: (SPD1 ^ RGA ^ CMG ^ GPS). We integrate of the MDM and bus models in a structural dependency model in Figure 9. Infusing this model with telemetry and dump log information allows us to determine dependencies between C&W events 5392 and 5014.

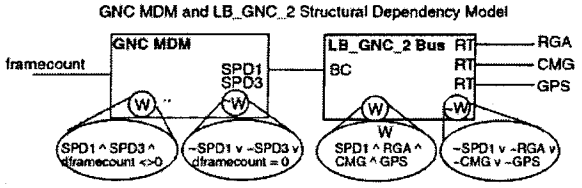

Figure 9. The structural dependency models of components covered by CW events 5014 and 5392.

6. MDM Simulation HW/SW Using Framecount

A closer look at the data from days 88 and 90 reveals that when the SPD1 network card failed, a software watchdog timer (WDT) tripped as well.. At present, the root cause of the WDT has not been determined [1]. By modeling the software execution and modification of memory (CVT) we hope to determine root causes for such errors. To perform tracing through software requires the capability to simulate the software processes and record their justifications. For example, in Figure 10., we show with the bold lines information dependencies between software tasks and shared memory elements of the CVT. Initially CVT(1,7) and CVT(2,6) are read by Ada Task1 at some framecount which writes out intermediate data product CVT(2,5). Ada Taskn reads this data product at a later framecount to produce CVT (3,2). In this manner, CVT(3,2) depends on CVT(1,7) and CVT(2,6).

The second scenario demonstrates this capability with a simulation which propagates a message a through an MDM (from BIA to SPD) by utilizing a CVT memory element. We have developed hardware and software components for this simulation and will highlight the software task component in this paper..

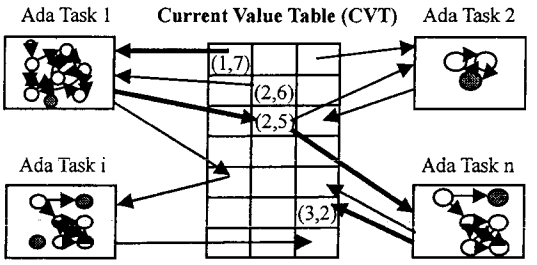

Figure 10. Memory location (3,1) depends upon memory locations (1,7) and (1,6) through location (2,5).

The scenario begins with the power supply status nominal (#1) and power-in nominal (#2). The power supply is turned on (#3) to provide power to the rest of the MDM including the processor (IOCU= BIA + DRAM) as well the SPD cards. The BIA is initialized as an RT (#4,5) followed by the SPD card which is reset and reinitialized (#6,7). When the MDM framecount starts (#8), Ada tasks execute when the framecount is within their nominal range (#9,13). The synchronized execution of a set of Ada tasks controls the propagation of information through the MDM. In the scenario, the information enters the MDM via the BIA input (from an upper tier computer or from MCC) (#9). A 1553 Ada task is scheduled to transfer the BIA value to its CVT. The CVT is given a write command (#11), with value to be stored on the write line. Once the memory latches, the Ada task stops executing and the value is available on CVT read line (#12). Two increments of the framecount later, and then another 1553 Ada task is executed which copies the CVT read data to SPD (#13) ready for propagation to a lower tier MDM.

We simulate the software at a high-level, not modeling the internals of the software but only the timing

as well as the inputs and outputs of the software. We assume that any output of a software routine could be dependent on all inputs. To define the timing, duration of the Ada task and elements of the CVT used by these tasks, requires an analysis of the Ada source code and its documentation[22,23,24,26,27].

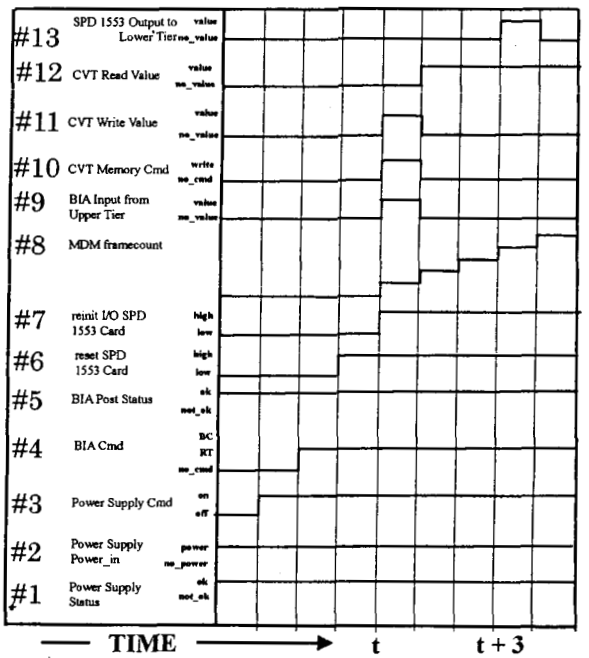

Figure 11. An MDM simulation to propagate data from BIA_in to SPD_out in 3 framecount increments.

Each Ada task is designed to execute on particular framecounts. The framecount is an integer number from 0 to 99, where each integer increment is 0.1 sec. This is how ISS ensures that Ada tasks do not read/write the CVT at the same time. We introduce metric time into MBR models to model the timed propagation of information through the C&DH system by augmenting traditional components from model-based reasoning with temporal preconditions. (Fig 12.). This allows us to explicity control the propagation of information in the structural dependency models. Instead of components propagating their data product at every tick of the diagnosis/simulation engine cycle, propagation only occurs when the framecount is within nominal range as defined by the Ada task component (i.e. start_time < framecount < stop_time).

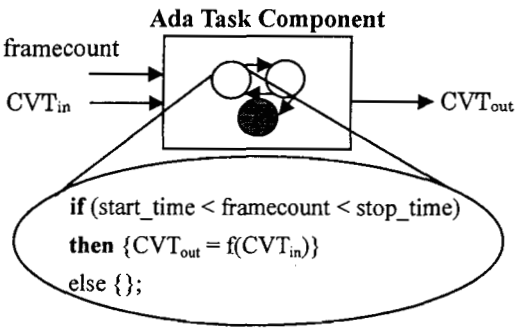

Figure 12. Each Ada task executes only when framecount is within time window: [start_time stop_time].

7. Discussion

We have presented our preliminary work towards defining an integrated hardware/software FDIR system to augment the ISS C&W system. We have addressed unexplained C&DH PRACAs as well as false-positive C&W rates, two serious issues in ISS data handling. We find that though ISS acknowledges that the state space of the software is too large to test at design time[6], no solution is provided for runtime software V&V. Advances in model-based diagnosis (MBD) to include software components [11,12,13] provides the roadmap for integrated hardware and software FDIR. Traditionally it is easier to model hardware components, because the physics underlying these models has been developed over hundreds of years (Kirchoff's laws, Bernoulli's equations), while paradigms for modeling software components is still being developed today (UML).

Scale-up challenges using simulation methods of MBD led us to develop synergistic strategies to utilize both the TEAMS and L2 tools. TEAMS [17,29,30,32] has been used for diagnosis of aerospace vehicles including portions the ISS and C&DH, while L2[15,16] has flown on the Deep Space 1 spacecraft with its single string 1553 bus network. Still, due to their coarse abstractions of the underlying structural dependency models, in the future we will utilize detailed simulations using the ISS software emulator MADE [23].

This work is related to CRANS (Configurable Realtime Analysis System)[14] dependency tracking, real-time monitoring, mission control tool. In CRANS, structural topologies of the system are encoded in logical form similar to fault tress. It is useful for

determining upstream causes and downstream effects of failued ORUs (orbital replacement units). But CRANS is difficult to configure, does not directly tie in with the C&W system and does not capture software dependencies. Future work could address automatic methods to configure CRANS from domain knowlegde in our tools. We also relate to Program Execution Montitor (PEM) cells [21] , as a "specialist software module to assist with detailed analysis" for state eastimation and recovery options as well as provide structural dependency models which can be used to generate PEM cells. Both [21] and [30] share out goal of augmenting the C&W system. The challengeall of these approaches face including the approach we have presented today, is to ensure that the systems can easily be configured for the current ISS stage[9].

5. Reference

[1] McCabe, J.J., *PRACA #3019 GNC-2 MDM Transitioned To Diagnostics - Unexplained Anomaly*

[2] Holloway, T., Waddell, B., *ISS Program Manager's Recommend for Program Operating Plan 2002*, Presentation to HQ 21 June 2002

[3] NASA Aerospace Safety Advisory Panel, Annual Reports for 2001,2

[4] Fletcher, D. P., Alena, R. *A Scalable, Out-of-Band Diagnostics Architecture for International Space Station Systems Support*, IEEE Aero 2003

[5] Liu,C.L., Layland J.L. *Scheduling Algorithms for Multi-Programming in a Hard Real-Time* ACM Vol 20, 1, 1973, pp. 46-61

[6] NASA-GB-1740.13-96 *Software V&V Challenges*

[7] Owens, D. , Dempsey, R .*Caution and Warning Systems* Brief NASA JSC DF25 CDH 06/02

[8] Problem Reports and Correction Action http://iss-wwww.jsc.nasa.gov/ss/issapt/praca

[9] O'Hagan, Brian ISS ODIN(C&DH) mission controller – personal communication 9/02

[10] JSC-48516-E1&E2 *ISS Ground Handbook All Expedition Flights MOD*, Sept 2000

[11] Hamscher, W.C., R. Davis. *Diagnosing Circuits with State: An inherently unconstrained problem* AAAI-84

[12] Mayer, W., et al. *Observations and Results gained from the JADE Project. DX '02*

[13]Wotawa, F. *On the Relationship Between Model-based Debugging and Program Slicing*, Artificial Intelligence Journal 135 (2002), pp125-143

[14] CRANS Users Manual – NASA JSC MOD

[15]Williams, B., Nayak P.P., *A Model-based approach to reactive self-configuring systems*. AAAI95.

[16] Kurien, J. , Nayak. P.P *Back to the Future for Consistency-based Trajectory Tracking*. AAAI-97

[17] Deb, S., Domagala, C.,Ghosal, S.,Patterson-Hine A., Alena , R. *Remote Diagnosis of the International Space Station utilizing Telemetry Data* SPIE April 2001

[18] D684-10500-04B CDH ADD Vols. 1-4.

[19] C&W Fault Trees: http://hsi.jsc.nasa.gov/Cwad/

[20] D684-11379-01 *MADE* Design Document 3/02

[21] Wang, L. *Cockpit Automation Architecture: Advanced Automation with Advanced Caution and Warning* NASA JSC/ER2 Jan 03

[22] S684-11032 *ISS Software Requirements: R2,, Command and Control (C&C) MDM (CSCI)* 11/01

[23]D684-11106-01 *ISS Command and Control Software , Software User's Manual R2* 10/01

[24] D684-10056-01K *ISS Prime Contractor Software Standards and Procedures Specification* 12/00

[25] D684-10177-01F *ISS Mission Build Facility Standard Output Definition*

[26]NAS15-10000 *Software Rqmtss Spec for the MDM Boot and Diagnostics Firmware of C&DH*

[27]D684-11111-01 *ISS Command and Control Software, Software Top Level Design Document R3*

[28] JSC 28721 *User's Guide for the Portable Computer System (PCS)* ISS MOD May 2001

[29] Deb S., Pattipati K., Raghavan V., Shakeri M., Shrestha R. *Multi-Signal Flow Graphs: A Novel Approach for System Testability Analysis and Fault Diagnosis* IEEE AES Systems Magazine, May 1995

[30] Aaseng G., Cavanaugh K.,Deb S. *An Intelligent Remote Monitoring Solution for the ISS*. IEEE 2003

[31] 3.101 Malfunction Methodology (Caution/6A – ALL/FIN 1/MULTI) NASA JSC MOD

[32] Deb S., Mathur A., Willet P, Pattipati K. *De-centralized Real-time Monitoring and Diagnosis* Proc. IEEE SMC Conference 1998