Livingstone Model-Based Diagnosis of Earth Observing One

Sandra C. Hayden [*] Adam J. Sweet[†]
QSS Group, Inc, Moffett Field, CA, 94035, USA

Scott E. Christa[‡]
AerospaceComputing, Inc., Moffett Field, CA, 94035, USA

The Earth Observing One satellite, launched in November 2000, is an active earth science observation platform. This paper reports on the development of an infusion experiment in which the Livingstone 2 Model-Based Diagnostic engine is deployed on Earth Observing One, to demonstrate the capability of monitoring the nominal operation of the spacecraft under command of an on-board planner, and to demonstrate on-board diagnosis of spacecraft failures. Design and development of the experiment, specification and validation of diagnostic scenarios and models, and the integration and test approach are presented.

I. Introduction

IN this experiment, Livingstone was uploaded to the Earth Observing One (EO-1) satellite to conduct diagnostic tests. EO-1 was developed and is operated by NASA Goddard Space Flight Center (GSFC). Livingstone is a diagnostic engine developed at NASA Ames Research Center (ARC) by the Model-Based Diagnosis and Recovery group and Brian Williams of MIT [1]. The original Livingstone flew on Deep Space 1 (DS1) as part of the Remote Agent autonomy experiment (RAX) in 1999 [2]. Since then, the MDBR group has created the next version of Livingstone, called Livingstone 2 (L2). As a technology infusion experiment, L2 and the spacecraft diagnostic models have been integrated with the Autonomous Sciencecraft Experiment (ASE) [3]. ASE was developed by NASA Jet Propulsion Laboratory (JPL), and first ran on-board EO-1 on September 20, 2003. The autonomy software consists of the Continuous Activity Scheduling, Planning, Execution and Replanning (CASPER) planner; the science event detection software and the Spacecraft Command Language (SCL), developed by Interface and Control Systems (ICS). SCL provides an executive, a database and the software bridge to the spacecraft's 1773 data bus. L2 provides a diagnosis component to ASE, not included before.

A range of development and integration activities were undertaken to support the experiment. The major challenges encountered and their resolutions are discussed. Tasks include development of L2 models of the EO-1 spacecraft and instrumentation, and failure scenario definition, based on knowledge acquired from GSFC. A Real-Time Interface (RTI) and corresponding modeling methodology were developed to account for communication delays and physical transients in the system. L2, models of the spacecraft, and the RTI were integrated with SCL and the CASPER planner, first on a PowerPC embedded system and then on EO-1's ground test bed, a pair of M-5 processors with the Virtual Satellite (VSat) simulation system. Diagnostic scenarios were validated prior to upload on the integrated system. System engineering of the overall autonomy software included allocation of VxWorks task priorities, SCL software bus bandwidth, CPU and RAM resources. In early September 2004, the combined L2 and ASE software was uploaded to EO-1, and checkout procedures of the L2 software began.

II. Livingstone Diagnostic Technology

The Livingstone algorithm and component-connection model are introduced here. A Livingstone diagnosis system consists of two main parts, a generalized inference engine and a domain-specific model. When Livingstone is deployed on different devices or vehicles, the inference engine does not change; only a new model needs to be developed. Livingstone uses a qualitative representation, propositional logic, to model the target system. The target

[*] Group Manager, Computational Sciences Division, NASA Ames Research Center, M/S 269-3, AIAA Professional Member.
[†] Research Engineer, Computational Sciences Division, NASA Ames Research Center, M/S 269-1, AIAA Professional Member.
[‡] Systems Engineer, NASA Ames Research Center, M/S 247-2

system may be physical, such as the spacecraft hardware, or logical, such as the spacecraft software. The model is used to predict the states of system components given their initial state, commands which affect the system, and possible mode transitions. If there is a discrepancy between observed and predicted behavior, this generates conflicts in Livingstone's internal belief state. These conflicts are then used to focus the search for component modes (including failure modes), which are consistent with the observed state of the world and the possible mode transitions in the model. This process is known as *conflict-directed best-first search*. The set of component modes, which is found to satisfy the constraints expressed in the model, is termed the mode vector.

A Livingstone component-connection model describes nominal and failure modes for components in terms of the propositional constraints that must hold in those modes. The connections are the constraints that must hold due to interactions between the components. Transitions between modes of a component are triggered by guard conditions such as commands, similar to a finite state machine representation. The model is qualitative due to the underlying propositional logic representation. Constraints are expressed as discrete variable-value pairs. Any real-valued sensor data must be transformed into qualitative data, "binned" by software called monitors, before being used by Livingstone. For the failure modes, the likelihood of the failure is indicated by a rank, an approximation of the prior probability. The mode vector describes the overall state of the system.

III. Architecture of the Diagnostic Experiment

The experiment architecture and configuration are shown in Figure 1.

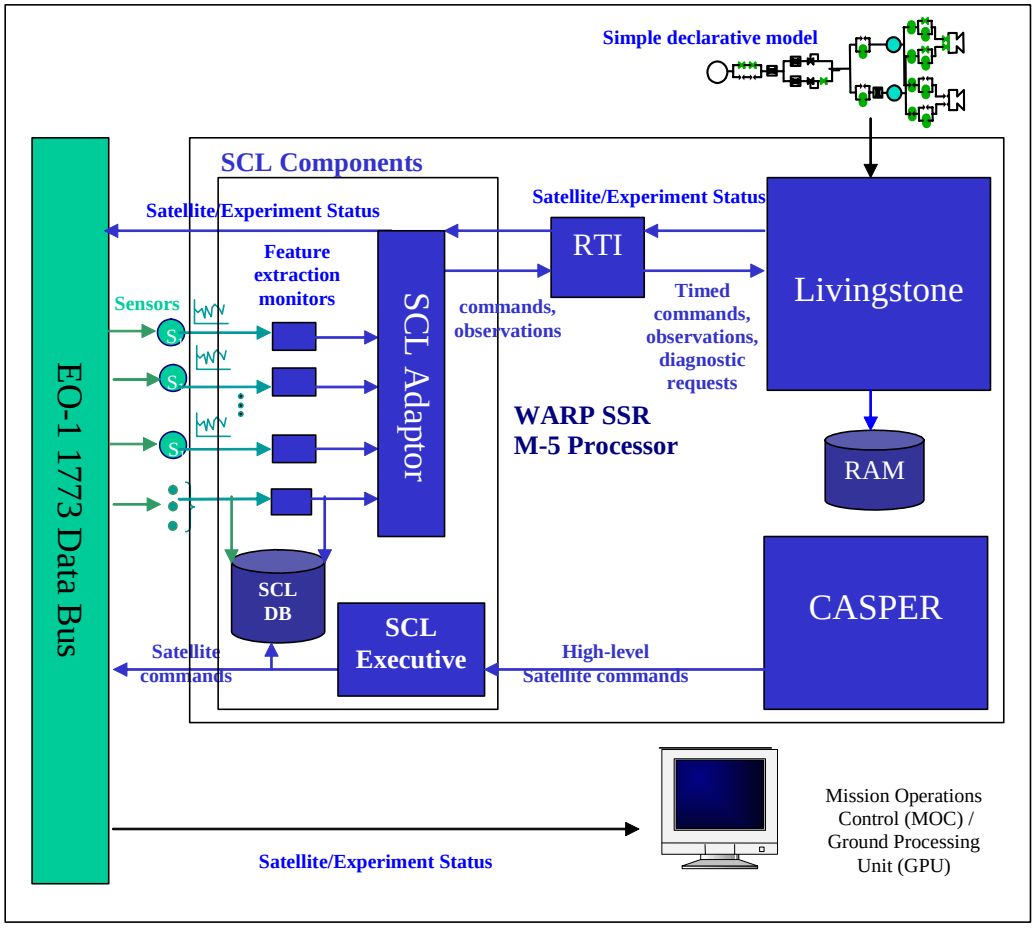

Figure 1: Architecture of the L2 on EO-1 Experiment.

L2 was integrated with the Autonomous Sciencecraft Experiment software architecture and infrastructure, and uploaded to the Wideband Advanced Recorder Processor (WARP) Mongoose-5 (M-5) processor on-board the EO-1 satellite. The experiment has the capability to process spacecraft telemetry and to downlink diagnostic health status telemetry for monitoring and display at the Mission Operations Center (MOC).

Briefly, the CASPER planner generates high-level plan scripts and sends them to the SCL Executive that generates a sequence of commands to execute the plan. The SCL Software Bridge connects applications on the WARP M-5 processor to the 1773 spacecraft data bus for all telemetry processing, including decommutation of incoming telemetry frames and support for telemetry downlink. Incoming data may be stored in the SCL Data Repository, with database triggers for notification of executed commands and received observations to subscriber processes such as L2.

L2 performs diagnosis using a qualitative model of the target system to predict observations given the commands issued to the system, postulating diagnoses to explain discrepant observations. A diagnosis consists of a group of failure candidates, their constituent modes and the likelihood of each candidate. The alternate candidates form an ambiguity group.

IV. Livingstone Model Development

An L2 model contains the device- or vehicle-specific information used in diagnosis. L2 models are created in a component-based manner: first components are defined, then connected together to create the overall system model. Components can also be contained in other components in the model (in a hierarchy), although internally L2 treats the model as flat.

As previously mentioned, L2 models are discrete. Variables can take a finite number of values such as "low", "medium", and "high", and components can contain a finite number of modes such as "on", "off", "failed on", and "failed off". Each component mode specifies qualitative constraints between the internal variables and the vehicle telemetry.

To perform diagnosis, L2 uses the commands given to the system to indicate component mode changes. Using the constraints in the model, L2 generates the expected telemetry values of the new system mode and compares them to the actual telemetry values. If the expected values and the actual values disagree, L2 determines that a fault has occurred. It then searches the space of component fault modes to find those consistent with the current observations; a diagnosis containing the set of consistent component faults is returned.

In general, the process of creating an L2 model has four steps: knowledge acquisition, scope definition, model creation, and model testing. In practice, these steps are often iterated before a model is complete.

A. Knowledge Acquisition

Knowledge acquisition is the process by which a modeler gathers information about a device or vehicle. It is usually the most time-consuming part of creating an L2 model. It simply takes time to understand the components of a vehicle and how they behave in nominal and fault conditions. In addition, if a system is still in the design stage, the information to capture in a diagnosis model is in flux or may not exist. Usually several sources of information are used to gain the knowledge needed: design specifications, schematics and Failure Modes, Effects and Criticality Analysis (FMECA), and, if available, the system designers themselves.

For EO-1, the models were created by the group at Ames, supported by the GSFC engineers. The forms of documentation mentioned above were used, as well as mission timelines and the EO-1 Spacecraft User's Guide. EO-1 has been flying for several years, which makes knowledge acquisition easier: the satellite hardware is not changing, the documentation is more complete, and the engineers have years of experience operating the spacecraft.

B. Scope Definition

Determining the scope of an L2 model involves deciding which vehicle components and component faults to include in the model, and the level of detail in which to model them. The scope of the EO-1 L2 model is a subset of the spacecraft components most relevant to the science data collection sequence: the two imaging instruments, called the Hyperion and the Advanced Land Imager (ALI), and the data recording device, called the WARP. To ease the integration of LEO-1 with ASE, the model scope was chosen to require only a subset of the commands and telemetry already made available to ASE by SCL. The telemetry values in use by ASE are mostly discrete "status bit" values. As a result, the EO-1 model is fairly high-level. More detailed models could be developed with additional work and by incorporating additional telemetry values (including continuous values).

C. Model Creation

After gaining knowledge of the vehicle and deciding the scope of the model, the model creation begins. As mentioned previously, while most real systems exhibit continuous behavior, L2 models are discrete; the main challenge in creating a model is creating a discrete representation of the system useful for diagnosis. Given that most

of the EO-1 telemetry observations used by the model are already discrete, creating a discrete representation was straight-forward.

The EO-1 model was created using L2's graphical model creation tool, called Stanley. It contains the three main subsystems described above: the ALI, the Hyperion, and the WARP. The subsystem structure is shown in the following diagrams, with interactions between components as connections. Component nominal and fault modes, transitions between modes, and propositions which hold in the modes and at the subsystem level are not shown for brevity.

The Hyperion model is shown in Figure 2.

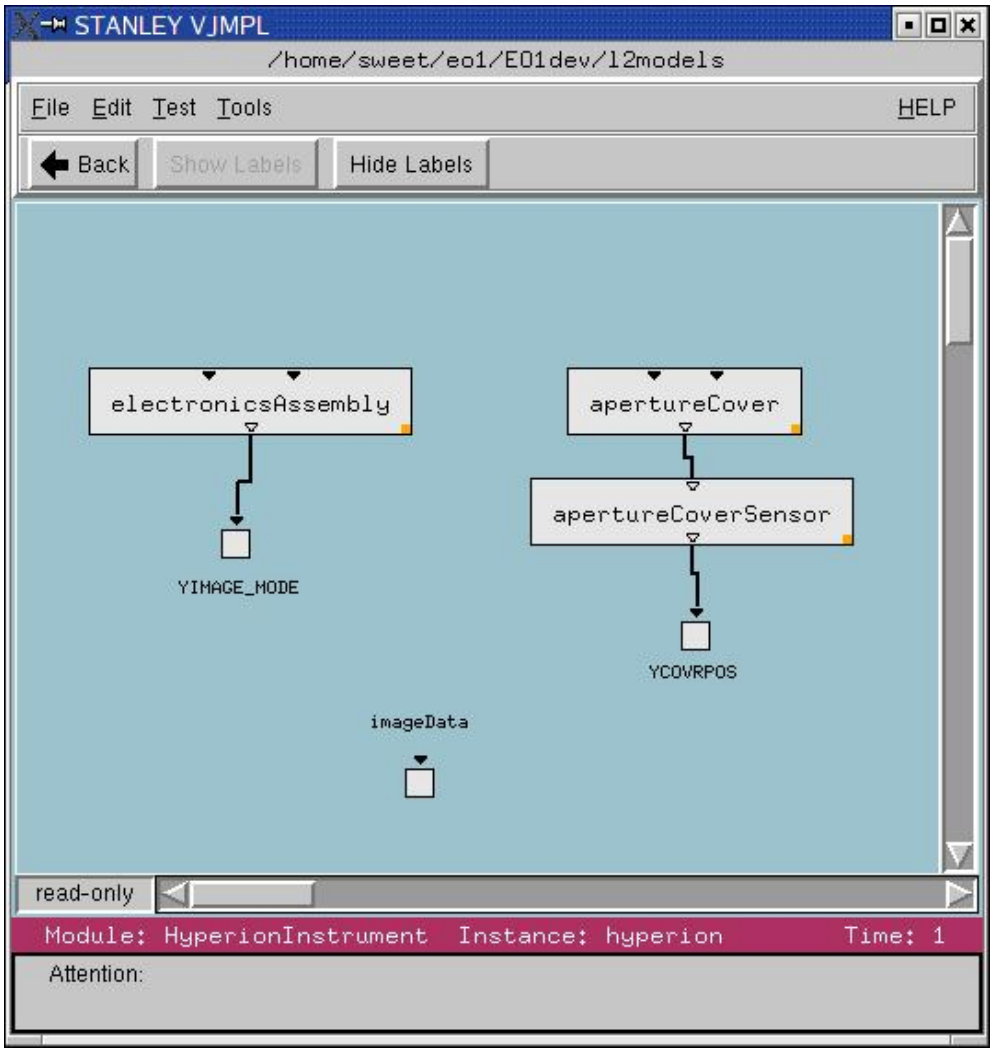

Figure 2: L2 model of the Hyperion

There are three components modeled in the Hyperion; the main aperture cover, which opens to image the earth, the aperture cover sensor, which measures the aperture cover's position, and the electronics assembly, containing the imaging electronics. The *imageData* variable represents what type of image is being taken. It is set based on the modes of the *electronicsAssembly* and the *apertureCover*: NO_IMAGE if the *electronicsAssembly* is disabled, DARK_IMAGE if the electronics are enabled but the *apertureCover* is closed, and EARTH_IMAGE if the electronics are enabled and the aperture cover is open.

The ALI model is shown in Figure 3.

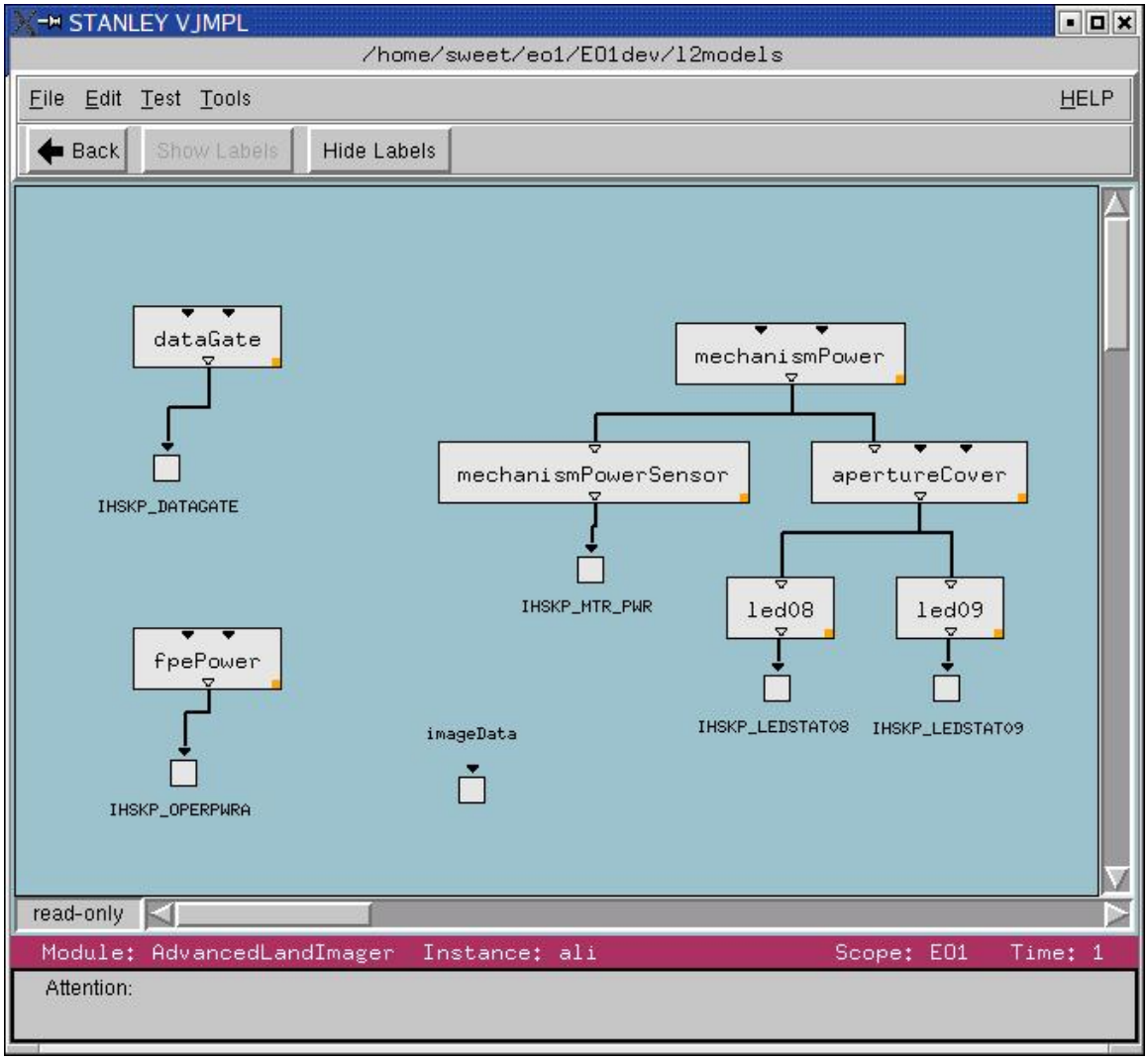

Figure 3: L2 model of the ALI

There are seven components contained in the model of the ALI. First, the *fpePower* represents the focal plane electronics power, which must be enabled in order to take an image. The *dataGate* status indicates if an image is currently being acquired. The *apertureCover* acts as the lens cap of the instrument; it is normally closed to protect the instrument and to take dark calibration images, and open when taking images of the earth. The *mechanismPower* component supplies power to the aperture cover, allowing it to move. Again, the *imageData* variable represents what type of image is being taken. It is assigned according to the modes of the *fpePower*, *dataGate*, and *apertureCover*: if the *fpePower* or *dataGate* is disabled, it is set to NO_IMAGE; otherwise, if the aperture cover is closed, it is set to DARK_IMAGE, and if the *apertureCover* is open it is set to EARTH_IMAGE. Finally, three of the sensors are modeled: the *mechanismPowerSensor* which reports the state of the mechanism power, and two light-emitting diode (LED) indicators, which indicate the state of the *apertureCover*. The multiple sensors surrounding the *apertureCover* were explicitly modeled (and allowed to fail) because the semi-redundant information will allow L2 to find multiple hypotheses when a single fault occurs in the subsystem, one of the key features demonstrated in the experiment.

Finally, the WARP model is shown in Figure 4.

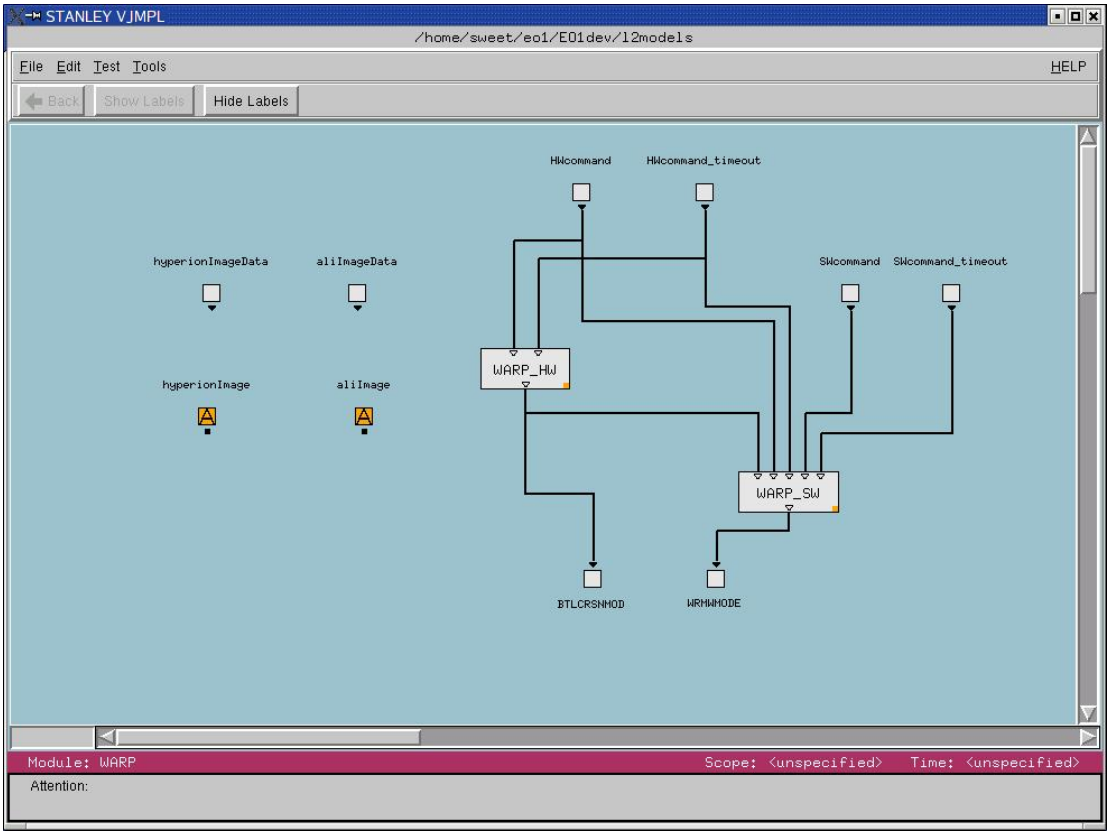

Figure 4: L2 model of the WARP

The WARP is modeled as two components, the WARP_HW and the WARP_SW. The WARP_HW models the device's hardware modes, and the WARP_SW models the software modes. As evidenced by the many connections between the two components, the hardware modes and software modes are closely related – not all combinations of modes are valid.

D. Testing the Model

Models can be tested directly in the Stanley GUI model development environment. Execution of the models involves L2 in a stand-alone mode. A diagnostic scenario, consisting of a sequence of commands and telemetry observations, exercises the model within Stanley.

For testing the L2 model of the EO-1 satellite, the basic scenario is the same as the model's scope: the satellite's imaging timeline or Data Collection Event (DCE). More specifically, the scenario is composed of the commands and telemetry observations sent to the Hyperion, ALI, and WARP. This sequence is briefly described below:

- Components set to image collection mode
- Dark calibration image taken
- ALI and Hyperion aperture covers opened
- Earth image taken
- ALI and Hyperion aperture covers closed
- Dark calibration image taken again
- Components set to standby mode

Seventeen Stanley scenarios were created in total, one representing the nominal data collection event, one for a dual data collect (two successive earth images) and one to test each of the fault modes in the L2 model. Each of the fault scenarios is based on the nominal scenario but with only the required telemetry modified to inject the fault.

V. Integration and Test on the EO-1 Test Beds

A. Remote Testing on the Strings

Initial integrated testing was conducted on PowerPC embedded systems. A series of tests were defined, incrementally building up the software. The 'A' tests integrated L2, the model and the RTI. The 'B' series in parallel tested the SCL Adaptor, SCL database and project, with SCL. Test C integrated the L2 and SCL software components. Test D had the addition of CASPER and the Science software, as well as a CASPER simulation of the satellite to generate sensor telemetry and respond to commands issued by the planner.

Once the PowerPC tests were completed, the testing was moved to the flight-hardware test beds. These test beds consist of two M-5 processors, three PCs, and communication hardware—which all together was called a 'string'. The Command and Data Handling (C&DH) and the WARP are both M-5 processors, identical to those used on-board the satellite. On the strings, the WARP also executes its own flight software for data recording and playback activities. Available CPU is restricted on the radiation-hardened M-5 processors, which are very slow at around 12 MIPS. Running on the string test beds significantly increases the flight readiness of the software, as this is a far more rigorous environment than the PowerPC.

Goddard has oversight of the development and maintenance of three such strings, contracted to Microtel and Hammers Corporation. String 1 is the test bed which is closest in configuration to the satellite. JPL has possession of String 2 for ASE testing. Ames can remotely access String 1 and String 3 at Goddard to execute tests. When Ames integration testing on String 3 at Goddard was near completion, verification tests were expanded to String 1 to run on hardware as close to the satellite as possible.

A schematic of a string is given in Figure 5 showing the different parts and how they connect.

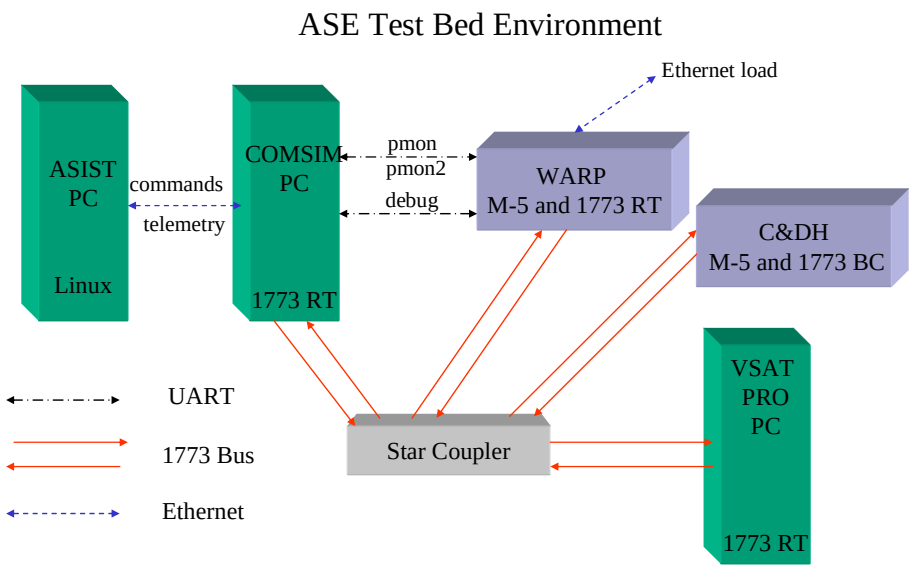

Figure 5: Test Bed Hardware Configuration

The WARP M-5 processor is where the experimental software resides. The C&DH is the 1773 Bus Controller (BC) which talks to other 1773 Remote Terminals (RT) such as the WARP, passing commands and receiving data. VSAT Pro is a virtual satellite that simulates the satellite hardware, such as the satellite's attitude, instruments and power. The Star Coupler is the 1773 network hub through which all the test bed hardware communicates.

The Linux machine runs the ground station software, called Advanced Spacecraft Integration and System Test (ASIST). ASIST communicates with the satellite, uplinking ground commands and receiving satellite telemetry. COMSIM simulates the low-level communication hardware that receives commands and telemetry. It can be used to

capture the serial output coming from the WARP for debugging purposes. COMSIM is also used to upload the software to the WARP via the Ethernet, which can also be done with another PC using TFTP.

B. Test Automation

Most scenarios took about three hours to complete and required over 2,000 keystroke entries. With this volume of input, human error becomes a significant factor. This prompted the automation of the test process. A scripting language called Expect was used to automate testing of the L2 software on the M-5 test bed at Goddard. Expect is an extension of the Tool Command Language (Tcl), used for automating user input to other applications. The name "Expect" comes from the idea of send/expect sequences popularized by uucp, kermit and other modem control programs.

Expect automated everything from starting up the software on the M-5 test bed, to running L2, to downloading the log files and translating them into human-readable text. The only test step not yet automated is the reboot of the M-5; this is already performed for the PowerPC hardware. Automation of the test procedure eliminated the majority of mistakes and increased the amount of testing accomplished.

C. CPU Metrics

The VxWorks Spy utility was used to measure CPU utilization per task, on the PowerPC only. This functionality is not supported on the M-5 processors, which are black boxes. A function was written to initialize the Spy interrupt clock at 1000 ticks per second and to collect data at five second intervals. The task's output was directed to a file on the PowerPC target. Upon completion of the test, the report file was downloaded to the host computer and parsed with a Tcl script to extract the individual task CPU usage data and save it to a Microsoft Excel file. The data collected represented the number of clock ticks each task executed, out of the 1000 possible samples per five seconds. This was then turned into a percentage of CPU time for each task.

Although the capability was developed for measuring CPU usage on the PowerPC, the schedule did not allow for gathering metrics. If further development of the software increases CPU consumption, metrics will become necessary to enable optimization of performance.

VI. Test Results on the Test Beds

A. L2 Unit Test Results

In the L2 unit tests, L2 is executed in standalone fashion using the Stanley model development environment. The diagnostic result from L2 for fault scenario FS06 as visualized in Stanley is shown in Figure 6.

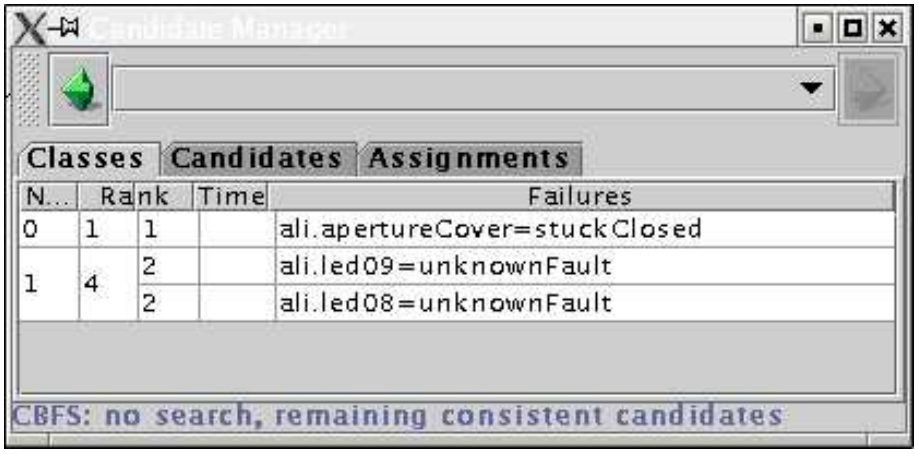

Figure 6: Screenshot of L2 unit test diagnosis

The relevant information in the table is that there are 2 candidates in the diagnosis of this fault. Each candidate is a possible explanation of the current observations. The first candidate contains a single fault, that the ALI's aperture cover is stuck closed. The second candidate contains two faults, that both of the LED sensors have failed. Each candidate is a possibility, according to the observations, but the single-fault candidate is more likely to have occurred than the double fault (as indicated by the lower number in the "Rank" column. Here, the two LED sensors

were measuring the position of the aperture cover. Hence, L2's diagnosis is that either the aperture cover is *stuckClosed*, or both of the sensors measuring the cover position have failed. This split of "component failed or sensors failed" is a common result when using L2.

The results for all of the L2 unit tests are given in Table 1. The criteria for success are as follows:

1) The diagnosis contains the injected fault as a candidate
2) All other candidates in the diagnosis are also possible given the commands and observations

As evident from the table, the L2 unit tests completed successfully for all scenarios.

Table 1: L2 Unit Test Results

Scenario ID	Final Diagnosis Candidate(s) and Component Fault(s)	Correct?
Eo1Nominal	None	Yes
Eo1DualNominal	None	Yes
Eo1FS01_AliDataGateFailedDisabled	ali.dataGate=failedDisabled ali.dataGate=unknownFault	Yes
Eo1FS02_AliDataGateFailedEnabled	ali.dataGate=failedEnabled ali.dataGate=unknownFault	Yes
Eo1FS03_AliMechanismPowerFailedDisabled	ali.mechanismPower=failedDisabled	Yes
Eo1FS04_AliMechanismPowerFailedEnabled	ali.mechanismPowerSensor=unknownFault ali.mechanismPower=failedEnabled	Yes
Eo1FS05_AliMechanismPowerSensorFailed	ali.mechanismPowerSensor=unknownFault	Yes
Eo1FS06_AliApertureCoverFailedClosed	ali.apertureCover=failedClosed ali.led08=unknownFault ali.led09=unknownFault	Yes
Eo1FS07_AliApertureCoverFailedOpen	ali.apertureCover=failedOpen ali.led08=unknownFault ali.led09=unknownFault	Yes
Eo1FS08_AliApertureCoverFailedIntermediate	ali.apertureCover=failedIntermediate ali.led08=unknownFault	Yes
Eo1FS09_AliLed09Failed	ali.led09=unknownFault	Yes
Eo1FS10_AliLed08Failed	ali.led08=unknownFault	Yes
Eo1FS20_HyperionApertureCoverFailedOpen	hyperion.apertureCover=failedOpen hyperion.apertureCoverSensor=unknownFault	Yes
Eo1FS21_HyperionApertureCoverFailedClosed	hyperion.apertureCover=failedClosed hyperion.apertureCoverSensor=unknownFault	Yes
Eo1FS23_HyperionElectronicsError	hyperion.electronicsAssembly=error hyperion.electronicsAssembly=unknownFault	Yes
Eo1FS24_HyperionApertureCoverSensorFailed	hyperion.apertureCover=failedOpen hyperion.apertureCoverSensor=unknownFault	Yes
Eo1FS35_WarpFailedToRecord	warp.software=unknownFault	Yes

B. Integrated Test Results

The diagnostic output of the integrated testing on the test beds is stored in integer-encoded log files. The same output will be generated onboard EO-1. The log files will be compressed and downlinked to ground for decompression, decoding to human-readable text and analysis. The results of the integrated tests for String 1 are given below in Table 2.

Table 2: String 1 Integrated Test Results

Scenario ID	Final Diagnosis Candidate(s) and Component Fault(s)	Correct?
Eo1Nominal	None	Yes
Eo1DualNominal	None	Yes
Eo1FS01_AliDataGateFailedDisabled	None	No
Eo1FS02_AliDataGateFailedEnabled	ali.dataGate=failedEnabled ali.dataGate=unknownFault	Yes
Eo1FS03_AliMechanismPowerFailedDisabled	ali.mechanismPower=failedDisabled ali.mechanismPowerSensor=unknownFault	Yes
Eo1FS04_AliMechanismPowerFailedEnabled	ali.mechanismPowerSensor=unknownFault ali.mechanismPower=failedEnabled	Yes
Eo1FS05_AliMechanismPowerSensorFailed	ali.mechanismPowerSensor=unknownFault	Yes
Eo1FS06_AliApertureCoverFailedClosed	ali.apertureCover=failedClosed ali.led08=unknownFault ali.apertureCover=failedIntermediate	Yes
Eo1FS07_AliApertureCoverFailedOpen	ali.apertureCover=failedOpen ali.led08=unknownFault ali.led09=unknownFault	Yes
Eo1FS08_AliApertureCoverFailedIntermediate	ali.apertureCover=failedIntermediate ali.led08=unknownFault	Yes
Eo1FS09_AliLed09Failed	ali.led09=unknownFault	Yes
Eo1FS10_AliLed08Failed	ali.led08=unknownFault	Yes
Eo1FS20_HyperionApertureCoverFailedOpen	hyperion.apertureCover=failedOpen hyperion.apertureCoverSensor=unknownFault	Yes
Eo1FS21_HyperionApertureCoverFailedClosed	hyperion.apertureCover=failedClosed hyperion.apertureCoverSensor=unknownFault	Yes
Eo1FS23_HyperionElectronicsError	hyperion.electronicsAssembly=error hyperion.electronicsAssembly=unknownFault	Yes
Eo1FS24_HyperionApertureCoverSensorFailed	hyperion.apertureCover=failedOpen hyperion.apertureCoverSensor=unknownFault	Yes
Eo1FS35_WarpFailedToRecord	warp.software=unknownFault	Yes

In the integrated test, 16 out of 17 scenarios completed successfully. The reason that FS01 failed the test is due to timing latencies of the actual system. The ALI data gate is commanded enabled, but commanded disabled again before the "enabled" telemetry is received. Therefore, L2 sees the same final disabled mode and does not have timely information to detect that the intermediate mode transition failed. L2 assumes no faults exist until evidence to the contrary is received; in this case that assumption results in a missed diagnosis or false negative for FS01.

Some other minor differences in the results also exist. In FS06, L2 reported a different double-fault candidate. This candidate is as likely as the one found in the L2 unit test. Because of CPU restrictions, L2 is restricted from exhausting all possible candidates. Here, it simply found this double-fault candidate first. If L2 was not limited by CPU, it would have returned both double-fault candidates (for a total of 3 candidates) in both the unit test and the integrated test.

VII. Conclusion

Over the past year, the project has gone from initiation to deployment on-board a spacecraft. Models of the satellite were developed from scratch and diagnostic scenarios validated on a series of test beds of increasing fidelity. A new Real-Time Interface and transient modeling methodology were employed to enable the software to run on a real-world system, with reduced complexity and complete independence of the RTI from the model specification. The project team learned about the satellite, about operations procedures, and how to coax delicate hardware and firmware systems into a working state. The L2 software has been uploaded to the satellite and tests are about to begin.

Much work lies ahead. A future paper will report on results of the on-board validation tests. Further important work remains on implementing recovery once a diagnosis is made. The models could be extended to cover additional subsystems, and to diagnose known faults that have occurred on-board the spacecraft. Performance improvements can be made to approach hard real-time requirements. This work will significantly contribute to the maturation of model-based diagnosis and improve the chances for adoption of this critical technology for many missions and applications.

References

1. B. Williams and P. Nayak, "A Model-based Approach to Reactive Self-Configuring Systems". *Proceedings of Thirteenth National Conference on Artificial Intelligence (AAAI'96)*, Portland, Oregon, 1996.
2. N. Muscettola, P. Nayak, B. Pell and B. Williams, "Remote Agent: To Boldly Go Where No AI System Has Gone Before". *Artificial Intelligence*, Vol 100, Best of IJCAI 97.
3. S. Chien, R. Sherwood, D. Tran, R. Castano, et al., "Autonomous Science on the EO-1 Mission". *Proceedings of the Seventh International Symposium on Artificial Intelligence, Robotics and Automation in Space*, Nara, Japan, 2003.

Biographies

Sandra Hayden is Principal Investigator and Project Manager of the Livingstone on EO-1 (LEO-1) Infusion Experiment, and a Group Manager for QSS Group Inc. Founded the QSS Software Process Improvement Network and lead CMMi pre-appraisals of selected QSS tasks. Architected and developed Integrated Vehicle Health Management (IVHM) for reusable launch vehicles (RLVs), under the Space Launch Initiative program. Ames Project Manager, under the Next Generation Launch Technology program, for the PITEX project (Propulsion IVHM Technology Experiment for X-vehicles), an IVHM application for the X-34 RLV Main Propulsion System. Her interests include model-based diagnosis and flight-critical systems. Prior experience is on software engineering for large, mission and safety-critical Ada software systems such as the Canadian Automated Air Traffic System (CAATS) and a 1553 System Data Bus for a submarine; and operations software for undersea diamond mining. Ms Hayden received an MS in Computer Science from Simon Fraser University, Vancouver, Canada.

Adam Sweet is a research engineer in the Computational Sciences Division at NASA Ames Research Center, under contract to QSS Group Inc. He graduated with an MS in Mechanical Engineering from UC Berkeley in 1999, and has since worked at Ames modeling and simulating physical systems. His focus has been in robotics, hybrid system simulation, and model-based diagnosis.

Scott Christa is a systems engineer at NASA Ames Research Center under contract to AerospaceComputing, Inc. He graduated with a Bachelor of Science in Aeronautics from San Jose State University and holds an Airline Transport Pilot (ATP) and flight instructor certificates. Although he doesn't fly for a living, he specializes in programming embedded system using anything from high-level languages down to assembler code.