Transient Region Coverage in the Propulsion IVHM Technology Experiment

Edward Balaban[1]
balaban@email.arc.nasa.gov

Adam Sweet[1]

Anupa Bajwa[1]

William Maul[2]

Chris Fulton[2]

Amy Chicatelli[2]

[1]NASA Ames Research Center
Moffett Field, CA 94035

[2]NASA Glenn Research Center at Lewis Field
Cleveland, OH 44315

Abstract

Over the last several years researchers at NASA Glenn and Ames Research Centers have developed a real-time fault detection and isolation system for propulsion subsystems of future space vehicles. The Propulsion IVHM Technology Experiment (PITEX), as it is called follows the model-based diagnostic methodology and employs Livingstone, developed at NASA Ames, as its reasoning engine. The system has been tested on flight-like hardware through a series of nominal and fault scenarios. These scenarios have been developed using a highly detailed simulation of the X-34 flight demonstrator main propulsion system and include realistic failures involving valves, regulators, microswitches, and sensors. This paper focuses on one of the recent research and development efforts under PITEX – to provide more complete transient region coverage. It describes the development of the transient monitors, the corresponding modeling methodology, and the interface software responsible for coordinating the flow of information between the quantitative monitors and the qualitative, discrete representation in Livingstone.

1. Introduction

The ability to perform meaningful diagnosis during the transient period immediately following a command can be critical for any diagnostic system: a large percentage of failures can occur in that time period, and the capacity to quickly detect and isolate a problem can mean the difference between a mission remediation and mission failure. In the case of complex physical systems, such as a Main Propulsion System (MPS) for a reusable launch vehicle, performing such a diagnosis can be difficult, since there is usually a significant period of instability - called a transient - following a command. This is the inherent lag between the issuance of a command and the steady state response of the system, since a physical system needs time to settle down in its new state.

The previous approach used in PITEX was to wait out that transient period for a predetermined amount of time, and then perform a diagnosis. All components of the same type had same timeout period. This period was set to cover the worst case scenarios and therefore delayed the diagnostic response from the PITEX system in most cases.

The new approach, reported here, is to dynamically determine transient characteristics that would indicate when the system is stable and could therefore be diagnosed by the PITEX reasoning engine. This capability would improve the diagnostic time in the majority of situations. Two additional concepts were also pursued as a result of this new approach.

The first was redefining the way physical systems are modeled in PITEX, in order to reduce the complexity of the Real-Time Interface (RTI), the software that serves as a conduit between the monitors and Livingstone. As the diagnostic coverage increased, the RTI became too complex and system specific. System information, such as the interactions between components of different subsystems, had to be explicitly described in the code. A simplified RTI, freed of such dependencies, could be adopted for other Livingstone applications without significant changes.

The second concept was the demonstration that Livingstone reasoning engine could be applied to any existing constraints in the transient period. The reasoning world for Livingstone can now be divided into steady-state and transient regions; sets of relevant features can be monitored in such regions and constraints applied based on the anticipated response of the system to a command. This is an important notion because it enables the PITEX diagnostic system to provide complete coverage throughout a given time period.

While the work described in this paper does not yet constitute a complete solution for transient region issues, it is an important step in expanding the range of Livingstone applications to beyond the discrete steady-state domain. This paper provides an overview of the PITEX system and its diagnostic domain – the X-34 MPS, then goes into the details on the previous and current approaches to handling the transient regions. Three specific test cases are presented that illustrate the potential capability of the new approach and, finally, some additional areas that need further investigation are reviewed.

2. PITEX Overview

This section begins an overview of PITEX with a brief historical summary of the project. The summary is followed by an overview of the X-34 flight demonstrator that contains descriptions of the vehicle, the MPS, and the nominal mission profile. Next, the diagnostic system is defined in terms of the software components and the design reference mission.

2.1 Historical Perspective

The development effort under the PITEX project has improved and enhanced the capabilities of the model-based diagnostic system that was developed under the NASA IVHM Technology Experiment for X-Vehicles (NITEX) project. NITEX was a Pathfinder Experiment that was developed by Ames Research Center, Glenn Research Center, and Kennedy Space Center as a real-time fault detection system of the X-34 MPS. The main objectives of PITEX became the continued enhancement of diagnostic technologies that are relevant to 2nd Generation Reusable Launch Vehicle (RLV) subsystems and the assessment of the real-time performance of the developed diagnostic solution. The program has been funded under the Space Launch Initiative and, most recently, by the Next Generation Launch Technology effort.

There have been several development periods that matured the PITEX software system into its present form today. During each phase of development, the capabilities of the software were improved and enhanced by focusing on key expansion areas, such as the scalability of the system, its handling of sensor noise and sensor failures, among others. One of the areas of continuous interest is the improvement of system response to failures in order to provide accurate and timely diagnostic information. The work presented here falls under that area.

2.2 X-34

The X-34 program [1] was a joint effort by industry and government to design, develop, and test a fully reusable vehicle that would demonstrate technologies and operating concepts applicable to future RLV systems. A nominal X-34 mission would include five general phases: pre-flight, captive carry, powered flight, post-flight, and landing. During pre-flight, the necessary ground operations are performed, such as propellant loading. In the next mission phase, captive carry, the X-34 is carried down the runway and up to the required launch altitude attached to an L-1011 carrier aircraft. Once captive carry is completed, the X-34 is released from the L-1011, its engine is started, and the powered portion of the flight begins. The vehicle eventually reaches the cruise altitude of 250,000 feet, where it flies at the speed of Mach 8. After powered flight is completed, the engine is shut down, and any excess propellants are dumped overboard. The X-34 then flies as a glider before it lands at a conventional runway. If a mission is terminated, the X-34 was designed to dump all propellants and still land safely. More details about X-34 operation can be found in [2].

The X-34 program was suspended in the spring of 2001. The work on developing an advanced diagnostic system for the MPS of the X-34 was continued, however, both in the hopes that the flight program will be revived and because a large amount of work, including detailed simulations of the MPS, had already been completed and deemed applicable to other similar propulsion systems. The X-34 MPS, being a complete, modern, yet not overly complicated propulsion system, provided a good development test bed.

2.3 X-34 Main Propulsion System

The MPS is responsible for providing the thrust that the RLV needs to meet the requirements of a mission. It is powered by liquid oxygen (LOX) and RP-1 rocket fuel and provides for the loading, storing, delivering, and disposing of these propellants. Within the MPS, there are many subsystems that carry out these functions: the propellant tanks, the LOX feed, fill and dump system, the RP-1 feed, fill, and dump system, the vent system, the pressurization system, and the pneumatic and purge system. The main engine for the X-34 vehicles was never completely developed and for development purposes is treated as a load on the rest of the MPS. For the PITEX application, the X-34 MPS was scoped to include only the pneumatic system, the pressurization system, the LOX subsystem, and the RP-1 subsystem. Since the purge system and the reaction control system were not included, this restricted the number of components modeled and monitored while still offering unique processing challenges.

To further scope the PITEX demonstration, one specific segment of an X-34 flight profile was selected. The captive carry portion was selected due to crew safety considerations of the piloted L-1011. During this phase of operation, the X-34 is carried to the required launch altitude of 38,000 feet while it is attached to the underside of an L-1011 aircraft. The engine is not running, and most of the subsystems of the MPS are in a quasi-static state. The primary functions for those subsystems that are operating are the following:

- the vent/relief system prevents over-pressurization of the tanks and provides propellant conditioning for the LOX;
- the LOX and RP-1 feed systems deliver propellant for engine bleed and thermal conditioning of engine components in the case of LOX.

The Design Reference Mission [2] for the captive carry phase is divided into a set of distinct phases. Throughout the first half hour of captive carry, the MPS is locked-up. During this period, thresholds are selected for the vent/relief system and the pressurization system so that they are inactive under nominal conditions. After this lock-up phase, the vent/relief system is activated to provide LOX conditioning. For two hours, this process maintains the nominal temperature and pressure in the LOX tanks within the predefined thresholds. Once the two hours are completed, the pressurization system is enabled. The RP-1 tank is pressurized and the RP-1 bleed process is performed. This process removes all the air and purge gases from the propellant fuel feed lines in preparation of engine ignition. Thereafter, the LOX tanks are pressurized and the LOX chill-down and bleed process is performed. This process introduces the feed line and engine components to the LOX propellant, expelling air and purge gases, as well as thermally conditioning these components.

2.4 Diagnostic System Overview

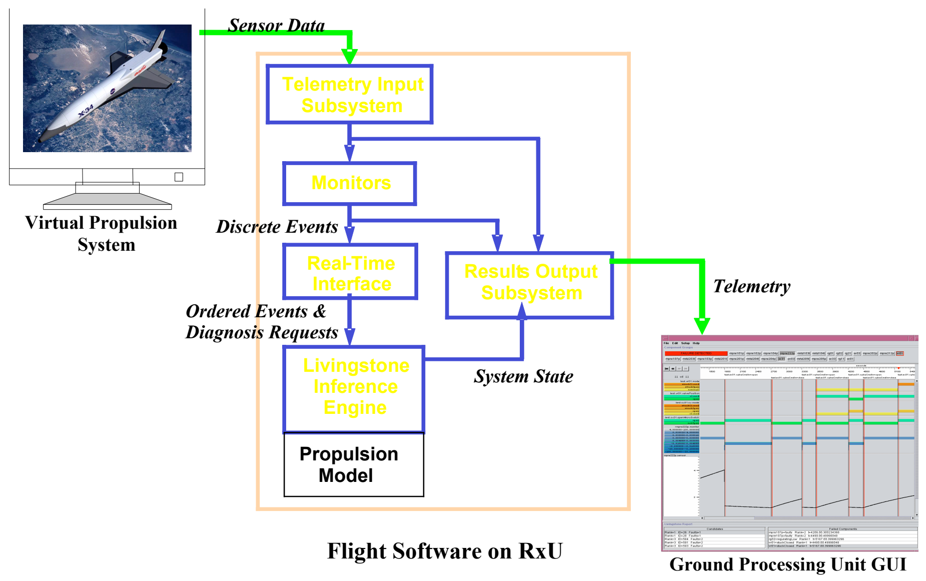

Sensor Data
Telemetry Input Subsystem
Monitors
Discrete Events
Real-Time Interface
Results Output Subsystem
Telemetry
Ordered Events & Diagnosis Requests
Livingstone Inference Engine
Propulsion Model
System State
Virtual Propulsion System
Flight Software on RxU
Ground Processing Unit GUI

Figure 0. Diagnostic System Architecture

The purpose of PITEX [3] has been to demonstrate the successful diagnosis of faults by using a real-time diagnostic software system. One key achievement has been the implementation of an architecture that can quickly diagnose faults in a quantitative, continuous domain, such as the MPS, using a qualitative, discrete inference engine, such as Livingstone [4, 5].

PITEX is an integrated software package that consists of the Telemetry Input System, Monitors, Real-Time Interface (RTI), Livingstone, Results Output System (ROS), and Ground Processing Unit (GPU). The overall architecture of the diagnostic system is shown on Figure 0. The virtual propulsion system simulates the sensor data associated with a particular mission phase and nominal or failure scenario. These data are stored in flat files prior to diagnostic system testing. The TIS reads in these data sets and stores the information so as to provide access to the modules in the same manner and time frame as that experienced by the system during real-time operations on an actual test or flight. After the TIS stores the data on-line to simulate a data sweep, the data are accessed by the Monitor software where pertinent features of the propulsion system are extracted and the quantitative information of the system is transformed into qualitative information. This information is then passed through the RTI to Livingstone, where system-level diagnostics are performed using a high-level qualitative model of the propulsion system. The diagnostic output is collected by the ROS and sent to the GPU for display.

2.5 Diagnostic Modeling

In model-based diagnosis, there is often a distinction between the actual model and the diagnostic engine – the part of the program that carries out the reasoning. The Livingstone inference engine follows this path and uses a separate model of the client system, its controller commands, and sensor observations [4, 5]. A model consists of a number of components, each having a set of nominal modes (e.g. "on" and "off"), and a set of fault modes (e.g. "stuck on", "stuck off"). Transitions between modes are also modeled. Transitions between nominal modes are assumed to be a result of a controller command and therefore modeled explicitly. Transitions to fault modes are assumed to be able to occur at anytime, from any nominal mode, and are, therefore, not modeled.

In addition to the X-34, several other applications have used Livingstone as their diagnosis engine: DS-1 Remote Agent Experiment (JPL/ARC) [6], X-37 Electro-Mechanical Actuators [7], an in-situ propellant production testbed (KSC) [8], a ship's cooling system (JHU/APL), Space Shuttle Main Engine (Honeywell), Command & Data Handling System of the International Space Station (ARC – ongoing) [9]. All of these used the *steady state* discrete modes (such as "on" or "off") to model the behavior of the system.

3. Evolution of PITEX Transient Methodology

The original PITEX software was intended to apply diagnostic reasoning on steady-state observations only and was unable to infer upon observations that occurred during transient periods. To ensure that observations provided to the diagnostic software were steady, latency periods were established based empirically on simulation data. These latency periods were event specific; meaning that for each known event there was a specified period of time in which the system would return to a steady state. There are several problems with this approach. The first one is that it is not entirely reliable; any variation in nominal system response would potentially expose the diagnostic engine to erroneous inputs. This approach also requires that all the events encountered by the system are known in advance, including fault events, along with their expected settling times. Finally, there are events that require latency periods on the order of tens of seconds and during these periods the diagnostic system is essentially blinded. There are dynamic indicators or signatures available for the diagnostic system during these latency periods and incorporating these into the diagnostic process would be extremely beneficial.

Previous PITEX efforts, in order to minimize the size and impact of these latency regions, focused on modifying signal processing algorithms used by the Monitors and expanding the logic policies of the RTI. The Monitors adopted statistical methodologies in order to provide observations faster and with a higher degree of confidence. This provided smaller latency periods and robustness to normal sensor variations, such as noise. The RTI logic policies were modified to allow intermediate diagnostic analyses to be performed during the latency periods by relaxing the constraints only on those sensors that were not expected to have settled yet. To achieve this, some of the Livingstone model information, such as the relationships among the components and sensors of the system, was replicated in the RTI. This allowed the RTI to identify and continually monitor the subsystems not impacted by the current event, as well as to perform preliminary fault analysis using sensors that respond quickly to the event, rather than waiting for the entire sensor suite to stabilize.

While these modifications lessened the impact of the latency periods, there were several problems remaining. By incorporating subsystem relationships into the RTI, the later was becoming very domain specific, which would make adaptation of PITEX work to other systems difficult. In addition, the approach still required a priori information about the latency periods, with all the downsides associated with that. Any unexpected but nominal deviation in the response of the system to specific events could result in invalid diagnostic results. Finally, even though these latency periods were reduced, the monitored system was still diagnostically inaccessible during them.

In the spring of 2003, PITEX team focused on redesigning the monitors, the Livingstone model, and the RTI in order to deal better with the challenges of transient coverage. These new modules, combined with the base PITEX code, formed the Transient Framework. In the Transient Framework, instead of having the RTI un-assigning the sensor observations that are expected to transition, the constraints that a component places on those sensor observations within the diagnostic engine are suspended during the transient period. The sensor observations associated with the subsystem are thus allowed to fluctuate. However, components in other subsystems are still enforcing constrains on their sensor values and may be diagnosed. Thus, the Transient Framework retains the advantage of diagnosing subsystems that are expected to be in a steady state. The corresponding model is now larger, as it contains the additional transient modes, however the RTI may now be domain-independent since the domain-specific subsystem relationships are no longer repeated within it. Furthermore, the transient modes may contain specific constraints on the transient period for different components, allowing for a diagnosis of the subsystem even while in the transient phase. The sections below describe this new approach in more detail.

4. Transient Model

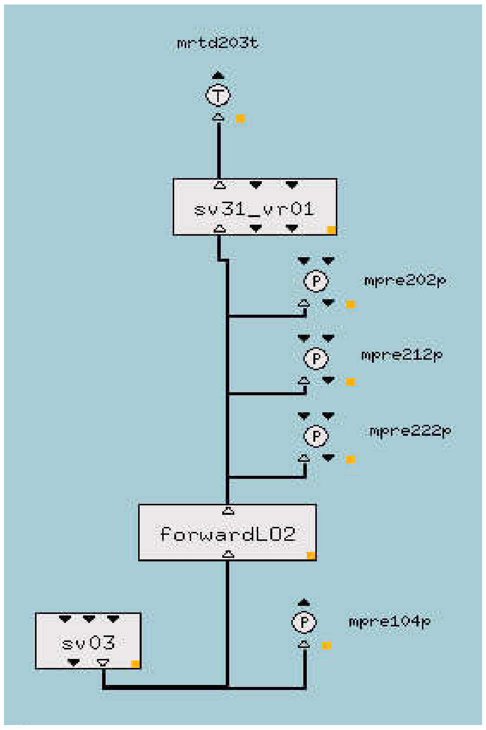
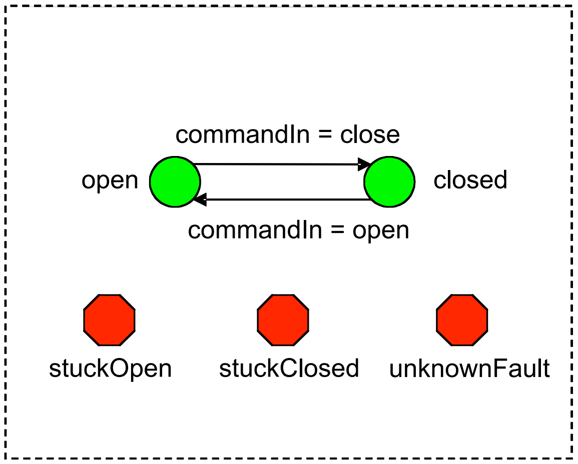


Figure 0. Transient Model

The model used in the transient coverage work was created using Stanley, a Livingstone modeling environment developed at NASA Ames. For the first iteration, a fragment of the PITEX X-34 MPS model was used. Its schematic is shown on Figure 0. The fragment covers the forward liquid oxygen tank (forwardLO2) and parts of its pressurization and vent systems. Solenoid Valve #3 (SV03) introduces helium into the tank to keep the pressure in it constant while the oxygen is being consumed by the engine. The line from SV03 to the tank is monitored by a pressure sensor MPRE104P. A vent/relief valve, VR01, releases the excess gaseous oxygen which boils off during the captive carry MPS lockup. This action is designed to prevent a potentially catastrophic tank explosion. A smaller solenoid valve, SV31, operates the pneumatic VR01 with the help of the helium from the pressurization system. For the reasons of simplicity, in the model used by the Transient Framework the two valves are combined into a single component, SV31_VR01. This design decision removed some failure modes that will need to be addressed in a more comprehensive model, but otherwise left the behavior of the vent system unchanged.

Figure 0 goes into more detail on SV31_VR01 component. The schematic on the left shows how this valve was implemented in Livingstone previously, the one on the right shows the transient representation. As mentioned earlier, the valve model now contains the transient “opening” and “closing” states in addition to the usual “open” and “close”. These transient states were left largely unconstrained for now; they only include the constraints stipulating that the transient period needs to complete successfully. If the transient monitor reports that the transient is not occurring or is abnormal, it will conflict with the nominal transient mode and trigger a fault



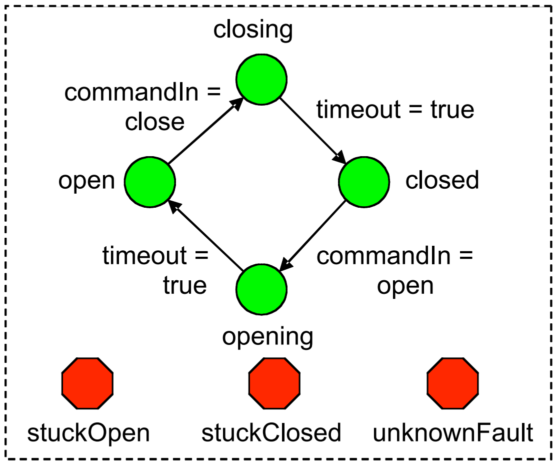


Figure 0. Base PITEX and Transient Valve Models

diagnosis. Otherwise, the steady-state constraints associated with the "open" and "closed" modes will be suspended, allowing the sensor values to fluctuate during the transition. The steady-state constraints are reinstated when the transient is detected to be over.

5. Transient Monitor

The initial task of this monitor during the transient period is to categorize the system response as “expected”, “absent” or “anomalous” by monitoring a sensor or group of sensors which should reflect dynamic behavior whenever a related command is issued. A follow-up task is to determine when the transient period has completed by deciding when dynamic behavior in the same set of sensors has sufficiently subsided.

For the purposes of this first study, one of the three redundant pressure sensors on the vent line, MPRE202P, was selected to monitor transient behavior for open commands to VR01. Monitoring of the transient behavior for the open command was selected because the sensor behavior is very distinct during that time.

When VR01 is commanded open, MPRE202P senses a pressure drop, which then settles out. At the time of the command, the transient monitor calculates the average value of MPRE202P over the data frame when the command was issued. A 0.5 second delay is then introduced. This delay was empirically derived based on the observed time required to see a significant change in MPRE202P. At the end of the delay, an average of the current data frame is again calculated and compared to the previous value. If the delta between the values surpasses a threshold, then the RTI is notified that the system is responding. Otherwise, the RTI is told that the transient has completed.

If the system is responding, the monitor continues to calculate averages over successive data frames and compares the current average with a value calculated, nominally, 250 milliseconds earlier. These two values are at the endpoints of a moving window of data. The transient is considered to be over when the absolute value of this new delta falls below a steady state value. At this time the RTI is notified that steady state has been achieved.

While this algorithm is relatively simple, the concept could be extended to use multiple sensors to detect both “system responding” and “transient over” events. In addition, detection of different features, such as a drift or spike, could be used as an indicator of system response within any or all of the sensors selected for the task. The extension of the algorithm would need to be balanced against the real-time requirements of the system.

6. Transient Real-Time Interface

The new Transient RTI has two main modes of operation – steady state and commanded transition. In the steady state, the RTI buffers incoming observations and checks whether any of them are for components that already have previous observations stored. If that is the case, the new observation replaces the old one. This cycle continues until a command arrives.

When that happens, the commanded transition mode is entered and a software timer is set. In the case of the Transient RTI, this timer serves as a backup mechanism in case the transient monitor fails to provide an indication that the system has reached a steady state. Livingstone is then informed of the command, which puts the commanded component into a less constrained, intermediate state (for a valve that would be “closing” or “opening”). Next, the observations are collected as they were before the command, until the transient monitor indicates that the transient period has indeed begun. That indication usually comes about 0.5 seconds after the command. The RTI then issues a Livingstone diagnostic request to check that the component has entered the intermediate, transient mode correctly. The observations are continued to be buffered until the transient period ends due to one of the following reasons:

Transient over
This is the nominal way to end a transient period. The transient monitor detects that a sensor has reached a steady state and the transient period has most likely ended.

Therefore, it sends a "transient over" observation. Once the RTI receives this observation, it commands the component to assume its final state, uploads all the observations from the buffer to Livingstone, and then requests a diagnosis.

Abnormal transient
When the transient detector notices a problem with one of the transient conditions monitored, such as a sudden, unexpected rise or fall in value, it issues an "abnormal transient" observation. That prompts the RTI to end the transient period early by sending the appropriate command to Livingstone. It then uploads all the accumulated observations and requests a diagnosis.

Timeout
If the transient monitor fails to send either a "transient over" or an "abnormal transient" message within a predetermined period of time, the timer task issues a timeout notice, and the RTI ends the transient period in a manner similar to the "Transient Over" and "Abnormal Transient" cases.

7. Relevant Execution Scenarios

This section summarizes the cases where using the adopted transient approach can prove beneficial:

System does not respond to a command or an event
In this case the transient detector does not notice any change in the behavior of the system after the mandatory 0.5 second wait, so it informs the RTI that the transient is complete. The RTI can then proceed to send Livingstone all the observations received up to that point and request a diagnosis. This request happens much sooner than if it were to wait for the predefined timeout period to elapse.

An anomaly occurs during the transient period
The transient period starts out nominally and the RTI is informed to that effect. However, at some point prior to steady state, the monitors detect an anomaly in sensor telemetry. The observed anomaly could be as simple as the absolute pressures exceeding a predefined or controlled threshold value. The monitors then issue a spontaneous observation to that effect and Livingstone can provide diagnostic analysis even during the transient regions, thereby eliminating, or at least reducing, the blind regions of the diagnostic system.

The transient period ends successfully, but sooner than usual.
The transient period ends successfully, but sooner than anticipated, and this is detected by the transient monitor. The RTI sends observations accumulated up to that point to Livingstone and the latter then requests a diagnosis without waiting for the latency period to end. This eliminates the often unreliable dependency on *a* priory information about the length of transient periods.

8. Testing

The test cases, presented in Table 1 are based on the existing NITEX/PITEX fault scenarios adapted for testing of the Transient Framework. They were selected to correspond to the execution scenarios described in the previous section. No changes to the original fault scenarios were required, except for the second test case, where NITEX/PITEX did not have a direct equivalent. To produce it, a transient anomaly was injected at 6186 seconds into the execution of the nominal scenario.

9. Results and Discussion

It is, of course, difficult to compare the results obtained for the Transient Framework to the previous PITEX results, even given the similarity of the scenarios. The model used in the Transient Framework is significantly smaller than the latest PITEX model used. This theoretically reduces the search time needed by Livingstone. PITEX was tested on 25 scenarios compared to Transient Framework's three. Still, a side-by-side look at the two result sets does present a promising trend for the transient approach:

Test Case 1 - Nominal:
Two criteria were considered to compare performance in this case. The first one is absence of false positive errors in diagnosis. If diagnosis is performed prematurely, before the critical observations have settled, Livingstone may detect inconsistencies with the expected state and report a fault (or faults). Both the base PITEX system and the Transient Framework satisfied this criterion. The second criterion is how soon the diagnostic system was able to confirm the nominal state after a command. Only the open commands for SV31/VR01 were reviewed, since these are the commands currently monitored by the transient detector. There are 7 such commands issued in the 9000 seconds of the nominal scenario. The base version of PITEX required an average of 3.13 seconds after the command was issued to confirm that it was successful (the minimum was 3.08 seconds, maximum – 3.17 seconds). For the Transient Framework the numbers were 2.51 seconds average diagnostic time, 2.10 minimum,

Transient Execution Scenario	**NITEX/PITEX Scenario**	**Description**
Test Case 1. The transient period ends successfully, but sooner than anticipated	Nominal	Captive carry portion of an X-34 flight
Test Case 2. An anomaly occurs during a transient period	Transient anomaly	Same as Nominal, only a transient anomaly is introduced after a command at 6186.0 seconds
Test Case 3. System does not respond to a command or an event	LOX tank vent relief valve stuck closed	LOX tank vent relief servo valve SV31 sticks closed at 5167.4 seconds

and 3.17 seconds maximum..

This demonstrates the ability of the diagnostic system to dynamically conclude when the system had steady state. The PITEX solution at 3+ seconds only involves rapid responding observations, microswitches, with the remaining observations being incorporated much later. This is due to the *a priori* uncertainty required by the PITEX system to ensure system stability. The Transient Framework diagnosis involves all available observations immediately because it has determined that the system has achieved steadiness directly.

Test Case 2 - Transient Anomaly:
This scenario did not have an exact equivalent in the NITEX/PITEX set and thus could not be directly compared to the previous results. However, the diagnostic response time obtained was less or equal to the smallest time measurable by PITEX - 0.08 seconds - which is as good as can be expected. The test case demonstrated the enhanced capability of the diagnostic system to accept and process dynamic features and apply those features to dynamic constraints during the transient period. Previous PITEX models would have completely ignored this type of information and would have been incapable of providing any diagnostic reasoning during the transient periods.

Test Case 3 - LOX tank vent relief valve stuck closed:
This was the easiest scenario to compare against the previous results since it remained unchanged and did not contain an injected fault. The base PITEX code took 3.08 seconds to detect and diagnose the fault; the Transient Framework code needed only 0.52 seconds. In this test case, the failure is a non-event; the system was commanded to perform, but there was no response. Because the current Transient Framework constantly evaluates the system after 0.5 seconds to determine if it has achieved steadiness, the diagnostic inference engine is able to evaluate the entire suite of observations sooner than the earlier PITEX version, which based its diagnosis at 3.08 seconds on only a partial set of observations.

While the above test scenarios cover a significant portion of situations occurring during transient periods, in order to provide a more complete coverage for physical systems such as MPS, failure scenarios which are rarer, yet frequently more difficult to diagnose must be considered. While extending the software to accommodate such cases was outside the scope of this effort, some preliminary discussions to identify the problems and possible solutions have already been held. These cases include the handling of rapid commanding sequences, overlapping command, and spontaneous failures.

Rapid Commanding
This is the situation when a command for the same component comes before a steady state for the previous command is achieved. For instance, a valve is still in the transient period for the open command, when a close command comes in. One possible way to address this problem is to create a model where transitions between intermediate states (such as "opening" and "closing") are allowed. This way the model can be switched directly from one transient period into another, without having to send potentially inconsistent observations to Livingstone in the meantime.

Overlapping Commands
In this case a command to one subsystem is issued while the transient period initiated by a command to another subsystem is still in progress. The Transient Framework may be well suited for this type of a situation since most of the constraints pertaining to the commanded component have already disabled for the transient timeout and thus should have no influence on a different subsystem.

Spontaneous failures
Sometimes failures in a physical system do not happen as a result of commanded transitions, but occur in between them, during what ordinarily would be tranquil periods. For example, a pipe could burst, a sensor can fail, or valve can spuriously transition from being open to being closed. In this case, only the observations are seen by the Monitors and RTI and without an associated cause to the observation changes (a particular commanded event), it can be difficult to determine how long to wait for system to reach a steady state. In the majority of situations, spontaneous observations can be treated the same way as a command when performing diagnosis and are expected to be handled well by the current design of the Transient Framework. A difficulty could arise if a command for the same subsystem is received during the transient timeout for the spontaneous observation. The system response to the command could mask or be masked by the spontaneous failure. Monitor observations may become inconsistent and result in a false positive fault diagnosis. Out of the problems described in this section, this is likely the most complex one to resolve.

From a broader perspective, several areas have been identified as promising for further research. For instance, the transient monitors need to evolve to provide more comprehensive and precise diagnostic coverage by combining inputs from several different sensors, such as pressure, temperature, and microswitch readings. Care, however, should be exercised as to not shift too much of the diagnostic responsibilities into the monitors by explicitly hardcoding some of the algorithms there and thus negating certain benefits of the model-based diagnostic approach.

Another research direction that is worth investigating is integration of the Transient Framework with BEAM, a signal pattern recognition system developed at JPL. Operating alongside with PITEX monitors, BEAM can assist with accurately identifying the boundaries and anomalies of transient periods. In the past BEAM has successfully been combined with PITEX to disambiguate diagnosis on a fault scenario [10].

Finally, scalability of the Transient Framework must be verified by applying it to larger, multiple subsystem models, such as the PITEX model of the entire X-34 MPS.

10. Concluding Remarks

The transient modeling approach, where intermediate, loosely constrained modes for commanded components are created, is a promising way to provide diagnostic coverage for the periods of instability that usually follow a command in a physical system. A diagnostic system for X-34 flight demonstrator MPS created using this methodology was both simpler in design than the earlier versions and provided faster diagnostic times.

Three test cases were developed and used to demonstrate the potential of this approach over the previous PITEX software design. In each case the transient modeling approach provided earlier and more comprehensive diagnostic solution. The new approach enables reasoning on features occurring within the transient period and is more robust, requiring no assumptions concerning system response durations.

While there are several research topics remaining to be investigated to ensure that this approach is viable, the expectation of the Transient Framework is very positive.

Acknowledgements

The authors would like to acknowledge the contributions of all of the other PITEX and NITEX team members at the NASA Kennedy Space Center, the NASA Ames Research Center, and the NASA Glenn Research Center. Their contributions were instrumental in the successful completion of this work. The authors would also like to acknowledge the support and guidance of the NGLT IVHM Project Office managed by the NASA Ames Research Center.

Bibliography

[1] P. K. Sgarlata and B. A. Winters, "X-34 Propulsion System Design," *33rd AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, July 6-9, 1997.
[2] R. H. Brown, Jr. and F. J. Darrow, Jr., "X-34 Main Propulsion System Design and Operation," *34th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, July 13-15, 1998.
[3] C. Meyer, H. Cannon, E. Balaban, C. Fulton, W. Maul, A. Chicatelli, A. Bajwa, E. Wong, "Propulsion IVHM Technology Experiment Overview", 2003 *IEEE Aerospace Conference Proceedings*
[4] J. Kurien and P. Nayak, "Back to the Future for Consistency-based Trajectory Tracking," *Proceedings of 7th National Conference on Artificial Intelligence*, 2000.
[5] A. Bajwa and A. Sweet, "The Livingstone Model of a Main Propulsion System," 2003 *IEEE Aerospace Conference Proceedings*
[6] D. Bernard et al, "Spacecraft Autonomy Flight Experience: The DS-1 Remote Agent Experiment," *AIAA-99-4512*, 1999.
[7] M. Schwabacher, J. Samuels, and L. Brownston, "The NASA Integrated Vehicle Health Management Technology Experiment for X-37," *SPIE AeroSense*, 2002.
[8] C. Goodrich and J. Kurien, "Continuous Measurements and Quantitative Constraints – Challenges Problems for Discrete Modeling Techniques," *I-SAIRAS Proceedings*, 2001.
[9] P. Robinson, M. Shirley, D. Fletcher,R. Alena, D. Duncavage, C. Lee, "Applying Model-Based Reasoning to the FDIR of the Command & Data Handling Subsystem of the International Space Station*," iSAIRAS 03* 2003
[10] H. Park, H. Cannon et al, "Hybrid Diagnostic System: Beacon-based Exception Analysis for Multimissions – Livingstone Integration," *Society for Machinery Failure Prevention Technology Conference,* April 2004.