Institute for Software Integrated Systems
Vanderbilt University
Nashville, Tennessee, 37212

The Inertial Measurement Unit Example: A Software Health Management Case Study

Abhishek Dubey , Nagabhushan Mahadevan, Gabor Karsai

TECHNICAL REPORT

ISIS-12-101

Febraury, 2012

The Inertial Measurement Unit Example: A Software Health Management Case Study

Abhishek Dubey, Nagabhushan Mahadevan, and Gabor Karsai
Institute for Software-Integrated Systems
Vanderbilt University
Nashville, TN 37212, USA

No Institute Given

Abstract. This report captures in detail a Two-level Software Health Management strategy on a real-life example of an Inertial Measurement Unit subsystem. We describe in detail the design of the component and system level health management strategy. Results are expressed as relevant portions of the detailed logs that shows the successful adaptation of the monitor/ detect/ diagnose/ mitigate approach to Software Health Management.

1 Introduction and Motivation

Software has become the key enabler for a number of core capabilities and services in modern systems [17]. For example, a modern car contains around 20 million lines of code, while just the flight control software of modern aircraft like F-22 and F-35 contains $1.7 - 5.7$ million lines of code [6]. Given the scale of the software systems, it is not hard to appreciate the challenge of ensuring correct behavior, especially in avionics where software malfunctions have caused a number of incidents in the past, including but not limited to those referred to in these reports: [3,4,9,18]. [21] provides an excellent discussion on the complexity in avionics software.

The state of the art for critical software development includes process standards such as DO-178B [7] and the emerging standards such as DO-178C [10]. However, it is known that software can contain latent defects or bugs that can escape the existing rigorous testing and verification techniques and manifest only under exceptional circumstances. These circumstances may include faults in the hardware system, including both the computing and non-computing hardware. Often, systems are not prepared for such faults.

State of the art for safety critical systems is to employ software fault tolerance techniques that rely on redundancy and voting [5,15,24]. However, it is clear that existing techniques do not provide adequate coverage for problems such as common-mode faults and latent design bugs triggered by other faults. Additional techniques are required to make the systems self-managing, i.e. they have to provide resilience to faults by adaptively mitigating the functional effects of those faults.

Self-adaptive systems must be able to adapt to faults in software as well as the hardware (physical equipment) elements of a system, even if they appear simultaneously. Conventional Systems Health Management is associated with the physical elements of the system, and includes anomaly detection, fault source identification (diagnosis), fault effect mitigation (at runtime/ online during operation), maintenance (offline), and fault prognostics (online or offline) [12,19]. Software Health Management (SHM), borrows concepts and techniques from Systems Health Management and is a systematic extension of classical software fault tolerance techniques. Srivastava and Schumann provide a good motivation for Software Health Management in [23]. SHM is performed at run-time, and just like Systems Health Management it includes detection, isolation, and mitigation to remove fault effects. SHM can be considered as a dynamic fault removal technique [2]. While Systems Health Management also includes prognostics, Software Health Management could possibly be extended in that direction as well, but we have not investigated it yet.

We have developed an approach and model-based support tools for implementing software health management functions for component-based systems. The foundation of the architecture is a real-time component framework that defines a component model for ARINC-653 systems[1] [8]. This framework brings the concept

[1] ARINC-653 (Avionics Application Standard Software Interface) is a specification for space and time partitioning in Safety-critical avionics Real-time operating systems. It allows to host multiple applications of different software levels on the same hardware in the context of an Integrated Modular Avionics architecture.[1,20]

of temporal isolation, spatial isolation, strict deadlines from ARINC-653 and merges it with the well-defined interaction patterns described in CORBA Component Model [26]. The health management in the framework is performed at two levels. The Component-level Health Manager (CLHM) provides localized and limited service for managing the health of individual software components. A higher-level System Health Manager (SLHM) manages the health of the overall system.

SLHM includes a diagnosis engine that uses a Timed Failure Propagation (TFPG) model automatically synthesized from the component assembly; the engine reasons about fault effect cascades in the system, and isolates the fault source components. This is possible because the data / behavioral dependencies and hence the fault propagation across the assembly of software components can be deduced from the well-defined and restricted set of interaction patterns supported by the framework. Once the fault source is isolated, the necessary system level mitigation action is taken. Similar approaches can be found in [14,25]. The key difference between those and our work is that we apply an online diagnosis engine coupled with a two-level mitigation scheme. Furthermore, this approach is applied to hard real-time systems where all processes run within finite time bounds and are continuously monitored for deadline violations. This includes, the health management processes.

Our approach is supported by a model-based design environment where developers can create models of the system and its components, as well as specify how fault mitigation will take place. A suite of software generators produce glue code that allows developer-supplied functional code or 'business logic' to form a collection of applications that run on an ARINC-653 platform.

In this report, we use the example of an Inertial Measurement Unit (IMU) system to demonstrate the novel contributions of this approach which include

- Model-based development of component-based systems for ARINC-653 platform.
- Automatic synthesis of monitoring code that is executed with the component operations.
- Automatic synthesis of diagnosis information from the system design models.
- Automatic synthesis of the mitigation code based on system specification.
- Generation and configuration of the distributed architecture required to operate the components in parallel with the component and system level health managers.

2 IMU Case Study

Figure 1 shows the logical system assembly for the IMU. This system consists of several subsystems described below. This case study can be repeated by the readers by downloading the tools from `https://wiki.isis.vanderbilt.edu/mbshm/index.php/Main_Page`. The design tools come with the IMU model as an example. Also included is the business logic code and instructions to repeat the experiment.

2.1 GPS Subsystem

A GPS subsystem consists of two components, GPS Receiver and GPS processor, see Figure 1. This is similar to the Sensor and GPS components shown earlier in the paper. While the GPS receiver emulates a software sensor providing the hardware readout, the GPS processor implements a Kalman Filter. Details of the implementation of Kalman Filter are not discussed here. When the GPS processor has an updated position, it sends a 'tick' out of its publisher port with the understanding that any subscriber component can fetch the update synchronously at a later time. Table 1 shows the real-time properties of both components in the subsystem. Overall, the GPS subsystem runs at a frequency of $0.1Hz$. In this particular system, there are two instances of the GPS subsystem, a primary and a backup, called the secondary subsystem. In this particular case study, the GPS coordinates were read from a table by the receiver component.

Monitors GPS processor's facet port is configured with a post-condition to evaluate the correctness of its internal position data. GPS processor does not have a component level health manager. Therefore, by default all detected anomalies are propagated to the system health manager and a local action of IGNORE is actuated.

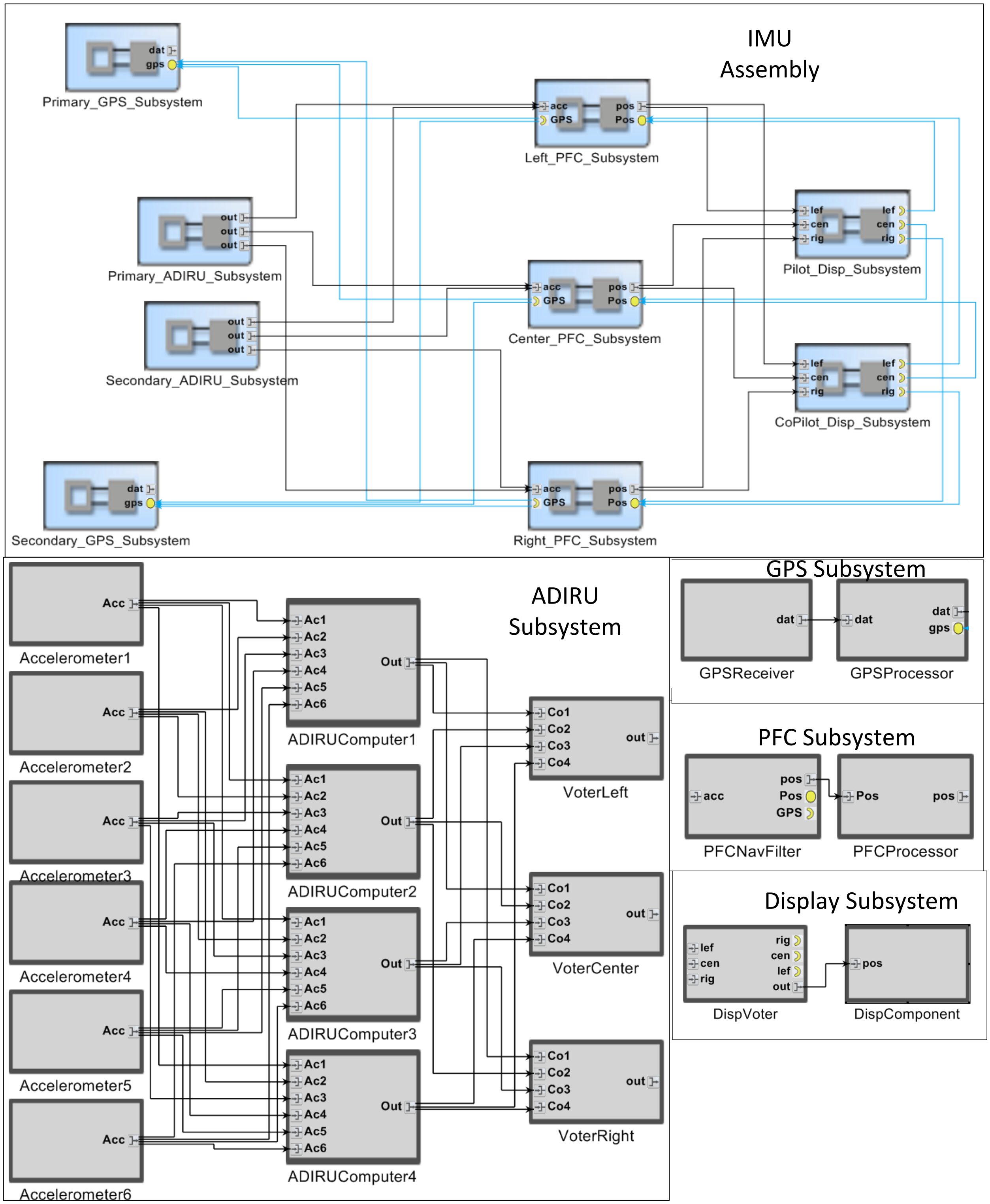

Fig. 1. Model of an IMU in ACM

Table 1. Details Of the Internal Port/ARINC-653 Processes of Components in the IMU system

Type	Component	Port	Period	Deadline	DeadlineType
Publisher	ADIRUAccelerometer	Acceleration	1.0	1.0	HARD
Publisher	GPSReceiver	data_out	10.0	10.0	HARD
Facet	GPSProcessor	getGPSData	-1	-1.0	HARD
Publisher	GPSProcessor	data_out	1.0	1.0	HARD
Consumer	GPSProcessor	data_in	10.0	10.0	HARD
Publisher	ADIRUProcessor	Output	-1.0	-1.0	HARD
Consumer	ADIRUProcessor	Ac1	-1.0	-1.0	HARD
Consumer	ADIRUProcessor	Ac2	-1.0	-1.0	HARD
Consumer	ADIRUProcessor	Ac3	-1.0	-1.0	HARD
Consumer	ADIRUProcessor	Ac4	-1.0	-1.0	HARD
Consumer	ADIRUProcessor	Ac5	-1.0	-1.0	HARD
Consumer	ADIRUProcessor	Ac6	-1.0	-1.0	HARD
Internal	ADIRUProcessor	ComputationTask	1.0	1.0	HARD
Publisher	ADIRUVoter	output	-1.0	-1.0	HARD
Consumer	ADIRUVoter	Co1	-1.0	-1.0	HARD
Consumer	ADIRUVoter	Co2	-1.0	-1.0	HARD
Consumer	ADIRUVoter	Co3	-1.0	-1.0	HARD
Consumer	ADIRUVoter	Co4	-1.0	-1.0	HARD
Internal	ADIRUVoter	VotingTask	1.0	1.0	HARD
Facet	PFCNavFilter	getPosData	-1	-1.0	HARD
Receptacle	PFCNavFilter	getGPSData	-1	-1.0	HARD
Publisher	PFCNavFilter	position	-1.0	-1.0	HARD
Consumer	PFCNavFilter	acc	-1.0	-1.0	HARD
Internal	PFCNavFilter	invokeGPS	10.0	10.0	HARD
Publisher	PFCProcessor	posReady	-1.0	-1.0	HARD
Consumer	PFCProcessor	Position	-1.0	-1.0	HARD
Receptacle	DispVoter	getPosData	-1	-1.0	HARD
Receptacle	DispVoter	getPosData	-1	-1.0	HARD
Receptacle	DispVoter	getPosData	-1	-1.0	HARD
Publisher	DispVoter	output	-1.0	-1.0	HARD
Consumer	DispVoter	centerPosToken	-1.0	-1.0	HARD
Consumer	DispVoter	rightPosToken	-1.0	-1.0	HARD
Consumer	DispVoter	leftPosTick	-1.0	-1.0	HARD
Internal	DispVoter	VotingTask	1.0	1.0	HARD
Consumer	PositionDisplay	position	-1.0	-1.0	HARD

2.2 ADIRU Subsystem

The architecture of the ADIRU subsystem (see Figure 1) in this case study is based on the Air Data Inertial Reference Unit (ADIRU) used on a Boeing 777 aircraft [16,22]. An ADIRU provides airspeed, angle of attack, altitude as well as inertial position and attitude information to other flight systems. The primary design principle in Boeing 777's ADIRU Architecture is multiple levels of redundancy. There are two ADIRU units: primary and secondary. The primary ADIRU is divided into 4 Fault Containment Areas (FCA), with each FCA containing multiple Fault Containment Modules (FCM): accelerometers (6 FCM), gyros (6 FCM), processors (4 FCM), power supplies (3 FCM), ARINC 629 bus (3 FCM). The ADIRU system was designed to be serviceable, with capability to tolerate up to one fault in each FCA without any maintenance. Systems can fly with two faults, but it necessitates maintenance upon landing.

In 2005, the ADIRU unit of a Malaysian Air flight was responsible for an inflight upset. Post-flight analysis [3] revealed that in 2001 accelerometer 5 in the primary ADIRU had failed with high output values and was subsequently marked as faulty. However, because there was only one failure no maintenance was requested on the unit, but the status of failed unit was recorded in on-board maintenance memory. However, on the day of the incident, a power cycle on the primary ADIRU occurred, during flight. Upon reset, the processors did not check the status of the on-board memory and hence did not regard accelerometer 5 as faulty. Thereafter, a second in-flight fault was recorded in the accelerometer 6 and was disregarded. Till the time of the incident the ADIRU processors used a set of equations for acceleration estimation that disregarded the values measured by accelerometer 5. However, the fault in accelerometer 6 necessitated a reconfiguration to use a different set of estimation equations. At this point, they allowed the use of accelerometers 1 to 5 as accelerometer 5 was not regarded as faulty, passing the abnormal high acceleration values to all flight computers. Due to common-mode nature of the fault, voters allowed the incorrect accelerometer data to go out on all channels. This high value was used by primary flight computers, although a comparison function used by the flight computers lessened the effect. In summary, a latent software bug and the common-mode

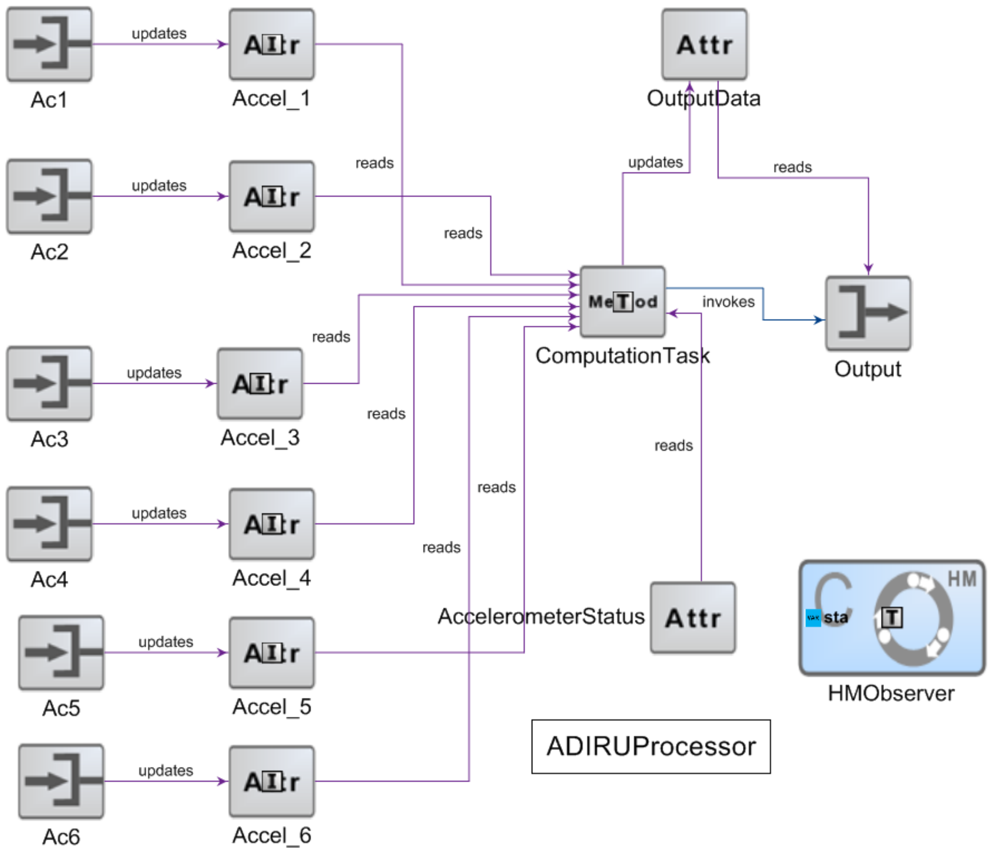

Fig. 2. ADIRU Processor

nature of the accelerometer fault bypassed the redundancy checks and caused the effect to cascade into a system failure [11].

In this particular case study, gyros are not emulated. There are six instances of accelerometer components. Each accelerometer component has a periodic publisher that publishes its data every 1 second, see table 1. Published data consists of a linear acceleration value measured in the axis of the accelerometer and a time stamp. All accelerometers measure in directions perpendicular to the six faces of a dodecahedron centered at the origin of the body coordinate system. Running the code generator of the ACM framework produces the code for all accelerometers. The only portion supplied by the developer is the function that is called in every cycle to produce the data. We use a lookup table to simulate actual sensor measurements, configured for each experiment. All acceleration values are fed to the four ADIRU processors. Figure 2 shows the internal data and control flow of the processor. Each processor consists of six aperiodic consumers (AC1-AC6), each connected to one accelerometer. Accelerometer data is cached internally in six state variables. ADIRU processor also contains a periodically triggered internal method for computing body acceleration value. During this step, the processor solves a set of linear regression equations to estimate the body linear acceleration. If processor is aware of a fault in one of the accelerometers or aware of the staleness of corresponding cached data, it can ignore that particular observation and use the other 5 for performing regression. Note that it needs at least four observations for performing the regression.

Output of each ADIRU processor is the body axis data and is published every second to the three voter components. The voters consume these data with three consumers. Each voter uses a median algorithm to choose the middle values and outputs it to flight computer. Like GPS, there is a backup for the ADIRU subsystem in the IMU assembly.

Monitors In our framework, the design tools allows the system designer to deploy monitors which can be configured to detect deviations from expected behavior, violations in properties, constraints, and contracts of an interaction port or component. Based on these monitors, following discrepancies can be currently identified:

- *Lock Time Out*: The framework implicitly generates monitors to check for resource starvation. Each component has a lock (to avoid interference among callers), and if a caller does not get through the lock within a specified timeout it results in starvation. The value for timeout is either set to a default value equal to the deadline of the process associated with component port or can be specified by the system designer.
- *Data Validity violation* (only applicable to consumers): Any data token consumed by a consumer port has an associated expiration age. This is also known as the validity period in ARINC-653 sampling ports. We have extended this to be applicable to all types of component consumer ports, both periodic and aperiodic.
- *Pre-condition Violation*: Developers can specify conditions that should be checked before executing. These conditions can be expressed over the current value or the historical change in the value, or rate of change of values of variables (with respect to previously known value for same parameter) such as
  1. the event-data of asynchronous calls,
  2. function-parameters of synchronous calls, and
  3. (monitored) state variables of the component.
- *User-code Failure*: Any error or exception in the user code can be abstracted by the software developer as an error condition which they can choose to report to the framework. Any unreported error is recognized as a potential unobservable discrepancy.
- *Post-condition Violation*: Similar to pre-conditions, but these conditions are checked after the execution of function associated with the component port.
- *Deadline Violation*: Any execution started must finish within the specified deadline.

These monitors can be specified via (1) attributes of model elements (e.g. Deadline, Data_Validity, Lock time out), (2) via a simple expression language (e.g. conditions). The expressions can be formed over the (current) values of variables (parameters of the call, or state variables of the component), their *change* (delta) since the last invocation, their *rate* of change (change divided by a time value). Table 2 provides the summary of anomalies that can be observed on a component port and the component as a whole.

In this example, monitors are configured to track the resource usage (CPU time) of the publishers / consumers in the Components associated with Accelerometers, ADIRU processors, Voters and Display components. The publisher port in each Accelerometer component is configured with a monitor to observe the published data via a post-condition. These monitors fire if the published data from the associated Accelerometer appears to be Stuck-High or Stuck-Low or show a rapid change in value that is more than the established norms. All the consumer ports in each of the ADIRU-processors, Voters and Display components have a specified Data-Validity time and the associated monitors trigger when the age of the incoming data (i.e. the difference between the current time and the time stamp on the data) is more than the specified Data-Validity time.

In addition to the monitors specified above, the ADIRU processor components look for the absence of published data on each of the consumer ports, connected to one of the six accelerometers. This is done by observing the lack of the *ENTRY*/*EXIT* events from these ports within a pre-specified timeout period, see Figure 3. It shows portions of the state machine specification monitoring the events for accelerometer 1. Once a missing data is detected, the status is set to 0. The `status` array, indexed from 0 and having six elements, captures the state of all six channels. Five similar state machines are used for observing the other accelerometers, in parallel.

CLHM in the ADIRU assembly All Accelerometer and Display components have local health managers. In case of the Accelerometers, the CLHM, see Figure 4, is configured to issue an IGNORE command when the post-condition violation is detected in the publisher.

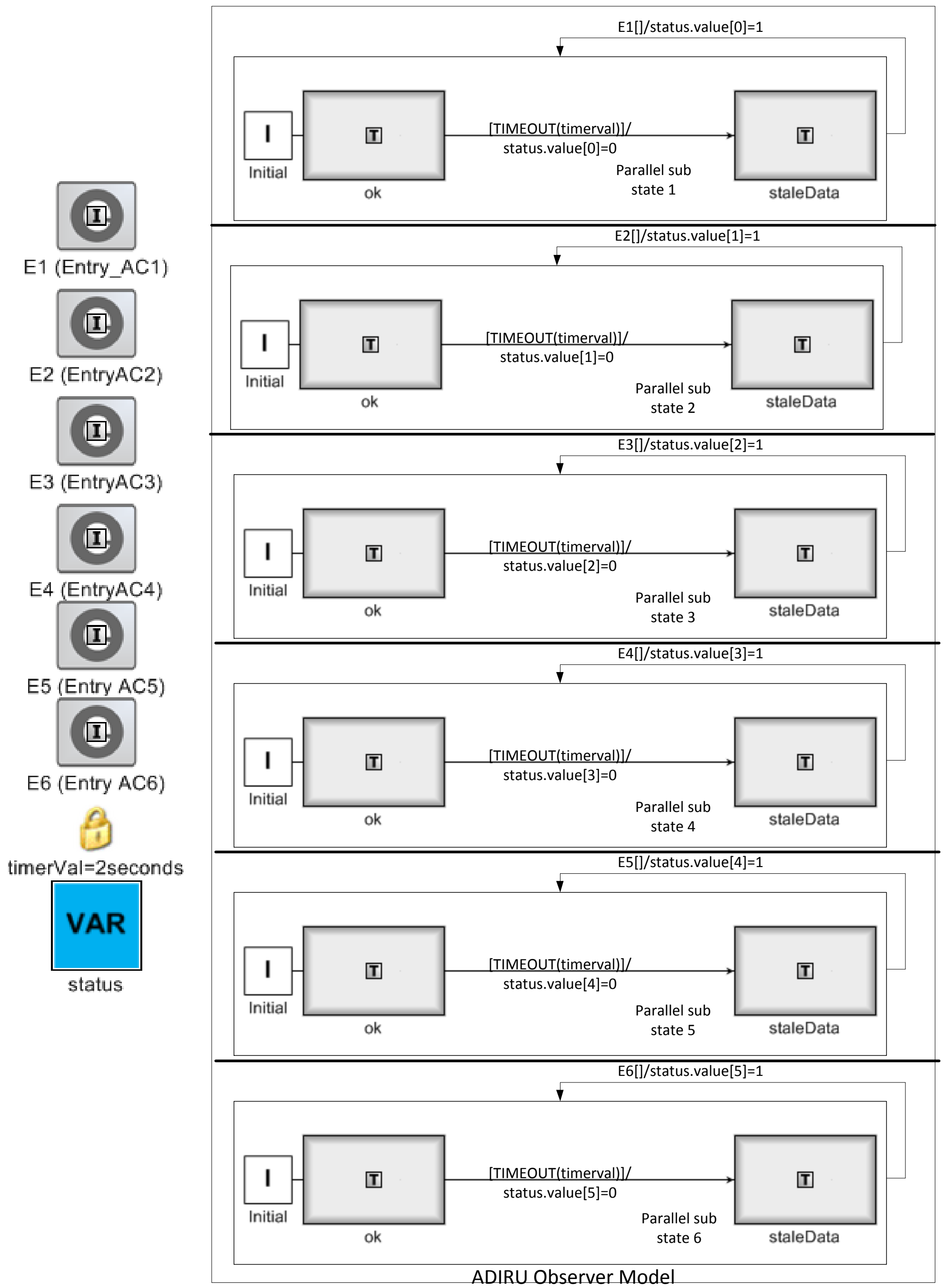

Fig. 3. Observer inside the ADIRU processor: This observer ensures that only latest data, received within last 2 seconds is used for computation.

2.3 PFC Subsystem

The PFC subsystem shown in Figure 1 emulates the flight computer which uses the body acceleration data fed by the ADIRU to track the planes inertial position. Table 1 describes the real-time properties of the ports and components in this subsystem. It receives the input from one of the voters of ADIRU subsystem. Typically, this subsystem also receives the information from Gyros to track the vehicle rotation, pitch and yaw. However, in this particular example we are restricting ourselves to a non-rotating vehicle. Given that

Table 2. Monitoring Specification. Comments are shown in italics.

<table>
<tr><td><Pre-condition>::=<Condition></td></tr>
<tr><td><Post-condition>::=<Condition></td></tr>
<tr><td><Deadline>::=<double value> <i>/* from the start of the process associated with the port to the end of that method */</i></td></tr>
<tr><td><Data_Validity>::=<double value> <i>/* Max age from time of publication of data to the time when data is consumed*/</i></td></tr>
<tr><td><Lock Time Out>::=<double value> <i>/* from start of obtaining lock*/</i></td></tr>
<tr><td><Condition>::=<Primitive Clause><op><Primitive Clause>|<Condition><logical op><Condition>| !<Condition> | True| False</td></tr>
<tr><td><Primitive Clause>::=<double value>| Delta(Var)| Rate(Var)|Var
/* A Var can be either the component State Variable, or the data received by the publisher, or the argument of the method defined in the facet or the receptacle*/</td></tr>
<tr><td><op>::= < | > | <= | >= | == | !=</td></tr>
<tr><td><logical op>::=&& | ||</td></tr>
</table>

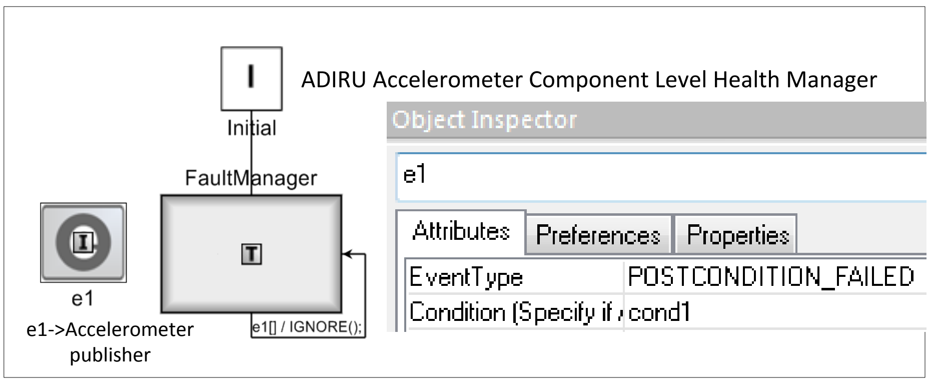

Fig. 4. CLHM State-Machine of Accelerometer. Event e1 occurs when the post-condition of the associated port fails.

inertial system using the body acceleration values tends to drift over time, the PFC Nav Filter component uses a receptacle port to fetch more accurate but slowly refreshing GPS data at a rate of $0.1Hz$ to reset its initial condition. The IMU system is configured with three PFC subsystems. All of them are active in parallel.

Monitors The acceleration consumer port of PFC Nav Filter component is configured with a pre-condition to detect rapid changes in the data fed to these consumers consistent with the physical limits on acceleration and jerk (rate of change of acceleration) in each one of the body axes. It also has a post-condition to check the correctness of the data received from GPS. There is no health manager configured in the component. Lack of health manager implies that all local anomalies are ignored and reported to the system health manager.

2.4 Display Subsystems

There are two instances of the display subsystem. The display subsystem receives the notification from each PFC subsystem that a position update is available. This information is buffered internally. Periodically, at a rate of 1 Hz, it uses three receptacles to fetch the data from the PFC components. Upon receipt of the data, it computes the median value and displays it. Both instances are active in the assembly at the same time.

Note 1. It should be noted that the communication patterns used in the case study, a published data update tick, followed by a synchronous call for fetching the data via a receptacle is intentional. Our goal is to show the richness of interaction patterns that are possible within the ACM framework.

Table 3. SLHM Functions. Here c denotes the component name and s denotes a subsystem name. Unless otherwise specified usage of the subsystem name in a command implies apply to all contained components.

Action	Semantics
IS_FAULTY (c|s)	Returns true if the component is faulty.. A subsystem is marked as faulty if the minimum number of components required for work is not available.
IS_NOT_FAULTY (c|s)	Returns false if the component or subsystem is faulty
RESET (c|s)	Instructs the component to execute its Reset method.
STOP (c|s)	Instructs the component to switch to Inactive mode. Component stops executing the functionality of all its ports. If subsystem is argument, command is applied to all its components
START (c|s)	Instructs the component to switch to Active mode. Component starts executing the functionality of all its ports.
DISABLE _OUTPUT (c|s)	Instructs the component to switch to Semi-Active mode. Only Consumer and Provides port are operational.
REWIRE (c,i,pc)	i: Interface Name, pc: Provider Component Name. This command Instructs Component (c) to switch its receptacle Interface (i) to connect to the approriate facet interface in another component (pc).
CHECK POINT (c|s)	Instructs the component to Checkpoint its current state-variables.
RESTORE (c|s)	Instructs the component to Restore its state-variables from the Checkpoint.

2.5 Generated TFPG Model

The generated TFPG model is too large to be displayed graphically. Instead, we have provided the complexity information of the graph here. Overall, there were 114 activation modes, 120 root failure sources, 234 unobserved discrepancies, 34 observed discrepancies or alarms, and 821 failure propagation links. This generated TFPG model is used by the reasoner in the Diagnosis Engine component. When new data is received from the Alarm Aggregator component, the reasoner generates a set of hypotheses that best describe the cause for the alarms. As new alarms are received it updates the hypotheses. The hypotheses with the best metric (Plausibility, Robustness) are regarded as the most plausible explanation. Further, if a system-level mitigation strategy is specified, then the component containing the source failure modes is identified and the information is passed on to the component hosting the system-level mitigation strategy.

2.6 System Health Manager

System level mitigation strategy is modeled as a hierarchical timed state machine. Table 3 lists the statements (functions) that can be used in the state machine to express the guard conditions (to check if a component is faulty) and actions (mitigation commands). These strategies are reactive in nature and aim to restore the functionality by cold/ warm reset or shifting to redundant component.

In this example, we augmented the system assembly shown in Figure 1 with the three SLHM components. Due to complexity and number of interconnections a visual rendering of the augmented assembly has not been provided.

Figure 5 shows the internal of system fault mitigation strategy. Only strategies concerned with the ADIRU and GPS subsystem are shown in the figure. Initially, system is setup such that only the primary ADIRU subsystem and the primary GPS subsystems are active. It can be seen from the state machine that if the diagnoser implicates Accelermoter5 as faulty, the SLHM sends it a stop command. When two or more accelerometers fail in the primary ADIRU, the SLHM sends a command to stop the primary subsystem and start the secondary subsystem. Figure 1 shows that the network connection between the three flight computers and the secondary already exists. It was just inactive when the system started. Upon receipt of the start command, this links become active and the system starts using the secondary ADIRU. If in future the primary ADIRU subsystem is mitigated such that there is no fault, then the SLHM can revert back to it if the Secondary system fails.

The mitigation in case of GPS subsystem involves the use of REWIRE command, see Table 3. This command is preceded by shutting down the primary subsystem and activating the secondary system. Rewire directs all three flight computers to use the secondary system for GPS update. Once the SLHM sends a command it waits for the alarm aggregator and the diagnoser to update their internal model of the system state. If a confirmation from affected component is not received within a bounded time that component is marked as faulty.

In a hard real-time system all time and network bandwidth has to be preallocated to solve the scheduling problem. Thus, during mitigation it is not possible to instantiate new components. However, a board or a component can be started and a communication link activated, given that its worst case load was accounted during system design.

Table 4. Deployment details of the IMU assembly. Given the hyperperiod and partition periods are same, each partition is executed only once in a hyperperiod.

Module Name	Host Name	Partition	Duration(s)	Period(s)	Subsystem	Component
System Hyper period =1 sec						
SecondaryRest	durip04	sPart3Rest	0.25	1.0	Secondary_ADIRU	VoterRight
SecondaryRest	durip04	sPart3Rest	0.25	1.0	Secondary_ADIRU	ADIRUComputer3
SecondaryRest	durip04	sPart3Rest	0.25	1.0	Right_PFC	PFCProcessor
SecondaryRest	durip04	sPart1Rest	0.25	1.0	Secondary_ADIRU	VoterLeft
SecondaryRest	durip04	sPart1Rest	0.25	1.0	Secondary_ADIRU	ADIRUComputer1
SecondaryRest	durip04	sPart4Rest	0.23	1.0	Secondary_ADIRU	ADIRUComputer4
SecondaryRest	durip04	sPart4Rest	0.23	1.0	Center_PFC	PFCProcessor
SecondaryRest	durip04	sPart4Rest	0.23	1.0	Center_PFC	PFCNavFilter
SecondaryRest	durip04	sPart2Rest	0.25	1.0	Secondary_ADIRU	ADIRUComputer2
SecondaryRest	durip04	sPart2Rest	0.25	1.0	Secondary_ADIRU	VoterCenter
SecondaryRest	durip04	sPart2Rest	0.25	1.0	Right_PFC	PFCNavFilter
SecondaryAcc	durip02	sPart1	0.16	1.0	Secondary_ADIRU	Accelerometer1
SecondaryAcc	durip02	sPart2	0.16	1.0	Secondary_ADIRU	Accelerometer2
SecondaryAcc	durip02	sPart3	0.16	1.0	Secondary_ADIRU	Accelerometer3
SecondaryAcc	durip02	sPart4	0.16	1.0	Secondary_ADIRU	Accelerometer4
SecondaryAcc	durip02	sPart5	0.16	1.0	Secondary_ADIRU	Accelerometer5
SecondaryAcc	durip02	sPart6	0.15	1.0	Secondary_ADIRU	Accelerometer6
PrimaryAcc	durip06	pPart5	0.16	1.0	Primary_ADIRU	Accelerometer5
PrimaryAcc	durip06	pPart4	0.16	1.0	Primary_ADIRU	Accelerometer4
PrimaryAcc	durip06	pPart4	0.16	1.0	Secondary_GPS	GPSProcessor
PrimaryAcc	durip06	pPart3	0.16	1.0	Primary_ADIRU	Accelerometer3
PrimaryAcc	durip06	pPart3	0.16	1.0	Secondary_GPS	GPSReceiver
PrimaryAcc	durip06	pPart2	0.16	1.0	Primary_ADIRU	Accelerometer2
PrimaryAcc	durip06	pPart2	0.16	1.0	Primary_GPS	GPSProcessor
PrimaryAcc	durip06	pPart6	0.15	1.0	Primary_ADIRU	Accelerometer6
PrimaryAcc	durip06	pPart1	0.16	1.0	Primary_ADIRU	Accelerometer1
PrimaryAcc	durip06	pPart1	0.16	1.0	Primary_GPS	GPSReceiver
PrimaryRest	durip05	pPart1Rest	0.25	1.0	Primary_ADIRU	ADIRUComputer1
PrimaryRest	durip05	pPart1Rest	0.25	1.0	Primary_ADIRU	VoterLeft
PrimaryRest	durip05	pPart2Rest	0.25	1.0	Primary_ADIRU	ADIRUComputer2
PrimaryRest	durip05	pPart2Rest	0.25	1.0	Primary_ADIRU	VoterCenter
PrimaryRest	durip05	pPart2Rest	0.25	1.0	Left_PFC	PFCNavFilter
PrimaryRest	durip05	pPart3Rest	0.25	1.0	Primary_ADIRU	ADIRUComputer3
PrimaryRest	durip05	pPart3Rest	0.25	1.0	Primary_ADIRU	VoterRight
PrimaryRest	durip05	pPart4Rest	0.22	1.0	Primary_ADIRU	ADIRUComputer4
PrimaryRest	durip05	pPart4Rest	0.22	1.0	Left_PFC	PFCProcessor
Display	durip03	CoPilotPart	0.5	1.0	CoPilot_Disp	DispVoter
Display	durip03	CoPilotyPart	0.5	1.0	CoPilot_Disp	DispComponent
Display	durip03	PilotPart	0.5	1.0	Pilot_Disp	DispVoter
Display	durip03	PilotPart	0.5	1.0	Pilot_Disp	DispComponent
SHMModule	durip09	SHMPartition	1.0	1.0	SLHM	AlarmAggregator
SHMModule	durip09	SHMPartition	1.0	1.0	SLHM	DiagnosisEngine
SHMModule	durip09	SHMPartition	1.0	1.0	SLHM	ResponseEngine

2.7 Deployment

Table 4 shows the deployment detail of the IMU assembly. Overall, in this case study six modules were used. Each module was sub-divided in to partitions and deployed on to one core of a machine in our lab. Details of the partition such as periodicity and duration and the component instances allocated to the partition are also given in the table. The system health manager components were deployed in a single partition on a separate module synchronizing the execution of all other modules. The runtime framework ensures that all modules run in a synchronized manner with the specified system-wide hyper period of 1 second. At the start of each hyper period a system health manager module sent a synchronization message to each module manager, which executes the module schedule. This is similar to the technique in the TTP/A protocol [13].

We deployed all the six modules in the IMU assembly (Table 4) on six computers in our lab, deployed in an isolated subnet. These computers were running the ARINC Component runtime. Upon initialization, all modules synchronized with the system module that ran the diagnoser and system response/ mitigation engine. Thereafter, each module cyclically scheduled its partitions. All modules resynchronized with the system module at the start of each hyper period. The code necessary for this distributed synchronization was auto generated from the ADIRU deployment model in which each module was mapped to a physical core on a processor.

2.8 Execution

Failure Scenario In this case study we injected following failures one by one: (1) A fault was introduced in the Primary ADIRU Accelerometer6 code such that its data is stuck high. (2) Then, a similar fault was injected in the Primary ADIRU Accelerometer5. (3) Subsequently, fault was injected in Accelerometer 4. Since 3 accelerometers are now faulty, primary ADIRU cannot provide the body acceleration data. (4) Finally, a fault was introduced in the Primary GPS Processor such that the position jumps high.

Table 5 shows relevant portions of the log file formatted to be displayed as a table. Column 2 is the name of the partition from which the message was recorded, Column 3 is the relative time in seconds since the start of experiment, event 1. Last Column is the descriptive message. At events 2, 10, and 20 three accelerometers failure were injected. It can be noted that there is always an approximate one second shift between the time of injection and time when alarm is registered on the system health manager. This is because the alarm aggregator buffers the alarms in a hyper period (1 second) before sending it to the diagnoser. This is to increase the likelihood that all related alarms are processed together. It can be seen that at event 25 more than two accelerometers from primary ADIRU were registered as faulty. This causes the stop messages to be sent to all components in the primary ADIRU subsystem, event 26. This is followed by start of Secondary ADIRU subsystem (27). Then, Confirmation from all components of primary and secondary system is received (abstracted into single event, event 29). Since Secondary ADIRU was in semi-active state, its accelerometers were already running and hence they ignore the start command sent by the health manager. It should be noted that each of these confirmations are also received with a delay due to the buffering action of Alarm Aggregator. Events 30 - 47 show the injection of GPS fault and the final recovery, including rewiring of PFC receptacles to the secondary GPS subsystem.

Table 6. Formatted Log file from the IMU execution.

	Part	Unix Time	Message
1	SHM	1306811930.14	Ready
2	pPart6	1306811972.11	Injecting Failure in Primary Acclerometer6
3	SHM	1306811973.138385746	ALARM AM_Primary_ADIRU_Subsystem_Accelerometer6_Acceleration _POSTCONDITION_FAILURE
4	SHM	1306811973.160051000	DIAGNOSER : DIAGNOSIS RESULTS : FAULTY COMPONENT Primary_ADIRU_Subsystem_Accelerometer6 Subsystem Primary_ADIRU_Subsystem
5	SHM	1306811973.160613885	SHM: SETTING COMPONENT Primary_ADIRU_Subsystem_Accelerometer6 TO FAULTY STATUS
6	SHM	1306811973.160999370	SHM: COMMAND STOP SENT TO COMPONENT Primary_ADIRU_Subsystem_Accelerometer6 Inactivated: Component Accelerometer6
7	SHM	1306811975.183065786	DIAGNOSER : DIAGNOSIS RESULTS : FAULTY COMPONENT Primary_ADIRU_Subsystem_Accelerometer6 Subsystem Primary_ADIRU_Subsystem
8	SHM	1306811975.183663159	SHM: COMPONENT Primary_ADIRU_Subsystem_Accelerometer6 EXECUTED PREVIOUS COMMAND
9	SHM	1306811975.184238910	SHM: SETTING COMPONENT Primary_ADIRU_Subsystem_Accelerometer6 TO FAULTY STATUS
10	pPart6	1306811982.122613333	[From Primary_ADIRU_Subsystem_Accelerometer5]Injecting Failure
11	SHM	1306811983.138341453	DIAGNOSER : ALARM RECEIVED : AM_Primary_ADIRU_Subsystem_Accelerometer5_Acceleration _POSTCONDITION_FAILURE Alarm::Publisher Acceleration:Component Accelerometer5:Subsystem Primary_ADIRU_Subsystem
12	SHM	1306811983.182080758	DIAGNOSER : DIAGNOSIS RESULTS : FAULTY COMPONENT Primary_ADIRU_Subsystem_Accelerometer6 Faulty: Component Accelerometer6:Subsystem Primary_ADIRU_Subsystem
13	SHM	1306811983.182210971	DIAGNOSER : DIAGNOSIS RESULTS : FAULTY COMPONENT Primary_ADIRU_Subsystem_Accelerometer5 Faulty: Component Accelerometer5:Subsystem Primary_ADIRU_Subsystem:Subsystem Primary_ADIRU_Subsystem

continued on next page

	Partition	Time	Message
1	SHM	0.00	Ready
2	pPart6	41.97	[From Primary_ADIRU_Subsystem_Accelerometer6]Injecting Failure
3	SHM	43.00	ALARM RECEIVED : AM_Primary_ADIRU_Subsystem_Accelerometer6_Acceleration_POSTCONDITION_FAILURE
4	SHM	43.02	FAULTY COMPONENT Primary_ADIRU_Subsystem_Accelerometer6 Subsystem Primary_ADIRU_Subsystem
6	SHM	43.02	COMMAND STOP SENT TO COMPONENT Primary_ADIRU_Subsystem_Accelerometer6
7	SHM	45.04	FAULTY COMPONENT Primary_ADIRU_Subsystem_Accelerometer6 Subsystem Primary_ADIRU_Subsystem
8	SHM	45.05	COMPONENT Primary_ADIRU_Subsystem_Accelerometer6 EXECUTED PREVIOUS COMMAND
10	pPart6	51.98	[From Primary_ADIRU_Subsystem_Accelerometer5]Injecting Failure
11	SHM	53.00	ALARM RECEIVED : AM_Primary_ADIRU_Subsystem_Accelerometer5_Acceleration_POSTCONDITION_FAILURE
12	SHM	53.04	FAULTY COMPONENT Primary_ADIRU_Subsystem_Accelerometer6 Faulty: Component Accelerometer6
13	SHM	53.04	FAULTY COMPONENT Primary_ADIRU_Subsystem_Accelerometer5 Faulty: Component Accelerometer5
15	SHM	53.04	COMMAND STOP SENT TO COMPONENT Primary_ADIRU_Subsystem_Accelerometer5
16	SHM	55.10	FAULTY COMPONENT Primary_ADIRU_Subsystem_Accelerometer6 Subsystem Primary_ADIRU_Subsystem
17	SHM	55.10	FAULTY COMPONENT Primary_ADIRU_Subsystem_Accelerometer5 Subsystem Primary_ADIRU_Subsystem
19	SHM	55.10	COMPONENT Primary_ADIRU_Subsystem_Accelerometer5 EXECUTED PREVIOUS COMMAND
20	pPart6	56.03	[From Primary_ADIRU_Subsystem_Accelerometer4]Injecting Failure
21	SHM	58.00	ALARM RECEIVED : AM_Primary_ADIRU_Subsystem_Accelerometer4_Acceleration_POSTCONDITION_FAILURE
22	SHM	58.09	FAULTY COMPONENT Primary_ADIRU_Subsystem_Accelerometer6 Subsystem Primary_ADIRU_Subsystem
24	SHM	58.09	FAULTY COMPONENT Primary_ADIRU_Subsystem_Accelerometer4 Subsystem Primary_ADIRU_Subsystem
25	SHM	58.09	FAULTY COMPONENT Primary_ADIRU_Subsystem_Accelerometer5 Subsystem Primary_ADIRU_Subsystem
26	SHM	58.09	COMMAND STOP ISSUED TO SUBSYSTEM Primary_ADIRU_Subsystem: Primary_ADIRU_Subsystem_Accelerometer1, Primary_ADIRU_Subsystem_Accelerometer2, Primary_ADIRU_Subsystem_Accelerometer3, Primary_ADIRU_Subsystem_Accelerometer4, Primary_ADIRU_Subsystem_ADIRUComputer1,Primary_ADIRU_Subsystem_ADIRUComputer2,Primary_ADIRU_Subsystem_ADIRUComputer3,Primary_ADIRU_Subsystem_ADIRUComputer4,Primary_ADIRU_Subsystem_VoterLeft, Primary_ADIRU_Subsystem_VoterRight,Primary_ADIRU_Subsystem_VoterCenter
27	SHM	58.09	COMMAND START ISSUED TO SUBSYSTEM Secondary_ADIRU_Subsystem: Secondary_ADIRU_Subsystem_Accelerometer1, Secondary_ADIRU_Subsystem_Accelerometer2, Secondary_ADIRU_Subsystem_Accelerometer3, Secondary_ADIRU_Subsystem_Accelerometer4, Secondary_ADIRU_Subsystem_Accelerometer5,Secondary_ADIRU_Subsystem_Accelerometer6,Secondary_ADIRU_Subsystem_ADIRUComputer1,Secondary_ADIRU_Subsystem_ADIRUComputer2,Secondary_ADIRU_Subsystem_ADIRUComputer3,Secondary_ADIRU_Subsystem_ADIRUComputer4,Secondary_ADIRU_Subsystem_VoterLeft, Secondary_ADIRU_Subsystem_VoterRight,Secondary_ADIRU_Subsystem_VoterCenter
29	SHM	60.88-71.20	Primary_ADIRU_Subsystem and Secondary_ADIRU_Subsystem EXECUTED PREVIOUS COMMANDS
30	pPart2	91.58	[From Primary_GPS_Subsystem_GPSProcessor]gps_data_src injecting fault
31	SHM	94.00	ALARM RECEIVED : AM_Primary_GPS_Subsystem_GPSProcessor_gps_data_src_getGPSData_POSTCONDITION_FAILURE
33	SHM	101.69	FAULTY COMPONENT Primary_ADIRU_Subsystem_Accelerometer6
34	SHM	101.69	FAULTY COMPONENT Primary_ADIRU_Subsystem_Accelerometer4
35	SHM	101.69	FAULTY COMPONENT Primary_ADIRU_Subsystem_Accelerometer5
36	SHM	101.69	FAULTY COMPONENT Primary_GPS_Subsystem_GPSReceiver
38	SHM	101.69	FAULTY COMPONENT Primary_GPS_Subsystem_GPSProcessor
39	SHM	101.69	COMMAND STOP ISSUED TO SUBSYSTEM Primary_GPS_Subsystem: Primary_GPS_Subsystem_GPSReceiver, Primary_GPS_Subsystem_GPSPRocessor
40	SHM	101.69	COMMAND START ISSUED TO SUBSYSTEM Secondary_GPS_Subsystem: Secondary_GPS_Subsystem_GPSReceiver, Secondary_GPS_Subsystem_GPSProcessor
41	SHM	101.69	COMMAND REWIRE_INTEFACE SENT TO COMPONENT Left_PFC_Subsystem_PFCNavFilter
43	SHM	101.69	REWIRE COMMAND DETAILS: INTERFACE Left_PFC_Subsystem_PFCNavFilter::GPSDataSource NEW-PROVIDER-COMPONENT Secondary_GPS_Subsystem_GPSProcessor
44	SHM	101.69	COMMAND REWIRE_INTEFACE SENT TO COMPONENT Right_PFC_Subsystem_PFCNavFilter : DETAILS: INTERFACE Right_PFC_Subsystem_PFCNavFilter::GPSDataSource NEW-PROVIDER-COMPONENT Secondary_GPS_Subsystem_GPSProcessor
45	SHM	101.69	COMMAND REWIRE_INTEFACE SENT TO COMPONENT Center_PFC_Subsystem_PFCNavFilter : DETAILS: INTERFACE Center_PFC_Subsystem_PFCNavFilter::GPSDataSource NEW-PROVIDER-COMPONENT Secondary_GPS_Subsystem_GPSProcessor
47	SHM	105.85-106.7	Command Execution Acknowledgement

Table 5. Formatted Log file. Event 1 at 1306811930.14 Unix Time

continued from previous page

	Part	Unix Time	Message
14	SHM	1306811983.182696581	SHM: SETTING COMPONENT Primary_ADIRU_Subsystem_Accelerometer6 TO FAULTY STATUS
15	SHM	1306811983.182770950	SHM: SETTING COMPONENT Primary_ADIRU_Subsystem_Accelerometer5 TO FAULTY STATUS
16	SHM	1306811983.183137684	SHM: COMMAND STOP SENT TO COMPONENT Primary_ADIRU_Subsystem_Accelerometer5 Inactivated: Component Accelerometer5
17	SHM	1306811985.237113225	DIAGNOSER : DIAGNOSIS RESULTS : FAULTY COMPONENT Primary_ADIRU_Subsystem_Accelerometer6 Subsystem Primary_ADIRU_Subsystem
18	SHM	1306811985.237253693	DIAGNOSER : DIAGNOSIS RESULTS : FAULTY COMPONENT Primary_ADIRU_Subsystem_Accelerometer5 Subsystem Primary_ADIRU_Subsystem
19	SHM	1306811985.237794079	SHM: COMPONENT Primary_ADIRU_Subsystem_Accelerometer5 EXECUTED PREVIOUS COMMAND
20	SHM	1306811985.238387117	SHM: SETTING COMPONENT Primary_ADIRU_Subsystem_Accelerometer6 TO FAULTY STATUS
21	SHM	1306811985.238463380	SHM: SETTING COMPONENT Primary_ADIRU_Subsystem_Accelerometer5 TO FAULTY STATUS
22	pPart6	1306811986.165013846	[From Primary_ADIRU_Subsystem_Accelerometer4]Injecting Failure
23	SHM	1306811988.138344947	DIAGNOSER : ALARM RECEIVED : AM_Primary_ADIRU_Subsystem_Accelerometer4_Acceleration_POSTCONDITION_FAILURE Alarm::Publisher Acceleration:Component Accelerometer4:Subsystem Primary_ADIRU_Subsystem
24	SHM	1306811988.226906017	DIAGNOSER : DIAGNOSIS RESULTS : FAULTY COMPONENT Primary_ADIRU_Subsystem_Accelerometer6 Subsystem Primary_ADIRU_Subsystem
25	SHM	1306811988.227034156	DIAGNOSER : DIAGNOSIS RESULTS : FAULTY COMPONENT Primary_ADIRU_Subsystem_Accelerometer4 Subsystem Primary_ADIRU_Subsystem
26	SHM	1306811988.227139408	DIAGNOSER : DIAGNOSIS RESULTS : FAULTY COMPONENT Primary_ADIRU_Subsystem_Accelerometer5 Subsystem Primary_ADIRU_Subsystem
27	SHM	1306811988.227648089	SHM: SETTING COMPONENT Primary_ADIRU_Subsystem_Accelerometer6 TO FAULTY STATUS
28	SHM	1306811988.227722922	SHM: SETTING COMPONENT Primary_ADIRU_Subsystem_Accelerometer4 TO FAULTY STATUS
29	SHM	1306811988.227793248	SHM: SETTING COMPONENT Primary_ADIRU_Subsystem_Accelerometer5 TO FAULTY STATUS
30	SHM	1306811988.228152141	SHM: COMMAND STOP ISSUED TO SUBSYSTEM Primary_ADIRU_Subsystem Inactivated: Subsystem Primary_ADIRU_Subsystem
31	SHM	1306811988.228296000	SHM: COMMAND STOP SENT TO COMPONENT Primary_ADIRU_Subsystem_Accelerometer1 Inactivated: Component Accelerometer1Inactivated: Subsystem Primary_ADIRU_Subsystem
32	SHM	1306811988.228532016	SHM: COMMAND STOP SENT TO COMPONENT Primary_ADIRU_Subsystem_Accelerometer6 Inactivated: Component Accelerometer6Inactivated: Subsystem Primary_ADIRU_Subsystem
33	SHM	1306811988.228754993	SHM: COMMAND STOP SENT TO COMPONENT Primary_ADIRU_Subsystem_Accelerometer2 Inactivated: Component Accelerometer2Inactivated: Subsystem Primary_ADIRU_Subsystem
34	SHM	1306811988.229002154	SHM: COMMAND STOP SENT TO COMPONENT Primary_ADIRU_Subsystem_Accelerometer3 Inactivated: Component Accelerometer3Inactivated: Subsystem Primary_ADIRU_Subsystem
35	SHM	1306811988.229231811	SHM: COMMAND STOP SENT TO COMPONENT Primary_ADIRU_Subsystem_Accelerometer4 Inactivated: Component Accelerometer4Inactivated: Subsystem Primary_ADIRU_Subsystem
36	SHM	1306811988.229479328	SHM: COMMAND STOP SENT TO COMPONENT Primary_ADIRU_Subsystem_Accelerometer5 Inactivated: Component Accelerometer5Inactivated: Subsystem Primary_ADIRU_Subsystem
37	SHM	1306811988.229704612	SHM: COMMAND STOP SENT TO COMPONENT Primary_ADIRU_Subsystem_ADIRUComputer1 Inactivated: Component ADIRUComputer1Inactivated: Subsystem Primary_ADIRU_Subsystem
38	SHM	1306811988.229953940	SHM: COMMAND STOP SENT TO COMPONENT Primary_ADIRU_Subsystem_VoterLeft Inactivated: Component VoterLeftInactivated: Subsystem Primary_ADIRU_Subsystem
39	SHM	1306811988.230183390	SHM: COMMAND STOP SENT TO COMPONENT Primary_ADIRU_Subsystem_ADIRUComputer2 Inactivated: Component ADIRUComputer2Inactivated: Subsystem Primary_ADIRU_Subsystem
40	SHM	1306811988.230411190	SHM: COMMAND STOP SENT TO COMPONENT Primary_ADIRU_Subsystem_VoterCenter Inactivated: Component VoterCenterInactivated: Subsystem Primary_ADIRU_Subsystem
41	SHM	1306811988.230636827	SHM: COMMAND STOP SENT TO COMPONENT Primary_ADIRU_Subsystem_ADIRUComputer3 Inactivated: Component ADIRUComputer3Inactivated: Subsystem Primary_ADIRU_Subsystem
42	SHM	1306811988.230865748	SHM: COMMAND STOP SENT TO COMPONENT Primary_ADIRU_Subsystem_VoterRight Inactivated: Component VoterRightInactivated: Subsystem Primary_ADIRU_Subsystem
43	SHM	1306811988.231091917	SHM: COMMAND STOP SENT TO COMPONENT Primary_ADIRU_Subsystem_ADIRUComputer4 Inactivated: Component ADIRUComputer4Inactivated: Subsystem Primary_ADIRU_Subsystem
44	SHM	1306811988.231309720	SHM: COMMAND START ISSUED TO SUBSYSTEM Secondary_ADIRU_Subsystem Activated: Subsystem Secondary_ADIRU_Subsystem
45	SHM	1306811988.231441068	SHM: COMMAND START SENT TO COMPONENT Secondary_ADIRU_Subsystem_Accelerometer5 Activated: Component Accelerometer5Activated: Subsystem Secondary_ADIRU_Subsystem
46	SHM	1306811988.231678584	SHM: COMMAND START SENT TO COMPONENT Secondary_ADIRU_Subsystem_Accelerometer1 Activated: Component Accelerometer1Activated: Subsystem Secondary_ADIRU_Subsystem
47	SHM	1306811988.231907056	SHM: COMMAND START SENT TO COMPONENT Secondary_ADIRU_Subsystem_Accelerometer4 Activated: Component Accelerometer4Activated: Subsystem Secondary_ADIRU_Subsystem
48	SHM	1306811988.232140229	SHM: COMMAND START SENT TO COMPONENT Secondary_ADIRU_Subsystem_Accelerometer2 Activated: Component Accelerometer2Activated: Subsystem Secondary_ADIRU_Subsystem
49	SHM	1306811988.232390930	SHM: COMMAND START SENT TO COMPONENT Secondary_ADIRU_Subsystem_Accelerometer6 Activated: Component Accelerometer6Activated: Subsystem Secondary_ADIRU_Subsystem
50	SHM	1306811988.232645527	SHM: COMMAND START SENT TO COMPONENT Secondary_ADIRU_Subsystem_Accelerometer3 Activated: Component Accelerometer3Activated: Subsystem Secondary_ADIRU_Subsystem
51	SHM	1306811988.232882717	SHM: COMMAND START SENT TO COMPONENT Secondary_ADIRU_Subsystem_VoterRight Activated: Component VoterRightActivated: Subsystem Secondary_ADIRU_Subsystem
52	SHM	1306811988.233121047	SHM: COMMAND START SENT TO COMPONENT Secondary_ADIRU_Subsystem_ADIRUComputer3 Activated: Component ADIRUComputer3Activated: Subsystem Secondary_ADIRU_Subsystem

continued on next page

continued from previous page

	Part	Unix Time	Message
53	SHM	1306811988.233366532	SHM: COMMAND START SENT TO COMPONENT Secondary_ADIRU_Subsystem_VoterLeft Activated: Component VoterLeftActivated: Subsystem Secondary_ADIRU_Subsystem
54	SHM	1306811988.233608990	SHM: COMMAND START SENT TO COMPONENT Secondary_ADIRU_Subsystem_ADIRUComputer1 Activated: Component ADIRUComputer1Activated: Subsystem Secondary_ADIRU_Subsystem
55	SHM	1306811988.233833918	SHM: COMMAND START SENT TO COMPONENT Secondary_ADIRU_Subsystem_ADIRUComputer2 Activated: Component ADIRUComputer2Activated: Subsystem Secondary_ADIRU_Subsystem
56	SHM	1306811988.234076470	SHM: COMMAND START SENT TO COMPONENT Secondary_ADIRU_Subsystem_VoterCenter Activated: Component VoterCenterActivated: Subsystem Secondary_ADIRU_Subsystem
57	SHM	1306811988.234314897	SHM: COMMAND START SENT TO COMPONENT Secondary_ADIRU_Subsystem_ADIRUComputer4 Activated: Component ADIRUComputer4Activated: Subsystem Secondary_ADIRU_Subsystem
58	SHM	1306811991.017957452	SHM: COMPONENT Primary_ADIRU_Subsystem_Accelerometer1 EXECUTED PREVIOUS COMMAND
59	SHM	1306811991.018033766	SHM: COMPONENT Primary_ADIRU_Subsystem_Accelerometer2 EXECUTED PREVIOUS COMMAND
60	SHM	1306811991.018104460	SHM: COMPONENT Primary_ADIRU_Subsystem_Accelerometer3 EXECUTED PREVIOUS COMMAND
61	SHM	1306811991.018172888	SHM: COMPONENT Primary_ADIRU_Subsystem_Accelerometer4 EXECUTED PREVIOUS COMMAND
62	SHM	1306811996.353463118	SHM: COMPONENT Secondary_ADIRU_Subsystem_VoterLeft EXECUTED PREVIOUS COMMAND
63	SHM	1306811996.353574715	SHM: COMPONENT Secondary_ADIRU_Subsystem_VoterCenter EXECUTED PREVIOUS COMMAND
64	SHM	1306811996.353644921	SHM: COMPONENT Secondary_ADIRU_Subsystem_VoterRight EXECUTED PREVIOUS COMMAND
65	SHM	1306812001.337392119	SHM: COMPONENT Primary_ADIRU_Subsystem_VoterLeft EXECUTED PREVIOUS COMMAND
66	SHM	1306812001.337514104	SHM: COMPONENT Primary_ADIRU_Subsystem_ADIRU Computer1 EXECUTED PREVIOUS COMMAND
67	SHM	1306812001.337587265	SHM: COMPONENT Primary_ADIRU_Subsystem_VoterCenter EXECUTED PREVIOUS COMMAND
68	SHM	1306812001.337654861	SHM: COMPONENT Primary_ADIRU_Subsystem_ADIRU Computer2 EXECUTED PREVIOUS COMMAND
69	SHM	1306812001.337720781	SHM: COMPONENT Primary_ADIRU_Subsystem_VoterRight EXECUTED PREVIOUS COMMAND
70	SHM	1306812001.337792430	SHM: COMPONENT Primary_ADIRU_Subsystem_ADIRU Computer3 EXECUTED PREVIOUS COMMAND
71	SHM	1306812001.337859599	SHM: COMPONENT Primary_ADIRU_Subsystem_ADIRU Computer4 EXECUTED PREVIOUS COMMAND
72	pPart2	1306812021.722214929	[From Primary_GPS_Subsystem_GPSProcessor]gps_data_src injecting fault
73	SHM	1306812024.138428206	DIAGNOSER : ALARM RECEIVED : AM_Primary_GPS_Subsystem_GPSProcessor _gps_data_src_getGPSData_POSTCONDITION_FAILURE Alarm::Method getGPSData:Provided gps_data_src:Component GPSProcessor:Subsystem Primary_GPS_Subsystem
74	SHM	1306812031.823803072	DIAGNOSER : DIAGNOSIS RESULTS : FAULTY COMPONENT Primary_GPS_Subsystem_GPSReceiver Subsystem Primary_GPS_Subsystem
75	SHM	1306812031.823958923	DIAGNOSER : DIAGNOSIS RESULTS : FAULTY COMPONENT Primary_ADIRU_Subsystem_Accelerometer6 Subsystem Primary_ADIRU_Subsystem
76	SHM	1306812031.824066362	DIAGNOSER : DIAGNOSIS RESULTS : FAULTY COMPONENT Primary_GPS_Subsystem_GPSProcessor Subsystem Primary_GPS_Subsystem
77	SHM	1306812031.824177648	DIAGNOSER : DIAGNOSIS RESULTS : FAULTY COMPONENT Primary_ADIRU_Subsystem_Accelerometer4 Subsystem Primary_ADIRU_Subsystem
78	SHM	1306812031.824286538	DIAGNOSER : DIAGNOSIS RESULTS : FAULTY COMPONENT Primary_ADIRU_Subsystem_Accelerometer5 Subsystem Primary_ADIRU_Subsystem
79	SHM	1306812031.824915858	SHM: SETTING COMPONENT Primary_GPS_Subsystem_GPSReceiver TO FAULTY STATUS
80	SHM	1306812031.825002192	SHM: SETTING COMPONENT Primary_ADIRU_Subsystem_Accelerometer6 TO FAULTY STATUS
81	SHM	1306812031.825087094	SHM: SETTING COMPONENT Primary_GPS_Subsystem_GPSProcessor TO FAULTY STATUS
82	SHM	1306812031.825164800	SHM: SETTING COMPONENT Primary_ADIRU_Subsystem_Accelerometer4 TO FAULTY STATUS
83	SHM	1306812031.825236589	SHM: SETTING COMPONENT Primary_ADIRU_Subsystem_Accelerometer5 TO FAULTY STATUS
84	SHM	1306812031.825562303	SHM: COMMAND STOP ISSUED TO SUBSYSTEM Primary_GPS_Subsystem Inactivated: Subsystem Primary_GPS_Subsystem
85	SHM	1306812031.825705730	SHM: COMMAND STOP SENT TO COMPONENT Primary_GPS_Subsystem_GPSReceiver Inactivated: Component GPSReceiverInactivated: Subsystem Primary_GPS_Subsystem
86	SHM	1306812031.825984165	SHM: COMMAND STOP SENT TO COMPONENT Primary_GPS_Subsystem_GPSProcessor Inactivated: Component GPSProcessorInactivated: Subsystem Primary_GPS_Subsystem
87	SHM	1306812031.826209854	SHM: COMMAND START ISSUED TO SUBSYSTEM Secondary_GPS_Subsystem Activated: Subsystem Secondary_GPS_Subsystem
88	SHM	1306812031.826349636	SHM: COMMAND START SENT TO COMPONENT Secondary_GPS_Subsystem_GPSReceiver Activated: Component GPSReceiverActivated: Subsystem Secondary_GPS_Subsystem
89	SHM	1306812031.826591712	SHM: COMMAND START SENT TO COMPONENT Secondary_GPS_Subsystem_GPSProcessor Activated: Component GPSProcessorActivated: Subsystem Secondary_GPS_Subsystem
90	SHM	1306812031.826868861	SHM: COMMAND REWIRE_INTEFACE SENT TO COMPONENT Left_PFC_Subsystem_PFCNavFilter
91	SHM	1306812031.827022578	SHM: REWIRE COMMAND DETAILS: INTERFACE Left_PFC_Subsystem_PFCNavFilter::GPSDataSource NEW-PROVIDER-COMPONENT Secondary_GPS_Subsystem_GPSProcessor
92	SHM	1306812031.827116318	SHM: COMMAND REWIRE_INTEFACE SENT TO COMPONENT Right_PFC_Subsystem_PFCNavFilter

continued on next page

continued from previous page			
	Part	**Unix Time**	**Message**
93	SHM	1306812031.827267327	SHM: REWIRE COMMAND DETAILS: INTERFACE Right_PFC_Subsystem_PFCNavFilter::GPSDataSource NEW-PROVIDER-COMPONENT Secondary_GPS_Subsystem_GPSProcessor
94	SHM	1306812031.827358278	SHM: COMMAND REWIRE_INTEFACE SENT TO COMPONENT Center_PFC_Subsystem_PFCNavFilter
95	SHM	1306812031.827498855	SHM: REWIRE COMMAND DETAILS: INTERFACE Center_PFC_Subsystem_PFCNavFilter::GPSDataSource NEW-PROVIDER-COMPONENT Secondary_GPS_Subsystem_GPSProcessor
96	SHM	1306812035.983657688	SHM: COMPONENT Primary_GPS_Subsystem_GPSReceiver EXECUTED PREVIOUS COMMAND
97	SHM	1306812035.983770669	SHM: COMPONENT Primary_GPS_Subsystem_GPSProcessor EXECUTED PREVIOUS COMMAND
98	SHM	1306812035.983849759	SHM: COMPONENT Left_PFC_Subsystem_PFCNavFilter EXECUTED PREVIOUS COMMAND
99	SHM	1306812035.983928751	SHM: COMPONENT Secondary_GPS_Subsystem_GPSProcessor EXECUTED PREVIOUS COMMAND

3 Conclusion

In summary, the report demonstrated the effectiveness of a two-level health management strategy by replicating a real-life episode involving on an aircraft subsystem - Inertial Measurement Unit (IMU). The design and implementation of the example realistic IMU system involved a real-time component model that introduces component-based software engineering techniques into real-time systems. Components, their interfaces, and interactions are explicitly modeled, and these models are annotated with observable pre- and post-conditions, as well as timing requirements. An anomaly detection system is constructed from these specifications which performs the monitoring on the software system, and, if needed, triggers a health management (mitigation) action. Health management happens first on the component level and then if required at the system-level. The mitigation at the component level is facilitated by a designer-specified reactive state machine. At the system level, a diagnosis process is triggered first which isolates the source of the cascading fault. The diagnosis result in turn trigger the system level mitigation strategy - either reactive or delibartive - which is responsible for restoring the system functionality.

4 Acknowledgments

This report is based upon work supported by NASA under award NNX08AY49A. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Aeronautics and Space Administration. The authors would like to thank Paul Miner, Eric Cooper, and Suzette Person of NASA Langley Research Center for their help and guidance on the project.

References

1. ARINC specification 653-2: Avionics application software standard interface part 1 - required services. Tech. rep.
2. Avizienis, A., Laprie, J.C., Randell, B., Landwehr, C.: Basic concepts and taxonomy of dependable and secure computing. IEEE Transactions on Dependable and Secure Computing 1(1), 11–33 (Jan 2004)
3. Bureau, A.T.S.: In-flight upset; 240km NW Perth, WA; Boeing Co 777-200, 9M-MRG. Tech. rep. (August 2005), `http://www.atsb.gov.au/publications/investigation_reports/2005/AAIR/aair200503722.aspx`
4. Bureau, A.T.S.: AO-2008-070: In-flight upset, 154 km west of Learmonth, WA, 7 October 2008, VH-QPA, Airbus A330-303. Tech. rep. (October 2008), `http://www.atsb.gov.au/publications/investigation_reports/2008/AAIR/aair200806143.aspx`
5. Butler, R.: A primer on architectural level fault tolerance. Tech. rep., NASA Scientific and Technical Information (STI) Program Office, Report No. NASA/TM-2008-215108 (2008), `http://shemesh.larc.nasa.gov/fm/papers/Butler-TM-2008-215108-Primer-FT.pdf`
6. Charette, R.: This car runs on code. IEEE Spectrum, February (2009)
7. DO-178B, Software considerations in airborne systems and equipment certification. RTCA, Incorporated (1992)
8. Dubey, A., Karsai, G., Mahadevan, N.: A component model for hard real-time systems: CCM with ARINC-653. Software: Practice and Experience 41(12), 1517–1550 (2011), `http://dx.doi.org/10.1002/spe.1083`
9. Greenwell, W.S., Knight, J., Knight, J.C.: What should aviation safety incidents teach us? In: SAFECOMP 2003, The 22nd International Conference on Computer Safety, Reliability and Security (2003)

10. Jaffe, M., Busser, R., Daniels, D., Delseny, H., Romanski, G.: Progress report on some proposed upgrades to the conceptual underpinnings of do-178b/ed-12b. In: System Safety, 2008 3rd IET International Conference on. pp. 1–6. IET (2008)
11. Johnson, C.W.;Holloway, C.: The dangers of failure masking in fault-tolerant software: Aspects of a recent in-flight upset event. In: 2nd IET Systems Safety Conference, The IET, Savoy Place, London, UK. pp. 60–65. IEE (2007)
12. Johnson, S., Gormley, T., Kessler, S., Mott, C., Patterson-Hine, A., Reichard, K., Scandura Jr, P.: System Health Management: With Aerospace Applications. John Wiley & Sons, Inc (2011)
13. Kopetz, H., Holzmann, M., Elmenreich, W.: A universal smart transducer interface: TTP/A. In: Object-Oriented Real-Time Distributed Computing, 2000. (ISORC 2000) Proceedings. Third IEEE International Symposium on. pp. 16 –23 (2000)
14. de Lemos, R.: Analysing failure behaviours in component interaction. Journal of Systems and Software 71(1-2), 97 – 115 (2004)
15. Lyu, M.R.: Software Fault Tolerance, vol. New York, NY, USA. John Wiley & Sons, Inc (1995), `http://www.cse.cuhk.edu.hk/~lyu/book/sft/`
16. Mcintyre, M.D.W., Sebring, D.L.: Integrated fault-tolerant air data inertial reference system (1994)
17. Potocti de Montalk, J.: Computer software in civil aircraft. In: Digital Avionics Systems Conference, 1991. Proceedings., IEEE/AIAA 10th. pp. 324 –330 (oct 1991)
18. NASA: Report on the loss of the mars polar lander and deep space 2 missions. Tech. rep., NASA (2000), `ftp://ftp.hq.nasa.gov/pub/pao/reports/2000/2000_mpl_report_1.pdf`
19. Ofsthun, S.: Integrated vehicle health management for aerospace platforms. Instrumentation Measurement Magazine, IEEE 5(3), 21 – 24 (Sep 2002)
20. Prisaznuk, P.: Arinc 653 role in integrated modular avionics (IMA). In: Digital Avionics Systems Conference, 2008. DASC 2008. IEEE/AIAA 27th. pp. 1.E.5–1 – 1.E.5–10. IEEE (2008)
21. Sha, L.: The complexity challenge in modern avionics software. In: National Workshop on Aviation Software Systems: Design for Certifiably Dependable Systems (2006)
22. Sheffels, M.: A fault-tolerant air data/inertial reference unit. In: Digital Avionics Systems Conference, 1992. Proceedings., IEEE/AIAA 11th. pp. 127 –131 (Oct 1992)
23. Srivastava, A., Schumann, J.: The Case for Software Health Management. In: Fourth IEEE International Conference on Space Mission Challenges for Information Technology, 2011. SMC-IT 2011. pp. 3–9 (August 2011)
24. Torres-Pomales, W.: Software fault tolerance: A tutorial. Tech. rep., NASA (2000), `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.32.8307`
25. Wallace, M.: Modular architectural representation and analysis of fault propagation and transformation. Electron. Notes Theor. Comput. Sci. 141(3), 53–71 (2005)
26. Wang, N., Schmidt, D.C., O'Ryan, C.: Overview of the CORBA component model. Component-based software engineering: putting the pieces together pp. 557–571 (2001)

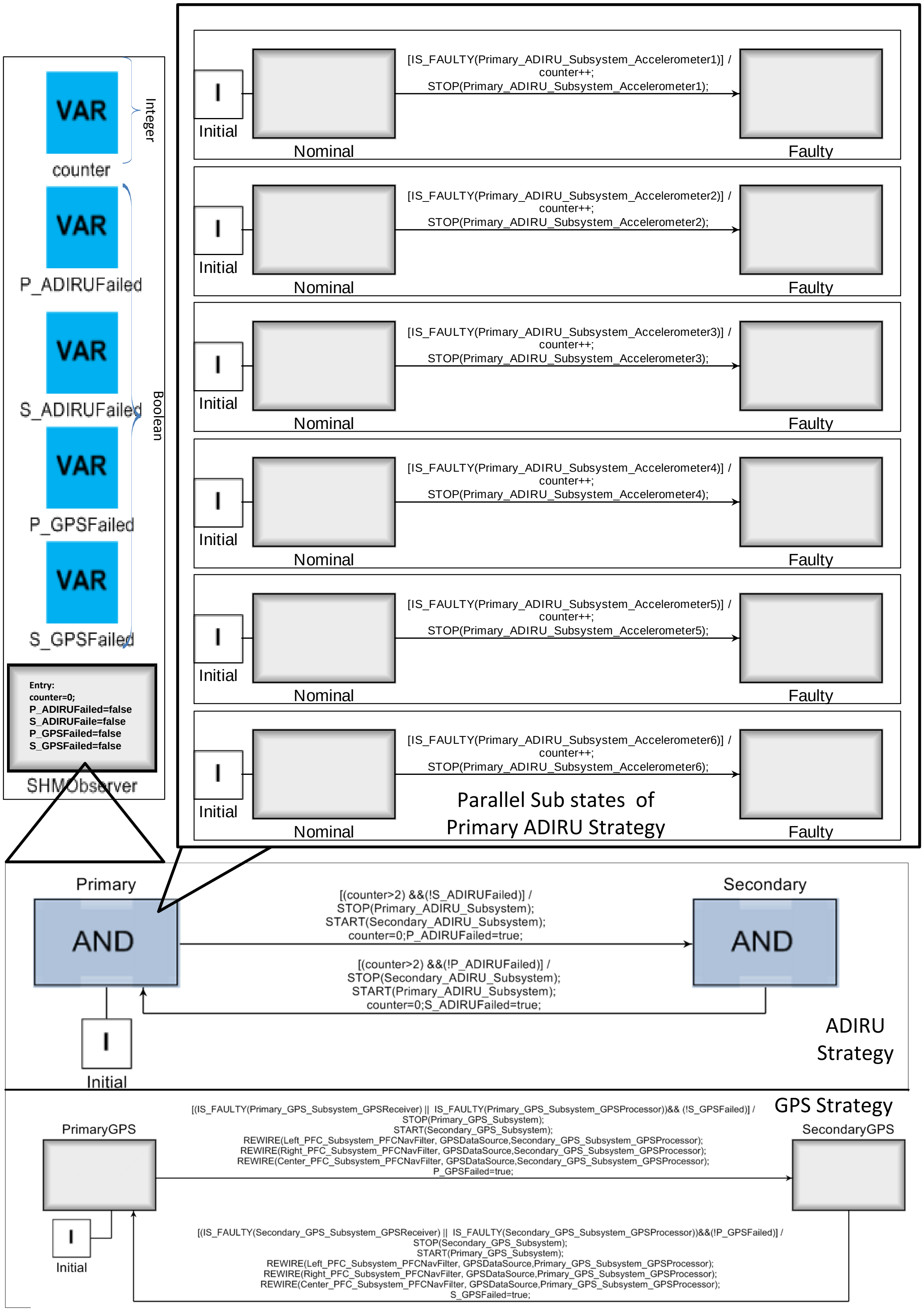

Fig. 5. Part of SHM State Machine from the IMU Assembly. The state machine for mitigating problems with secondary is similar to the primary.