CHEMICAL ENGINEERING TRANSACTIONS
VOL. 33, 2013
A publication of AIDIC
The Italian Association of Chemical Engineering
Online at: www.aidic.it/cet
Guest Editors: Enrico Zio, Piero Baraldi

ISBN 978-88-95608-24-2; **ISSN** 1974-9791
DOI: 10.3303/CET1333109

Testing Method of Integrated Modular Avionics Health Monitoring

Huinan Zhang*, Shihai Wang, Bin Liu, Xiaoxu Diao

Beihang University of China, School of Reliability and system engineering,Beijing,100191, Peoples R China, Science&Technology on Reliability&Environment Engineering Laboratory,Beijing,100191,Peoples R China.
holyjiangnan@126.com

Avionics software is safe-critical embedded software and its architecture is evolving from traditional federated architectures to Integrated Modular Avionics (IMA) to improve resource usability. ARINC653, as a standard widely employed in the avionics industry, supports partitioning concepts in accordance with the IMA philosophy. Monitoring the health of certain aerospace structures has been shown to be a key step in reducing the life cycle costs for structural maintenance and inspection. Since the health of the structures ultimately determines the health of a vehicle, health monitoring is also an important prerequisite for improved aviation safety. In this paper, we present the preliminary results from our ongoing project on designing and evaluating architectures for integrated structural health monitoring. To insure the development of the avionics software constructed on ARINC653 operating system with high reliability and efficiency, we propose a model-driven design methodology based on Architecture Analysis &Design Language (AADL) for HM module of ARINC653 system. This paper presents an approach for the modeling, verification and implementation of ARINC653 systems using AADL. It details a modeling approach exploiting the new features of AADL version 2 for the design of ARINC653 architectures. It also proposes modeling patterns to represent the safety mechanisms of the HM module of Integrated Modular Avionics. Thus, it assists system engineers to simulate and validate non functional requirements such as scheduling or resources dimensioning and then propose a method of testing the HM module of Integrated Modular Avionics using Fault Injection and Program Instrumentation Technology.

1. HM of IMA

Since the 1970s, with the rapid development of electronic technology, avionics accounted for more than 40% of the total cost of aircraft at the same time more and more complex structure of the system also presented a greater challenge to the reliability of the aircraft, 21 century avionics system has been facing serious challenges of cost, performance, and reliability.

With the continuous development of the technology of modern airborne equipment, software changes a lot. Avionics system software from the 1960s discrete avionics system, after the 1970s, data transmission, display, comprehensive, development of data processing to the 1990s data fusion, sensor fusion, integrated with the antenna of a new generation of integrated modular avionics systems. Integrated Modular Avionics(IMA) is the highest level of the structural development of the avionics, designed to reduce aircraft LCC, to improve avionics functionality and performance and to solve the software upgrading and hardware problem.

Health monitoring (HM) is a core content of the IMA system standard. HM is operated by O/S, responsible for monitoring and reporting system hardware, application software and operating system software defect and failure, help isolating system failures and preventing failures from spreading. Health monitoring deal with errors in the avionics system for diagnosis, classification, response, and recovery strategies, and health status monitoring through monitoring tables on the system, the module health monitoring table and partition health monitoring tables of the three health monitoring table management system health monitoring, a system error classification, classification, and press level dispatch module health monitoring

Please cite this article as: Zhang H., Wang S., Liu B., Diao X., 2013, Testing method of integrated modular avionics health monitoring, Chemical Engineering Transactions, 33, 649-654 DOI: 10.3303/CET1333109

tasks, partition health monitoring tasks, process health monitoring treatment process scheduling error recovery action

1.1 Health Monitoring table

Health monitoring is managed by three health monitoring table, including the table of system health monitoring, the module health monitoring table, partition health monitoring table. These tables running by system integrators static configuration, access to part of the configuration as the system logic. System health monitoring table is made of error code, state level and the dispatch level. System level check system health monitoring tables based on the state of the system when the error code and error events, error events dispatch level, pursuant to which the level sent to different health monitoring tasks. The module health monitoring table is sent to the module-level health monitoring tasking failure, Recovery strategy formulated by the error code and error events system status, error handler (error recovery action) entrance composed. District Health Monitoring table for failure to dispatch to the partition level health monitoring tasking Recovery Strategy formulation, each partition has a separate partition health monitoring tables, error code, error events when the system status, error handler (error recovery action) inlet.

1.2 Error diagnosis

Error diagnosis, running by processor error diagnostic functions or operating system, processor diagnostic errors, is divided into three levels: the module level, partition-level, process-level system will diagnose errors and to determine the error code identification. System health monitoring table, determined by the error code and the system state level to the wrong dispatch level, and the error classification system, the state of the system is also divided into three, namely: the module level, area level, the process level. It can diagnose fault such as storage area protection faults, overflow, division by zero, time interrupts, input / output errors, operating system diagnostic errors configured fault, overtime and other mistakes,

1.3 Error response

The error response, relies on the diagnostic error occurring. when an error occurs, the system state District Health Monitoring table and module health monitoring table by the error code , the system status and specific error handler relationship. I.e.,the state of the system managed by the operating system, which includes the module is initialized, perform the module health monitoring task execution, non-health monitoring tasks or partition-related tasks, partition switching, health monitoring task execution, partition initialization, the core of the operating system partition related task execution, process management, process execution state.

1.4 Error recovery action

The module level, partition-level errors, module health monitoring tables and District Health Monitoring table are table-driven processing. Module the health surveillance table and the District Health Monitoring table by the error code and error event occurs when the state of the system, the error handler inlet composition sent to the module level or partition-level errors, the module health monitoring tasks or and partition health monitoring tasks according to error code, the system state in the the module health monitoring table and partition health monitoring table lookup to error handler entrance, go to a specific error handler to complete error recovery actions. Process-level error response is handled by the health monitoring process, the application can be connected to the error handler. Module level errors and partition-level error recovery activities is statically configured health monitoring table configuration. Module-level Recovery Action: Ignore (only the record, but no action is taken), error reporting, reported to the avionics system reset, shutdown, etc.; partition-level error recovery action: stop the partition ignored (only records, but not no action is taken), reporting errors, re-boot partition (cold or warm start); process-level error recovery action is connected to the application error handler is: ignore repeated verification, before the implementation of recovery activities, (n times), stop error process and re-initialized from the entry point, stop error process and start another process, stop error process, re-boot partition (cold or warm start), to stop the partition (set to idle state).

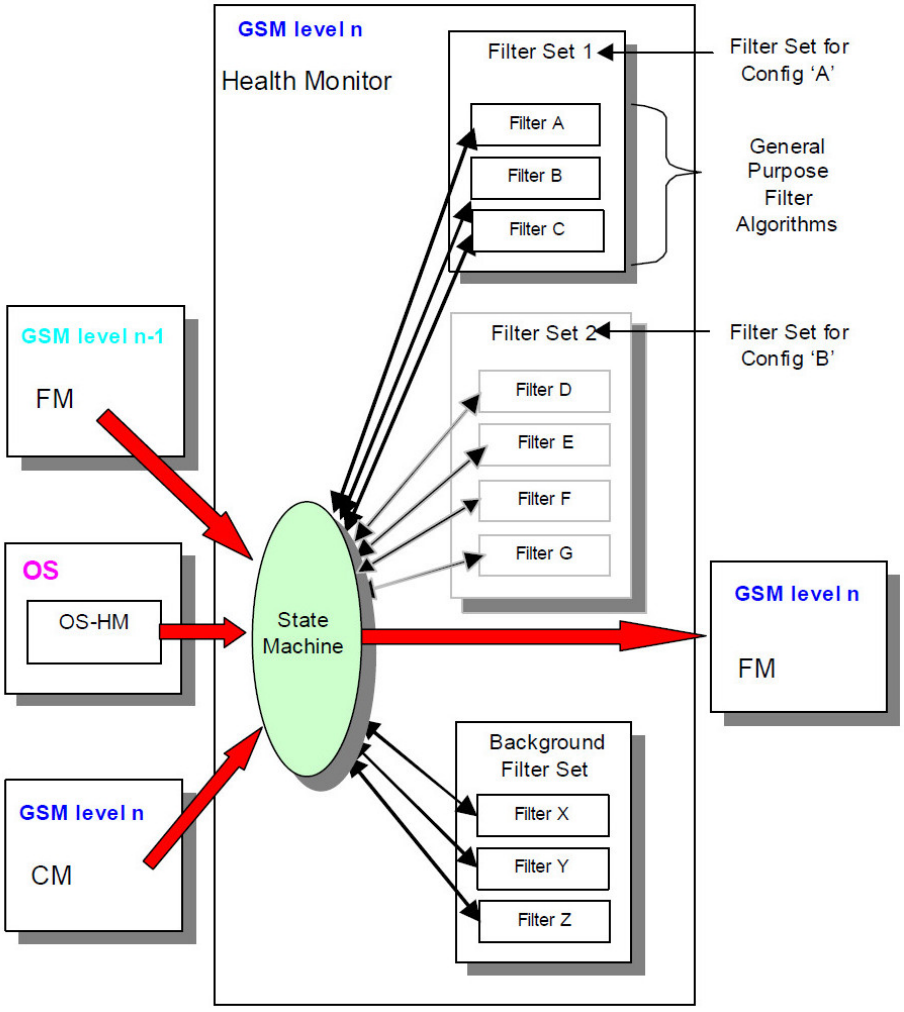

Figure 1: HM system

Embedded software test refers to the method through the use of software and hardware, simulate input and output signals of the tested software crosslinking system and its physical, software running of the simulation measured real environment, and the test carried out in the environment. The complexity and cost of the all-digital simulation are very high, and the reliability test requirements as close as possible to the software running the real situation, embedded software testing is generally used in the loop simulation environment, also known as the hardware (physical) in the loop (Hardware in the Loop) simulation. This simulation part of the system is described mathematical model and put it into a simulation model; another part of the physical (or physical models) way to introduce simulation loop.

Embedded software simulation testing environment needs to have high simulation accuracy, also need to meet the real-time conditions. Taking into account the embedded software generally have relatively strong real-time characteristics, so be sure to take advantage of the real-time environment simulation testing environment to achieve. In addition, the simulation test environment must support the interface signals used by the system hardware and communication protocols, as well as precise I/ O timing needs. So, the only real-time simulation in order to meet the necessary prerequisite for reliability testing of the demanding requirements of the system operating environment fidelity real-time and half-physical simulation.

2. AADL modeling

Architecture Analysis and Design Language AADL is a character and graphical language, jointly proposed by the SAE (Society for Automotive Engineers) Sub-Committee of Architecture Description Language, Embedded Computing Systems Committee, Avionics Systems hardware and software architecture for real-time systems design and analysis of key performance.

AADL development aimed at challenging resourcerestrictions and strict real-time response requirements of embedded systems modeling. These systems are generally considered to be highly reliable, potential applications include avionics, automatic control systems, flight management system, engine and power train control systems, medical equipment, industrial process control equipment, robotics, and aerospace applications. AADL provides a standard and accurate enough (the machine can handle) method of embedded real-time systems architecture modeling and analysis of system properties, and support systems to achieve predictability integration. AADL is defined for description of the system components, interfaces, and components portfolio standard methods, in order to facilitate the exchange of engineering data between multiple organizational and technical disciplines in embedded real-time systems. An accurate description of the machine can handle concepts and runtime architecture approach provides the framework for system modeling and analysis, and the convenience of automatic code generation, system architecture, and other development activities, can significantly reduce the design and implementation flaws. And AADL can be extended to support more applications, such extensions can be defined as the Annex a part of the core standards.

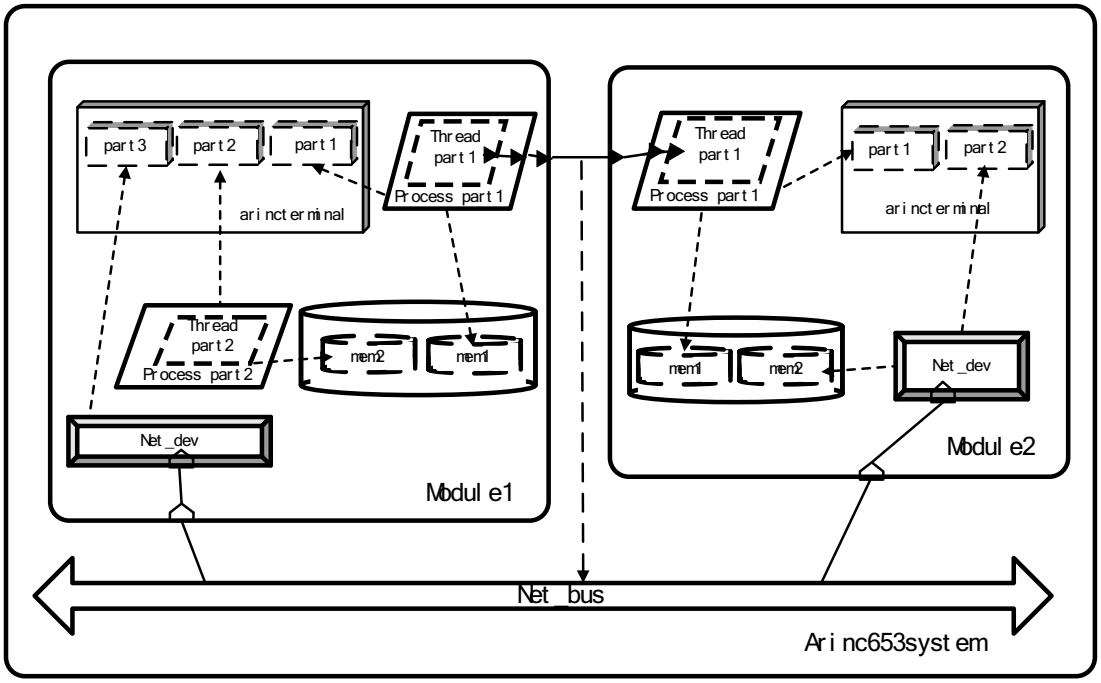

Figure 2: AADL model of ARINC653 system

AADL can be used to describe the system structure mapping software components to execution platform. It can be used to describe the functional interface components (such as data input and output) and the performance of key attributes of the components (such as real-time). The AADL is also used to describe the interactions between components, such as input and output of data on how to connect how to locate or application software components to execution platform components. By providing model-driven mode and the mode switching, AADL is able to describe the dynamic behavior of the run-time architecture. These describe the system designers to complete the analysis of the components and systems, for example, the system schedulability analysis, particle size analysis, credibility analysis. From these analyzes, the designer can barium l assessment balance and change of architecture. The benefits of doing so can be in the system before encoding to complete the analysis and evaluation of the performance of the system, to provide assurance for the further development of the system.

3. Software fault injection

Fault injection technology as a non-traditional software testing techniques is the fault in accordance with the specific fault model, the employer is conscious way, and impose specific fault in the system to be tested, in order to speed up the system error failure occurred. Software fault injection (referred SFI) is a method of using software technology to test the software system, first started in 1978 DeMillo proposed program mutation testing. The SFI can improve software quality assessment software level. Widely used in

a number of high-reliability software and in areas such as safety-critical software, such as aviation, aerospace, software testing.

The purpose of the SFI injection according to the type of fault is to observe the input and output of the system and to evaluate the robustness and reliability of the system, and ultimately avoid software failure and losses in practical applications. Typically, the type of fault can be injected into the: 1) memory failure; 2) processor failure; 3) communication failure; 4) the process of failure, such as deadlock, livelock, process cycle, the process hangs and excessive use of system resources messages, etc.; 5) failure, such as loss of a message, corrupted message, wait for the message timeout disorder information, the information copied and components; 6) network failure; 7) operating program failure; 8) code failure such as statements, variables, and so on.

The SFI general steps: 1) analysis of the current test object, process and test purposes; 2) using a certain method to generate fault use case; 3) analog fault type; 4) management fault injection process, such as fault coverage and recovery mechanisms, etc.; 5) running target system implemented fault injection; 6) analysis and evaluation of test results and test performance.

The the software dynamic environment fault injection is achieved through the establishment of the simulation model, and then insert fault injection unit inside the model fault injection. This is a non-intrusive fault injection, can be injected into the hardware failures and software faults Source code analysis, artificial or manually identify possible fault injection points, to modify the source code, such as modifying the assignment change execution branches to change the function call, use the compiler to recompile the source code modified, this generated when the program and the program under test is not a mirror, and then run the modified program to observe the health of the recording software, and analysis, thus confirming the validity fault injection. Repeatedly modify the source code, you always need to be recompiled to run again.

The software dynamically binary fault injection (source code). Software startup in debug mode, fault injection information defined in the script debugger, start the software debugger and load the debugger script, the script defined breakpoint location and fault injection operation, when the program runs, and each run to a breakpoint on the fault injection instructions execute scripts, record information, and then immediately release program. When the After all breakpoints fault injection, according to recording information, we can determine whether the fault injection. This injection process needs program started in debug mode, some loss in performance, fault injection for high-level language.

4. Conclusions

The traditional method of testing for health monitoring is no longer applicable, and the difficulties caused by the new features for health monitoring, put forward the ideas and methods of the following apply to the health monitoring test:

1) based on model-driven software testing methods , there needs to be a software testing modeling language, we can use the health monitoring software and its surrounding environment (IMA platform modeling and other software), initially built a testing model and testing environment, and we can solve the problem that health monitoring software cannot be run independently. Then we can use model-driven testing method to build test cases and run the testing process.

2) health monitoring system and the difference between ordinary avionics software is not a regular feature that it does not deal with and respond to the situation, but the fault of the software, in the design of test cases we cannot simply deal with its regular features, and more importantly, is the test for the identification of the software fault, the judgment and processing capabilities. Software failure occurring, after all, is a small probability event, we can't wait them occur randomly and test them; avionics software running environment and the actual testing environment is different, so the type and frequency of failure would be unable to accurately restored. This requires us artificially set that runtime failures happen in the aviation environment and its probability of occurrence in order to test for the identification of these failures, judgment and processing capacity. Preliminary research on IMA system health monitoring and fault injection technology are needed.

3) IMA system time partitioning scheduling is based on the running blueprint, the time scheduling, scheduling case of the normal time, the respective partition occupied in the time window of a certain time, each time window will stay out of some spare to cope with possible interruptions, this division is in the integration of the system running the blueprint good allocated, and mature software can be assigned. Encountered in scheduling the actions taken by the health monitoring may lead to blueprint unreasonably running, then we need to test these emergencies' running rationality and the consideration to be dealt with.

References

Aeronautical Radio, Inc. ARINC Specification 653-1 Avi-onics application software standard interface[S]. Annapolis: Aeronautical Radio, Inc, 2003.

Airlines Electronic Engineering Committee . Avionics Application Software Standard Interface Part 1 - Required Services [S] . 2551 Riva Rode , Annapolis , Maryland 21401 - 7435:ARINC , March 7 , 2006 .

Barry M., Horvath G., "Prototype Implementation of a Goal-Based Software Health Management Service," smc-it, pp.117-124, Third IEEE International Conference on Space Mission Challenges for Information Technology, 2009

Hertel T., Over H., Bludau H., Ertl G.,1994, Phys. Rev. B 50, 8126.

Kern K., 1994, The Chemical Physics of Solid Surfaces, vol. 7: Phase Transitions and Adsorbate Restructuring at Metal Surfaces, Eds. King D.A., Woodruff D.P., Elsevier, Amsterdam, the Netherlands.

Kjurkchiev N., Andreev A., 1990, Two-sided method for computation of all multiple roots of an algebraic polynomial, Serdica 15, 302-330 (in Russian).

Liu Jianjun, Zhong Shan, Ye Hong, 2009, Reliability modelingfor airborne equipment system using AADL[J]. Aeronautical Computing Technique, 39(2): 90 94.

SAE. AS 5506-2004 Embedded Computing Systems Committee,Aerospace Avionics Systems Division, Architecture Analysis andDesign Language(AADL)[S]. 2004.