Chapter

Design of Intelligent and Open Avionics System Onboard

Changqing Wu, Xiaodong Han and Yakun Wang

Abstract

The continuous development of space missions has put forward requirements for high performance, high reliability, intelligence, effective integration, miniaturization, and quick turn around productization of the electronic system of satellites. The complexity of satellites has continued to increase, and the focus of satellite competition has shifted from the launch of success shifts to communication capacity, performance indicators, degree of flexibility, and continuous service capabilities. So, the importance of onboard avionics system is becoming increasingly prominent. In the future, the advanced avionics system integrates most of the platform's electronic equipment. The design level of the system largely determines the performance of the satellite platform. This chapter focuses on the application requirements of the new generation of intelligent avionics system for future communication satellites and adopts an "open" architecture of "centralized management, distributed measurement and drive, and software and hardware 'modular' design" to build a universal, standardized, and scalable intelligent avionics system.

Keywords: satellite, avionics system, intelligent, open architecture, modular design, centralized management, reliability

1. Introduction

With the continuous advancement of electronics and computer technology, the functions and performance of spacecraft avionics system have also continuously improved, covering functions such as spacecraft remote measurement and remote management, energy management, thermal management, health management, payload information processing, and mission task management. Avionics system plays a core role in the realization of information sharing and comprehensive utilization, function integration, resource reorganization and optimization, and information processing and transmission [1]. It is the foundation for spacecraft to implement autonomous management and control and is also a bridge for communication management from a spacecraft to other spacecrafts and ground station [2].

The traditional spacecraft electronic system uses a layered centralized management control mode similar to a pyramid. It not only needs a large amount of data interaction between the management unit and the interface unit but also requires the management unit to process a large amount of underlying data, which makes the management unit overwhelmed. It severely limits the processing and support of high-level tasks by electronic systems. Moreover, the management unit is at the top of the "pyramid" of centralized management, which requires high reliability. Once

a failure occurs, the entire electronic system will fail. Thus, the centralized management method is no longer suitable for the needs of spacecraft development.

The satellite intelligent avionics system is an information processing and transmission system that uses computer network technology to interconnect satellite-borne electronic equipment on the satellite to achieve internal information sharing and comprehensive utilization, function integration, and resource reorganization and optimization. Utilizing onboard computers to complete satellite data management, control management, communication management, time management, energy management, and job management functions through unified scheduling of satellite missions. Its essence is the generation, identification, processing, analysis, transmission, and distribution of information process. The integrated satellite electronic system integrates the functions of the satellite platform electronic equipment, and its design level directly determines the performance of the satellite platform [3–5].

At present, satellite sub-systems mostly adopt independent design schemes, which decentralize satellite attitude control, propulsion control, thermal control, satellite-ground link communication, and power control functions. The onboard computer is responsible for tasks such as remote control, telemetry, program-controlled operation, thermal control, and time management. The attitude and orbit control computer are responsible for attitude and orbit (including propulsion control) control. Each sub-system such as power supply, thermal control, and digital transmission is equipped with corresponding lower-level computers responsible for telemetry acquisition and remote control of the respective sub-system. However, the satellite system designed using this approach is usually resulting in heavy weight, high power consumption, large volume (aka high size, weight, and power (SWAP)), complex interface relationships, weak system reconfiguration capabilities, and low functional density. In order to overcome the abovementioned shortcomings and make the satellite avionics system better meet the SWAP and flexible system configuration requirements of future missions, it is necessary to improve its design technology, that is, from the current independent design of each sub-system to the open and modular design of the entire satellite. Based on the principle of unified application, deployment and operation of hardware resources, and the full use of the various functions of the software, the information sharing of the entire satellite, simple system configuration, and overall performance optimization are realized.

This chapter focuses on the application requirements of the new generation of intelligent avionics system for future communication satellites, and adopts an open architecture of "centralized management, distributed measurement and drive, and software and hardware modular design." The universal, standardized, and scalable intelligent avionics system is built based on the basic modular elements of open hardware modules, open software components, and industry standardized internal and external busses.

2. System structure

This section introduces the intelligent open system architecture, including Sections 2.1, 2.2, 2.3, and 2.4. Section 2.1 introduces the overall architecture design; the system adopts the distributed design mode and completes the intelligent management of onboard tasks through the menu hardware architecture and open interface protocol. Section 2.2 discusses the hardware architecture of high-performance computing and introduces the onboard high-performance computing and the corresponding storage capacity from the main functions,

processing, storage, and radiation resistance. Section 2.3 describes the dynamic state reconfigurable task scheduling that improves the fault tolerance ability of the satellite network in view of the typical scenarios of the satellite integrated electronic system in the operation process. Section 2.4 discusses the design of software partition protection mechanism related to the next-generation avionics system and analyzes the requirements, design, and functions of partition protection, aiming to improve the robustness of the software system.

2.1 Overall architecture design

The architecture of the newly proposed next generation of intelligent communication satellite avionics system is shown in **Figure 1**. The avionics system architecture (ASA) is designed as a data bus (DB)-based real-time distributed computer system. ASA consists of one Satellite Management Unit (SMU), one Platform Integrated Services Unit (PFISU), one Payload Integrated Services Unit (PLISU), and a set of DB and auxiliary software. The SMU is the core of the avionics system. ASA controls the PFISU and PLISU by DB and connects with Telemetry and Telecommand Unit (TTU) to receive commands and send the telemetry data. PFISU and PLISU are the execution parts of the avionics system. PFISU and PLISU are used to command driver, signal sample, power distribution, heater control, pyrotechnic management, and interface management. To improve the reliability of avionics system, the SMU, PFISU, and PLISU will have built in redundancies. This avionics system supports the functions of satellite on-orbit dynamic registration, spatial data interaction, and routing and can solve the problem of user-oriented and task-oriented opening of satellite system.

2.2 High-performance computing hardware architecture

As the amount of data generated by satellite electronic equipment continues to increase, a large amount of data processing requirements place higher requirements on satellite information processing capabilities. The avionics system is the

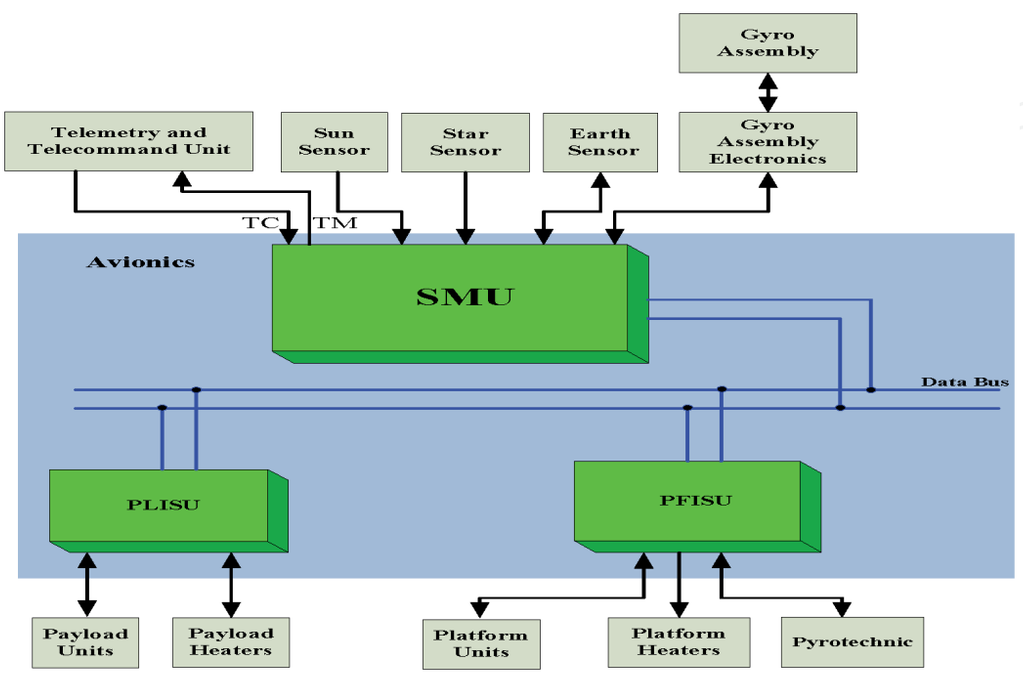

Figure 1.
Avionics system architecture diagram for satellite communication system.

information core of the whole satellite, especially for the requirements of intelligent satellite systems. Research on the realization of high-performance computing of avionics systems is an inevitable requirement [6, 7]. In order to improve the computing capacity of the avionics system, a high-performance onboard processor is utilized. The following introduces onboard high-performance computer from four aspects: main functions, processing, storage, and anti-irradiation.

i. Main functions:

- Uses redundant onboard computer supporting on-orbit reconstruction and reconfiguration for highly reliable avionics system.
- Supports Consultative Committee for Space Data Systems (CCSDS) telemetry and telecommand with optional radio-frequency channels. This feature allows the proposed intelligent avionics system design to be open interface using widely acceptable industry standards.
- Uses interface with external unit. Provide a brief description why is this important function.
- Provides secondary power distribution and discrete instructions to external units. Provide a brief description why is this important function.

ii. High-performance processors:

- 215 Dhrystone Million Instructions executed Per Second (DMIPS) and floating-point arithmetic unit
- L1 instruction cache and L1 data cache with Error Correcting Code (ECC) function
- Internal Random Access Memory (RAM), FLASH, and Electrically Erasable Programmable Read-Only Memory (EEPROM), with ECC function
- Contains basic software: BIOS and startup software

iii. High-performance memory:

- Volatile: 192 MB SDRM CPU, with error detection function
- Volatile: 64 MB SDRM IO, with ECC function
- Nonvolatile: 4GB FLASH, with ECC function

iv. Radiation resistance:

- Spaceborne components will not be locked due to space radiation.
- Dual-core processor lockstep technology is used for error detection.
- All memories have ECC function (RS code or EDAC).

Among them, the "lockstep" technology is a fault-tolerant computing technology. This technology uses the same, redundant hardware components and processes the same instructions at the same time. The core idea is to keep multiple central processing units (CPUs) and memories executing the same instructions accurately and synchronously by running synchronous comparisons in operation to improve the fault-tolerant computing capability of the avionics system.

2.3 Dynamically reconfigurable task scheduling

The two typical scenarios usually encountered by satellite avionics systems during operation are (a) a node fails or requires functional reorganization so that some tasks on this node need to be migrated to other nodes through the network and (b) the resource occupancy rate of a node is too high so that some tasks on this node will be migrated to other relatively idle nodes for execution. The avionics system is designed with networked real-time multitasking distributed system software, which can also implement dynamic reconfiguration of functions and task scheduling. The embedded system software running on each node in the network supports not only the local real-time multitasking scheduling but also the network operation capability. The avionics system supports function modification and function migration between nodes, which realizes the transformation to software-defined satellite functions, reduces the differences in hardware products, improves the fault tolerance of the intra-satellite network, and also meets the growing needs of intra-satellite networking [8].

The avionics system networked real-time multitasking distributed system software is shown in **Figure 2** and has the following characteristics:

- Application tasks are directly oriented to users. In order to complete a top-level function in a specific domain, the tasks are decomposed into functions of appropriate granularity. The software functions that multiple tasks will use are called domain public services. It is called public services in multiple fields, has a clear interface definition, can complete certain functions relatively independently, and adds service registration, management, control, and governance to provide strong support for space application tasks. The user's service

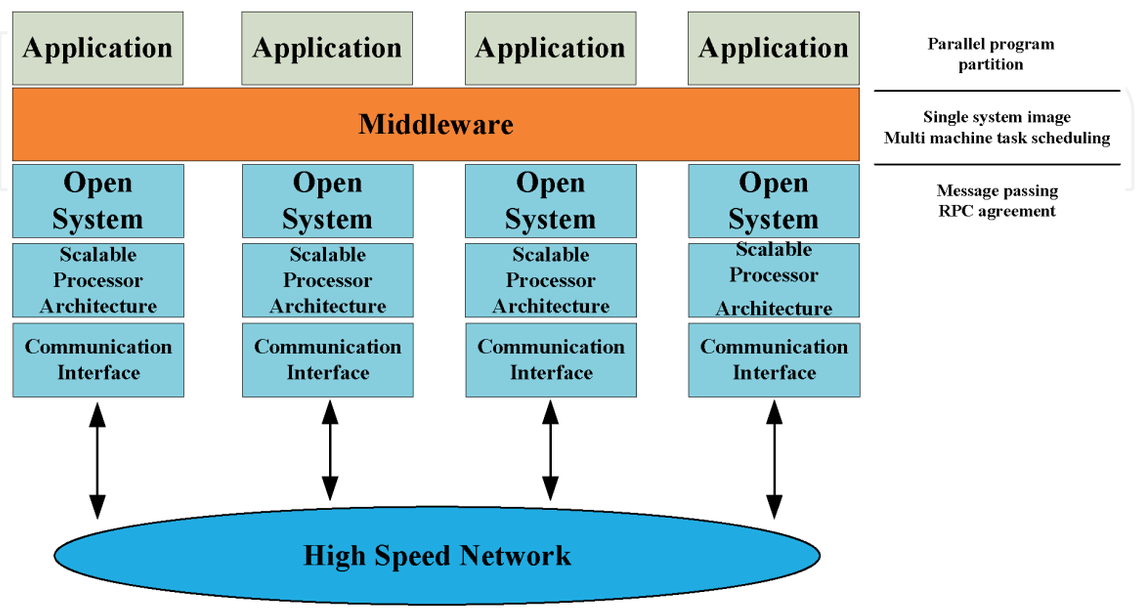

Figure 2.
Networked real-time multitasking distributed system software.

composition capability, that is, when the business needs of the day change and the service call is adjusted, can support the user to quickly combine services and form a new business process [9].

- The middleware layer serves as a transition layer between the above and the next [10]. By combining the characteristics of the tasks of each layer, the corresponding theme is designed, and the publish-subscribe technology is used to provide the entire application layer with access to various resources in the basic resource layer. The software layer implements operating system and communication protocol level shielding through the packaging of thread tasks, synchronization resources, memory access, IO operations, Ethernet, shared memory, and fiber-optic communications, providing access to the hardware device layer. By virtualizing the calculation, storage, and network resources of the basic resources, the basic resources as a service are realized, and the availability and scalability of the hardware resources are guaranteed. At the same time, the software is dynamically deployed for the hardware of the basic resources, including the automation of basic software and application software installation settings, maintenance and upgrades, etc., and provides the system with general basic services such as system reconstruction, software fault tolerance, data management, subscription release, etc.

2.4 Software partition protection mechanism design

i. Partition protection requirements

Avionics systems can implement multiple functions to share resources. The functional entities (which can be software modules, hardware modules) that share resources are called partitions. The partitions of the original avionics system shared resources, but sharing would bring potential problems described below:

- Multi-partition shared memory and IO: if one partition accidentally or maliciously rewrites the memory and IO of other partitions, it will cause the rewritten partition to fail.

- Multi-partition shared processor time: if a partition maliciously takes up processor time due to a failure, the related partition will crash.

- Multiple partitions share the same communication link: if a partition occupies channels too much, it will affect the bandwidth and real-time performance of other partitions.

Therefore, the design of system software architecture of the avionics system should meet the reliable partition protection to avoid the above problems.

ii. Partition protection design

The partition protection design of onboard system software in avionics systems includes the following three aspects:

- Space protection. For processors with Memory Management Unit (MMU) support, such as X86 processors, it is stipulated that the partition itself cannot directly access physical memory, and only virtual memory can be accessed through the MMU memory mapping table configured for each

partition by system software. For processors without MMU support, taking the Scalable Processor Architecture (SPARC) processor as an example, during partition initialization, the system software protects the SRAM by setting the privileged register to achieve partition space isolation.

- Inter-partition communication. Inter-division communication is another important content of partition space protection. Inappropriate inter-partition communication mechanisms can cause mutual interaction between partitions. ARINC-653 specification provides a communication mechanism that does not affect each other. Inter-partition communication within a single processor includes a memory buffer mechanism and a blackboard mechanism. The buffer mechanism mainly provides data communication between partitions, and the blackboard mechanism mainly provides sampling services between partitions. Taking the buffer mechanism as an example, the protection mechanism for inter-partition communication is the following: (1) The core system software allocates an inter-partition communication buffer (size, permission, and connection relationship between buffers) for each partition according to the blueprint information. Only the own buffer can be accessed. (2) The core system software manages the resources for communication between partitions in the privileged state and copies the information in the source buffer to the destination buffer according to the blueprint information.

- Partition time protection. Its strategies include the following: (1) The basic unit of scheduling is partitioning, and partition scheduling has no priority. The Main Time Frame (MTF) is used to statically define the scheduling order of each partition and the proportion window size. MTF is one of the blueprint contents. The core system software is configured according to the MTF, and multiple partitions are scheduled in a cyclic manner. (2) The resources (such as timers, stacks, and memory) and blueprints required for partition scheduling are in a privileged state, and the user partition cannot overwrite the partition scheduling resources.

iii. Advantages of partition protection

- Blueprint only registers the interfaces on the modules and then mounts them on the app as a whole. The purpose of blueprint itself is to organize the parallel coexistence of multiple modules and avoid registering modules directly on the app. In fact, it is more convenient for development and code maintenance, because ultimately all interfaces on views are still directly mounted on the app, which corresponds to the entire application; there is no obvious difference [11].

- Blueprint is not a pluggable application, because it is not a real application, but a set of operations that can be registered in the application and can be registered multiple times.

- At the same time, we cannot use multiple objects to manage and register, because this will cause each object to have its own configuration, which is not easy to manage.

- With blueprint, the application will be managed in the flask layer, share the configuration, and change the application object on demand through

registration. The disadvantage of blueprint is that once an application is created, it can only be unregistered by destroying the entire application object.

3. Menu system composition

Based on the menu design idea, the avionics system can realize the sharing of hardware modules and resources, task migration, and system reconstruction, enhance the tolerance and processing ability of the avionics system for faults, significantly improve the development efficiency and productization degree of the integrated electronic system, and make the integrated electronic system highly reliable. The intelligent spacecraft provides the necessary technical support, mainly including the contents that will be presented in the subsequent sections [12].

3.1 Satellite management unit

The Satellite Management Unit is an improved satellite research equipment. The design fully draws on the advantages of the previous satellite platform and has been optimized and expanded. The main completed functions are as follows:

i. Telecommand function: The SMU receives the telecommand instructions from the TTU and completes the distribution of the instructions.

ii. Telemetry function: The SMU collects its own telemetry and receive the indirect telemetry parameters collected by the bus terminal equipment through the 1553B bus. The SMU complete the framing processing according to the CCSDS standard and transmit it to the TTU through the serial port.

iii. Satellite autonomous management function: Using application software running on the SMU, functions such as energy management, thermal management, bus management, payload management, and pyrotechnics management are realized.

iv. On-orbit maintenance function: For the temporary adjustment of control parameters during the execution of on-orbit tasks, the SMU is able to modify the control parameters of the software. It also supports onboard software maintenance function of onboard software, which can realize the update and recovery of software modules.

v. Important data saving function: The SMU can periodically save and maintain important data of satellite sub-systems. When the internal configuration of the relevant sub-system changes or the corresponding module restarts, the SMU can send the important data stored internally to the corresponding module. Besides, when the SMU is reset or switched off, the SMU can restore the current working state through the important data.

vi. Fault Detection Isolation and Recovery (FDIR) function: The SMU provides the operating platform for satellite FDIR [13]. When the SMU is healthy, the SMU monitor key information such as the entire satellite's energy and thermal control. The SMU detects various failure conditions in real time and performs troubleshooting through the direct remote command interface or through the 1553B bus. The FDIR of the control sub-system is completed by the attitude control computer. Note that the most advanced 1553B bus can handle data rate

up to 10 Mbps. For data rate higher than 10 Mbps, industry trend is moving toward SpaceWire data bus that can handle data rate up to 400 Mbps.

The SMU adopts the dual-machine cold backup working mode. In the case of the autonomous switching enabled, when the on-duty machine fails, the failure-tolerant module automatically completes the switch of SMU and power off the faulty machine. In the case of the autonomous switching not enabled, when the on-duty machine fails, the power failure and machine switch is conducted by ground station.

3.2 Integrated service unit

The Integrated Service Unit (ISU) adopts modular hardware design [14, 15]. Each functional module is connected to the "bus interface management module" through an internal bus, which uses widely acceptable data bus such as 1553B bus. The ISU is mainly composed of a bus interface management module, a matrix instruction and matrix telemetry module, and an analog quantity acquisition and a discrete instruction output module.

The menu module is mainly composed of a bus interface management module, a matrix acquisition and command module, and an information acquisition and command module [16]. The module menu is shown in **Table 1** below.

The functions of each module are also modularized. The capability of each module is shown in **Table 2**.

After the module design is completed, the number of modules is determined according to the task requirements. Then, complete the assembly according to the standard interface. The number of modules menu is shown in **Table 3**.

3.3 Data bus network

The data bus network is the information transmission hub of the avionics system. Through the data bus network, distributed data acquisition, and instruction output, centralized operation and control are implemented, thereby improving the efficiency of system processing. The avionics system first-level bus is 1553B bus [17]. Note that for data rate higher than 10 Mbps, SpaceWire data bus is recommended.

Data exchange between SMU and ISU and other equipment realized through 1553B bus. The master–slave response mode of the 1553B bus is used to transmit platform command data and telemetry acquisition data. In the data exchange process of the first-level bus, the SMU always acts as the controller of the 1553B bus

Functions	Module menu
Telemetry/telecommand	Acquisition and Command-A module (AA)
	Acquisition and Command -B module (AB)
	Matrix acquisition and command module (AC)
Temperature acquisition	High-voltage heater control module (HH)
	Low-voltage heater control module (LH)
	Heater and distribution module (HD)
Pyrotechnic management	Pyrotechnic management module (CA)
Bus data transmission	1553B bus

Table 1.
Function module list.

Module	Module capabilities
AA	Analog/temperature measurement acquisition channel
	Small current discrete command output
	High current command drive circuit
AB	Analog/temperature measurement acquisition channel
	Small current discrete instructions
	Bi-level quantity acquisition
AC	Matrix instructions
	Switch status acquisition
HH	High-pressure heater power distribution
	Temperature measurement collection
LH	Low-voltage heater power distribution
	Temperature measurement
HD	High-voltage instrumentation and power distribution
	High-pressure heater power distribution
	Temperature measurement
CA	Pyrotechnic management

Table 2.
Modularized functions.

Demand	Quantification	AA	AB	AC	CA
Pyrotechnics management	80				2
High current instruction	30	1			
Low current instruction	100		1		
Analog acquisition	220	2	2		
Matrix acquisition	550			2	
Total (take the maximum)		2	2	2	2

Table 3.
Menu-style design.

and initiates the communication. ISU and other equipment, as the RT end, receive instructions and send collected telemetry data to the SMU.

4. Failure detection isolation and recovery (FDIR) design

In order to achieve autonomous and healthy operation of the satellite, the intelligent satellite system uses the FDIR software to monitor the status of the satellite in real time and diagnose and predict its working status and performance trends [18]. When a failure occurs, the FDIR software can locate the failure in time and determine which components are not working normally or the performance is degraded.

4.1 FDIR design goals and principles

Design goals:

i. Satellites can survive if any failure occurs.

ii. When a failure occurs, try to extend the mission time of the satellite and reduce the loss of mission interruption.

iii. The life of the satellite should be guaranteed: optimize fuel consumption and minimize system configuration and component losses.

The above three principles apply to the launch phase, the transfer orbit phase, and the on-orbit phase.

FDIR is an important component of the onboard software, which can perform on-orbit processing of failures, thereby reducing the impact of failures. However, not all on-orbit failures can be detected and processed. FDIR design should follow the following principles:

i. FDIR processing follows the single failure principle, that is, only one failure is processed at a time.

ii. Failures are divided into 0 to 4 levels according to their impact on satellites.

iii. The higher the failure level, the higher the processing priority. During a failure processing, if a higher-level failure occurs, the higher-level failure is processed first.

iv. The failures of the same level are processed in the order of occurrence.

v. All FDIR processing requires failure recovery instructions and failure processing records.

4.2 Failure levels

According to the impact of the failure on the satellite operation, the failure is categorized as follows:

i. System-level failure: failures that damage the functions and performance of the satellite system.

ii. Sub-system-level failure: the functions of the sub-system cannot be or are partially completed, or the main performance indicators and parameter values of the sub-system exceed the range required by the sub-system design. But it does not affect the main functions and performance of the system.

iii. Equipment-level failure: equipment functions cannot complete the main performance indicators, or parameter values exceed the range of equipment design requirements. But it does not affect the main functions and performance of the system.

iv. Module-level failure: a failure in which the module function cannot be completed, or the main performance indicators and parameter values exceed the range required by the component design. But it does not affect the main function and performance of the equipment.

According to the possible impact on components, functions, and systems, FDIR is designed for five failure levels from levels 0 to 4, according to the different sub-systems that each failure belongs to, including measurement and control FDIR,

avionics FDIR, and power supply and distribution FDIR as illustrated in **Figure 3**. The larger the number, the higher the fault level, and vice versa.

- Level 0 failure: a level 0 failure refers to a failure that occurs inside an equipment and can be recovered autonomously by the hot backup method inside the equipment without affecting other components of the system.

- Level 1 failure: a level 1 failure refers to the failure of a single equipment or module of each sub-system. After a level 1 failure occurs, the system will perform autonomous failure isolation and recovery according to the FDIR policy. If the failure isolation and recovery is successful, it has no impact on system tasks. The detection, isolation, and recovery of failures are implemented by application software.

- Level 2 failure: a level 2 failure refers to the functional level abnormality of the satellite sub-system. Under such failures, the system performance cannot meet the design requirements. For level 2 failures, the recovery strategies need to be implemented and related components need to be enabled or restarted. Level 2 failures can cause system performance degradation or temporary interruption of system tasks. Its failure detection, isolation, and recovery are performed by application software.

- Level 3 failure: a level 3 failure refers to the failure of the CPU hardware, which is detected by the hardware. After the failure occurs, it is switched to the backup CPU according to the failure handling strategy.

- Level 4 failure: a level 4 failure refers to the failure of the satellite to maintain the pointing to the ground in the on-orbit phase and requires sun capture processing.

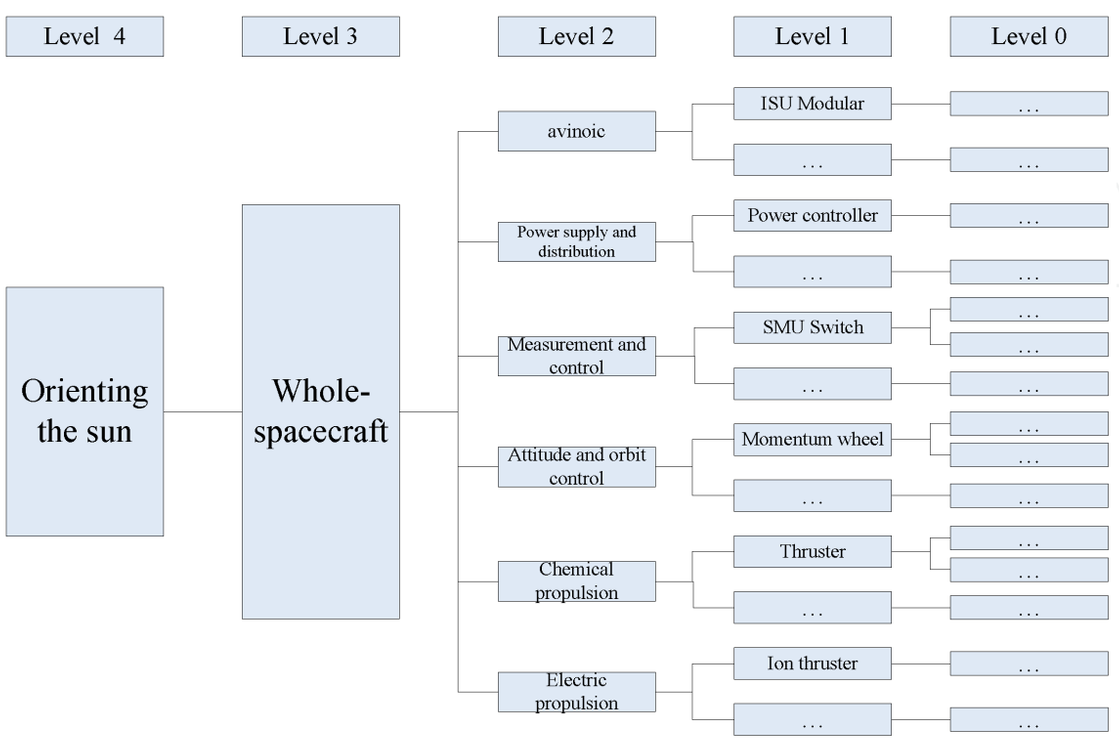

Figure 3.
Schematic diagram of satellites in orbit during their lifetime.

4.3 FDIR scheme

The software autonomously isolates the failure and rebuilds the system at the appropriate time according to the following FDIR scheme:

- High-level failure detection such as level 4 failure has priority over low-level failures such as level 3. When two or more failures are detected at the same time, the recovery sequence of high-level failures is performed preferentially. Once a high-level failure occurs, all detection of the same-level and low-level failures are suspended before the recovery sequence is completed.

- Only one failure recovery sequence is performed on the satellite at the same time, that is, all FDIR failure recovery is shielded during the execution of any failure recovery strategy. The sufficiency and necessity of the failure recovery sequence should be effectively verified by ground testing to minimize the interpretation during the sequence execution.

- After the execution of the failure recovery sequence is completed, the failure detection of the flight can be continued, but the failure recovery enable status should be set to disable. At the same time, the detection of other similar and low-level failures should be enabled. After confirming the working status of the products on the ground, reset the backup status and enable the FDIR recovery function.

- FDIR only detects the status of the on-duty module. When the FDIR enable flag is "disabled," the status of the module is not detected. When the FDIR enable flag is "enabled," the status of the on-duty module is detected. If the failure detection condition is met on the on-duty module and the FDIR recovery enable flag is "disabled," the health status of the module is set to unhealthy. If the failure detection condition is met on the on-duty module and the FDIR recovery enable flag is "enabled," it is determined whether the status of the on-duty module is the same as the backup module. If they are the same, do not perform the recovery operation and set the on-duty module as unhealthy. If they are different, perform the recovery operation and set the backup module to the on-duty status.

- For autonomous maintenance on the satellite, the "health status" of each module can only be changed from "healthy" to "unhealthy."

- For dual-machine hot standby equipment or modules, only the health status of non-duty module is detected, and no recovery is performed.

- Use its own fault-tolerant RAM and lower computer to save important data in time for state recovery after failure.

4.4 FDIR processing requirements for satellites in orbit

The FDIR requirements for each phase of the satellite are as follows:

i. Launch phase: allows failure detection and recovery of level 0 and level 1 failures.

ii. Transfer orbit phase: allows failure detection and recovery of level 0 to level 3 failures.

iii. On-orbit phase: allows failure detection and recovery of level 0 to level 4 failures.

4.5 FDIR processing

The processing flow of FDIR mainly includes four parts:

i. Judgment of processing conditions: First, determine the scope of failure detection according to the requirements of the satellite in orbit and ground control. Then, according to the validity of the telemetry data and the situation of the modules on duty, determine the FDIR project that can be used for failure detection.

ii. Fault detection: Determine whether a failure occurs based on the recognition characteristics.

iii. Comprehensive information processing: After a failure occurs, it is determined that whether the current situation of the satellite meets the recovery conditions. At the same time, in the case of multiple failures, priority judgment is required. Finally, determine the failures that can be recovered and the order of recovery.

iv. Failure recovery: According to the engineering and testing experience, perform corresponding recovery operations.

5. Impacts on next-generation avionics system

The intelligent avionics system adopts a system engineering method using modular and open design to uniformly design the information processing, control and management processes, hardware, and software, which is to realize the optimization of information and resource sharing. Based on the onboard computer and high-speed bus such as SpaceWire data bus, a set of information fusion systems and mechanisms is established. The system is a menu-style, modular, and extensible open service platform, which achieves a high degree of integration of various onboard software and hardware resources and can meet the requirements for different tasks.

The intelligent avionics system adopts the design concept of a modular menu system architecture to meet the needs of real-time, reconfigurable, autonomous planning, and intelligence of the system. With the SMU as the 1553B and SpaceWire bus controller for data rate less than 10 Mbps/for data rate more than 10 Mbps, respectively, and the ISU as the remote terminal, a distributed, master-slave, and menu-based satellite networks are constructed.

Satellites are designed with a network layout, which can design different menu network nodes on the bus network. After the payload capacity is strengthened, the network node can increase the corresponding payload processing unit. The SMU is used as the main processing computer to perform the main control of satellite services to form a master-slave network structure. The high-performance onboard processor enables the intelligent avionics systems with high-performance computing capabilities, which not only meets the data processing requirements but also lays the foundation for satellite intelligence. The intelligent avionics system adopts partition protection measures. Through the design of space protection, time protection, and partition communication, it provides reliable functional entities (such as

software modules or hardware modules) that share resources. Partition protection avoids the impact of other partitions under abnormal conditions such as single partition failure or malicious access. At the same time, the intelligent avionics system is equipped with real-time multitask distributed system software, which can realize the dynamic reconstruction of functions and tasks. For example, when a node fails or needs a functional reorganization, some tasks on that node will be migrated to other nodes. Or, the resource occupation rate of a node is too high, and some tasks on this node will be migrated to other relatively idle nodes for execution. This design can improve the failure tolerance of the intra-satellite network and achieve efficient resource allocation and scheduling. All information is collected into the SMU for comprehensive analysis and processing through the 1553B/SpaceWire bus network. For example, in the process of autonomous energy management, it is found that the battery discharge depth reaches 80%. If the control sub-system is still in the mode of pointing to the ground, it will seriously affect the safety of the satellite. At this time, the instructions should be sent in time to orient the satellite to the sun to ensure the safety of the satellite. The SMU can be fully applied to the satellite's autonomous information fusion processing, ensuring that the satellite can still guarantee normal communication services in the event of a major failure, and energy security in emergency situations.

6. Conclusion

The intelligent avionics system design is the key technology for future advanced satellites. The system design has adopted modular and open system architecture approach using an efficient computing hardware system to maintain multiple central processing units and memory executing instructions accurately and synchronously with high cost-performance and cost-efficiency ratio. This approach improves the failure tolerance of the next-generation avionics systems. The function modification capabilities and function migration between modules realize the transition to software-defined satellite. Furthermore, this approach also reduces the differences in hardware products and improves the failure tolerance of the satellite's internal network, which meets the ever-increasing networking requirements of the satellite.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China (No. 61972398).

IntechOpen

Author details

Changqing Wu, Xiaodong Han* and Yakun Wang
Institute of Telecommunication Satellite, China Academy of Space Technology, Beijing, China

*Address all correspondence to: willingdong@163.com

IntechOpen

References

[1] Sherwood R, Chien S, Tran D, et al. The EO-1 autonomous sciencecraft. In: Small Satellite Conference. Logan, UT: AIAA/USU; 2007

[2] Rabideau G, Tran D, Chien S, et al. Mission operations of earth observing-1 with onboard autonomy. In: IEEE International Conference on Space Mission Challenges for Information Technology. Pasadena, CA: IEEE; 2006

[3] Sherwood R, Chien S, Tran D, et al. Enhancing science and automating operations using onboard autonomy. In: International Conference on Space Operations (SpaceOps 2006). Rome, Italy: AIAA; 2006

[4] Sherwood R, Chien S, Tran D. Autonomous science agents and sensor webs: EO-1 and beyond. In: IEEE Aerospace Conference (IAC 2006). Big Sky, MT: IEEE; 2006

[5] Chien S, Sherwood R, Tran D, et al. Using autonomy flight software to improve science return on earth observing one. Journal of Aerospace Computing, Information, and Communication. 2005;**2**(4):196-216

[6] Chien S, Sherwood R, Tran D, et al. Lessons learned from autonomous sciencecraft experiment. In: Autonomous Agents and Multi-Agent Systems Conference (AAMAS 2005). Utrecht, Netherlands: AAMAS; 2005

[7] Rabideau G, Chien S, Sherwood R, et al. Mission operations with autonomy: A preliminary report for earth observing-1. In: International Workshop on Planning and Scheduling for Space (IWPSS 2004). Darmstadt, Germany: IWPSS; 2004

[8] Marie J. Spacebus 4000 avionics: Key features and first flight return. In: 24th AIAA International Communications Satellite System Conference (ICSSC). 2006

[9] Heymans P, Boucher Q, Classen A. A code tagging approach to software product line development. An application to satellite communication libraries. International Journal on Software Tools for Technology Transfer. 2012;**14**(5):553-566

[10] Boreli R, Ge Y, Iyer T, et al. Intelligent middleware for high speed maritime mesh networks with satellite communications. In: 9th International Conference on Intelligent Transport Systems Telecommunications (ITST). Harbin, China: IEEE; 2009

[11] Zhu H. Design of on-board software architecture based on design pattern. Computer Application. 2008;**25**(12):180-181

[12] Yu Z, Yang-Ming Z, Zheng-Liang H, et al. Fault-tolerant design of memory module for pica-satellite on-board computer. Journal of Astronautics. 2008;**29**(6):2057-2061

[13] Guiotto A, Martelli A, Paccagnini C. SMART-FDIR: Use of artificial intelligence in the implementation of a satellite FDIR. In: DASIA 2003. Prague, Czech Republic: DASIA; 2003

[14] Herpel HJ, Schuettauf A, Willich G. Open modular computing platforms in space — Learning from other industrial domains. In: IEEE Aerospace Conference. MT, USA: IEEE; 2016

[15] Mark H, Helene DM, Alexandre C. Requirements baseline for integrated modular avionics for space separation kernel qualification. In: Dasia-Data Systems in Aerospace. Barcelona, Spain: DASIA; 2015

[16] Panpan Z, Tingyuan G, Jianjun G, et al. Plug-and-play on-board computer system design based on BM3803 processor. Journal of Aerospace Engineering. 2013;**22**(06):92-96

[17] Kan W, Jingfei J, Mianjiang H, et al. Intelligent fault-tolerant 1553B bus system based on adaptive learning. In: IEEE International Conference on Communication Software and Networks. Xi an, China: IEEE; 2011. pp. 378-381

[18] SalarKaleji F, Dayyani A. A survey on fault detection, isolation and recovery (FDIR) module in satellite onboard software. In: International Conference on Recent Advances in Space Technologies (RAST). Istanbul, Turkey: IEEE; 2013