Hindawi
International Journal of Aerospace Engineering
Volume 2018, Article ID 7019838, 22 pages
https://doi.org/10.1155/2018/7019838

Research Article

Formal Verification Method for Configuration of Integrated Modular Avionics System Using MARTE

Lisong Wang, Miaofang Chen, and Jun Hu

College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, 29 Jiangjun Dadao, Nanjing 210016, China

Correspondence should be addressed to Lisong Wang; wangls@nuaa.edu.cn

Received 25 August 2017; Revised 18 January 2018; Accepted 24 January 2018; Published 22 April 2018

Academic Editor: Linda L. Vahala

The configuration information of Integrated Modular Avionics (IMA) system includes almost all details of whole system architecture, which is used to configure the hardware interfaces, operating system, and interactions among applications to make an IMA system work correctly and reliably. It is very important to ensure the correctness and integrity of the configuration in the IMA system design phase. In this paper, we focus on modelling and verification of configuration information of IMA/ARINC653 system based on MARTE (Modelling and Analysis for Real-time and Embedded Systems). Firstly, we define semantic mapping from key concepts of configuration (such as modules, partitions, memory, process, and communications) to components of MARTE element and propose a method for model transformation between XML-formatted configuration information and MARTE models. Then we present a formal verification framework for ARINC653 system configuration based on theorem proof techniques, including construction of corresponding REAL theorems according to the semantics of those key components of configuration information and formal verification of theorems for the properties of IMA, such as time constraints, spatial isolation, and health monitoring. After that, a special issue of schedulability analysis of ARINC653 system is studied. We design a hierarchical scheduling strategy with consideration of characters of the ARINC653 system, and a scheduling analyzer MAST-2 is used to implement hierarchical schedule analysis. Lastly, we design a prototype tool, called Configuration Checker for ARINC653 (CC653), and two case studies show that the methods proposed in this paper are feasible and efficient.

1. Introduction

Integrated Modular Avionics (IMA) [1, 2] represent real-time computer network airborne systems, which consist of a number of computing modules capable of supporting numerous applications of differing criticality levels. The IMA concept proposes an integrated architecture with application software portable across an assembly of common hardware modules. How to design and analyze an IMA system effectively is becoming one of the most important challenges in the domain of complex embedded system engineering in recent years.

ARINC653 specification [3, 4] is one kind of standard for IMA system implementation, and the configuration information of IMA system includes almost all details of the whole system architecture to make IMA system work correctly and reliably. The configuration information in IMA system is usually according to ARINC653 standards. It includes all the data information of abstract levels in system architecture, which can be used to configure the parameters of the IMA system such as hardware resources, the operating system interface, and the environment of application. Even more, when the hardware and software on computing platform change, we can only adjust system configuration information so that existing applications can effectively run on the new platform with consistency. Therefore, how to ensure the correctness of the system reconfiguration information is one of the key problems in the IMA/ARINC653 system design and maintenance. Since the schedule information of IMA is included in the system configuration information, the schedulability analysis needs to be checked within the configuration context.

Model-driven engineering (MDE) [5–7] is the mainstream methodology in the domain of system engineering and software engineering for the past ten years. Its basic idea is concentrating on system model design, model transformation, and model analysis and verification, to improve the capability and efficiency of development and maintenance of a complex engineering system. In the DO-178C, which is the latest version of airworthiness standards for aviation software, requirements of both a model-based development and formal methods have been officially proposed [8–10]. MARTE (Modelling and Analysis of Real-Time and Embedded Systems) [11, 12] is a real-time system modelling and analysis language published by Object Management Group (OMG), supporting modelling and analysis of functional modelling for design of complex real-time systems with consideration of time constraints, resource allocation, other nonfunctional properties, and so on.

In this paper, we focus on modelling and formal verification of configuration information of IMA/ARINC653 system based on MARTE. The remainder of the paper is organized as follows. In Section 2, we present an introduction of the typical IMA/ARINC653 system architecture, ARINC653 configuration information, and MARTE modelling elements. Section 3 defines semantic mapping from key concepts of configuration (such as modules, partitions, memory, process, and communications) to components of MARTE element and proposes a method for model transformation between XML-formatted configuration information and MARTE models. In the next section, a formal verification framework of ARINC653 configuration based on MARTE models is given, and Section 5 discusses a schedulable and validation approach of ARINC653 partition system. Also, we design a hierarchical scheduling strategy with consideration of characters of the ARINC653 system, and a scheduling analyzer MAST-2 [13, 14] is called to implement a hierarchical schedule analysis. In Section 6, we design a prototype tool called CC653 (Configuration Checker for ARINC653) for modelling and verification of the ARINC653 configuration file [15]. Finally, we give two examples to illustrate the effectiveness of our method.

2. ARINC653 and MARTE

2.1. ARINC653 System. The IMA architecture of an ARINC653 system is shown in Figure 1, which consists of several layers from top to bottom, that is, application software, ARINC653 interface, real-time operating system, hardware interface, and devices. Except those functions implemented by application software, an ARINC653 system module mainly provides mechanisms for execution of a safety-critical software, including partition management, process management, memory management, timing management, interpartition communication, intrapartition communication, and health monitoring.

The right side block of Figure 1 indicates that the configuration information is a very important part of ARINC653 system since it contains many system parameters with consideration of almost all layers from abstractions to real hardware devices. Specifically, ARINC653 configuration information can be divided into two types, module level and partition level, describing the interpartition and intrapartition resource allocation, respectively. Reference [16] introduces a table of configuration information in detail, including 13 module-level entries and 16 partition-level entries. In Table 1, we give an example of general configuration parameters of the module-level in which exactly one entry should be contained for each IMA module configuration. Data form of configuration information can be in different kinds formats, such as XML, Excel, and CSV. In our works, we consider configuration information with the XML format which is recommended in the standard ARINC653 specification [3, 4].

One of the main purposes of usage of configuration information is to improve portability and reusability of the IMA system; that is to say, people wish that an IMA system can be adapted quickly by reconfiguring existing configuration information to satisfy new requirements comparing with traditional federal avionic system, for example, when some resource allocations are changed, or some hardware substituted, or new application modules added. However, even if the system configuration can be descripted in a structured way by XML format, large numbers of entries in a real IMA system (maybe thousands of entries) challenge the maintenance of system configurations. It is necessary to find effective ways to ensure the correctness and integrity of the IMA system reconfigured, such as allocating more memory to a new added partition when free memory in the system module is not enough, which is an implicit safety problem that may not be discovered easily in the stage of reconfiguration, and warranting schedulability when a new partition is added or some time frame are adjusted due to changes in requirements. In addition to the size of system configuration mentioned, many factors also need to be considered since changes in the different types of configuration data may influence each other. It is not feasible to carry out overall validations of configuration information by manual inspection, especially in the case of reconfiguration. Therefore, we need a systematic framework to check IMA configuration information based on some standard models and processes.

2.2. MARTE. MARTE [11, 12] (Modelling and Analysis of Real-Time and Embedded Systems) is a domain-specific modelling language for specifying and analyzing real-time and embedded software applications and systems. MARTE is also becoming one of the industry standards in embedded system domain. In fact, it is one kind of complement standard of UML by refining standard UML concepts with more exact semantics interpretation for real-time computing. As shown in Figure 2, those elements of MARTE specification are structured around three main aspects: MARTE foundations, MARTE design model, and MARTE analysis model.

In brief, the foundation package in Figure 2 is used for description of basic concepts in a real-time and embedded system, such as priority, response time, execution cycle, and resource constrained. It is not only the foundation of the design and analysis model but also the core of the MARTE specification. The design model package provides modelling elements of higher abstraction layer, used for modeling the

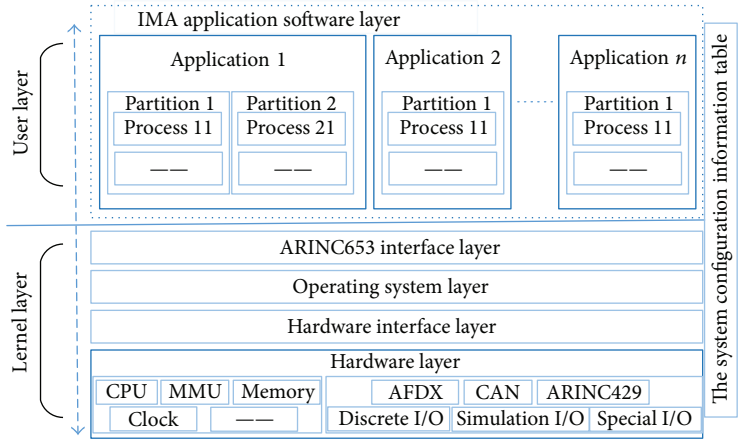


Figure 1: IMA architecture of an ARINC653 system.

Table 1: Example of general configuration information table of ARINC653 system.

Configuration entry types	Explanations
PARTITION_NB	Number of avionics application partitions
SYSTEM_PARTITION_NB	Number of system administration partitions
MAF_DURATION	Duration of the major time frame
CACHE_CONFIG	Cache configuration data
RAM_BEGIN	Starting address of memory for OS and drivers
RAM_SIZE	Memory size for the OS and drivers
CFG_AREA_BEGIN	Starting address of memory for configuration data
CFG_AREA_SIZE	Memory size for the configuration data
MAC_ADDRESS	MAC addresses for sending and receiving
MODULE_LOCATION	MAC addresses for sending and receiving

behavior of concurrent and real-time activities. The analysis model encapsulates model elements to analyze the performance and reliability of the system, specifically for energy consumption, memory usage and availability, reliability, and safety of the system. The design models and analysis models should be established by reusing the foundation concepts. These modelling and analysis mechanisms in MARTE provide a good way to analyze and check the IMA system configuration information.

3. Modelling ARINC653 Configuration with MARTE-2

In order to build MARTE models from ARINC653 configuration information, we need to define semantic mapping rules from different types of items in the ARINC653 configuration to MARTE model components, properties, or other modelling elements. Then we can design a systematic way for configuration modelling (it is also called a process of model transformation in this section). Since the details of IMA configuration are quite complicated, considering that all configuration information of the ARINC653 system in one time is not feasible, in this paper we just focus on how to build MARTE models for key concepts in the ARINC653 system (such as module, partition, process, partition communication, and health monitor) by establishing model transformation rules and corresponding examples.

3.1. Rules of Module Transformation. The "module" concept in the ARINC653 system refers to a computational node in an airborne computing network, in which several different application software are deployed and the real-time operating system conforms to ARINC653 specifications. Specifically, a module part of the configuration contains the descriptions of a processor, communication interface, memory, and other resource allocation of hardware, also including partition information of avionics application execution in the module. However, configuration information related to the inside of partitions is not included in this so-called "module-level" configuration. The typical module configuration data includes the system resource allocation for each partition and the partition-level scheduling strategy. Based on the semantic analysis, we can use the "hwProcessor" component in the MARTE specification to model those configuration information. The "hwProcessor" component specifies the runtime execution environment, including CPU scheduling, memory allocation, and BUS of communication connection. For instance, the name of a module (Module Name) can be defined by using a MARTE component attribute (descriptionHwProcessor) and module scheduling time slice (Major Frame) can be described by defining the component attribute (mainScheduler). Table 2 shows the module transition rules.

Figure 3(a) is a small example of the module configuration part of ARINC653 system in XML data. Generally, it includes two parts: partition information and scheduling information. The partition information specifies each partition information (partition name, partition ID), and the example module contains two partitions, named Part1 and Part2. The scheduling information specifies that the

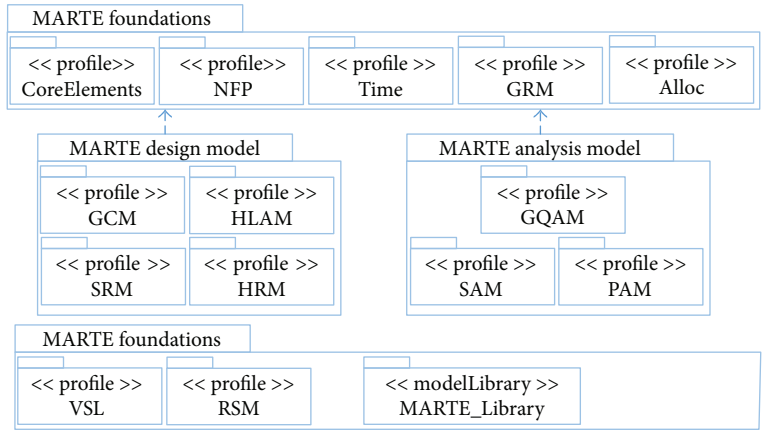


Figure 2: The basic modeling elements and structure of MARTE.

Table 2: Module transformation rules.

ARINC653 concepts		MARTE model elements
$\Phi(Module)$	$\rightarrow$	hwProcessor
$\Phi(Name)_{\text{module}}$	$\rightarrow$	descriptionHwProcessor
$\Phi(Schedules)_{\text{module}}$	$\rightarrow$	mainSchedulerHwProcessor

scheduling time of Part1 is 0.025 s and the scheduling time of Part2 is 0.020 s. Then the XML configuration is transformed to a MARTE model Figure 3(b), in which "Partition Scheduler" pointed out the partition-level scheduling information.

3.2. Rules of Partition Transformation. The "Partition" concept in the ARINC653 system is one of the key components of IMA. It requires that a set of avionics applications implementing different functions are integrated in the same module; the system management should make each software application running in respective partition in an independent way except for necessary communications among those partitions. That is to say, all partitions should be isolated one by one both in the dimension of time and space. Time isolation means that execution of different partitions should not interfere with each other in the corresponding configuration part of allocation of execution time on the module platform; for instance, scheduling cycle and execution time for each partition need to be preconfigured very carefully. Spatial isolation demands that different partitions should be assigned to different memory address, even in the stage of running time.

In MARTE specification, there are no concepts corresponding directly to the partition concept of ARINC653 system, but we can combine swSchedulingResource and ProcessingResource components together, also their attributes, to descript the configuration information of an IMA partition. For the time and space characteristics of a partition, we use the memoryPartition component in MARTE to indicate contents of the partition (such as thread and data). We also put swSchedulableResource component to be included inside of memoryPartition component in MARTE. The relationship between these two components can be defined by Allocate, and the value of attribute nature is spatial distribution.

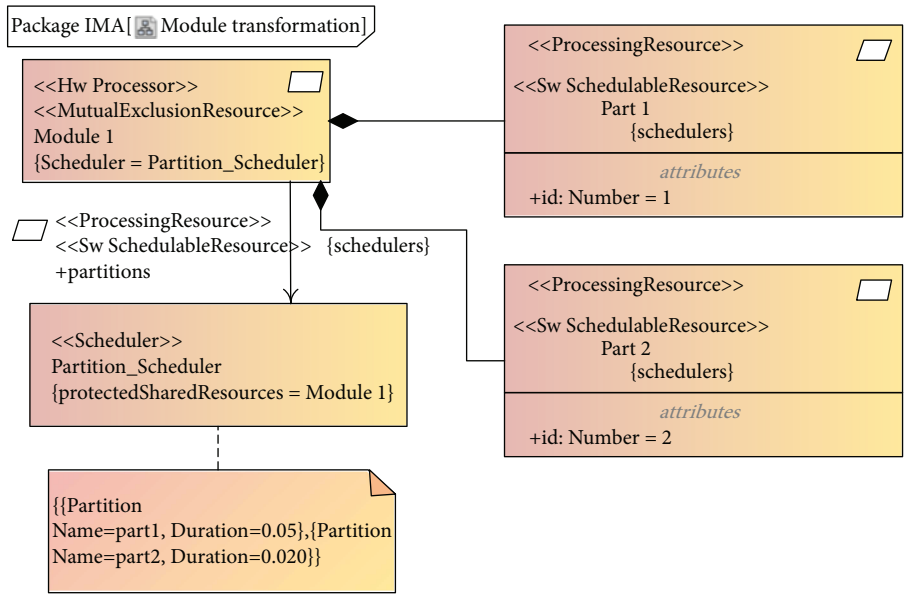
In addition, the concept of Memory in ARINC653 specification denotes demands of memory allocation on one module and also gives memory requirements for each partition in which each memory space is independent without affecting each other. Correspondingly, the hwMemory component in MARTE indicates system memory allocation information (such as size and category). The hwProcessor component can combine swSchedulingResource and hwMemory components together through setting its ownedHW attribute properly. At the same time, this binding points out the allocation of memory of swSchedulingResource component. Therefore, we can convert the ARINC653 memory directly to hwMemory component in MARTE, by defining a subcomponent hwMemory for each memoryPartition component to indicate the spatial isolation conditions in each partition on the same module. So, partition transformation rules are given in Table 3.

Figure 4(a) is an example of the partition configuration part of ARINC653 system in XML data. It contains two parts of partition information and memory information. The partition information indicates the name and ID of a partition and memory allocation information of the partition is required (such as a BSS segment size of $0 \times 10{,}000$ and TEXT segment size of $0 \times 40{,}000$). Then the transformation from XML configuration to a MARTE model is shown in Figure 4(b), in which "ownHW" points out the memory information required.

3.3. Rules of Process Transformation. "Process" in the ARINC653 system is a dynamic execution concept of application just like in a general system. It contains executable codes, application data, and resource of stacks. But ARINC653 defines that a process should be assigned in only one partition and one partition can contain multiple processes to complete the corresponding application functions. Inside of an isolated partition, there is another world, in which we can define process-level scheduling strategy (such as preemptive scheduling strategy), maximum response time, memory allocation, and other information, to control the execution of processes. For the purpose of modeling the

```
<Module Name="Module1">
  <Partition Name="Part1" Id="1">…</Partition>
  <Partition Name="Part2" Id="2">…</Partition>
  <Schedules>
    <Schedules Id="0" Name="Partition_Scheduler">
      <Partition Window PartitionNameRef="Part1" Duration="0.025"/>
      <Partition Window PartitionNameRef="Part2" Duration="0.020"/>
    …
    </Schedules>
  </Schedules>
</Module>
```

(a) Module configuration information of ARINC653



(b) Module configuration information of MARTE

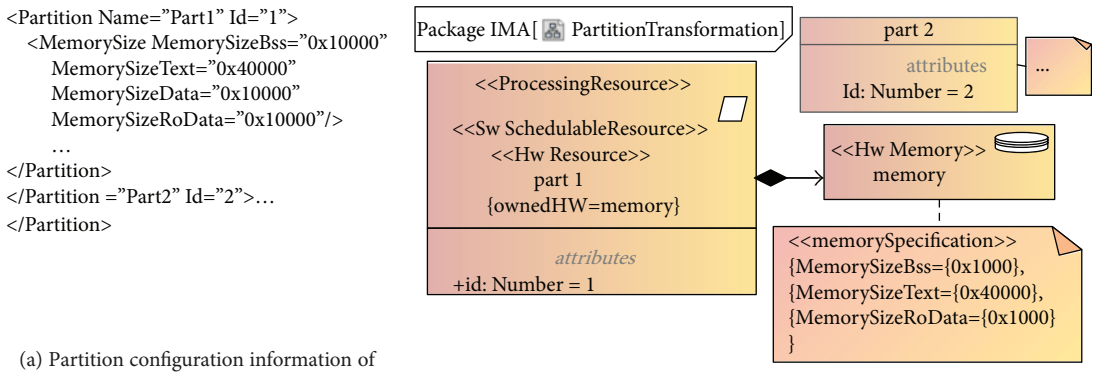
Figure 3: Module configuration transformation from ARINC653 to MARTE.

Table 3: Partition transformation rule.

ARINC653 concepts		MARTE model elements
$\Phi(Partition)_{\text{Module}}$	$\rightarrow$	$\left\langle \begin{matrix} swSchedulingResource, \\ ProcessingResource \end{matrix} \right\rangle_{hwProcessor}$ and satisfy corresponding memory allocation
$\Phi(Memory)_{\text{Module}}$	$\rightarrow$	$ownendHWhwProcessor$ The type of *ownedHW* is *hwMemory*
$\Phi(Memory)_{\text{Partition}}$	$\rightarrow$	$ownedHW_{\langle \text{swSchedulingResource}, ProcessingResource \rangle}$ The type of *ownedHW* is *hwMemory*

process concept of ARINC653, we consider swSchedulingResource component in MARTE which is defined as the most basic execution unit of the system triggered through the time period or external events, and also, the communications between threads can be described by port connection, subroutine calls, and shared data. Since they have the same semantics of executable units of a system, we can convert the process of ARINC653 to the swSchedulingResource component of MARTE. Those characteristics of an executable unit can be defined through MARTE element attributes, including execution cycle, execution deadline, execution time, and scheduling strategy. Considering that a process is included in some partition, we included the swSchedulingResource component in the ProcessingResource component by relationship. Process transformation rules are given in Table 4. Figure 5 is an example of process configuration transformation from part of the ARINC653 system in XML data to a MARTE model.

3.4. Rules of Health Monitoring Transformation. The health monitoring mechanism of ARINC653 system is one of the characteristics of IMA, which is responsible to detect, report, and control abnormal system behaviors in case some hardwires or software failure occur. In a safety-critical avionics



(a) Partition configuration information of ARINC653

(b) Partition configuration information of MARTE

Figure 4: Partition configuration transformation from ARINC653 to MARTE.

Table 4: Process transformation rule.

ARINC653 concepts		MARTE model elements
$\Phi(Process)_{\text{Partition}}$	$\rightarrow$	*swSchedulingResource*
		$\langle$ *swSchedulingResource*, *ProcessingResource* $\rangle$

application, any kind of fault occurrence should be detected in time and need to be isolated by a predefined strategy; furthermore, some recovery actions ought to be taken. According to the scope affected by faults, ARINC653 divide those faults into three levels: module, partition, and process. Also according to the cause of faults, multiple types of faults are further defined, for instance, floating-point error, stack overflow, and hardware error. Obviously, the error handling mechanisms are different (e.g., module restart, cold start, and warm start) due to different levels of faults.

In MARTE specification, there are also no concepts corresponding directly to the health monitoring of ARINC653 system, but we can add some new attributes of HM ERRORS and HM Actions for hwProcessor, memoryPartition, swSchedula-bleResource, and ProcessingResource components in MARTE, which are corresponding to the module recovery action HM Module Recovery Actions, the partition recovery action HM Multipartitions Actions, and the process recovery action HM Process Recovery Actions. Specifically, HM ERRORS defines fault types and HM Actions defines recovery actions. These attributes can be included in different components to represent different levels of ARINC653 system faults. Health monitoring transformation rules described above are given in Table 5.

3.5. Rules of Partition Communication Transformation. Intrapartition communication refers to the interactions among those processes assigned in the same partition. Intrapartition communication mechanisms generally include buffers, blackboards, semaphores, and events. Buffers and blackboards provide general interprocess communication (also synchronization), whereas semaphores and events are provided for interprocess synchronization. All intrapartition message passing methods must ensure atomic message access (for instance, partially written messages should not be read).

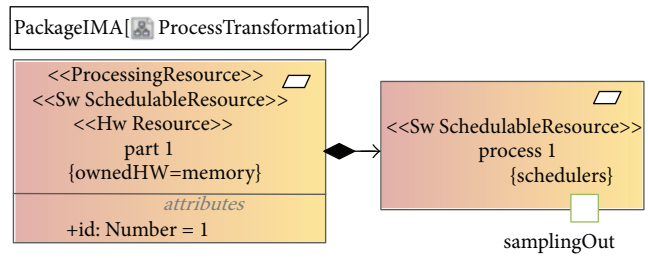
Since existing data in message queue will not be replaced by a new one of the next communication and ClientServer-Port component in MARTE exactly has the same semantic concept of buffer, we can convert a message queue (also called a buffer) of ARINC653 to a ClientServerPort component in MARTE which contains an attribute of data. Specifically, if the values of attribute kind and isAtuomic are provided and true, respectively, the ClientServerPort component will write data into the buffer; if the values of attribute kind and isAtuomic are required and true, respectively, the ClientServerPort component will read data from the buffer. Similarly, because the blackboards do not retain the last communication data, it can be converted to flowPort component of MARTE which represents the signal mechanism of ARINC653, and we can define the concurrent mechanism by adding attribute Concurrency Control Protocol to those components.

The interpartition communication is intended to facilitate communications between different ARINC653 application partitions residing on the same module or on different modules of a network. All interpartition communication is conducted via message passing. The basic mechanism linking partitions by messages is the channel. A channel defines a logical link between a message source and one or more destinations, and it also specifies two modes of message passing: sampling mode and queuing mode. In the sampling mode, a message remains in the source port until it is transmitted by the channel or it is overwritten by a new occurrence of the message; this allows the source partition to send messages at any time. In the queuing mode, each new instance of a message may carry uniquely different data and is not allowed to overwrite previous ones during the transfer. No message should be unintentionally lost in the queuing mode.

We can convert the queuing mode port to the ClientServer-Port component of MARTE which has the capability of receiving a set of data, inserting into a queue, similar to the characteristics of queuing mode in ARINC653. The sampling mode port is converted to a flowPort component. FlowPort has no data queue, and it has same communication semantic of sampling mode in ARINC653; that is, the

```
…
<Process Name=”process1” >
  <Ports><SamplingPort Name=”samplingOut”…/>
  </Port>
</Process>
…
<Partition Name=”part1” Id=”1”>
  <Process NameRef=”process1”></Process>
  …
</Partition>
```

(a) Process configuration information of ARINC653



(b) Process configuration information of MARTE

Figure 5: Process configuration transformation from ARINC653 to MARTE.

Table 5: Health monitoring transformation rules.

ARINC653 concepts		MARTE model elements
$\Phi(HM\_Errors)_{\text{Module}}$	$\rightarrow$	$HM\_Errors_{hwProcessor}$
$\Phi(HM\_Actions)_{\text{Module}}$	$\rightarrow$	$HM\_Actions_{hwProcessor}$
$\Phi(HM\_Errors)_{\text{Partition}}$	$\rightarrow$	$HM\_Errors_{\langle swSchedulingResource, ProcessingResource\rangle}$
$\Phi(HM\_Actions)_{\text{Partition}}$	$\rightarrow$	$HM\_Actions_{\langle swSchedulingResource, ProcessingResource\rangle}$
$\Phi(HM\_Errors)_{\text{Process}}$	$\rightarrow$	$HM\_Errors_{swSchedulingResource}$
$\Phi(HM\_Actions)_{\text{Process}}$	$\rightarrow$	$HM\_Actions_{swSchedulingResource}$

last received data will be discarded when the next data arrived. The partition communication transformation rules described above are given in Table 6.

Figure 6(a) is an example of part of the partition communication configuration of ARINC653 system in XML data. It contains two kinds of configuration of partition and communication. The partition information specifies each partition setup (partition name, partition ID, and so on) and communication ports of partitions. For instance, partition Pr2 has a sampling port named “pdatain”: the port refresh frequency is 0.025 seconds and the type is destination port. Partition Pr1 has a sampling port named “pdataout”: the port refresh frequency is 0.015 seconds and the type is source port. The communication information indicates each channel setup; this example contains only one channel. The source port of channel is “pdataout” port of pr1, and the destination port of channel is “pdatain” port of pr2, thus establishing a data stream communication connection. The transformation of MARTE model of the configuration information specified in Figure 6(a) is shown in Figure 6(b). Sampling refresh period points out the refresh frequency.

3.6. Modeling of Communication Virtual Link. Virtual link is a virtual logical link consisting of one to many sublinks, which mainly provides bandwidth resource isolation function. It is established through communication between different terminal nodes in AFDX network. The bandwidth resources of virtual links are guaranteed by some parameter setting, such as the length of the data frame, the gap of the bandwidth allocation, and the data transmission mode. The CommunicationMedia (CM) component in MARTE can represent the data transmission between the source terminal and the destination terminal, so the elementSize property in the CM component can represent the length of the data frame in a virtual link, using the capacity properties and elementSize attributes in the CM component to represent the bandwidth allocation gap parameters in the virtual link and using the transmMode attribute in the CM component to represent the data transmission pattern. There are three main modes of transmission: simplex, half-duplex, and full-duplex.

Figure 7 is an example of a communication virtual link MARTE model, which shows the property of data transmission between RDC and switch virtual link. The transmission data frame length is 11, the transmission capacity is 100 Mb/s, and the transmission mode is full-duplex.

3.7. AFDX Terminal Modeling. The AFDX terminal system provides the communication access port of the device to the AFDX network and is responsible for completing the communication tasks from the partition or equipment and sending and receiving the data. The terminal system sets relevant parameters according to the communication requirement and the data frame size in the communication link, including the parameters of the maximum data frame length and the minimum packet gap. The CommunicationEndPoint (CEP) component in MARTE represents the interface of communication elements passing through CM components and contains only one attribute packetSize, which matches the elementSize attribute in CM components. Therefore, the CEP component in MARTE is used to represent the AFDX terminal, and the packetSize attribute in the CEP component is used to represent the data frame size parameters of the terminal and the communication link.

As shown in Figure 8, the AFDX terminal module provides the communication access port from RDC to AFDX switch. The AFDX terminal is represented by the CEP component. The property packetSize represents the size of the transmission data frame and is consistent with the elementSize of the data frame size in the communication virtual link.

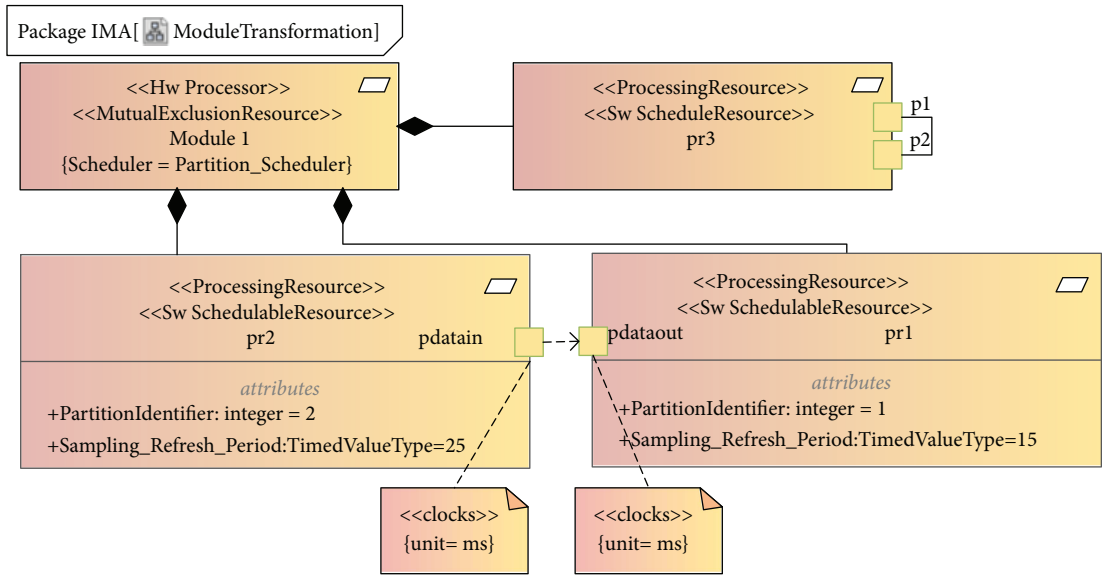
3.8. Modeling of Resource Equipment Module. The resource and device modules in IMA system include core processing module IPM, remote data concentrator RDC, AFDX switch, and a controller. Each module has its own independent processing unit to perform corresponding functions. IPM provides computing resources for applications that are residing in the partition, and AFDX switches provide communication resources to provide message forwarding function, and RDC is used as avionics input and output device for data collection

Table 6: Process communication transformation rules.

ARINC653 concepts		MARTE model elements
$\Phi(MessageQueue)_{Partition}$	$\rightarrow$	$ClientServerPort \left\langle \begin{matrix} swSchedulingResource, \\ ProcessingResource \end{matrix} \right\rangle$
$\Phi(Blackboards)_{Partition}$	$\rightarrow$	$flowPort \left\langle \begin{matrix} swSchedulingResource, \\ ProcessingResource \end{matrix} \right\rangle$
$\Phi(Sampling\_Port)_{Partition}$	$\rightarrow$	$ClientServerPort \left\langle \begin{matrix} swSchedulingResource, \\ ProcessingResource \end{matrix} \right\rangle$
$\Phi(Queuing\_Port)_{Partition}$	$\rightarrow$	$flowPort \left\langle \begin{matrix} swSchedulingResource, \\ ProcessingResource \end{matrix} \right\rangle$

```
<Partition PartitionName="pr2" PartitionIdentifier="1">
  <Sampling_Port Direction="DESTINATION" Name ="padatain" RefreshRateSeconds="0.0250"/>
</Partition>
<Partition PartitionName="pr1" PartitionIdentifier="2">
  <Sampling_Port Direction="SOURCE" Name ="padataout" RefreshRateSeconds="0.0150"/>
</Partition>
<Connection_Table>
  <Channel ChannelIdentifier="1">
    <Source>
      <Standard_Partition PortName="pdataout" PartitionName="pr1" PartitionIdentifier="2"/>
    </Source>
    <Destination>
      <Standard_Partition PortName="pdatain" PartitionName="pr2" PartitionIdentifier="1"/>
    </Destination>
  </Channel>
</Connection_Table >
```

(a) Process communication configuration information of ARINC653



(b) Process communication transformation rules

Figure 6: Process configuration transformation from ARINC653 to MARTE.

and data encapsulation. The hwProcessor MARTE component properties can be used to describe the current environment of the system, including the CPU resource allocation, scheduling memory allocation, and communication connection, so we can use hwProcessor MARTE components to model IPM module concept IMA, module name available

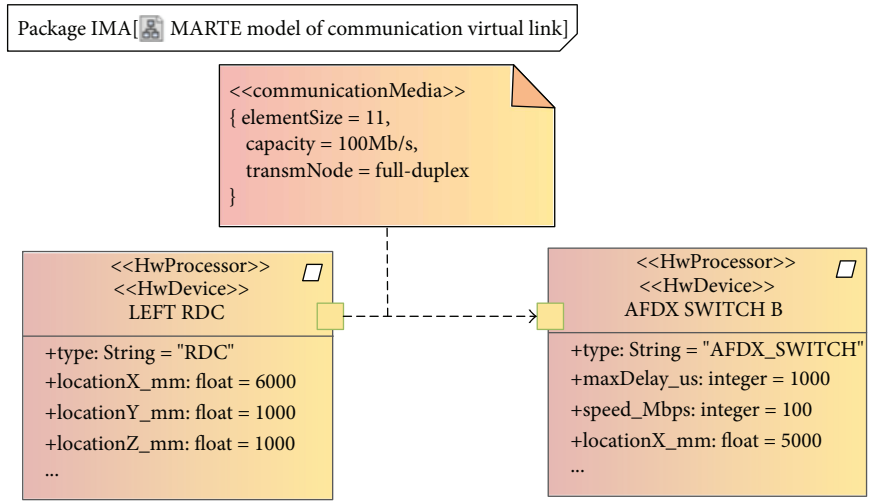


Figure 7: MARTE model of communication virtual link.

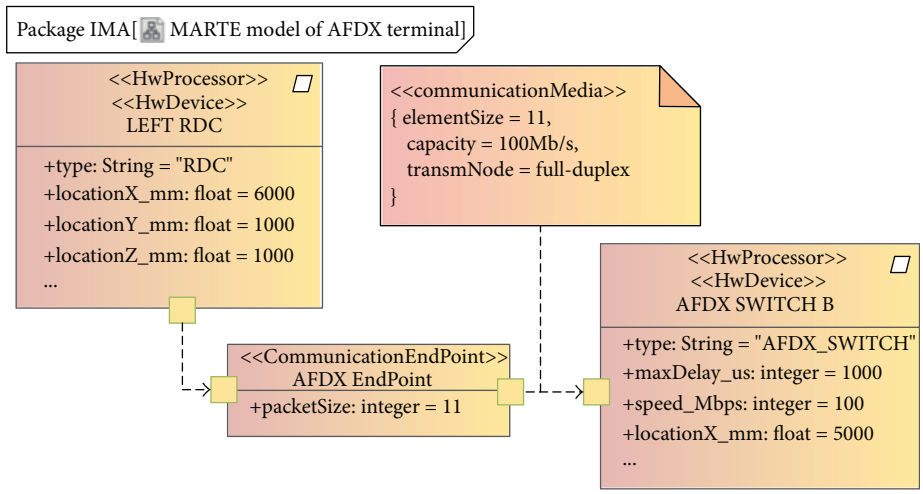


Figure 8: MARTE model of AFDX terminal.

description component attributes to define mainScheduler attributes in hwProcessor schedPolicy to set the partition between the scheduling strategy. At the same time, hwProcessor components and HwDevice components in MARTE can represent the concept of resource modules such as AFDX switches and RDC.

Figure 9 gives the description of MARTE model in IMA IPM, RDC, AFDX switches and controllers, and other resources equipment module. The IPM module and scheduling framework for 1000US contains two Psys partitions and Papp; RDC and AFDX switch and control by using hwProcessor and HwDevice components, including the maximum delay time, transmission rate, and refresh rate of property.

4. A Framework for Verification of ARINC653 System Configuration

Based on the above modeling rules for ARINC653 system configuration by transforming the XML format configuration data to corresponding MARTE elements, in this section, we propose a formal framework for verification of safety properties which should be considered during the stage of ARINC653 configuration design and maintenance. Figure 7 shows the main steps in this framework in which an open source environment Ocarina [17, 18] and a theorem proof tool are applied. Since up to now there are still no formal model verification techniques which can be directly applied to MARTE models, we have to transfer those MARTE models into corresponding AADL models which are inputs of Ocarina (those transformations rules can be referred to [12, 19]) for the purpose of applying the theorem proof tool effectively. In our future works, we will consider to establish formal analysis techniques directly addressing the MARTE models. In Figure 10, another input of Ocarina is safety properties of ARINC653 system configuration which are described in logic-based formal language REAL (Requirement Enforcement Analysis Language) [20].

Requirement Enforcement Analysis Language (REAL) is a kind of model checking language applied in specification and architecture design stage to analyze the validity of

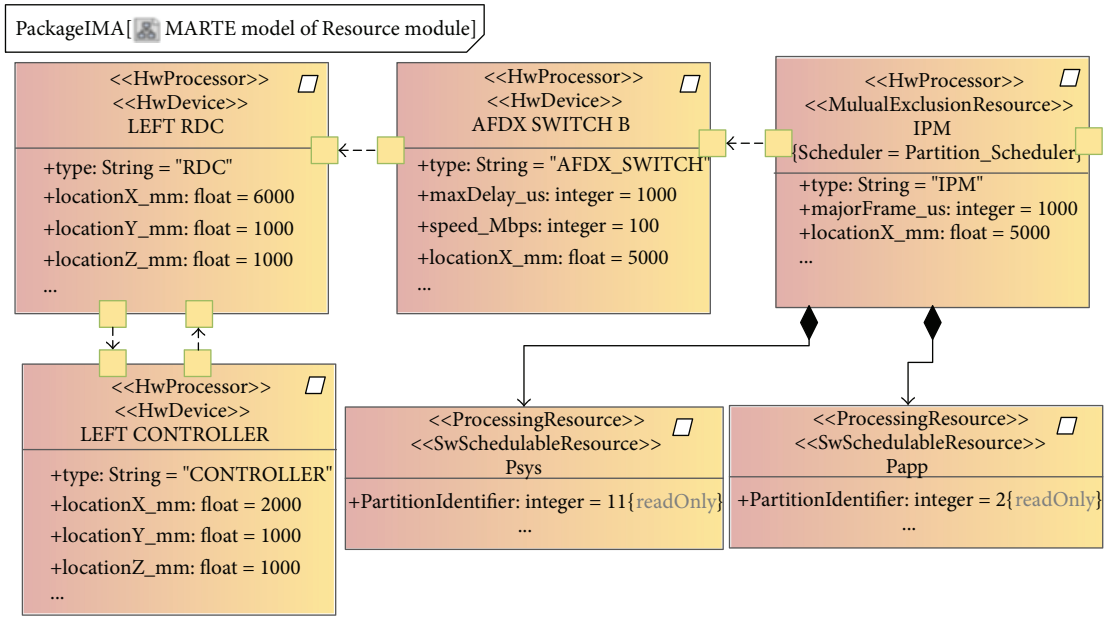


Figure 9: MARTE model of resource module.

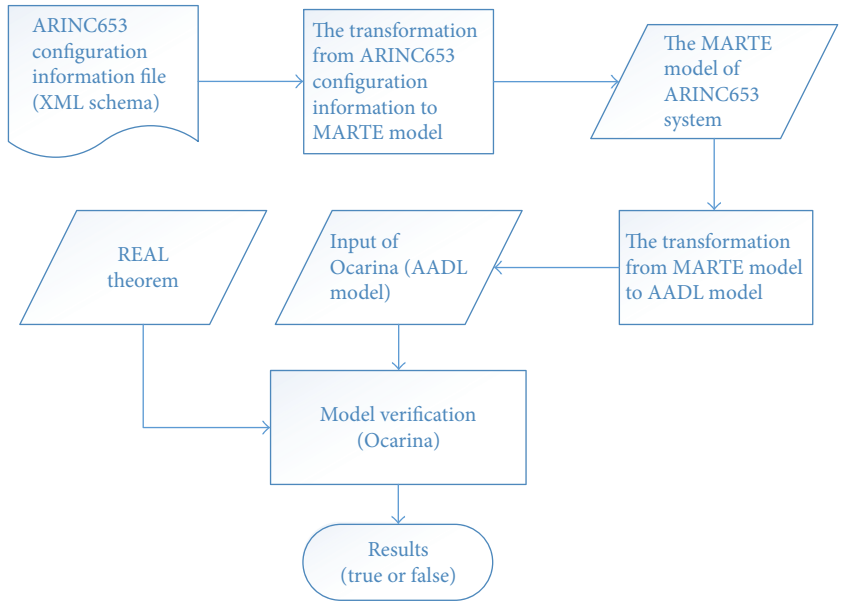


Figure 10: Validation framework of configuration information.

models. In essence, it is a language based on set operations, allowing establishment of a set of elements, to verify models by providing a set of first-order logic and Boolean expression statements. More examples of REAL theorems are shown below. As mentioned above, inputs of the verification framework of Figure 10 includes two parts: AADL models corresponding to the MARTE models which resulted from transformation of ARINC653 system configuration information and REAL theorems of description for safety properties of system configuration in the stages of specification and design. The REAL execution engine in Ocarina can verify whether the AADL models satisfy those properties with the form of first-order logic expressions and theorems. The following is a list of some typical examples of safety properties of ARINC653 system configuration, and we give the corresponding REAL theorems in the next subsections:

Timing Constraints. The requirements of time constraints must be guaranteed such that in a scheduling cycle, each partition is scheduled at least one time. The sum of all time allocations of each partition and one cycle time in a module is equal and so on.

Spatial Isolation. Memory must be allocated individually for each partition and so on.

Memory Allocation. The stack space, data space, and code space occupied by processes and threads in partition must be less than the application in the partition configuration and so on.

Health Monitoring. Health monitoring must ensure that each potential fault is banded with a corresponding recovery action, and a hierarchical structure of each level (module, partition, and process) should be associated with all kinds of recovery action and so on.

The following are the descriptions of safety properties of ARINC653 configuration with the form of REAL theorems.

4.1. Time Constraints. In brief, time constraints need to verify two kinds of constraints: the first one is that each partition of an ARINC653 module must be scheduled one time in each scheduling period and the second one is that the sum of all partition scheduling slices should equal to the time period of the module allocated. Figure 11 shows an example of second constraint, that is, the MARTE property Module Major Frame indicates the period of module-level schedule, and Partition Slots is a list of values, indicating time allocations for each partition in a module. What we need to check is whether each ARINC653 module of the Partition Slots value equals the value of Module Major Frame.

4.2. Spatial Isolation. Property of spatial isolation in essence needs to verify that one specific segment of memory should only be assigned to one partition which is shown as a REAL theorem in Figure 12. Specifically, Figure 12 identifies the main memory component of the system that is variable minimum, and then obtains all its subcomponent of memory, that is variable *partmen*. Finally, we get each subcomponent and assign them to the process component. What we need to check is whether the number of process component elements is equal to 1, for instance, one memory segment must be assigned to one partition.

4.3. Memory Allocation. Memory allocation involves lots of issues. It just gives an example of basic safety property as mentioned before: the stack space, data space, and code space occupied by processes and threads in one partition must be less than the application in the partition configuration. Figure 13 shows the corresponding REAL theorem which firstly identifies all threads in a process of one partition and finds out the necessary memory quantity of the partition process, that is, variable MEMS, then we need to check whether the stack space, data space, and code space occupied by processes and threads in one partition is less than the application in the partition configuration.

4.4. Health Monitoring. Example of property of health monitoring includes two kinds of constraint: the first one is when an error occurs inside a partition; it should not affect other partitions even if there are existing data communications between partitions. The first way to achieve this objective is for the receiver partition to be assigned to a lower critical level than the sender partition as shown in Figure 11. The second one is that faults inside ARINC653 system are divided into three different levels: module, partition, and process; one specific fault should be detected and covered at least in one level of the three.

```
theorem partition_execution
foreach cpu in Module_Set do
   vp:= {x in Partition_Set | Is_subcomponent_of(x,cpu)};
   check(float(property(cpu,"Module_Major_Frame"))=
      sum(property(cpu,"Partition_Slots")) and
Cardinal(vp)>0);
end;
```

Figure 11: REAL theorem of partition time constraints.

```
theorem memory_bound
foreach s in System_Set do
   mainmem := {y in Memory_Set | Is_subcomponent_of(y,s)};
   partmem := {x in Memory_Set | Is_subcomponent_of(x,mainmem)};
   partitions := {x in Process_Set | Is_Bound_to(x,partmem)};
   check(Cardinal(partitions) = 1);
end;
```

Figure 12: REAL theorem of spatial isolation.

```
theorem check_memory_requirements_partition
foreach prs in Process_Set do
   Thrs := {x in Thread_Set | Is_subcomponent_of(x,prs)};
   mems := {x in Memory_Set | Is_bound_to(prs,x)};
   check((sum(property(Thrs,"Source_Stack_Size")))
         +sum(propertyThrs,"Source_Data_Size")))
         +sum(property(Thrs,"Source_Code_Size")))
         <sum(property(mems,"Byte_count"))));
end;
```

Figure 13: REAL theorem of memory allocation.

Figure 14(a) shows the REAL theorem example of the property of health monitoring. Firstly, the theorem gets the ARINC653 partition execution environment by using function calls of communication in the system partition, then checks whether the value of Criticality (representing the partition critical level) of destination partition is less than the value of sender partition. Figure 14(b) also shows the second constraint of example, that is, by firstly getting the information of partition which contains the ARINC653 process to be analyzed and also the corresponding execution environment of the partition, then getting the module which contains corresponding partitions. After that, we need to check the fault list (the attribute is HM ERRORS) of the ARINC653 process, partition, and module and determine whether its critical level is according to the default fault list.

5. Schedulability Analysis of ARINC653 Module

After getting the MARTE model with the concept of time behavior related to the IMA system, we constructed the corresponding MAST-2 text model according to the MARTE model of the system and verified the schedulability of the system based on the MAST tools. The analysis

```
theorem check_partition_level
foreach srs in Partition_Set do
   thr := {x in Process_Set | Is_subcomponent_of(x,thr)};
   spart := {x in Partition_Set | Is_Bound_to(src,x)};
   dpart := {x in Partition_Set | Is_Connected_to(src,x)};
   check(cardinal(src)>0 and cardinal(dst)>=0 and
         (max(property(dpart,"Criticality"))
          <max(property(spart,"Criticality"))));
end;
```

(a)

```
theorem check_error_coverage
foreach thr in Partition_Set do
   prs := {x in Process_Set | Is_subcomponent_of(thr,x)};
   ps := {x in Partition_Set | Is_Bound_to(prs,x)};
   cpu := {x in Module_Set | Is_subcomponent_of(ps,x)};
   var errors := List("Power_Fail","Module_Config","Module_Init",
    "Module_Scheduling","Partition_Scheduling","Partition_Config",
    "Partition_Handler","Partition_Init","Deadline_Miss",
    "Application_Error","Numeric_Error","Illegal_Reguest",
    "Stack_Overflow","Memory_Violation","Hardware_Fault");
    var actual_errors := List(property(cpu,"HM_Errors") +
          property(thr,"HM_Errors") + property(ps,"HM_Errors"));
    check(Is_In(errors,actual_errors) and Is_In(actual_errors,errors));
end;
```

(b)

Figure 14: REAL theorems of health monitoring.

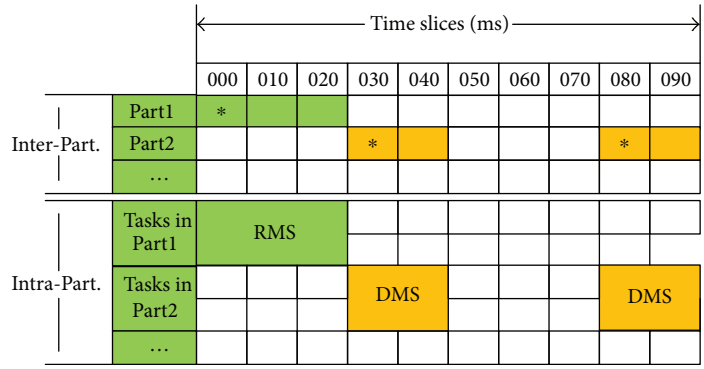


Figure 15: The scheduling model of IMA. ∗ means the start of schedule.

results show that the scheduling model of IMA system has two level scheduling characteristics of interpartition scheduling and intrapartition scheduling.

5.1. The Feature of Schedule Model for IMA System. In the IPM module of the IMA system, there are multiple application partitions and system partitions, each of which contains a set of respective execution tasks. In the IPM resource configuration phase, the CPU time resource will be allocated to each partition, and the allocation is carried out in the form of a time slice. The execution of all the partitions themselves and the tasks in the partition need to meet the allocated time slice constraints; here, the time slice contains two parameters, offset and duration. If the time slice's corresponding time is finished, the current operation will be interrupted until the next time slice is allocated to continue the corresponding task. Corresponding to partitions in between and tasks within the partition, the IMA system scheduling model consists of two level scheduling strategies, which are interpartition scheduling and intrapartition scheduling, respectively, as shown in Figure 15.

5.1.1. Interpartition Scheduling. The interpartition scheduling (called the first-level scheduling) in the IMA system is determined when the IPM resource is configured. Each partition is periodically allocated to a time slice, and the size of the time slice is denoted as duration. The characteristics of interpartition scheduling are as follows:

(a) Partition is the basic unit of scheduling.

(b) There is no priority between partitions.

(c) Interpartition scheduling is repeated in a period of time.

(d) Scheduling period is determined by the size of the time slice fixed in IPM resource allocation.

(e) At least all partitions are assigned to one partition window.

Interpartition scheduling needs to ensure the size and cycle of the time slice of all partitions and the main time framework to meet the application requirements of the avionics system, and window scheduling cannot be overlapped.

5.1.2. Intrapartition Scheduling. Intrapartition scheduling (called the second-level scheduling) refers to the process or intertask scheduling within a partition. The intrapartition scheduling is based on the corresponding task scheduling strategy to allocate time to the task process set. The task scheduling policy is a preemptive scheduling strategy, which is usually EDF (earliest deadline priority), LLF (idle time first), RMS (rate monotonic scheduling), DMS (deadline

monotonic scheduling), and so on. In the stage of IPM resource allocation, it is unable to allocate the time slice for each task, so it is impossible to directly decide whether the time slice of each task satisfies the requirements. In this case, the user needs to customize the scheduling policy and then use the MAST tool to determine the schedulability.

5.2. MAST-2 Model Description of IMA System. According to the IMA system's MARTE model description and related time requirement constraints, MAST-2 text model can be established and used as input of MAST schedulability decision tool to analyze schedulability of the system. MAST-2 model elements have been described in Section 2.2. The following will show how to describe the IMA system according to the construction of MAST-2 text model's corresponding MARTE model, mainly from the Processing_Resource and Scheduler attributes of Scheduling, _Resource, Operation, End_To_End_Flow, and MAST-2 model to describe the elements of IMA system.

5.2.1. The Elements of Processing_Resourse Model. Processing_Resource is used to describe the processing power of a hardware component, including executing a piece of code or forwarding a set of messages. The subtypes of Processing_Resource include Regular_Processor and AFDX_Link. Regular_Processor describes the ability of processor modules to execute applications, and AFDX_Link describes links between processors and switches that use AFDX protocol to transmit messages. So the IPM module in the MARTE model can be represented by the Regular_Processor type in Processing_Resource model, and AFDX switch network can be represented by AFDX_Link. Table 7 specifies that a MARTE model of a IPM module and a AFDX switch network module is transformed into a Processing_Resource model element.

5.2.2. SchedulerModel Elements. The Scheduler model element manages the applications or tasks assigned to the processor by using the appropriate scheduling policy, these scheduling policies include the fixed priority policy (denoted as Fixed_Priority_Policy), the earliest deadline first (abbreviate to EDF), and so on. Scheduler has a hierarchical structure, including Primary_Scheduler and Secondary_Scheduler. Therefore, the partition information and scheduling in the MARTE model can be represented by Primary_Scheduler in Scheduler, and the process information and scheduling information can be represented by Secondary_Scheduler in Scheduler. Table 8 provides a Scheduler text model for the corresponding interpartition scheduling and intrapartition scheduling:

5.2.3. Schedulable_Resource Model Elements. The Schedulable_Resource model elements describe a schedulable entity, such as a task executed in a processor or a communication task in the network. The subtypes of Schedulable_Resource are Thread and Communication_Channel. Thread describes the execution of a thread or task in the Regular_Processor model, and Communication_Channel describes the transmission of messages in the network. Therefore, in the MARTE model of IMA system, each process task can be described by Thread type. Table 9 is a Schedulable_Resource model element description of a computing task in IMA system.

Table 7: Processing_Resource model elements of MARTE.

```
–IPM Module
Processing_Resource (
    Type => Regular_Processor,
Name => processor1,
Max_Interrupt_Priority => 32,767,
Min_Interrupt_Priority => 1,
Speed_Factor => 1.00);
–AFDX Networks Module
Processing_Resource (
    Type => AFDX_Link,
    Name => AFDXNETWORK,
    Transmission => FULL_DUPLEX,
    Max_Packet_Size => 1,
    Min_Packet_Size => 0.00,
    Speed_Factor => 1.00);
```

5.2.4. Operation Model Elements. The Operation model element describes the processing capacity range of a computing task or message transmission execution, including the time range of task execution and the range of message size. The subtypes of Operation include Code_Operation and Message_Operation, which describe the processing of computing tasks and message delivery, respectively. Table 10 describes the MAST-2 model for performing the processing capabilities of a computing task Task1 and message transfer nettask1 in the IMA system.

5.2.5. End_To_End_Flow Model Elements. End_To_End_Flow model element describes the system implementation of activities triggered the relationships between conditions in a series of events, including three types, which are external event External_Events, internal event Internal_Events (including time requirement), and event processor Event_Handlers. The End_To_End_Flow model element can be used to describe the IMA partition scheduling model. Entry events belong to external events, its type is Periodic, and express that major time slicing is periodic distribution; the end event belongs to the internal event, its hard global deadline can represent the rotation cycle of major time slice of the partition scheduling, and the time slice of each partition is set between the entry event and the end event, and there is a Hard global deadline to represent the available time slices of the partition, respectively. The corresponding MAST-2 model description is shown in Table 11.

5.3. IMA System Schedulability Verification Framework. According to the characteristics of IMA system partition scheduling, this paper proposes a schedulability decision method based on the third-party tool MAST simulation method for IMA system time resource configuration verification. To aim at MARTE model for time resource configuration behaviors of system, the corresponding MAST-2 model text file can be constructed. At the same time, because the MAST scheduling strategy is scalable, we can choose the

Table 8: Scheduler model element description.

```
–Inter-partition scheduling
Scheduler(
    Type => Primary_Scheduler,
    Name => partition1,
    Host => processor1,
    Policy =>
        (Type => Fixed_Priority,
        Max_Priority => 32,767,
        Min_Priority => 1));
– Intra-partition scheduling
Scheduler(
    Type => Secondary_Scheduler,
    Name => task1,
    Host => partition1,
    Policy =>
        (Type => EDF,
        Is_Global => No));
```

Table 9: Scheduling_Resource model element description.

```
– Computing task: task1
Schedulable_Resource(
    Type => Thread,
    Name => task1,
    Schedulable_Parameters =>
        (Type => EDF,
        Worst_Context_Switch => 0.00,
        Avg_Context_Switch => 0.00,
        Best_Context_Switch => 0.00
        Is_Global => No),
        Scheduler => partition1));
```

Table 10: Operation model element description.

```
– computing task task1
Operation(
    Type => Code_Operation,
    Name => task1,
    Worst_Case_Execution_Time => 50.00,
    Avg_Case_Execution_Time => 50.00,
    Best_Case_Execution_Time => 50.00);
– message transfer
Operation(
    Type => Message_Operation,
    Name => nettask1,
    Max_Message_Size => 1,
    Avg_Message_Size => 1,
    Min_Message_Size => 1);
```

custom scheduling strategy according to requirements, such as adding relevant time constraints information into MAST-2 model files, and then using MAST tools for further schedulable analysis. Finally, we get the schedulability decision results and scheduling simulation Gantt chart. The concrete method framework is shown in Figure 16. We will give a specific example analysis on the IMA airborne water processing subsystem in Section 7.2.

Table 11: End_To_End_Flow model element description.

```
– Scheduling time attributes of partition P1, P2
End_To_End_Flow (
  Type => regular,
    Name => module,
    External_Events =>
    (Type => Periodic, – Time slice type is periodic
    Name => module_input,
    Period => 1000.000, – The cycle is 1000 μs
    Max_Jitter => 0.000,
    Phase => 0.000),
  Internal_Events =>
    (Type => Regular, – Scheduling time attributes of partition P1
    Name => p1_out,
    Timing_Requirements =>
      (Type => Hard_Global_Deadline,
      Deadline => 10000.000,
        Referenced_Event => module_input)), – timer starts
    (Type => Regular, – Attributes of Part.2
      Name => p2_out,
      Timing_Requirements =>
        (Type => Hard_Global_Deadline,
        Deadline => 10000.000,
        Referenced_Event => p1_out))),– Start the time from the
end of P1
Event_Handlers => –- Define the proces of time slice
(Type => Activity, –- From the module to the partition P1
    Input_Event => module_input,
    Output_Event => p1_out,
    Activity_Operation => p1,
    Activity_Server => p1_task),– Executing the P1 task
(Type => Activity, – From partition P1 to partition P2
    Input_Event => p1_out,
    Output_Event => p2_out,
    Activity_Operation => p2,
    Activity_Server => p2_task));– Executing the P2 task
```

6. CC653: A Configuration Checker for ARINC653 Module

Based on the above methodology of modelling, verification, and schedulability for ARINC653 system, we design a prototype tool CC653 (Configuration Checker for ARINC653), its functions include configuration information model extraction, configuration correctness verification, and schedulability verification for an IMA/ARINC653 module. It should be noted that our works up to now do not consider AFDX networking configuration which is another big issue [8] and it needs a different way to deal with. The architecture of CC653 is shown in Figure 17.

The purpose of the CC653 tool is prepared for stages of an ARINC653 module design and reconfiguration. Its framework is divided into three levels: application layer, business layer, and persistence layer.

(1) The application layer provides main functions, including MARTE model viewing and editing, scheduling analysis, safety property verification, and result displaying. Data cleaning module check whether the ARINC653 configuration files and the model files

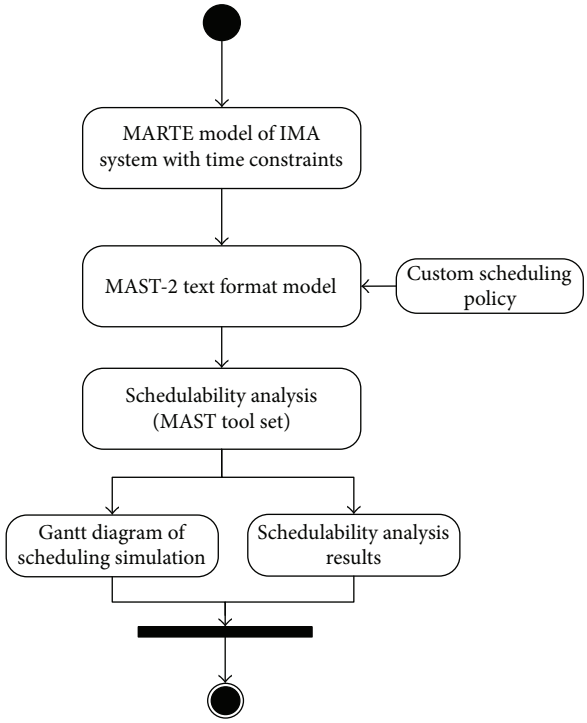


Figure 16: Process flow of IMA system schedulability verification.

are available, the data form is complete or not, and return the relevant error tips for the user. The verification results in dynamic displaying module will show users the multidimensional verification results, such as scheduling simulation results and scheduling simulation Gantt chart.

(2) The business layer is responsible for processing those instructions coming from the application layer, for instance, the transformation from XML formatted configuration information to MARTE models. On the other hand, data processing of the persistent layer are completed in this level. And business layer also needs to provide scheduling strategies and verification documents to the application layer.

(3) The persistence layer manages different kinds of data, such as reading the configuration file, accessing the MARTE model file, and generating the MAST-2 file.

The execution process of CC653 is described in Figure 18. Firstly, we need to parse the configuration file in XML format and save the corresponding MARTE model transformed according to the modelling rules. Then REAL theorems for safety properties of system configuration are constructed through requirement constraints, and the theorems are imported into Ocarina to verify with MARTE models. For schedule analysis, CC653 continue to convert information of task sets in MARTE configuration model into text format which can be identified by MAST tool, also custom schedule strategy. Finally, verification results and the corresponding error messages are output through the user interface. We can group those process steps into three modules as follows.

The module of configuration information model generation provides functions including ARINC653 configuration information (XML format) editing and parsing:

(1) MARTE model generation, configuration information conversion rule management, verification function selection, and other functions. The user can view the ARINC653 configuration information according to the integrated XML tool, and also display and analyse the converted MARTE model.

(2) The module of configuration information correctness verification provides the functions including viewing REAL theorems and calling Ocarina model analysis suite to analyze and verify the MARTE model generated from the module of configuration information model generation.

(3) The module of partition schedulability verification provides the functions including modelling the scheduling information, the conversion of MARTE model to the text recognized by MAST, and calling mast tool to check the schedulability of the converted model.

7. Case Study

7.1. Example of General Configuration Verification. Table 12 describes a partition scheduling information of ARINC653 module; Table 13 is a part of the XML configuration file which contains the configuration information Table 7 described. The total scheduling period of the current module is 0.2 seconds, it contains 5 partitions. The table and figure specify the starting time and execution time for each partition, that is, the starting time and size of the corresponding time window.

According to the model transformation method, we firstly convert the ARINC653 system configuration information illustrated in Figure 19 into MARTE models. Figure 19 partly shows the MARTE model. Partition P1, P2, P3, P4, and P5 were converted into the virtual processor component partition 1, partition 2, partition 3, partition 4, and partition 5; the allocation partition time slice are expressed in attribute slots allocation and partition slots. The former one specifies the sequence of partitions occupying the system clock and the latter one specifies the size of time slice. For the idle duration of system clock, there is no partition occupying the system clock, so firstly a free partition (partition idle) can be created, then the corresponding time slices are allocated to that partition.

Then we need to prepare REAL theorems according to the safety requirements to be verified. Considering that the example configuration described in Table 12 contains time information, we need to verify the time constraints of system partition. Here, we use the theorem (depicted in Figure 11 of Section 4) to check whether the size of module time slices are equal to the sum of partition time slices, and save the verification theorem as a file with the extension of "real." After that, we use the Ocarina tool to analysis the model and the result of output is shown in Figure 20. In that snapshot of

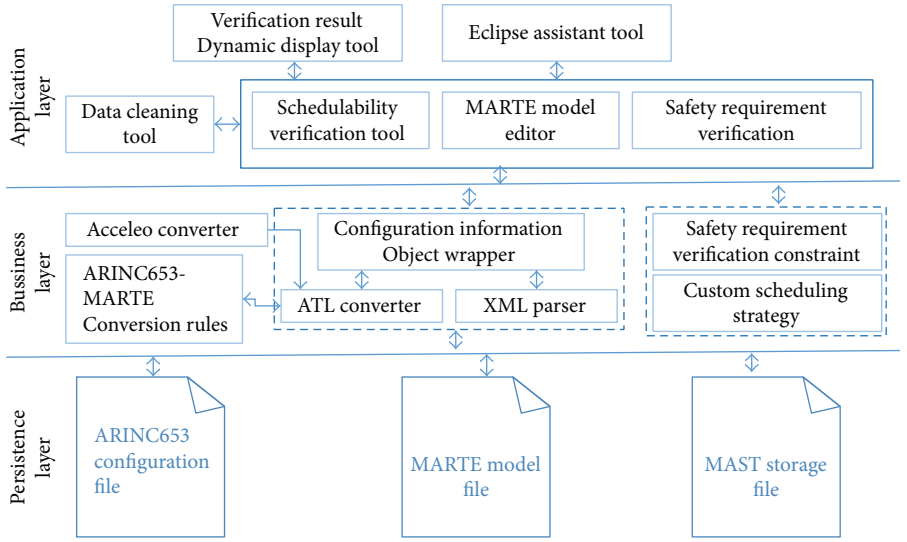


Figure 17: Architecture of prototype tool CC653.

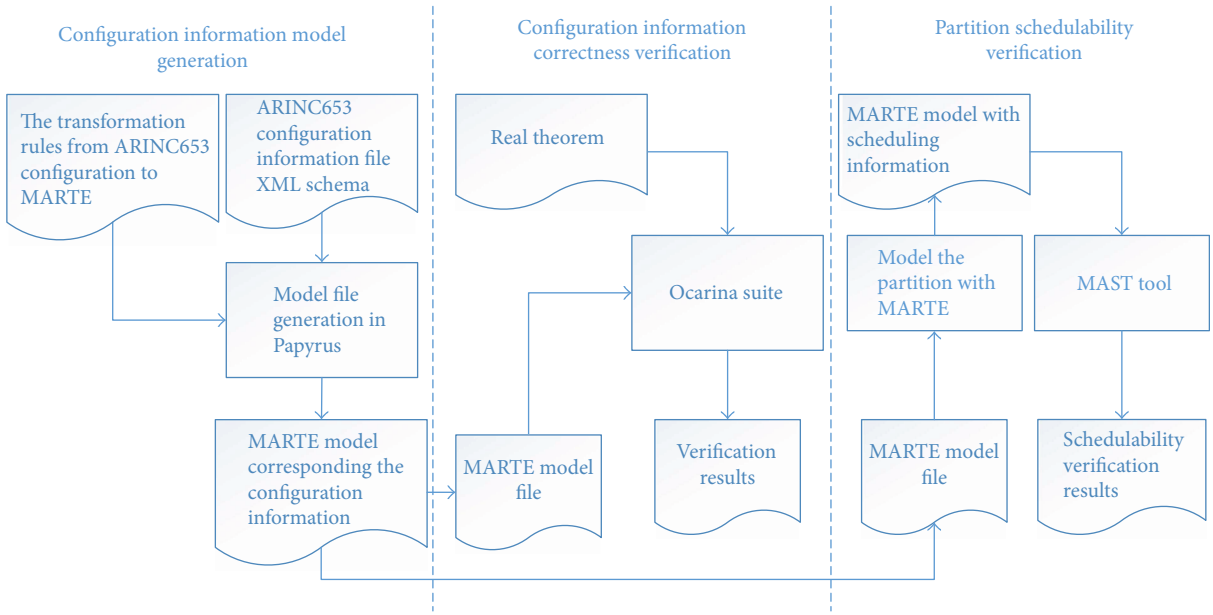


Figure 18: Execution process of CC653.

processing, the arinc653.real is a file name of the REAL theorem, and ARINC653 system is the input file name of Ocarina which corresponds to the MARTE configuration model.

The results of the primary analysis shows that the configuration contained in Table 12 is satisfied with the requirement of time constraints, namely, in a module scheduling period, the sum of partition time slice is less than or equal to the total of module time slices, so the verification process returns true (shown in Figure 20 that theorem resource is TRUE). However, an ARINC653 module maybe under reconfiguration in the stage of system maintenance; for instance, we may need to add a new partition P6 on the current module to support some new functions. The scheduling information of P6 after reconfiguration is shown in Table 14; that is, we add the allocation of time slice to P6. Now we should have to verify the time constrain of the reconfiguration file once again. After second verification process, we found that the configuration information contained in Table 14 is not satisfied with the requirement of time constraint, namely in a module scheduling period, the sum of partition time slice is greater than the total of module time slices (the sum of all data in line "window duration" is greater than 0.2 seconds), the results return false, which means that the configuration needs to adjust more carefully.

7.2. Example of Partition Schedulability. In this section, we will give an example of modelling the partition system schedule information with MARTE and the ARINC653 system

Table 12: Partition configuration schedule rules.

Window ID	Partition	Window offset	Window duration
1.1	P1	0.00	0.02
4.1	P4	0.02	0.01
2.1	P2	0.03	0.01
3.1	P3	0.04	0.03
4.2	P4	0.07	0.01
1.2	P1	0.1	0.02
4.3	P4	0.12	0.01
2.2	P2	0.13	0.01
3.2	P3	0.14	0.03
4.4	P4	0.17	0.01
5.1	P5	0.18	0.02

Table 13: Partition schedule configuration in XML format.

```
<Module_Schedule MajorFrameSeconds="0.200">
  <Partition_Schedule PartitionIdentifier="1"
        Partition_Name="system management"
        PeriodSeconds="0.100"
        PeriodDuriationSeconds="0.020">
    <Window_Schedule WindowIdentifier="101"
        WindowStartSeconds="0.0">
        WindowDuriationSeconds="0.020"
        PartitionPeriodStart="true"/>
    <Window_Schedule WindowIdentifier="102"
        WindowStartSeconds="0.1">
        WindowDuriationSeconds="0.020"
        PartitionPeriodStart="true"/>
  </Partition_Schedule>
  …
  <Partition_Schedule PartitionIdentifier="5"
        Partition_Name="IHVM"
        PeriodSeconds="0.200"
        PeriodDuriationSeconds="0.020">
    <Window_Schedule WindowIdentifier="501"
        WindowStartSeconds="0.180">
        WindowDuriationSeconds="0.020"
        PartitionPeriodStart="true"/>
  </Partition_Schedule>
</Module_Schedule>
```

schedulability analysis. This example shows how to use MARTE to model the schedule information of ARINC653 system with the help of custom scheduling strategy of MAST and use MAST for the simulation of scheduling. According to the simulation results, we can know that an ARINC653 module is schedulable or not.

In the aviation specification ATA-38, the main functions of the airborne water-and-waste (WAW) system are described, including the storage and transmission of clean water, storage, and removal of sewage.

Figure 21 gives an overview of the physical structure of the WAW, which contains two tanks for storing drinking water and sewage. Two signal controllers are also included in the system. Each controller contains a sensor and an actuator. The main functions of the system are scheduled to run on the IMA platform in the form of a partition application. Its main functions include calculating the available space, calculating the equilibrium state of the aircraft, and controlling the valve and the information of the output state to the cockpit. In addition, the current system parameters will be displayed in a cockpit display.

Figure 22 describes the structure of the hardware resource network of the WAW system. This architecture is based on the requirements of the high-level system and the information of the IMA platform. Two signal controllers in the figure are used to detect the various signals of the storage tank in the WWS system and then transmit it to the corresponding RDC (Remote Data Collector) in the form of analog signals. Another two RDCs are connected to those signal controllers and AFDX switches, respectively. The function of RDC is to convert and transmit the digital and analog signals. And two AFDX switches connect those devices into the AFDX network. An application partition and a system partition are scheduled to run in the IPM (Integrated Processing Module). In the application partition, the current system state is calculated based on the received data, and the control command is sent. The function of the system partition is mainly to monitor the whole system and the communication of the virtual link among different devices.

Table 15 shows schedule information of an interpartition and intrapartition task set in a WAW system. Papp denotes the application partition and Psys denotes the system partition. The main schedule framework duration is 10 ms. And scheduling strategy could be EDF and DMS. The time slices for partitions are 6 ms and 4 ms, respectively. For each partition, all kinds of task set parameters are included in the table: task, cycle, execution time and deadline, and so on.

According to the method given in the third section which shows how to transfer an IMA system into a MARTE model, we can establish the MARTE model of WAW system, as shown in Figure 23.

In this MARTE model, the hwProcessor and hwDevice components represent the AFDX switches, RDC, and signal controllers in the WAW system. The hwProcessor and MutualExclusionResource components represent the IPM processor module. The system partition and application partition in IPM are represented by the combination of swSchedulingResource and ProcessingResource components and related attributes, respectively. Each partition has two tasks and its scheduling properties are described with swSchedulingResource components. Finally, the time constraints corresponding to each task are represented by TimeConstraint component, including time slice and deadline.

The results of the analysis of the WAW's MARTE model by using MAST are shown in Figure 24. The left part of Figure 24 is the result of the schedulability analysis of the time resource configuration. It can be judged from the result that time resource allocation in the table can satisfy the requirement of WAW system, and there will not be any partition or process without being scheduled. The right part of Figure 24 is the Gantt diagram analysis for the scheduling simulation. From this diagram, according to the above time configuration, all processes and tasks can be scheduled, and there are no time window overlaps.

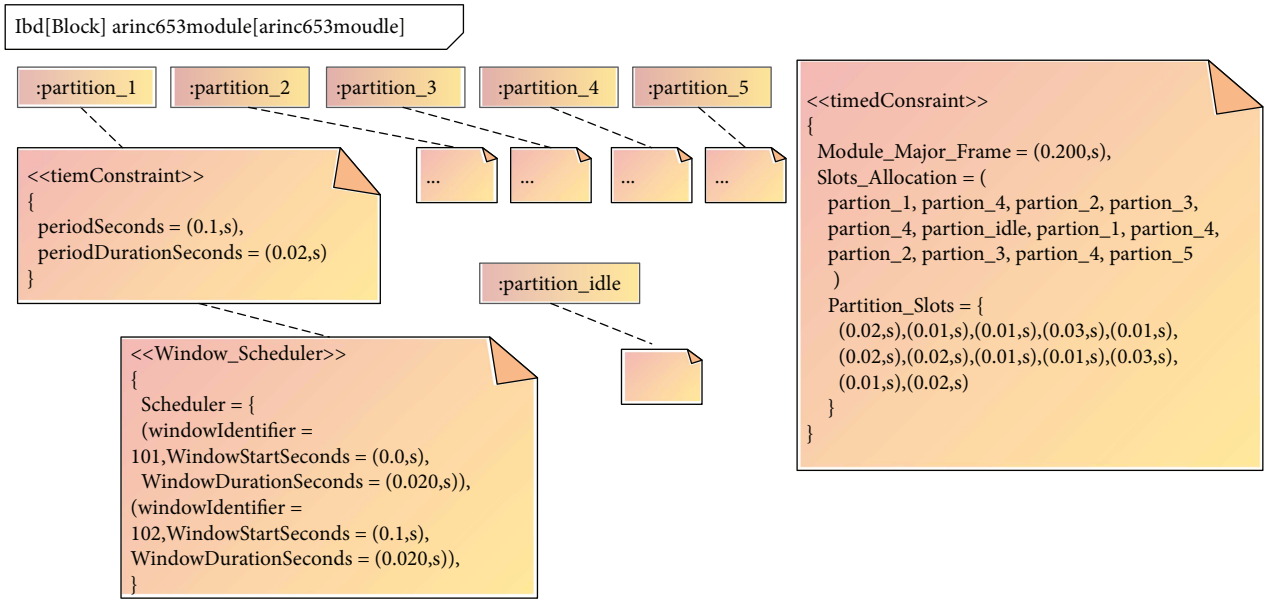


Figure 19: Partition configuration schedule model of MARTE.

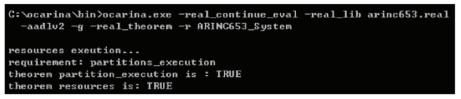


Figure 20: Result of partition configuration analysis.

Table 14: Reconfiguration table of partition information.

Window ID	Partition	Window offset	Window duration
1.1	P1	0.00	0.02
4.1	P4	0.02	0.01
2.1	P2	0.03	0.01
3.1	P3	0.04	0.03
4.2	P4	0.07	0.01
1.2	P1	0.1	0.02
4.3	P4	0.12	0.01
2.2	P2	0.13	0.01
3.2	P3	0.14	0.03
4.4	P4	0.17	0.01
5.1	P5	0.18	0.02
4.3	P5	0.08	0.03

That is, the time resource configurations in the table are verified and safe.

In the process of IMA system resource configuration, time resources could be reconfigured because of the changes of requirement of the system. For example, if we want to add an application partition Papp2 in the WAW system, we need to allocate a time slice and other resources for the Papp2. Table 16 shows the modified time resources of all partitions. At this point, the total time framework is still required to be 10 ms. After analysis by MAST, we can find that the allocation of time resources for Papp2 may lead to the situation that the sum of time slices of all partitions are greater than the main time frame. Therefore, this resource schedule framework will not be accepted.

8. Related Works

The related works are divided into the following four categories: the first one is related to the domain of model-driven engineering methods applied in the development of complex airborne systems in recent years; for instance, [21] introduced a verification analysis method based on model-driven framework for aviation electronic system and [22] provided a modelling analysis example based on the system modeling language (SysML). The second one is about the research of MARTE and AADL (architecture analysis and design language) [23] in the design stage of complex embedded systems. For example, [12] described the concept of consistency and the mapping relationship between MARTE subset and AADL; reference [19] established the model transformation method between MARTE and AADL. Third, related works include research on the structure and analysis methods for IMA/ARINC6523 system configuration information. For example, reference [24] described a model-driven development method of ARINC653 configuration table; reference [25] designed an IMA system reconfiguration verification method which is not based on model-driven methodology instead of directly processing configuration data set. The last one of the related works is about IMA system schedulability, that is, including [26] which described how to compute time consumed by the system based on the function of task time requirement and established a decision theorem of system schedulability; the effect of partition parameters on the real-time performance of task scheduling is analyzed by [27]. By comparison of the above related

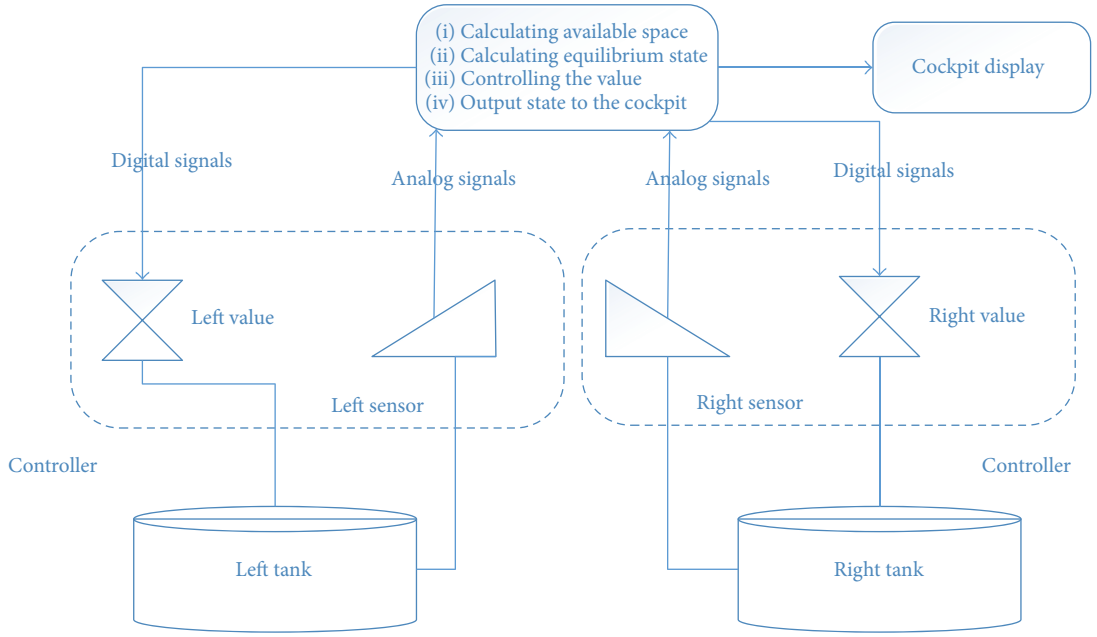


Figure 21: Physical structure of the water-and-waste system.

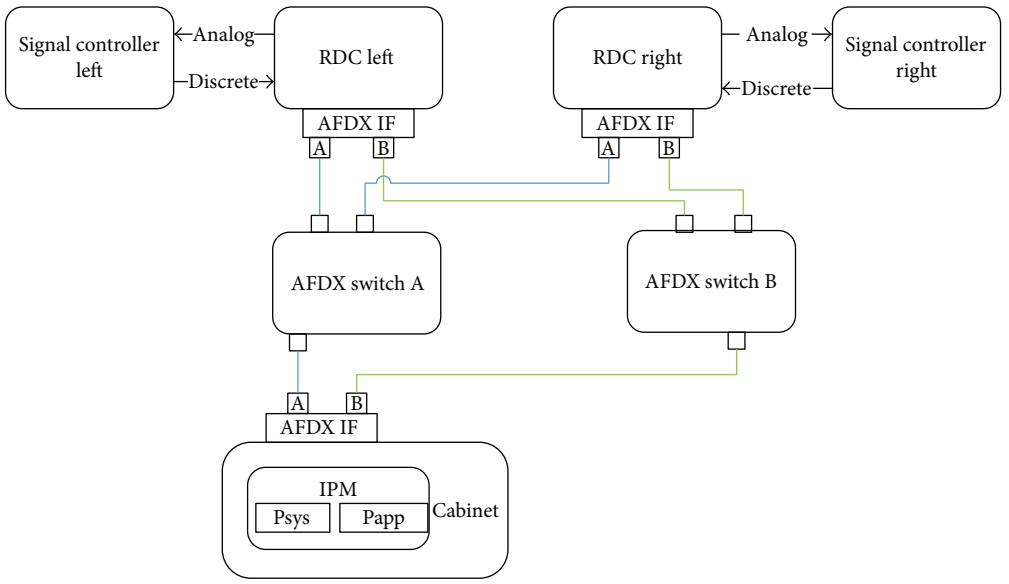


Figure 22: The resource architecture of water-and-waste system.

Table 15: Schedule information of WAW system.

Partitions	Time slice (ms)	Offset	Duration	Schedule strategy	Task sets			
					Task	Cycle	Execution time	Deadline
Papp	6	0	10	DMS	T1	10	3	10
					T2	5	1	5
Psys	4	0	10	RMS	T3	20	2	20
					T4	10	2	10

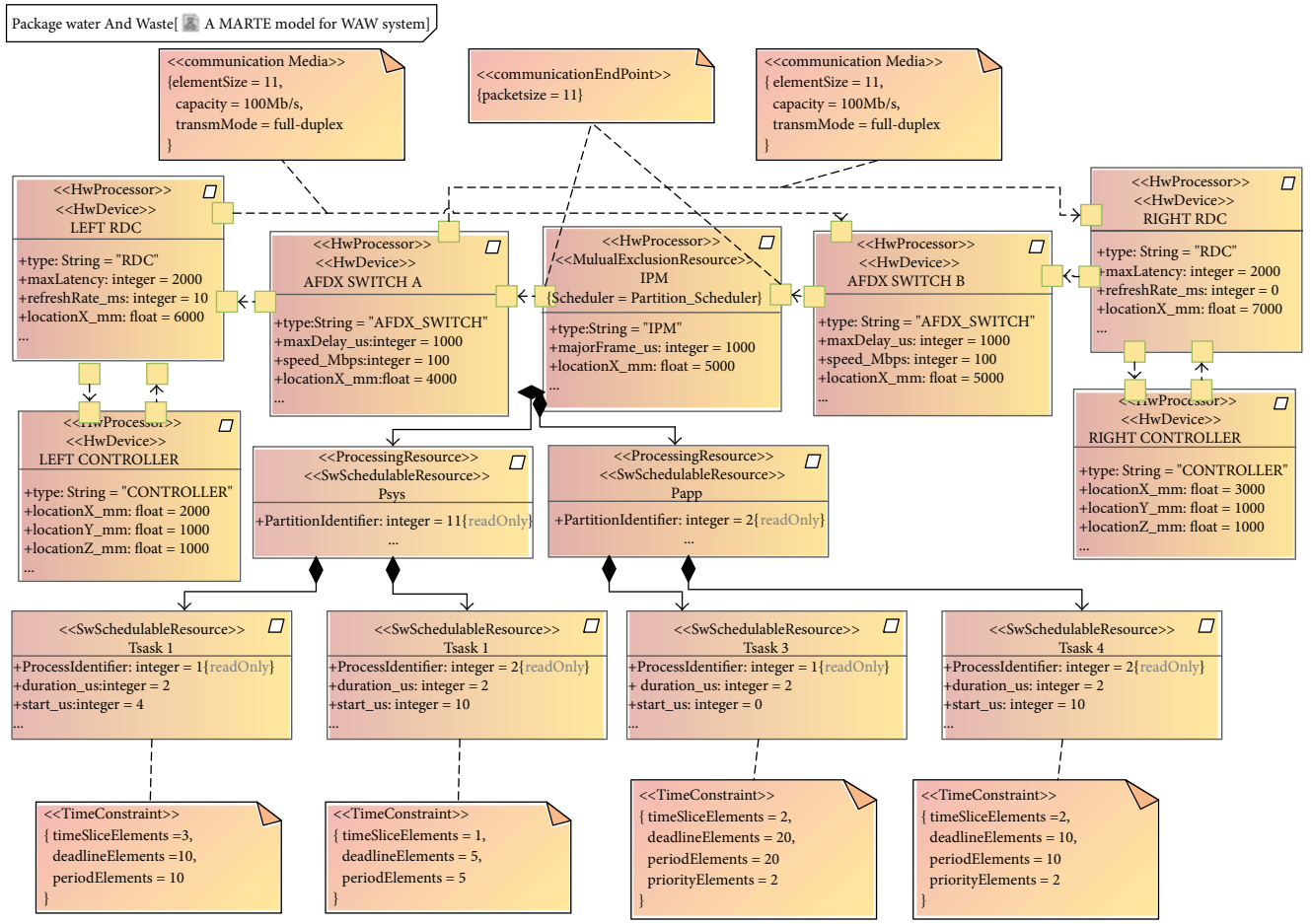


Figure 23: A MARTE model for WAW system.

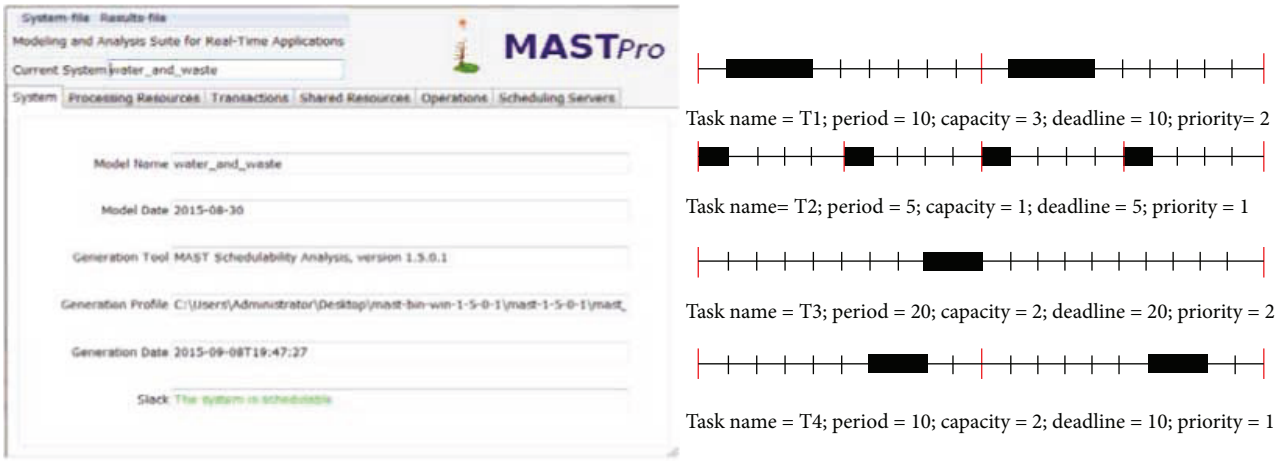


Figure 24: The analysis result of MAST for WAW model.

works, our works proposed a MARTE-based comprehensive methodology for IMA/ARINC653 system configuration modelling, formal verifications, and schedulability analysis and also designed a tool of CC653 which can be used both in the development stage and reconfiguration stage.

9. Conclusion

In this paper, we focus on modelling and verification of configuration information of IMA/ARINC653 system based on MARTE. Firstly, we define a semantic mapping from key

Table 16: Reschedule information of WAW system (time unit: ms).

Partitions	Slices (ms)	Offset	Cycle (ms)	Schedule strategy	Taskset			
					Task	Cycle	Execution time	Deadline
Papp	6	0	10	DMS	T1	10	3	10
					T2	5	1	5
Psys	4	0	10	RMS	T3	20	2	20
					T4	10	2	10
Papp2	2	0	10	DMS	T5	5	1	5
					T6	3	1	3

concepts of configuration (such as modules, partitions, memory, process, and communications) to components of MARTE element and propose a method for model transformation between XML-formatted configuration information and MARTE models. Then we present a formal verification framework for ARINC653 system configuration based on theorem proof techniques, including construction of corresponding REAL theorems according to the semantics of those key components of configuration information and formal verification of theorems for the properties of IMA, such as time constraints, spatial isolation, and health monitoring. After that, a special issue of schedulability analysis of ARINC653 system is studied. We design a hierarchical scheduling strategy with consideration of the characters of the ARINC653 system, and a scheduling analyzer MAST is called to implement hierarchical schedule analysis. Lastly, we design a prototype tool, called Configuration Checker for ARINC653 (CC653), and case studies of configuration of IMA system are provided for demonstration.

Future works include the following two aspects: (1) we need to further consider the modelling and verification of those secondary configuration information, and it is necessary to give formal definitions for those transformation rules between ARINC653 configuration data and MARTE models; (2) we need to apply our methods to a real avionics system development to obtain feedback and improvement and also further design formal verification techniques for MARTE model of configuration directly, without using third-party verification tools.

Conflicts of Interest

The authors declare no conflict of interest.

Authors' Contributions

Lisong Wang conceived and proposed the framework for configuration verification of IMA. Miaofang Chen performed the experiments and analyzed the results. Jun Hu designed the transformation rules and analyzed the case study.

Acknowledgments

The research work supported by National Basic Research Program of China (973 Program) nos. 2014CB744904 and 2014CB744903.

References

[1] G. R. Parr and R. Edwards, "Integrated modular avionics," *Air & Space Europe*, vol. 1, no. 2, pp. 72–75, 1999.

[2] C. B. Watkins and R. Walter, "Transitioning from federated avionics architectures to integrated modular avionics," in *2007 IEEE/AIAA 26th Digital Avionics Systems Conference*, Dallas, TX, USA, October 2007.

[3] A. E. E. Committee, *Avionics Application Software Standard Interface*, Aeronautical Radio, Annapolis, MD, USA, 1997.

[4] A. E. E. Committee, *Avionics Application Software Standard Interface Part 1-Required Services*, ARINC Document ARINC Specification, Annapolis, MD, USA, 2006.

[5] A. Rutle, A. Rossini, Y. Lamo, and U. Wolter, "A formal approach to the specification and transformation of constraints in MDE," *The Journal of Logic and Algebraic Programming*, vol. 81, no. 4, pp. 422–457, 2012.

[6] J. Hutchinson, M. Rouncefield, and J. Whittle, "Model-driven engineering practices in industry," in *ICSE '11 Proceedings of the 33rd International Conference on Software Engineering*, Waikiki, Honolulu, HI, USA, 2011.

[7] M. Vlter, T. Stahl, J. Bettin, A. Haase, and S. Helsen, *Model-driven soft-ware development: technology, engineering, management*, John Wiley & Sons, New York, NY, USA, 2013.

[8] L. Rierson, *Developing Safety-Critical Software: A Practical Guide for Aviation Software and DO-178c Compliance*, CRC Press, Boca Raton, USA, 2013.

[9] J. Rushby, "New challenges in certification for aircraft software," in *EMSOFT '11 Proceedings of the ninth ACM international conference on Embedded software*, pp. 211–218, Taipei, Taiwan, 2011.

[10] Y. Moy, E. Ledinot, H. Delseny, V. Wiels, and B. Monate, "Testing or formal verification: DO-178c alternatives and industrial experience," *IEEE Software*, vol. 30, no. 3, pp. 50–57, 2013.

[11] S. Graf, S. Gérard, Ø. Haugen, I. Ober, and B. Selic, "Modeling and analysis of real-time and embedded systems," in *International Conference on Model Driven Engineering Languages and Systems*, pp. 58–66, Montego Bay, Jamaica, 2006.

[12] M. Omg, *Modeling and Analysis of Real-Time and Embedded Systems Specification Version 1.1 Formal-11-06-02*, Object Management Group, Needham, MA, USA, 2011.

[13] M. G. Harbour, J. J. Gutiérrez, J. M. Drake, P. L. Martínez, and J. C. Palencia, "Modeling distributed real-time systems with MAST 2," *Journal of Systems Architecture*, vol. 59, no. 6, pp. 331–340, 2013.

[14] M. Gonzlez Harbour, J. G. Garca, J. P. Gutirrez, and J. D. Moyano, "Mast: modelling and analysis suite for real time applications," in *13th Euromicro Conference on, 2001 Real-Time Systems*, pp. 125–134, Delft, The Netherlands, 2001.

[15] L. Wang, Y. Zhou, M. Wang, and J. Hu, "A tool for IMA system configuration verification and case study," in *IISA 2017: Advances in Intelligent Systems and Interactive Applications*, pp. 254–260, Beijing, China, 2014.

[16] A. Ott, *System Testing in the Avionics Domain*, University of Bremen, Germany, 2007.

[17] M. Anderson and S. K. Shukla, "APECS code synthesis: extending ocarina for multi-threaded code synthesis from AADL models for safety critical applications," in *11th International Conference Networking, Sensing and Control (ICNSC)*, Miami, FL, USA, 2014.

[18] G. Lasnier, B. Zalila, L. Pautet, and J. Hugues, "Ocarina: an environment for AADL models analysis and automatic code generation for high integrity applications," in *International Conference on Reliable Software Technologies, Ada-Europe 2009*, pp. 237–250, Brest, France, 2009.

[19] S. Turki, E. Senn, and D. Blouin, "Mapping the MARTE UML profile to AADL," in *Proceedings of the 3rd International Workshop on Model Based Architecting and Construction of Embedded Systems (ACES-MB 2010)*, pp. 11–20, Oslo, Norway, 2010.

[20] K. Pohl, *Requirements Engineering: Fundamentals, Principles, and Techniques*, Springer Publishing Company, Berlin, 2010.

[21] M. O. Khan, M. Sievers, and S. Standley, "Model-based verification and validation of spacecraft avionics," in *Proceedings of the AIAA Infotech@ Aerospace Conferences*, Garden Grove, CA, USA, 2012.

[22] T. Weilkiens, *Systems Engineering with SysML/UML: Modeling, Analysis, Design*, Morgan Kaufmann, San Francisco, CA, USA, 2011.

[23] F. Kordon, J. Hugues, A. Canals, and A. Dohet, *Embedded Systems: Analysis and Modeling with SysML, UML and AADL*, John Wiley & Sons, Hoboken, NJ, USA, 2013.

[24] Á. Horváth and D. Varró, "Model-driven development of ARINC 653 configuration tables," in *2010 IEEE/AIAA 29th Digital Avionics Systems Conference (DASC)*, Salt Lake City, UT, USA, 2010.

[25] P. Hollow, J. McDermid, and M. Nicholson, "Approaches to certification of reconfigurable IMA systems," in *Proceedings 10th International Symposium of the International Coucil on Systems Engineering*, Minneapolis, Minnesota, 2000.

[26] H. Feng, S. Liru, and X. Huagang, "Two level task partition scheduling design in integrated modular avionics," *Journal of Beijing University of Aeronautics and Astronautics*, vol. 34, no. 11, pp. 1364–1368, 2008.

[27] Z. Tianran and X. Huagang, "Two-level hierarchical scheduling for hybrid real-time tasks in avionic systems," *Acta Aeronautica et Astronautica Sinica*, vol. 32, no. 6, pp. 1067–1074, 2011.