International Journal of Multimedia and Ubiquitous Engineering
Vol.9, No.4 (2014), pp.73-84
http://dx.doi.org/10.14257/ijmue.2014.9.4.08

Configuration Tool for ARINC 653 Operating Systems

Eu-Teum Choi[1], Ok-Kyoon Ha[2] and Yong-Kee Jun[1]

[1]*Department of Informatics, Gyeongsang National University*
[2]*Engineering Research Institute, Gyeongsang National University*

{*etchoi, jassmin, jun*}*@gnu.ac.kr*

Abstract

ARINC 653 Specification defines a standardized interface of real-time operating systems and an Application Executive (APEX) to develop the reliable applications for avionics based on Integrated Modular Avionics (IMA). The requirements of system platform based on ARINC 653 Standard are defined as configuration data and are integrated to the XML configuration file(s) in the real-time operating system. Unfortunately, existing configuration tools for integrating requirements do not provide checking the syntax errors of XML and verifying the integrity of input data for partitioning. This paper presents a configuration tool for ARINC 653 OS that consist of Wizard module which generates the basic configuration data for IMA based on XML Scheme of ARINC 653 Standard, XML and Partition Editor for the partitioning system of IMA, and Verification module which checks the integrity of input data and XML syntax with its visualization.

Keywords: *ARINC 653, IMA, partition OS, configuration, XML, verification*

1. Introduction

Avionic systems have been changing from Federated Systems to Integrated Modular Avionics (IMA) which purposes to reduce the weight of air-born systems and its power consumption by integrated management of the system. The ARINC 653 Specification [1-3] has been developed as a standardized interface definition of real-time operating system to simplify the development of IMA [4-9]. The ARINC 653 provides a strict and robust time and space partitioning [10-12] to guarantee the reliability of avionic systems by isolating the failures of the system. Configuration data [13] for the time and space partitions can be defined as the XML configuration file(s) that can be accessed only by system operation system.

System Integrator of air-born system usually employs editing tools to integrate all of the requirements of the system to the configuration data which bases on XML schema of the ARINC 653 standard. However, it is quite tedious activity to check the syntax errors of XML and the integrity of partition scheduling [14-16] during the integrating process. Moreover, existing configuration tools, which employ general purpose editing tool, do not provide any function to verify the syntax errors of XML and the integrity of input data for partitioning such as the allocation of resources and the scheduling of applications.

This paper presents a configuration tool for the ARINC 653 OS that consist of Wizard module which generates the basic configuration data for IMA based on XML Scheme of the ARINC 653 Standard, XML and Partition Editor for the partitioning system of IMA, and Verification module which checks the integrity of input data and

ISSN: 1975-0080 IJMUE

XML syntax with its visualization. XML Editor provides to check syntax errors of the XML configuration file based on the XML schema of the ARINC 653 standard, and Partition Editor verifies the integrity of partitioning with graphic user interface (GUI) by analyzing the input data of partitions.

The remainder of this paper is organized as follows. Section 2 introduces the concept of ARINC 653 for IMA platform and XML scheme for specifying ARINC 653 OS configuration. The design of our configuration tool which defines and verifies the configuration data of ARINC 653 OS is illustrated in Section 3, and Section 4 depicts the implementation of the configuration tool which consists of partition wizard, multi-page editor, outline view, and problem view. We conclude our argument of the paper in Section 6.

2. Background

Configuration data of ARINC 653 can be defined as the XML configuration file(s) to provide a strict and robust time and space partitioning system for IMA. This paper introduces the notion of the ARINC 653 for IMA platform and the XML Schema that defines the structure of the data needed to specify any ARINC 653 configuration.

2.1. ARINC 653 Standard

The ARINC 653 Specification [1[3] has been developed as a standardized interface definition of real-time operating system to simplify the development of Integrated Modular Avionics (IMA) [4-9]. This standard specifies an Application Executive (APEX) which provides services comprised of a set of fifty-one routines to enable the development of portable applications on an IMA platform. The main objective of the APEX is to provide a strict and robust time and space partitioning environment [10-12] allowing a processing unit known as *module* to host multiple applications independently in each partition. A module is managed by an operating system called *Module Operating System* (MOS).

The *temporal partitioning* is a strict time slicing which guarantees that only one application at a time is accessing the system resources including the processor according to a periodic scheduler. The *spatial partitioning* provides strict memory management by guaranteeing that a memory area allocated to a partition and its processes cannot be corrupted by another partition and its processes. Each partition is governed by a *Partition Operating System* (POS) which provides standard functionalities like scheduling, process management, and interaction with the MOS.

A partition consists of a set of concurrently executing processes, sharing access to the system resources with the help of a preemptive, priority-based scheduler. They are initialized at module starting time using an XML configuration file. Processes can communicate within a partition using shared buffers that provide a queue for message passing and blackboards that allow them to read, write and clear a single message. APEX also provides machine-dependent services such as interrupt handling, partition switching or inter-partition communication for interactions with the hardware. Figure 1 depicts the conceptual structure of ARINC 653.

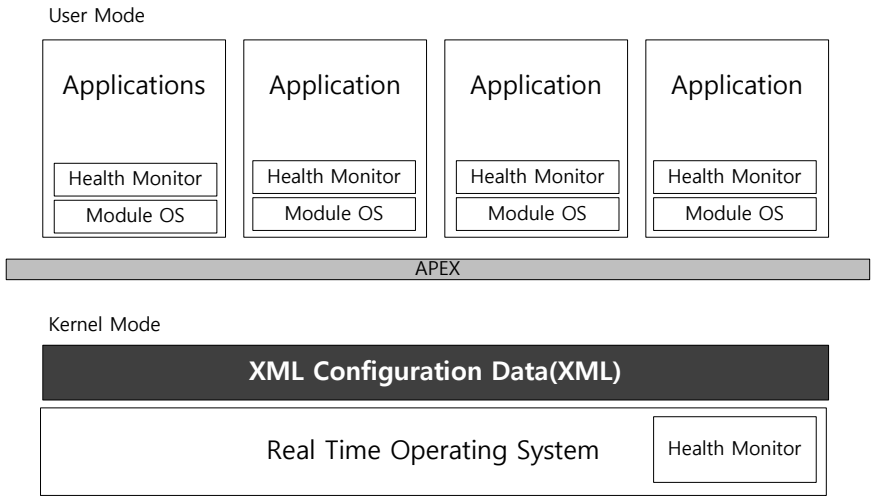


Figure 1. The conceptual architecture of ARINC 653

One of the most important features of ARINC 653 is indisputably its health monitor (HM) which has the responsibility to detect and provide recovery mechanisms for both hardware and software failures at the process, partition, and module levels. The objective of this hierarchical error handling framework is to contain and isolate faults before they propagate across the whole system. Typical faults detected by the HM include memory and processors errors, input/output and device errors or any other application level errors like division by zero, deadline misses or invalid system calls. If an application detects an error during its execution, it may report to the HM by invoking an APEX system call. As presented in the ARINC 653 specification, each error is characterized by an *error identifier* and a symbolic name. The specification defines a set of states in which a system can be at each point of time. These states can be for instance, partitions or modules initialization, process execution or error handling.

2.2. XML Schema for ARINC 653 OS

The configuration data [13] of ARINC 653 is static data areas that are accessed only by operating system, but they are not built as the part of the operating system. Therefore, they cannot be accessed directly by any application. The ARINC 653 Specification provides an extensible XML Schema that defines the structure of the data needed to specify any ARINC 653 configuration. The XML Schema consists of tagged pairs that describe the attributes of elements and their relationship to the whole.

Figure 2 shows the elements and their relationship used to define the XML Schema of ARINC 653. The square boxes represent the major elements and the attributes which are appeared by the text inside the box. From the figure, An ARINC 653 Module consist of seven major elements, one or more Partitions, a System HM table, a Module HM table, a Connection table, one or more Partition HM tables, one or more Partition Memory elements, and a Module Schedule. The requirements of the major elements are as follows:

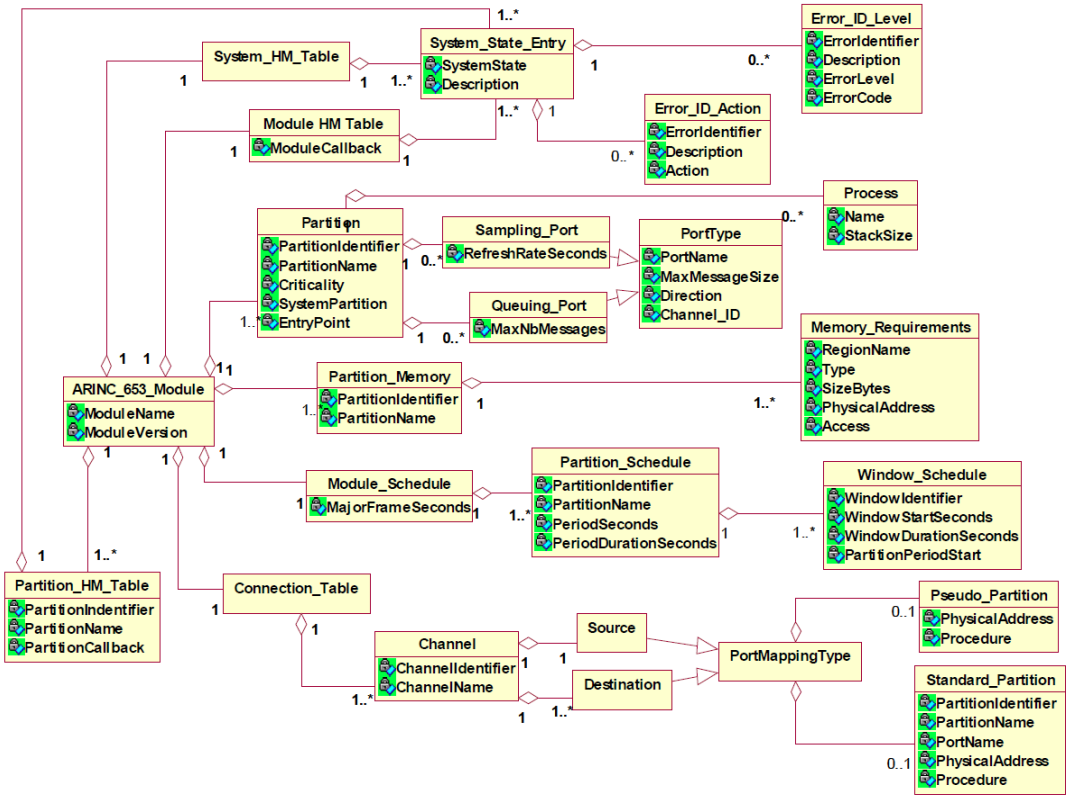


Figure 2. XML Schema Element Relationships (appear in [1])

- A Partition consist of five attributes, Partition Identifier, Partition Name, Criticality, System Partition, and Entry Point, and zero or more sampling and queuing port elements.
- Each sampling port or queuing port for Inter partition communications has a unique attribute and several common shared attributes defined by PortType.
- All of HM tables should be made up of one or more System state entry which includes Error ID Levels and Error ID Actions.
- The System state entry of the System HM tables is made up of one or more Error ID Levels. The id levels are defined as a distinguished module, partition, or process level errors. The process level errors are mapped to the predefined error codes.
- The Module and Partition HM tables are made up of one or more the System state entry that consists of one or more Error ID Actions. The Error ID Actions are the definition of error handlers what the system OS is to do when the error occurs in a specific system state.
- The Module Schedule of ARINC 653 is made up of Partition Schedule elements, and the attributes for the Module Schedule defines the major frame rate for each partition scheduling and the mapping of each partition to one or more partition windows.

- The Connection table defines the global channel identifications to their port connections. Each channel consists of one source and one or more destinations, where the source and destinations are either standard partitions or a pseudo partition.
- The definition of Partition Memory must characterize the regions of memory with proper codes or separated data that each partition has access to.

3. Design of Configuration Tool for ARINC 653 OS

The functional requirements for implementing avionics applications are integrated as the system configuration data with satisfying the requirements for the design of an air-craft. For guaranteeing safety critical executions of the applications, the ARINC 653 standard specifies the configuration tables to make sure of minimum functionalities, such as partition management, process management, time management, memory allocation, inter-partition communication, intra-partition communication, and health monitor, that can be provided by the system OS. This paper presents a configuration tool which considers the minimum functionalities for the ARINC 653 OS.

To create the immaculate configuration file by System Integrator, our configuration tool consider the following requirements.

- The configuration tool must provide a user interface (UI) to consist automatically partition schedules of ARINC 653.
- The tool must provide any method to verify the integrity of the partition schedules defined by user.
- The tool must consider the compatibility with integrated development environment (IDE) for avionics software.
- The created configuration files (or XMLs) by the tool must apply the ARINC 653 OS.

Figure 3 depicts overall architecture of our configuration tool that consider above requirements. This tool consists of three main parts, configuration part for creating and maintaining the partitions of the ARINC 653 OS, verification part which checks the syntax errors of the configuration data and verifies the integrity of the partition requirements, and reporting part to provide related issues to the partition configurations.

Configuration part consists of three modules, Partition Wizard, Partition Editor, and XML Editor. Partition Wizard defines the configuration data and creates a XML configuration file for partitions based on the system requirements and the configuration tables of the ARINC 653 standard. By this module, memory areas that can be accessed by each partition are allocated, and schedule methods are selected to archive robust execution of partitions. Partition Editor provides the summary of partition configurations from the XML file to modify the configurations including partition memory areas, partition schedules, and other partition properties. XML Editor is a text editing module considering the XML Schema of the ARINC 653 standard.

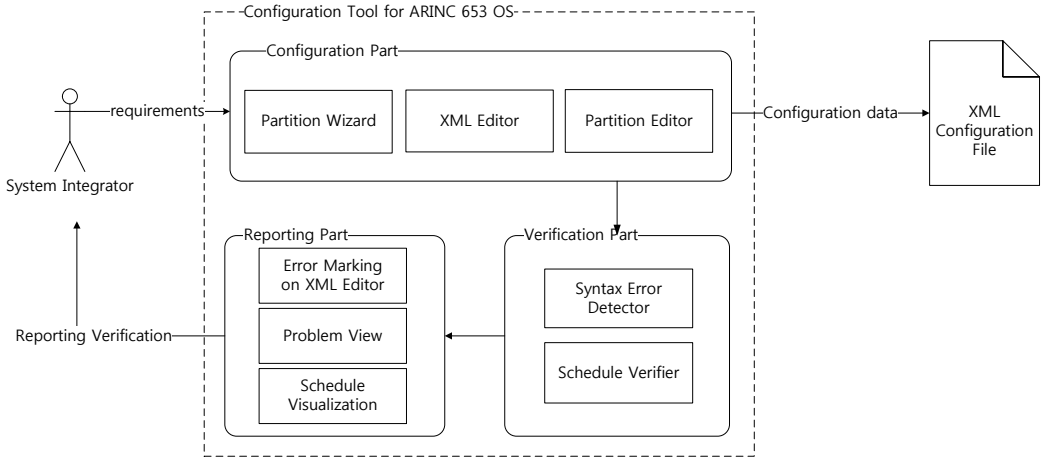


Figure 3. The architecture of our configuration tool

The verification part consists of the Syntax Error Detector and Schedule Verifier modules to verify the XML configuration file. The Syntax Error Detector checks existing errors in the context of the XML file by applying the XML schema. This module is internally behaved as an error handler of the XML Editor module. Schedule Verifier is a module for verifying the integrity of partitions by analyzing three kinds of scheduling configurations, such as module schedules, partition schedules, and window schedules. Reporting part consists of three kinds of error reporting modules, Error Marking, Problem View, and Schedule Visualization. The Error Marking and The Problem View indicate the syntax errors fixed by the Syntax Error Detector, and The Schedule Visualization graphically represents the status of scheduled partitions verified by Schedule Verifier.

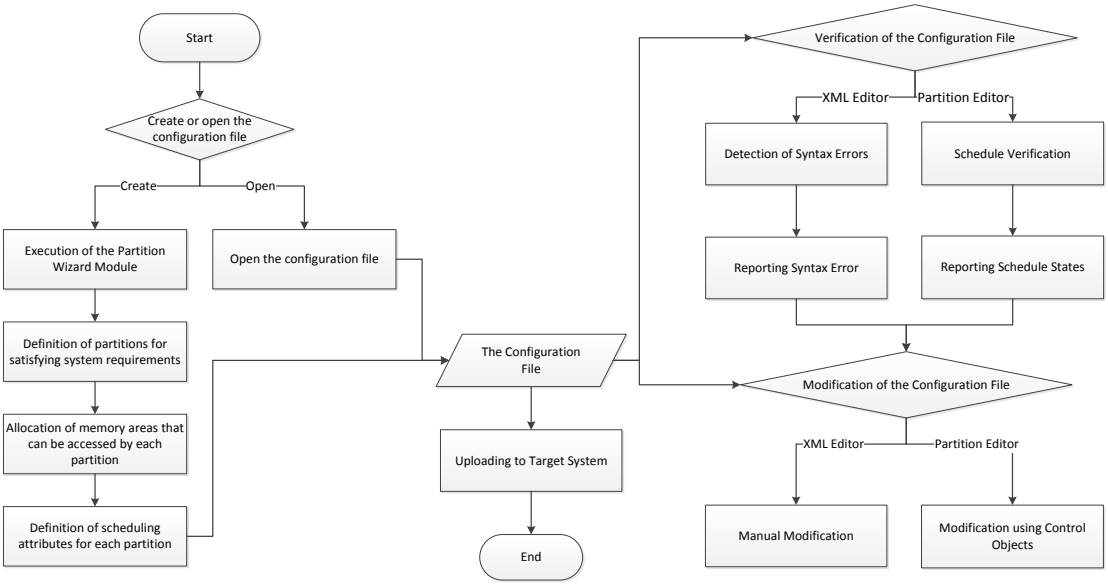


Figure 4. The processes of our configuration tool

Figure 4 depicts the processes of our configuration tool. The configuration files are created by the partition wizard module with three steps. These three steps define partitions of OS for

satisfying system requirements and scheduling attributes for each partition, and allocate memory areas that can be accessed by each partition. The configuration files created and opened by the partition wizard can be verified or modified by using the XML Editor or the Partition Editor. The XML Editor provides the way to modify manually the configuration files, and the Partition Editor provides an Interface to modify automatically the configuration files only by controlling related objects. To verify the reliability of the configuration files, the syntax errors are detected and reported by the XML Editor, and the validation of partition scheduling for ARINC 653 OS is checked by the Partition Editor through the visualization of the scheduling. The final configuration files that were completed three kinds of procedure (*i. e.*, automatic creation, manual or automatic modification, and automatic verification) are uploaded to a target system by the integrator of the system.

With our configuration tool, it can guarantee that the integrating system and application requirements are became more correct, because it automatically produces only verified configuration data for ARINC 653 OS. Thus, the tool is useful for the system integration of any air-craft as well as its facility.

4. Implementation

We implemented the configuration tool as a plug-in of the Eclipse Integrated Development Environment (IDE) [17-18]. For the implementation, we used Java programming language and the Eclipse Standard Development Kit (SDK) 3.8.1 for supporting Java Development Kit (JDK) 1.7, and carried on a system with Intel i5 3.4GHz CPU and 4GB main memory under Windows 7 OS. Figure 5 shows our configuration tool which consists of partition wizard, multi-page editor, outline view, and problem view for defining and verifying configuration data. In our implementation, we added the outline view that forms an overview list to provide the circumstantial information of important configuration data, such as partitions, partition memories, module schedule, and each HM table, and the problem view to provide the error information in the XML configuration file.

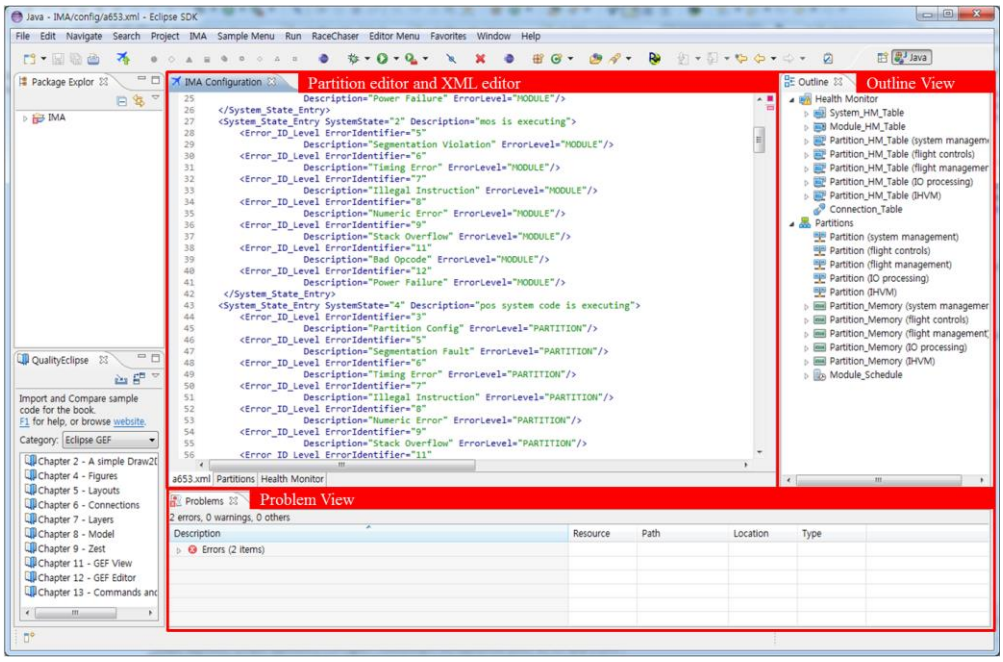
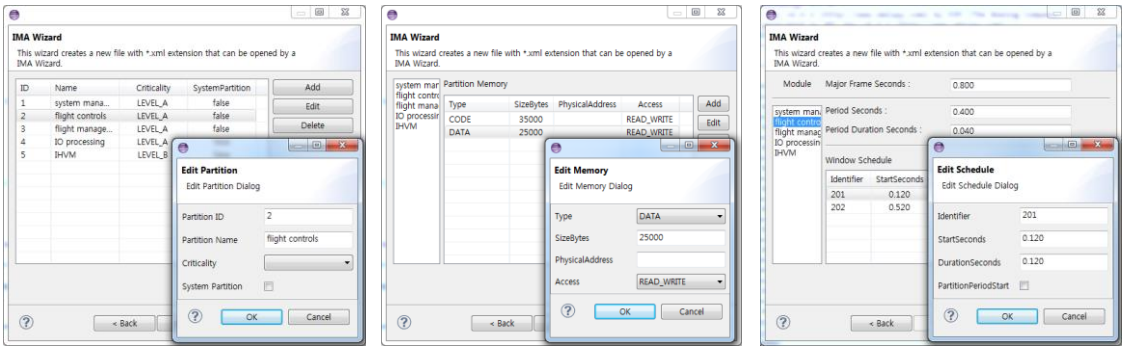


Figure 5. Overview of our configuration tool for ARINC 653 OS

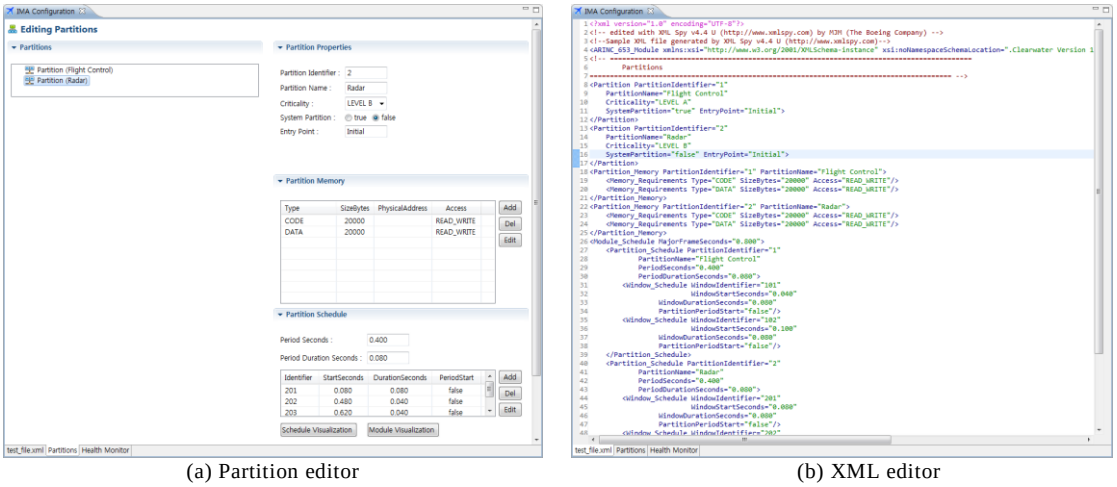


(a) Partition page (b) Partition memory page (c) Partition schedule page

Figure 6. The Partition Wizard of configuration tool

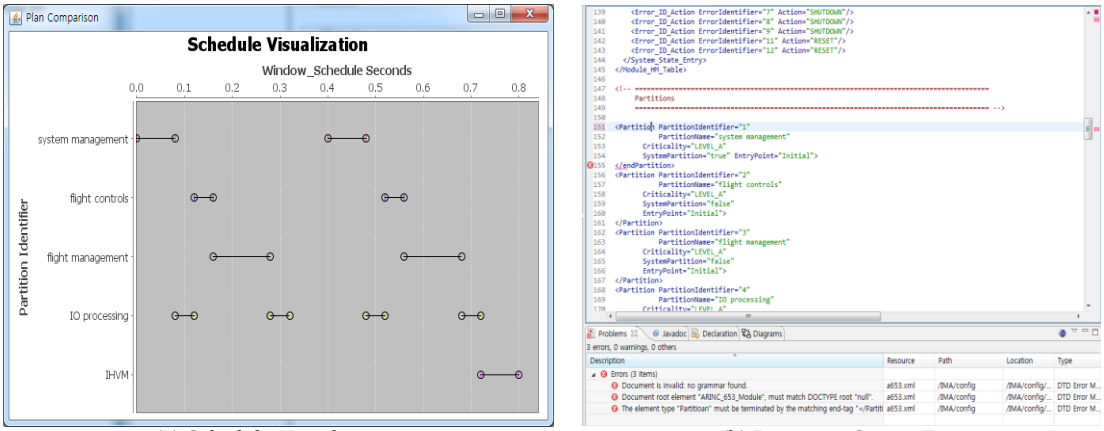
The Partition Wizard module is used in order to generate a XML configuration file through a sequenced set of pages. The sequenced pages of this module provide three steps to define partitions, partition memories, and partition schedules. Each page of the Partition Wizard module appears in Figure 6. Figure 6(a) is the first step of the wizard to define partitions for satisfying system requirements, and (b) is the second step to allocate memory areas that can be accessed by each partition. The final step, Figure 6(c), defines scheduling attributes for each partition and its windows. A XML configuration file will be recorded into an appointed directory after all of three steps.

The multi-page editor consists of the Partition Editor and the XML Editor to modify the XML configuration file which generated by Partition Wizard. Figure 7 shows each of the Partition Editor and XML Editor. The Partition Editor, Figure 7(a), summarizes the configuration data related to partitioning information which is defined in the XML file, and provides control objects to modify the configuration data. The XML Editor, Figure 7(b), shows source code of the XML file. This editor is more specific than general text editor for the configuration data, because this considers the XML schema of the ARINC 653 standard.



(a) Partition editor (b) XML editor

Figure 7. Partition Editor and XML Editor of our configuration tool



(a) Schedule Visualization (b) Reporting Syntax Error

Figure 8. Reported results for the analysis of errors in a configuration

Figure 8 shows three modules of the reporting part, the Error Marking, the Problem View, and the Schedule Visualization. Figure 8(a), the Schedule Visualization module, graphically verifies the integrity of partitions by analyzing three kinds of scheduling configurations, such as module schedule, partition schedule, and windows schedule. This module represents scheduling problems which analyzed by the Schedule Verifier module of the Verification Part. Figure 8(b) shows how both the Error Marking and the Problem View present the syntax errors of the XML file which are located by the Syntax Error Detector. The Error Marking module indicates the offending text with a squiggly under-line and an error icon on the left-hand ruler for the XML editor, and the Problem view display the details of the errors.

5. Conclusion

ARINC 653 Specification defines a standardized interface of real-time OS and an APEX to develop the reliable applications for avionics based on IMA. The requirements of system platform based on ARINC 653 Standard are defined as configuration data and are integrated to XML configuration file(s) in the real-time OS. Unfortunately, existing configuration tools for integrating requirements do not provide checking the syntax error of XML and verifying the integrity of input data for partitioning.

This paper presented a configuration tool which consists of three main parts for ARINC 653 OS. The XML configuration file can be easily generated and easily modified with our configuration tool. Moreover, the tool graphically represents each scheduling errors and reports syntax errors in the XML configuration file. Therefore, our configuration tool is helpful to integrate the system requirements of applications for avionics based on IMA as well as verifying the integrity of partitions. Future work includes improving the tool for supporting health managements of ARIINC 653, such as HM tables of each level and error handling.

Acknowledgements

This research was supported by the Korea Evaluation Institute of Industrial Technology (KEIT) under “the Development of Verification System for Mid-sized IMA Project” (10043591) funded by the Ministry of Trade, Industry & Energy and was also supported by the MSIP(Ministry of Science, ICT & Future Planning), Korea, under the “SW master's course of a hiring contract” support program (NIPA-2013-HB301-13-1004) supervised by the NIPA (National IT Industry Promotion Agency).

References

[1] Airlines electronic engineering committee (AEEC), Avionics Application Software Standard Interface - ARINC Specification 653 – part 1. (Supplement 2- Required Services), ARINC inc., **(2006)**.
[2] P. J. Prisaznuk, "ARINC 653 role in Integrated Modular Avionics (IMA)", Proceedings of the 27th Digital Avionics Systems Conference, **(2008)** October 26-30; St. Paul, MN.
[3] S. Santos, J. Rufino, T. Schoofs, C. Tatibana and J. Windsor, "A portable ARINC 653 standard interface", Proceedings of the 27th Digital Avionics Systems Conference, **(2008)** October 26-30; St. Paul, MN.
[4] O. -K. Ha, G. M. Tchamgoue, J. -B. Suh and Y. -K. Jun, "On-the-fly Healing of Race Conditions in ARINC-653 Flight Software", Proceedings of the 29th Digital Avionics Conference, **(2010)** October 3-7; Salt Lake City, UT.
[5] G. M. Tchamgoue, O. -K. Ha, K. -H. Kim and Y. -K. Jun, "A Framework for On-the-fly Race Healing in ARINC-653 Applications", International Journal of Hybrid Information Technology, SERSC, vol. 4, no. 2, **(2011)**, pp. 1-12.
[6] P. J. Prisaznuk, "Integrated modular avionics", Proceedings of the IEEE 1992 National, **(1992)** May 18-22; Dayton, OH.
[7] T. Schoofs, S. Santos, C. Tatibana and J. Anjos, "An integrated modular avionics development environment", Proceedings of the 28th Digital Avionics Systems Conference, **(2009)** October 23-29; Orlando, FL.
[8] P. Parkinson and L. Kinnan, "Safety-critical software development for integrated modular avionics", Embedded System Engineering, vol. 11, no. 7, **(2003)**.
[9] C. B. Watkins and R. Walter, "Transitioning from federated avionics architectures to Integrated Modular Avionics", Proceedings of the 26th Digital Avionics Systems Conference, **(2007)** October 21-25; Dallas, TX.
[10] N. Diniz and J. Rufino, "ARINC 653 In Space", Dasia 2005, EUROSPACE, Edinburgh, Scotland, **(2005)**.
[11] S. Han and H. Jin, "Full virtualization based ARINC 653 partitioning", Proceedings of the 30th Digital Avionics Systems Conference, **(2011)** October 16-20; Seattle, WA.
[12] J. Rufino, S. Filipe, M. Coutinho, S. Santos and J. Windsor, "ARINC 653 interface in RTEMS", Proceedings of the Data Systems In Aerospace, **(2007)** June; Napoli, Italy.
[13] A. Hrvath and D. Varro, "Model-Driven Development of ARINC 653 Configuration Tables", Proceedings of the 29th Digital Avionics Systems Conference, **(2010)** October 3-7; Salt Lake City, UT.
[14] Y.-H. Lee, D. Kim, M. Younis and J. Zhou, "Scheduling tool and algorithm for integrated modular avionics systems", Proceedings of the 19th Digital Avionics Systems Conference, **(2000)** October 7-13; Philadelphia, PA.
[15] L. Kinnan, J. Wlad and P. Rogers, "Porting applications to an ARINC 653 compliant IMA platform using VxWorks as an example", Proceedings of the 23rd Digital Avionics Systems Conference, **(2004)** October 24-28.
[16] S. H. VanderLeest, "ARINC 653 hypervisor", Proceedings of the 29th Digital Avionics Systems Conference, **(2010)** October 3-7; Salt Lake City, UT.
[17] A. Bolour, "Notes on the eclipse plug-in architecture", http://www.eclipse.org/articles/Article-Plug-in-architecture/plugin_architecture.html.
[18] C. Eric and D. Rubel, "Eclipse: building commercial-quality plug-ins", vol. 2, **(2006)**.

Authors

Eu-Teum Choi

He received the BS degree in Department of Applied Life Chemistry from Gyeongsang National University (GNU), South Korea. He is currently enrolled MS degree from the Department of Informatics in GNU. He worked as an engineer of IT department in Korea industry for two years. His research interests include distributed programming and its debugging, embedded system programs for avionics, and dependable systems.

Ok-Kyoon Ha

He received the BS degree in Computer Science under the Bachelor's Degree Examination Law for Self-Education from National Institute for Lifelong Education, and the MS and PhD degree in Informatics from Gyeongsang National University (GNU), South Korea. He is now a Research Fellow of Engineering Research Institute (ERI) in GNU. He worked as the manager of IT department in Korea industry for several years. His research interests include parallel/distributed programming and its debugging, embedded system programs, and dependable systems. Dr. Ha is a member of Korean Institute of Information Technology (KIIT) and Korea Institute of Information Scientist and Engineers (KIISE).

Yong-Kee Jun

He received the BS degree in Computer Engineering from Kyungpook National University, and the MS and PhD degree in Computer Science from Seoul National University. He is now a full professor in the Department of Informatics, Gyeongsang National University, where he had served as the first director of GNU Research Institute of Computer and Information Communication (RICIC), and as the first operating director of GNU Virtual College. He is now the head of GNU Computer Science Division and the director of the GNU Embedded Software Center for Avionics (GESCA), a national IT Research Center (ITRC) in South Korea. As a scholar, he has produced both domestic and international publications developed by some professional interests including parallel/distributed computing, embedded systems, and systems software. Prof. Jun is a member of Association for Computing Machinery (ACM) and IEEE Computer Society.