ASSESSMENT OF ARINC 653 AND ITS POTENTIAL APPLICABILITY IN RSAF CONTEXT

KHOO Wee Tian

ABSTRACT

In all software development standards, every piece of software needs to go through regression testing when bugs are discovered and fixed during software development. The difficulty has been in conducting the change impact analysis of a software change and determining the adequate amount of regression testing required to ensure that the software has been tested adequately and that no unintended changes have been introduced. ARINC[1] 653 was developed by the aviation industry to address this concern. This article assesses the features of ARINC 653 and how it makes a change impact analysis easier. This article also assesses ARINC 653's potential applicability in the Republic of Singapore Air Force context.

Keywords: ARINC 653, software regression testing, partitioning, real-time operating system, avionics

INTRODUCTION

Software is present in almost every piece of equipment used today. In the case of software in safety critical equipment or in systems such as aircraft, the safe and proper functioning of the software are important requirements. To this end, software development standards have evolved steadily over the years with more and more emphasis placed on the certification of safety critical software. One of the most prevalent safety critical software development standards in use for aircraft today is RTCA[2]/DO-178B Software Considerations in Airborne Systems and Equipment Certification. This standard has been adopted by the Federal Aviation Administration (FAA) as well as the European Aviation Safety Agency (EASA), where it is also known as EUROCAE ED-12B, to certify software used in commercial aircraft.

Repeated testing is required for every piece of software when bugs are uncovered and fixed during the various phases of software development. This is known as regression testing. In general, the methodology adopted by software developers and software certification bodies to determine the scope of regression testing involves conducting some form of change impact analysis. However, conducting such an analysis requires a good knowledge of the software architecture and the design of the software of the specific aircraft or system which resides with the software developer, usually the original equipment manufacturer (OEM). Without this knowledge, software certification bodies – not to mention buyers of such equipment or systems such as the Republic of Singapore Air Force (RSAF) – would find it quite difficult to perform such analyses especially for equipment or systems with legacy software which are generally less modular in nature. This often results in a regression test scope which is much more conservative, hence requiring more time, effort and resources.

This is a major concern for the aviation industry as more and more aircraft are being designed with increasingly more software content leading to time-consuming and costly software testing and regression testing efforts. To address this, aviation experts have come together to develop the ARINC 653 specification Avionics Application Software Standard Interface for a Real Time Operating System (RTOS) interface definition (Aeronautical Radio, Incorporated, 2006-2012) as an enabler in the development of Integrated Modular Avionics (IMA) (Prisaznuk, 2008). In the years since the development of this specification, software based on ARINC 653 has been implemented in both the Airbus A380 and the Boeing 787, achieving certification by EASA and FAA respectively.

This article evaluates the features of ARINC 653 and explores how ARINC 653 makes a change impact analysis easier, thereby allowing the scope of the regression tests to be more deterministic and reduced. This article also looks at how ARINC 653 may be potentially applied in the context of the RSAF.

WHAT IS ARINC 653?

Background

The aviation industry developed ARINC 653 as a standardised RTOS interface definition between the RTOS of an avionics computer resource and the application software. This benefits both the software developers as well as the hardware platform suppliers. This standardisation effort was sponsored by the airline user community, through ARINC, and involved many interested parties.

To meet the software certification requirements of RTCA DO-178B / EUROCAE ED-12B, software specialists convened and identified the following specific needs for avionics equipment:

a) **Safety-critical –** as defined by FAR[3] Part 25.1309

b) **Real-time –** responses must be within a prescribed time period

c) **Deterministic –** the results are predictable and repeatable

ARINC 653's RTOS interface definition supports these needs. The standardised RTOS interface establishes a known interface boundary for avionics software development, thus allowing independence of the avionics software applications and the underlying core software. This enables concurrent development of application software components and the RTOS. The same applications can also be made portable among ARINC 653-compliant platforms. The standard RTOS interface definition also enables the underlying hardware platform and the core software to evolve independently of the software applications (Prisaznuk, 2008).

Partitions

The key concept introduced by ARINC 653 is the idea of a well-defined partition. It creates a kind of container for a software application and guarantees that execution of the application is both spatially – in terms of memory allocation – and temporally isolated. The partitions are divided into two categories: application partitions and system partitions. The application partitions execute avionics applications. They exchange data with the environment by means of a specific interface called APEX (Application/Executive). The system partitions are optional and their main role is to provide services that are unavailable in APEX, such as device drivers or fault management. Figure 1 shows a generic logical structure of the RTOS (Samolej, 2011).

In essence, this RTOS structure ensures that each partition in which an application is running has its allocated processing time and memory. The occurrence of issues such as lockups,

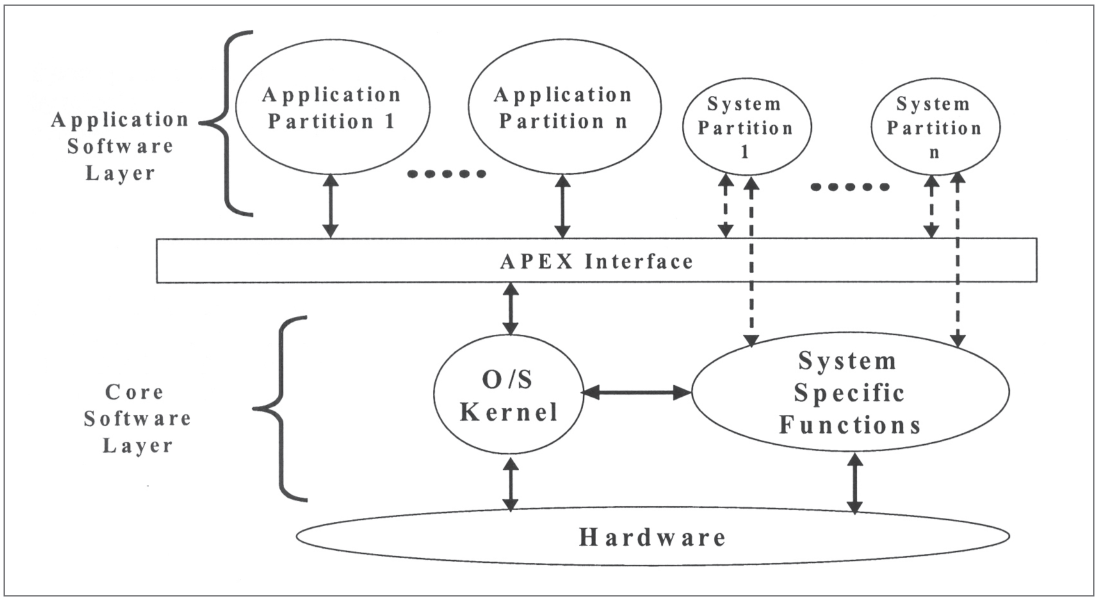

Figure 1. Generic ARINC 653 logical RTOS structure (Aeronautical Radio, Incorporated, 2010)

freezes, endless loopings and overruns within one partition will not prevent another partition from operating. However, it is still possible for running partitions to suffer from “data starvation” due to the faulty partitions ceasing to deliver the necessary data (Lagoy & Gauer, n.d.).

Processes that belong to one partition may be executed concurrently. A separate partition-level scheduling algorithm is responsible for this. A major time frame, activated periodically, is defined for each module. Each partition in this module receives one or more time partition windows to be executed within this major time frame, and this is illustrated in Figure 2. Generally, time partition windows constitute a static cyclic executive. Real-time tasks executed within the partition can be scheduled locally according to priority-based policy. The order of the partition windows is defined in a separate configuration record of the system.

Inter-application, or inter-partition, communication is based on the concept of ports and channels. The applications do not have information on which partition the data is sent to. The applications send and receive data via ports that are connected virtually by channels (Samolej, 2011).

Hardware-Software Module Architecture

The ARINC 653 specification also includes recommendations on micro-processor module architecture for the dedicated RTOS. The general diagram of this architecture is shown in Figure 3. Each module includes one or more micro-processors. The hardware structure may require some modification of the core operating system to allow the system to integrate with the hardware though this modification is not required for the APEX interface.

All processes that belong to one application partition (real-time tasks) must be executed on one micro-processor. It is forbidden to allocate them to different micro-processors within the module or split them between modules. The application program should be portable between processors within the module without any modifications of the interface to the core operating system.

The module also has an important element called the Health Monitor (HM). HM is an operating system component that monitors hardware, operating system and application faults and failures. Its main task is to isolate faults and prevent failure

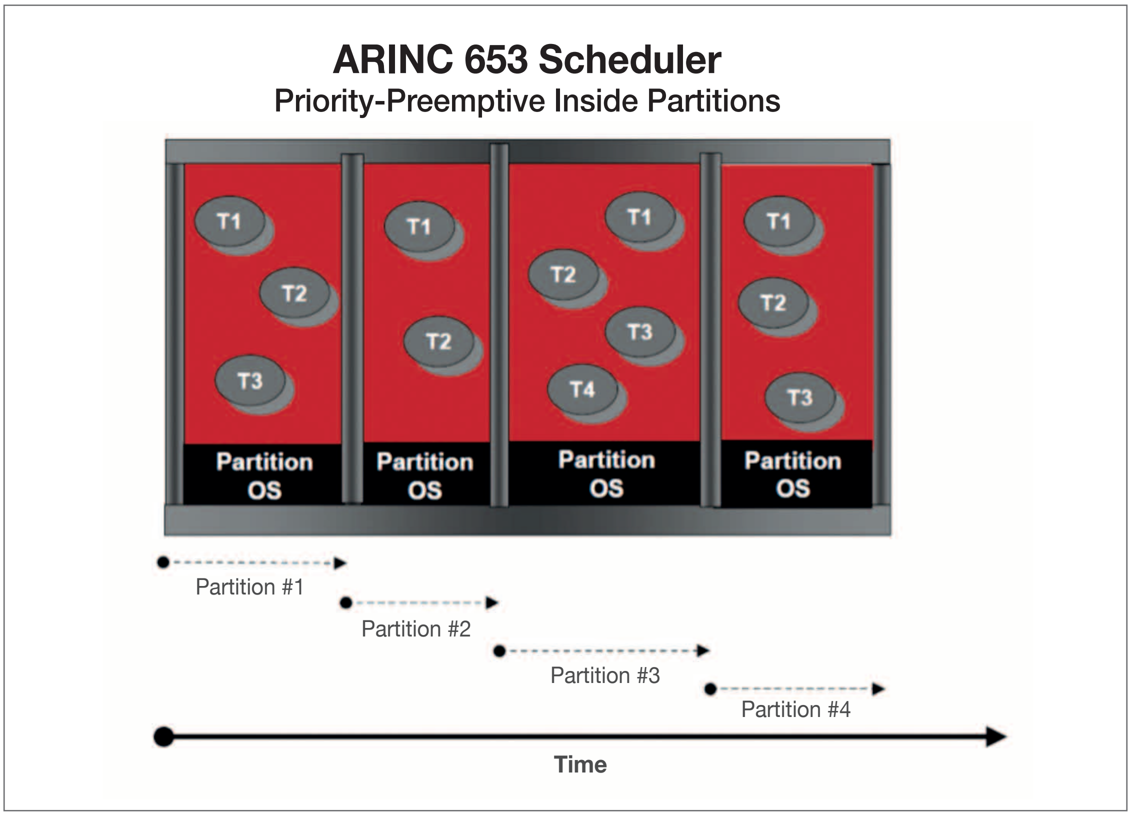

Figure 2. Example of ARINC 653 scheduler (Wind River Inc., 2008)

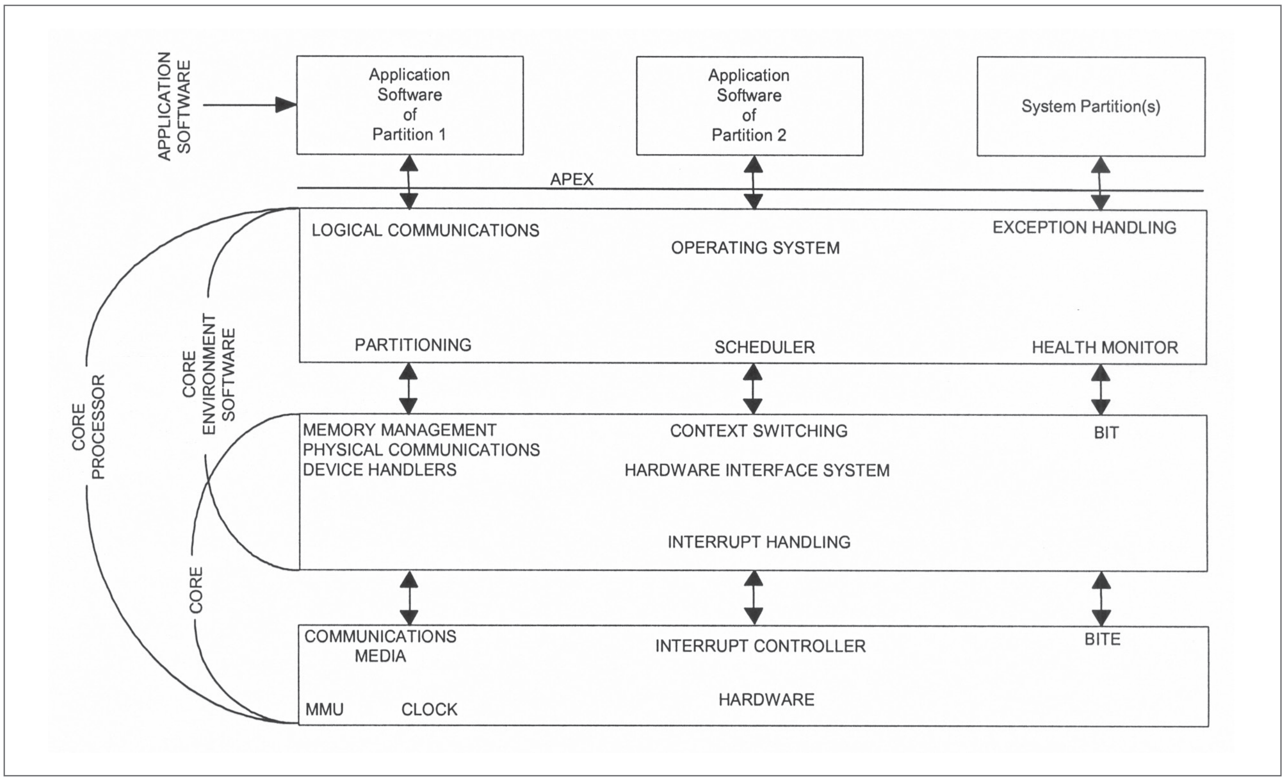

Figure 3. Hardware-software architecture of typical ARINC 653 module (Aeronautical Radio, Incorporated, 2010)

propagation. For example, the HM can restart a partition when it detects an application fault.

Applications, or partitions, can be developed by different providers. The integrator of all the applications and modules collects data regarding resources, timing constraints, communications ports and exceptions defined in each partition. The collected data is then transferred to configuration records. The configuration record for each module is an XML document interpreted during compilation and consolidation of the software (Samolej, 2011).

APEX Interface

APEX interface definition is the main part of ARINC 653. The APEX specifies how to create platform-independent software that fulfils ARINC 653 requirements. The main components are:

a) Partition Management – The APEX interface provides a separate set of functions enabling the user to determine the actual partition mode and change it. The application may start the partition after the creation of all application components. It is also able to obtain the current partition execution status. Inter-partition health monitoring procedures can stop or restart the partition.

b) Process Management – The application can be constructed as a set of soft or hard real-time processes, scheduled according to priorities. APEX process management services can:

i. create process and collect process status or ID
ii. start, stop, suspend or resume the process
iii. prevent process pre-emption
iv. change the process priority

c) Time Management – The APEX manages both aperiodic and periodic processes. Periodic processes are activated regularly. Additionally, a separate parameter called “time capacity” is attached to each process. It defines the time frame within which each task must finish computations. When a process is started, the deadline is set as the current time plus time capacity. The operating system periodically checks whether the process is completed within the allotted time. Each process has a priority, and during any rescheduling event, the operating system always selects the highest priority process that is in the “ready state” for execution.

d) Memory Management – There are no memory management services in APEX because partitions and their associated memory spaces are defined during system configuration. The whole memory is allocated statically to partitions and processes before the partition or application starts.

e) Inter-partition Communication – Inter-partition communication is based on queuing port and sampling port communication units. The queuing port provides the inter-partition message queue, whereas the sampling port shares variables between the ports. During system integration, the ports are connected by means of channels defined in system configuration tables. The ports communicate with other partitions or device drivers within the module or exchange data between modules.

f) Intra-partition Communication – The synchronisation of processes belonging to one partition may be achieved by the appropriate application of counting semaphores and events. The inter-process communication within the partition (intra-partition communication) is implemented by means of APEX buffers (shared message queues) and APEX blackboards (shared variables). It is possible to define a time-out period within which a process waits for data if the data is not immediately available. The process may be blocked for the specified time only.

g) Health Monitoring – The ARINC 653 HM is an advanced exception handling engine. Three error levels are defined:

i. process level which affects one or more processes in the partition
ii. partition level with only one partition affected
iii. module level which affects all partitions within the module

The health management framework is hierarchical, in which errors that cannot be handled at the level where they occur are propagated up to the next level. Both partition level and module level errors are handled by a set of procedures installed by the system integrator. The process level errors can be handled by the application software developer. A separate sporadic task called "error handler" can be registered for each of the partitions. When the HM detects an error at the process level, it calls the error handler. The handler recognises the error and, depending on the error, can:

i. log it
ii. stop or restart the failed process
iii. stop or restart the entire partition
iv. invoke the registered error handler process for the specific error code (Samolej, 2011)

Addressing DO-178B Certification

Examples of RTOSes that meet ARINC 653 requirements are LynuxWorks LynxOS-178 RTOS, LynuxWorks LynxOS-SE RTOS, Sysgo PikeOS and Wind River VxWorks 653 (Samolej, 2011).

However, besides an ARINC 653-compliant RTOS, the RTOS should come with DO-178B qualified development tools such as compilers and qualified verification tools up to the required design assurance level of the software to ease the DO-178B certification effort for the software. These tools should ideally also be able to provide evidence for the DO-178B certification.

ASSESSMENT OF ARINC 653

To be able to reduce the scope of regression testing, testing should ideally only be done for the parts of the software that are changed as well as the parts which are unchanged but are affected indirectly by the change. However, in many cases, especially in non-modular software, regression testing of other areas of the software would also be conducted to ensure that unintended changes have not been introduced into those areas. This approach often results in a large regression test scope.

For software using ARINC 653, the specification dictates that software applications are written as "partitions". The ARINC 653 RTOS ensures that the partitions are spatially (i.e. in terms of memory space allocation) and temporally (i.e. in terms of scheduling processor usage) separated. ARINC 653 specifies that each partition be allocated a predetermined area of memory. These unique memory areas are identified based on the requirements of the individual partitions and can vary in size and access rights. At most, only one partition has write access to any particular area of memory. Memory partitioning is ensured by prohibiting memory accesses outside of a partition's defined memory area. This partition memory allocation is defined in configuration tables. For temporal separation, the partitions are scheduled on a fixed, cyclic basis by the ARINC 653 RTOS. The RTOS maintains a major time frame of fixed duration, which is repeated periodically throughout the module's runtime operation. Partitions are allocated one or more predefined partition windows within this major time frame. See Figure 4 for an example of partition windows scheduling. The order of partition activation is defined in configuration tables. This provides a deterministic scheduling methodology to ensure that individual partitions have uninterrupted access to common resources during their assigned time periods (Aeronautical Radio, Incorporated, 2010). Any issues within one partition

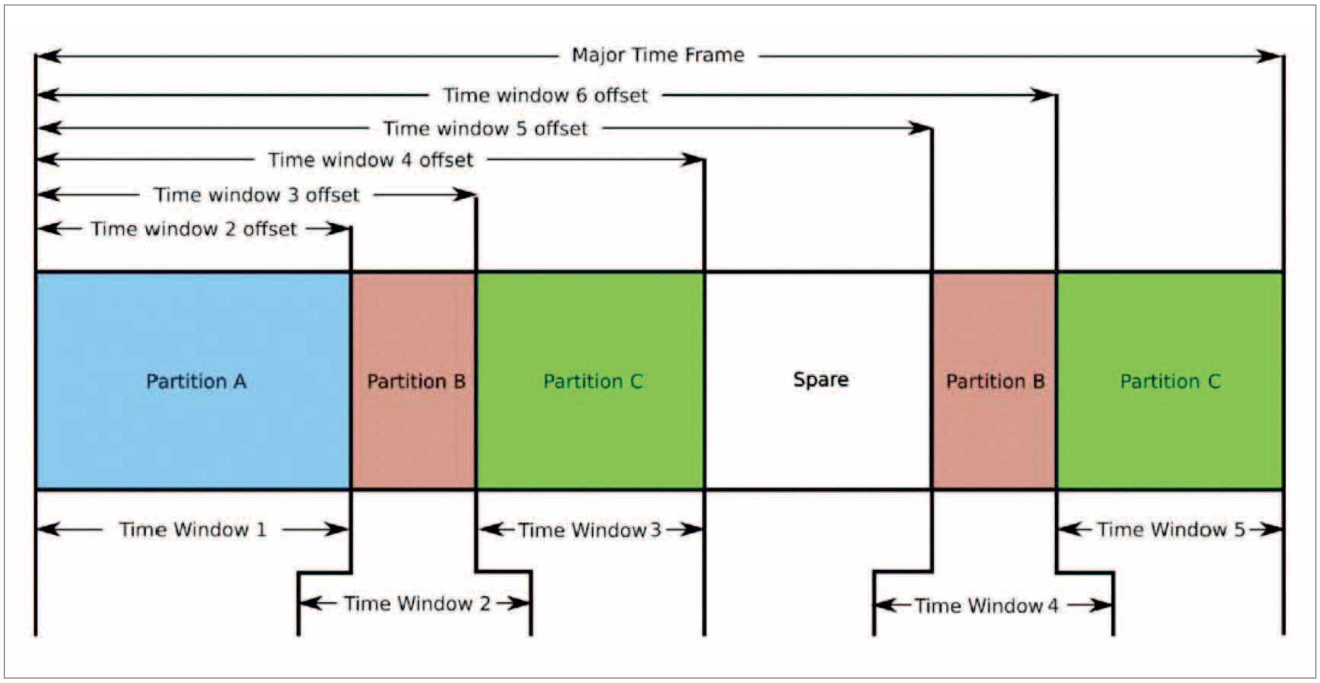

Figure 4. Example of ARINC 653 partition time windows (Aeronautical Radio, Incorporated, 2010)

will not prevent another partition from operating. Hence, this partitioning feature of ARINC 653 ensures the modularity of the software application in each partition with respect to the software applications in other partitions. In essence, these features allow each partition to be likened to and treated as individual Line Replaceable Units (LRUs) in a federated avionics architecture.

In ARINC 653, inter-partition communications is implemented via ports and channels. The data exchanged are called messages. A port is an access point in the partition to send or receive messages from other partitions either within the same module or from other modules. Channels connect the sending and receiving ports. Ports and channels are virtual and the underlying transport layer can be implemented via memory sharing, physical bus communications or other means. From the perspective of a partition, APEX communication services support the transmission and receipt of messages via ports and are the same regardless of the system boundaries crossed by the communicated message. The ARINC 653 RTOS also ensures the integrity and sequence of arrival of the messages as well as its atomic nature (i.e. either the whole message is received or nothing is received). Similar to spatial and temporal partitioning, the core module resources required to manage the ports and channels are defined in configuration tables (Aeronautical Radio, Incorporated, 2010). Besides supporting the reusability and portability of application software, this feature also makes it easy to identify which other partitions could potentially be affected by a change in one partition. For instance, this could be done by looking at the channels exiting the changed partition and identifying which partitions are receiving messages from this changed partition.

These features of ARINC 653 could allow the regression test scope to be limited to just the partition that is changed and the partitions that could potentially be affected by this change. If DO-178B qualified development and verification tools which could identify dependencies and provide certification evidence are also available to support the particular ARINC 653 RTOS, it would also be possible to identify the specific changes and their linkage to other affected areas within the application software so as to further reduce the regression test scope for a particular partition.

This assessment assumes that the traditional federated architecture of the aircraft's avionics and its associated software are replaced by a single IMA system with software based on ARINC 653. FAA (2010) defines an IMA system as a shared set of flexible, reusable and interoperable hardware and software resources that, when integrated, form a platform that provides services. In cases where the federated architecture is only replaced partially by an IMA system or by more than one IMA system, the benefits of ARINC 653 may not be reaped fully as limiting the regression test scope to just the changed partition and the partitions that could be affected by this change would not be sufficient. Regression testing between an IMA system and the remaining LRUs of the federated architecture (in the case where the federated architecture was only partially replaced by the IMA system) or between IMA systems (in the case where the federated architecture is replaced by more than one IMA system) would be required.

POTENTIAL APPLICABILITY IN RSAF CONTEXT

Like many other air forces, the RSAF has its own airworthiness certification body to certify the airworthiness of the aircraft and systems which are acquired and inducted into its inventory or upgrades implemented on its existing fleet. Certification of software is delegated by this body to a subordinate committee called the Software Modification Committee (SMC).

Today, none of the software used in the RSAF's fleet is based on ARINC 653. The SMC has to deal with software ranging from those developed to legacy standards such as DOD-STD-2167A to more recent standards such as DO-178B. Although the RSAF and the DSTA project management teams (PMTs) have subject matter expert knowledge of software in general, they may not have detailed knowledge of the specific software designs of these aircraft and systems in their position as buyers. This is exacerbated by the fact that the legacy software of the aircraft and systems do not support partitioning and that the OEMs usually do not provide sufficient information for the PMTs to review the change impact analysis for the SMC's approval. In some cases, this has resulted in the SMC taking a more conservative approach by adopting the test scope originally used by the aircraft OEM to certify the aircraft or system for the RSAF. This results in a larger regression test scope especially in cases where safety-critical software is involved. This consequently results in more time, effort and resources required to release a new software version.

With the advent of ARINC 653, IMA and their increasing use in aircraft, it is a possibility that ARINC 653 based software may be introduced into the RSAF's fleet. Depending on the extent of the IMA and ARINC 653 based software implemented in the aircraft, this could mean that the PMTs could take advantage of the features in ARINC 653 to ensure that the regression test scope approach taken for software changes in these aircraft is adequate. The PMTs should be able to verify or request the necessary information to authenticate the comprehensiveness and accuracy of a change impact analysis and determine the adequacy of the resulting regression test scope for any software change proposal. This is especially if the ARINC 653 RTOS used comes with DO-178B qualified development and verification tools. The resulting benefits are reduced time, cost and resources required to release a new software version for the introduction of new capabilities to the RSAF or to fix critical bugs that limit or constrain current capabilities.

POSSIBLE FUTURE WORK

To better prepare the RSAF and the PMTs for the introduction of ARINC 653 based software, one approach could be to use a pilot project to implement ARINC 653 based software on a small scale. This approach would enable the RSAF and the PMTs to familiarise themselves with the features of ARINC 653 and to verify its benefits in terms of facilitating the reduction in regression testing. At the same time, this project could also identify any challenges involved and come up with mitigating measures to address them before the actual induction of the first ARINC 653 based aircraft into the RSAF's inventory.

CONCLUSION

It is assessed that the features of ARINC 653 do provide the resulting software applications with a structure that makes change impact analyses and the determination of the scope of regression testing easier and more deterministic. This is especially so if the ARNC 653 RTOS used comes with DO-178B qualified development and verification tools.

The RSAF could acquire aircraft with IMA and ARINC 653 based software in the future and take advantage of these ARINC 653 features to better manage regression testing efforts.

ACKNOWLEDGEMENTS

The author would like to thank DHAEL (Engineering), AELD-RSAF ME7 Ang Hang Guan and AELD-RSAF ME6 Lim Kim Hai for their guidance and inputs.

REFERENCES

Aeronautical Radio, Incorporated. (2010, November). *Avionics application software standard interface, required services* (ARINC Specification 653: Part 1). Annapolis, MD: Aeronautical Radio, Incorporated.

Aeronautical Radio, Incorporated. (2007, January). *Avionics application software standard interface, extended services* (ARINC Specification 653: Part 2). Annapolis, MD: Aeronautical Radio, Incorporated.

Aeronautical Radio, Incorporated. (2006, October). *Avionics application software standard interface, conformity test specification* (ARINC Specification 653: Part 3). Annapolis, MD: Aeronautical Radio, Incorporated.

Aeronautical Radio, Incorporated. (2012, June). *Avionics application software standard interface, subset services* (ARINC Specification 653: Part 4). Annapolis, MD: Aeronautical Radio, Incorporated.

Federal Aviation Administration (FAA). (2010). *Integrated modular avionics development, verification, integration, and approval using RTCA/DO-297 and Technical Standard Order-C153* (Advisory Circular AC 20-170). Washington, DC: FAA.

Lagoy, A. and Gauer, T. A. (n.d.). *IV&V on Orion's ARINC 653 flight software architecture*. Retrieved from http://www.nasa.gov/centers/ivv/pdf/482470main_2530_-_IVV_on_Orion_ARINC_653_Flight_Software_Architecture_100913.pdf

Prisaznuk, P. J. (2008). *ARINC 653 role in integrated modular avionics (IMA)*. 27th Digital Avionics Systems Conference, St. Paul, MN, USA.

Samolej, S. (2011). ARINC specification 653 based real-time software engineering. *e-Informatica Software Engineering Journal, 5*(1), 39-49. doi: 10.2478/v10233-011-0029-x

Wind River Inc. (2008). *ARINC 653 an avionics standard for safe, partitioned systems*. Retrieved from http://www.computersociety.it/wp-content/uploads/2008/08/ieee-cc-arinc653_final.pdf

ENDNOTES

[1] ARINC was incorporated as Aeronautical Radio, Incorporated in 1929. It was chartered by the Federal Radio Commission (which later became the Federal Communications Commission) to serve as the airline industry's single licensee and coordinator of radio communication outside of the government. Over the years, ARINC has broadened its scope to provide engineering solutions in the aerospace and defence, aviation, airports, government, networks, security and transportation industries worldwide.

[2] RTCA (known as Radio Technical Commission for Aeronautics), originally founded in 1935, and until its re-incorporation in 1991 as a private not-for-profit corporation is a US volunteer organisation that develops technical guidance for use by government regulatory authorities and by industry. It has over 200 committees and overall acts as an advisory body to the FAA. RTCA is sponsored as a Federal Advisory Committee by the US DOT Federal Aviation Administration. Guidance documents are developed and drafted by a Special Committee (SC) and are based on a consensus developed within the SC charged with responsibility for the given document. Despite the loosely defined requirements of membership in RTCA, the guidance documents are based on expert technical opinion.

[3] FAR (Federal Aviation Regulations) Part 25.1309 defines the requirements for equipment, systems and installations in civil aircraft administered by the FAA.

BIOGRAPHY

KHOO Wee Tian is Head Engineering (Air Systems). He provides technical guidance to programme managers in the area of avionics architecture and integration. Wee Tian has extensive experience in the acquisition and upgrade of air platforms. He is also a member of the RSAF airworthiness committees. Wee Tian was a member of the F5 Upgrade Project Team as well as the Mission Computer Team which won the Defence Technology Prize Team (Engineering) Award in 1999 and 2008 respectively. He was also a member of the Longbow Apache AH-64D acquisition team which was awarded the Gruppo Agusta International Fellowship 2002 Award by the American Helicopter Society. Wee Tian graduated with a Bachelor of Engineering (Electrical and Electronics Engineering) degree with Honours from the then Nanyang Technological Institute in 1986. He further obtained a Master of Science (Electrical Engineering) degree from the National University of Singapore in 1993.