Advanced Science and Technology Letters
Vol.77 (UNESST 2014), pp.33-36
http://dx.doi.org/10.14257/astl.2014.77.07

Design of Visualizing Event Synchronization for Race Conditions in ARINC-653 Applications

Myeong-Sin Kang[1], Bong-Jun Paeng[2], Ok-Kyoon Ha[3], and Yong-Kee Jun[2]

[1] Aero Master Corporation, 345, Haeansaneop-ro, Sacheon, Republic of Korea
[2] Department of Informatics, Gyeongsang National University, 501, Jinjudea-ro, Jinju, Republic of Korea
[3] Engineering Research Institute, Gyeongsang National University, 501, Jinjdea-ro, Jinju, Republic of Korea
myeong41@nate.com, paeng0316@gmail.com, {jassmin, jun}@gnu.ac.kr

Abstract. This paper presents a visualization tool that provides the overall information of process synchronization based on the event services in ARINC-653 applications. The tool visualizes logical synchronization among processes and race conditions by the violations of synchronization order, such as the priority inversion, through analyzing the aspect of the process execution and the accesses to the shared resources of each partition considering event services. We evaluated our visualization tool using synthetic programs for ARINC-653 on a simulation system for integrated modular avionics.

Keywords: ARINC-653, IMA, reliability, race conditions, visualization.

1 Introduction

The ARINC-653 standard [1] for integrated modular avionics (IMA) [2] defines four intra-partition communication services: buffer, blackboard, semaphore, and event. The event services of ARINC-653 allows synchronization objects to provide access to the shared partition resources for concurrent processes. ARINC 653 applications using the event synchronization may lead to concurrency bugs, such as race conditions [3], because of their non-deterministic executions by the event services which re-schedule processes and reset the event services. Races in ARINC 653 applications occur when concurrent processes access to a shared resource without explicit synchronization, such as event, or cannot be guaranteed their execution order. Race conditions must remove from the applications, because they may lead to serious results, such as priority inversion, or mysterious behaviors of the applications.

2 Background

ARINC-653 standard specifies two communication mechanisms for intra-partition communication [1]. The first one allows communication among processes on a

ISSN: 2287-1233 ASTL

partition via buffers and blackboards. The other is for process synchronization such as semaphores and events. An event in ARINC-653 application is a synchronization object used to notify the occurrence of a condition to processes which may wait for it and to control process accesses to the shared partition resources.

It is hard to debug race conditions because there are many possible execution paths of the program and a lot of the defects are hard to reproduce. Hence, the visualization for the executions of the applications may offer effective debugging environments with intuitively understanding. The previous visualization tools which are included in Integrated Development Environment for major real time Operating Systems, such as VxWorks [4], Integrity [5], and LynxOS [6], offer the occurrence of synchronizations and the state transitions of processes using graphical symbols and lines. However, for debugging ARINC 653 applications, these tools need to analyze the process state transition considering event synchronization and to ratiocinate the occurrence of accesses to shared resources from source codes, because they do not consider race conditions.

3 Design of Visualizing Event Synchronization

We use some symbols to indicate effectively the aspect of processes changed by event synchronization, the accesses to shared resources and their racy information occurred during an execution of ARINC 653 applications. The symbols appear in Fig. 1. The symbols classify into the event synchronization symbols considering event services for ARINC 653, such as SET_EVENT, RESET_EVENT, WAIT_EVENT, the access symbols for read/write accesses to shared resources, and the process control flow symbols for understanding the execution order of processes with synchronization.

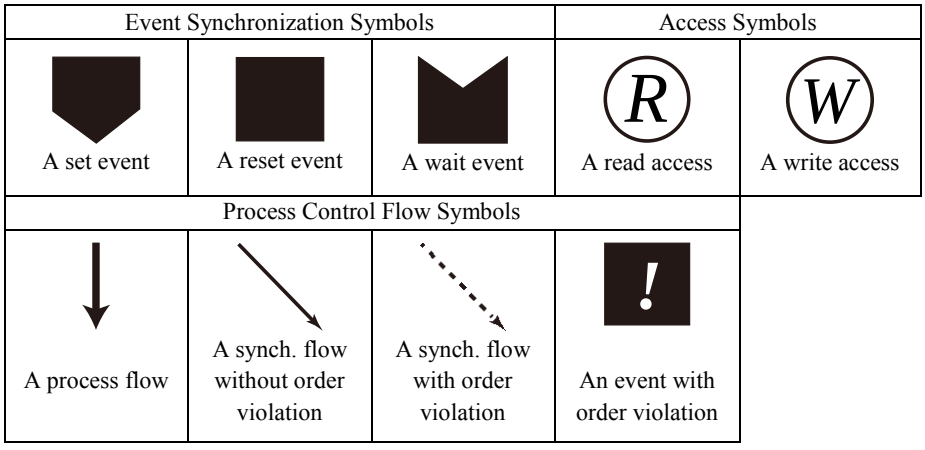

Fig. 1. Visualization Symbols

The process control flow symbols consider order violations as the implicit and the explicit event synchronization to locate race conditions. The detail view of each symbol is as follows:

- ***A Set Event Symbol*** defines that a specified event was requested to set, and means that all processes waiting on that event was moved from the waiting state to the ready state.
- ***A Reset Event Symbol*** indicates an event was requested to change the EVENT STATUS from the up state to down state.
- ***A Wait Event Symbol*** indicates that an event was requested to move the current process from the running state to waiting state when the event was placed in down state.
- ***Read/Write Access Symbol*** indicates that a process was accessed to a shared resource in a partition to read/write any data.
- ***A Synchronization Flow Symbol without order violation*** indicates that a solid arrow from a set event symbol to a wait event symbol.
- ***A Synchronization Flow Symbol with order violation*** indicates that a dashed arrow from a set event symbol to a wait event symbol.
- ***An Event Symbol with order violation*** indicates an event was nested one or more order violations.

4 Implementation

We implemented our visualization tool based on simple notions which appeared in Section 3 using Java and SWT (Standard Widget Toolkit) [7] which provides flexibly cope with platform environment and fast graphic processing. The implementation was carried on a single board computer system with Intel Xeon Dual-core 2 CPUs and 8GB main memory under a simulation system for integrated modular avionics (SIMA). Fig. 2 shows the overall interface of implemented visualization tool.

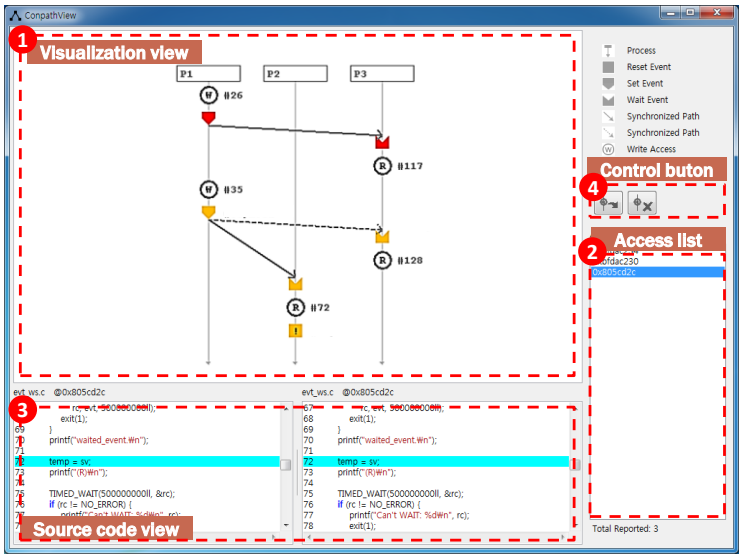

Fig. 2. The interface of the visualization tool

The tool consists of three views: *Visualization View* which is an actual visualization panel to display an execution of ARINC 653 application with the designed symbols (① in Fig. 2), *Access List* which lists shared resources to highlight interesting one (② in Fig. 2), and *Source Code View* which provides the original source does to display where a race condition ran with the execution by indicating each of two lines that related to the racy pair of accesses (③ in Fig. 2). It provides *Control Buttons* (④ in Fig. 2) to indicate additional information, such as the line number of the events and the object name of the events, or to reset extended information on the visualization view.

5 Conclusion

Ensuring the reliability of safety critical software, such as ARINC-653 applications, is difficult because the potential for subtle interactions between concurrent processes can cause race conditions which are hard to detect and reproduce. This paper introduced a visualization technique to help understanding and effective debugging with scalable graphical symbols, and designed a visualization tool that provides the overall information of process synchronization based on the event services in ARINC-653 applications. Moreover, the tool provides an effective approach to debug race conditions by indicating locations of the defects.

Acknowledgments. This work was supported by the BK21 Plus Program (Research Team for Software Platform on Unmanned Aerial Vehicle, 21A20131600012) funded by the Ministry of Education(MOE, Korea) and also 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.

References

1. Airlines Electronic Engineering Committee (AEEC): Avionics Application Software Standard Interface - ARINC Specification 653 - Part 1 (Supplement 2 - Required Services), ARINC (2006)
2. Ha, O.-K., Tchamgoue, G.M., Suh, J.-B., Jun, Y.-K.: On-the-fly Healing of Race Conditions in ARINC-653 Flight Software. In: Proceedings of the 29th Digital Avionics Conference. pp. 5.A.6-1 - 5.A.6-11. IEEE, Salt Lake City (2010)
3. Helmbold, D.P., McDowell, C.E.: A Taxonomy of Race Conditions. J. Parallel Distrib. Comput. 33, 159-164 (1996)
4. Wind River, Workbench, http://www.windriver.com/procucts/workbench/
5. Green Hills, Multi, http://www.ghs.com/products/
6. Lynux Works, Liminosity, http://www.lynuxworks.com/products/
7. Eclipse, The standard Widget Toolkit, http://www.eclipse.org/swt/

