Transactions of the Korean Nuclear Society Autumn Meeting
Busan, Korea, October 27-28, 2005

A Study on Fault-Tolerant Software Architecture for COTS-Based Dependable System

Y. M. Kim, C. H. Jeong
Regulation Research Division, Korea Institute of Nuclear Safety
ymkim@kins.re.kr

1. Introduction

Recently, with the rapid development of digital computers and information processing technologies, nuclear instrument and control (I&C) systems which needs safety-critical function have adopted digital technologies. Also, use of commercial off-the-shelf (COTS) software in safety-critical system has been incremented with several reasons such as economical efficiency and technical problems. But, it requires a considerable integration effort and brings about software quality and safety issues. COTS software is usually provided as a black box that cannot be modified.

The biggest problem when we integrate such a product into dependable systems is the reliability of COTS software. There is no guarantee that the software will perform its function correctly. It may have bugs or unidentified components.

Recently, the method of software verification and validation (V&V) is accepted as a way to assure the dependability of new-developed safety-critical nuclear I&C software. But, because of the limitation of COTS software, software V&V can't be applied as rigorously as new-developed software.

There are considerable attentions into describing software architecture with respect to there dependability properties [5]. In this paper, we present fault-tolerant software architecture using the C2 architectural style. The remainder of the paper is organized as follows: Section 2 discusses background work on the COTS software in nuclear I&C, software fault tolerance and C2 architectural style. Section 3 describes the architecture for fault-tolerant COTS-based software. Finally, we discuss the conclusion and future work.

2. Background

2.1 COTS Software in Nuclear I&C

Safety evaluation process of new-developed software of nuclear I&C system have been performed through the software V&V during software life cycle. But, there may exist situations in which safety systems which will be designed in part with COTS item. In that case, the COTS dedication process shall be accomplished [3]. In preliminary phase, the risks and hazards are evaluated and the safety functions are identified. In secondary phase, the COTS item is evaluated for acceptability using detailed acceptance criteria.

In the case of COTS software, due to the intellectual properties, we may not acquire documents in relation to development, past examples and operating experience. So, the verification and validation of the COTS software is performed very restrictedly.

Specially, in case of real-time operating systems, there are no clear criteria about evaluation of suitability, operating experience, systems of problem report and compensation tests. So, it only depends on engineering judgments.

2.2 Software Fault Tolerance

Because we are not able to get error-free software, software fault tolerance has been issued continuously [1]. The root cause of software design errors is the complexity of software.

For some applications such as nuclear I&C systems, software safety is more emphasized than software reliability. Fault tolerant technologies used in those applications are aimed at preventing a disaster. Single version software fault tolerance techniques are system structuring and atomic action, fault detection, exception handling and others. Multi-version techniques are assumed that software which was built differently. So if one of the redundant versions fails, at least one of the others should provide an acceptable output. There are recovery block, N-version programming, self-checking programming, consensus recovery block and others.

Software fault-tolerance technologies using multi-version COTS products are presented at [6]. In [6], they exampled the NFS service which is developed using different COTS operation systems and file systems.

2.3 Integration using C2 Style

C2 is a component- and message-based architectural style for constructing flexible and extensible software systems [2] [4]. In Figure 1, C2 has components linked together by connectors (message routing devices). This architecture emphasizes weak binding between components, so it provides high integration ability.

Components have two ports which are top and bottom. A top port of a component can be connected with bottom port of a connector and a bottom port of a component can be connected to a top of a connector. But, there are no limits on the number of components or connectors that may be connected to a connector. In C2 style, direct communications between components are not permitted. Components can communicate with each other through connected connectors. Because all messages are exchanged asynchronously and shared address space is not assumed, integration problems can be greatly simplified.

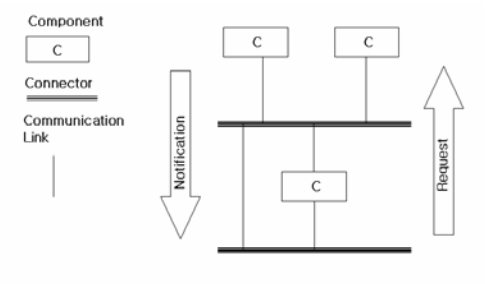

Figure1. Basic Architecture of C2 Style

3. Fault-tolerant COTS Integration Architecture

3.1 Fault-detection Middleware

Our approach, we treat COTS software as an untrustworthy software component. In [2], a COTS component is wrapped inside a C2 Component. We also use this wrapping technology. A fault-detection middleware (FDM) is a wrapper. The FDM makes the COTS component of a fault-tolerant C2 component, so it can be integrated with other components independently and flexibly and provides robustness by fault detection and fault isolation capabilities.

The FDM monitors all of the information flow which is in and out of the COTS component and protects the system that the fault can't be propagated to external components.

3.2 Fault-tolerant COTS software integration architecture

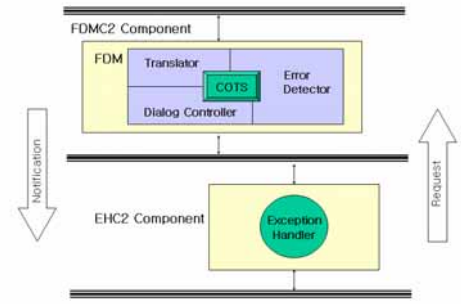

Figure2. Fault-tolerant COTS software integration architecture

Figure 2 shows the architecture for a fault-tolerant COTS component integration. It consists of a Fault Detection Middleware C2 component (FDMC2) and an Exception Handler C2 component (EHC2). A FDM is wrapping a COTS component. It is the FDMC2.

In the FDM, there are a Translator, a Dialog Controller and an Error Detector.

The Translator resolves inconsistency problems between communication components such as data encryptions, message names, the number of parameters and parameter types and orders.

The Dialog Controller controls all incoming and out coming messages. It communicates with connectors. The Dialog Controller maps the request messages from the other components to the proper messages for the internal FDM objects and the COTS component. The Error Detector detects errors of COTS component. All in/out message of COTS passes the Error Detector. All predefined COTS errors and exceptions are detected by the Error Detector.

The EHC2 has the Exception Handler objects.

The Exception Handler manages the exceptions which can be occurred by COTS. The exceptions which can be happened are as follows [1];

- Interface exceptions are signaled by a component when it detects an invalid service request.
- Local exceptions are signaled by a module when its error detection mechanisms find an error in its own internal operations.
- Failure exceptions are signaled by a module after it has detected an error which its fault processing mechanisms have been unable to handle successfully.

4. Conclusion

From now, we presented fault-tolerant software architecture for dependable system such as safety nuclear I&C system. We used C2 style which provided independency and extensibility and used middleware for fault detection and isolation.

This approach has a weak point which can't protect unexpected errors. Our future work will be concentrated on the resolution of it and we intend to show the case study using our architecture.

REFERENCES

[1] Wilfredo Terres-Pomale. Software Fault Tolerance: A Tutorial. Langley Research Center, Virginia, 2000

[2] Nenad Medvidovic, Peyman Oreizy, and Richard N. Taylor. Reuse of Off-the-Shelf Components in C2-Style Architectures. Proceedings of the 1997 Symposium on Software Reusability, Boston, USA, May, 1997

[3] IEEE Std. 7-4.3.2, IEEE Standard for Criteria for Digital Computers in Safety Systems of Nuclear Power Generating Stations, 2003

[4] Taylor, R. N., Medvidovic, N,. Anderson, K. M., Whitehead, E. J., Jr., Robbins, J. E., Nies, K. A., Oreizy, P and Dubrow, D. L., A Component- and Message-Based Architectural Style for GUI Software. IEEE Transactions on Software Engineering, Vol.22, No.6, pp.390-406, 1996

[5] Saridakis,T., Issarny, V. Developing Dependability Systems using Software Architecture. In Proc, 1st Working IFIP Conf. on Software Architecture, pp. 83-104, February 1999.

[6] Miguel Castro, Rodrigo Rodrigues, Barbara Liskov, BASE: Using Abstraction to Improve Fault Tolerance. ACM Transactions on Computer Systems, Vol.21, No.3, August 2003, Pages 236-269