Advances in Engineering Software 55 (2013) 32–44

Contents lists available at SciVerse ScienceDirect

Advances in Engineering Software

journal homepage: www.elsevier.com/locate/advengsoft

A proposal and verification of a software architecture based on LabVIEW for a multifunctional robotic end-effector

José Marcos Silva Anjos, Guilherme Kisseloff Coracini, Emília Villani *

Instituto Tecnológico de Aeronáutica, São José dos Campos, São Paulo 12228-900, Brazil

ARTICLE INFO

Article history:
Received 1 February 2012
Received in revised form 12 September 2012
Accepted 22 September 2012
Available online 31 October 2012

Keywords:
Discrete event systems
LabVIEW
Model checking
Model-based testing
Multifunctional end-effector
Software verification

ABSTRACT

This paper proposes a software architecture based on LabVIEW for controlling discrete event systems. The proposed architecture is an adaptation of the producer–consumer design pattern. This work uses the control software of a multifunctional robotic end-effector as a test-bed for analyzing the applicability of the software architecture and its limitations and advantages. This case study demonstrates the effectiveness of the architecture for dealing with the integration of multiple functionalities in the control system. For this case study, the validation of the architecture is performed using two verification techniques: (1) a formal verification using timed automata and the UPPAAL model checker and (2) the CoFI (Conformance and Fault Injection) method for defining the set of test cases to check the software product. Both verification techniques identified errors that were introduced into the control system during the programming phase.

1. Introduction

This paper discusses the problem of designing and verifying control software for discrete event systems in LabVIEW. The motivation for this work originated from a practical application in the aircraft industry, the development of the control software for a multifunctional robotic end-effector. The FARE (Fuselage Assembly Robotic End-effector) is part of the ASAA (Aircraft Structure Assembly Automation) project, a partnership between the Brazilian aircraft industry and the Aeronautics Institute of Technology (ITA).

The FARE behavior is characterized primarily as a discrete event system with time constraints. The use of LabVIEW as the programming language for the FARE control system is a requirement of the aircraft manufacturer because of its reduced development time, scalability and ease of integration with hardware devices. LabVIEW is a graphical programming environment that has been extensively used in instrumentation and control applications. However, examples of discrete event control systems implemented in LabVIEW are not common.

The development processes of discrete event control systems, usually based on modeling techniques such as automata and Petri nets, should be adapted to the graphical programming language of LabVIEW. As off-the-shelf solutions for modeling discrete events systems, LabVIEW has a state diagram toolkit for the implementation of finite state machines and a recently introduced statechart module. Because of the limitations of these solutions for systems with large numbers of states and transitions, this work presents a proposal based on the producer–consumer design pattern.

The contribution of this work is the proposal, application and validation of a software architecture for discrete event control systems. This work uses the FARE as a test bed for analyzing the applicability of the software architecture and its limitations and advantages. A discussion regarding which technique should be used to validate discrete event control systems developed in LabVIEW is also a contribution of this paper. Two verification techniques are used and analyzed in this paper: (1) a formal verification using timed automata and the UPPAAL model checker and (2) the CoFI (Conformance and Fault Injection) method for defining the set of test cases used to check the software product.

The validation approach follows the steps illustrated in Fig. 1. Starting from the software requirements, a partial version of the FARE control software is developed. A corresponding model for timed automata is developed and verified. The errors detected in the timed automata model are used for correcting the software. Concurrently, tests of the CoFI method are developed using the requirements specified by the aircraft manufacturer and applied to the partial version of the end-effector control software. The errors detected are compiled in a list of lessons learned, which is used for the development of the full version of the FARE control

* Corresponding author. Address. Instituto Tecnológico de Aeronáutica, Praça Marechal Eduardo Gomes, 50, Vila das Acácias, CEP 12.228-900, São José dos Campos, SP, Brazil. Tel.: +55 12 3947 5864; fax: +55 12 3947 6965.

E-mail addresses: jmarcos.anjos@gmail.com (J.M.S. Anjos), guilherme.coracini@gmail.com (G.K. Coracini), evillani@ita.br (E. Villani).

0965-9978/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.advengsoft.2012.09.004

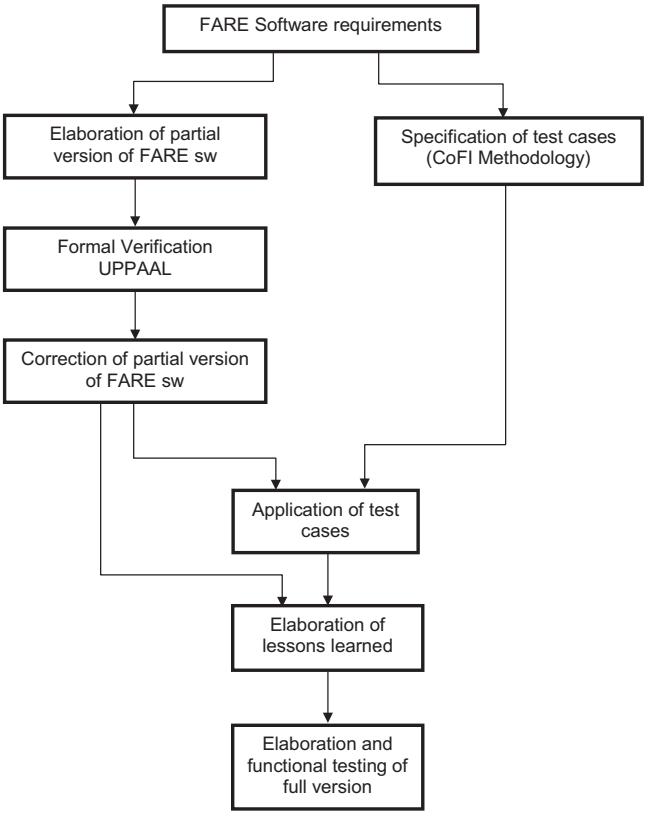

Fig. 1. Validation approach.

software. The full version is submitted to functional tests for final validation.

This work is organized as follows. The next section discusses related works. Section 3 describes the proposed architecture. Section 4 presents the FARE and the requirements for its control software. Then, Section 5 describes the verification of the partial version of the FARE software using the UPPAAL model checker and the application of the CoFI model-based testing method. Section 6 discusses some conclusions and future work.

2. Related work

The review of related work focuses on two topics: the control of discrete event systems and development of software for discrete event systems in LabVIEW.

One of the most important contributions for the control of discrete event systems is the supervisory control theory (SCT) proposed by Ramadage and Woham [20]. In the SCT, the plant is modeled as a language generator specified over an alphabet of events. The controller, called the supervisor, is synthesized to fulfill a given specification. The supervisor restricts the behavior of the plant by disabling controllable events.

Since 1989, many works by different authors have extended and complemented the SCT. In this section, we focus on some works that discuss modularity and implementation issues that may arise when applying the SCT to industrial systems.

Dietrich et al. [8] address the problem of having a controller that not only disables controllable events but also chooses an event from the enabled ones. The chosen event is interpreted as a command given to the plant. This controller derived from the supervisor is called an implementation of the supervisor. The work discusses the problems, such as a blockage in the implementation even if the original abstract supervisor is non-blocking, that may arise because of the arbitrary choice of an implementation.

Miremadi et al. [17] propose a more expressive modeling formalism for treating large industrial systems. The proposal includes a bottom-up structure in which the model of the plant and the specifications are a result of the composition of sub-plants and sub-specifications. The connection between the sub-models is performed via the synchronization of shared events and shared variables. The shared variables are used as guard conditions and actions that are associated with the occurrence of events. The incorporation of shared variables addresses the problems that may arises when the condition for a state change in one sub-model depends on the current state of another sub-model. In a case study, the proposal is applied to a flexible manufacturing cell in which automated guided vehicles (AGVs) transport parts between different stations.

Hellgren et al. [12] discuss the implementation of a modular supervisory controller in a programmable logic controller (PLC). They propose a parameterization of the supervisor model that results in a deterministic model suitable for implementation. The implementation language used in the work is sequential function charts (SFCs). A monolithic SFC implementation is obtained using the modular supervisor.

Modularity is also exploited by De Queiroz and Cury [7]. The authors present an extension of the SCT that addresses modular specification and modular plants. Instead of employing a monolithic supervisor for the entire plant, they employ a modular supervisor for each specification. Issues related to the implementation of the modular supervisor in a PLC are discussed, and a final structure for the control program is proposed. In the example presented by the authors, the language of the control program is the Ladder Diagram. Leal et al. [14] also discuss the implementation of modular supervisors in PLCs using the Ladder Diagram programming language. Among the issues discussed in this work is the treatment of events in a PLC scan cycle.

Oliveira et al. [18] present a method to check the consistency of the implementation of a control system in Ladder Diagram that uses ISA 5.2 binary logic diagrams for the corresponding specification. The specification and the implementation are transformed into eXtensible Markup Language (XML) files. Then, timed automata are generated automatically from the XML files according to the syntax and semantics of the UPPAAL tool. Finally, conformance testing is performed on the automata models to determine whether the implementation is conforms to the specification. Farines et al. [10] present an automatic approach to model and verify PLC programs written in Ladder Diagram using the formal language FIACRE. The purpose is to guarantee the satisfaction of generic and application-oriented properties.

One important difference between the previous works and that proposed in this paper is that almost all the previous works consider a controller implementation in PLCs and, therefore, focus on PLC programming languages, such as SFC and Ladder Diagram. One consequence of using LabVIEW is that because it is not a language dedicated to describe discrete event systems, the automatic translation from LabVIEW to a formal language such as Petri nets or automata is not trivial. In Section 5.1, this problem is discussed using the FARE as an example. The purpose of translating the LabVIEW program to automata is to perform formal verification of properties and to assure the controller conforms to the specification.

One important point common among these works and the proposal of this paper is the focus on modularity as a way of dealing with large systems. In this paper, modularity is achieved by decomposing the plant into modules and then designing the controller for each module as a consumer loop process, as detailed in Section 3. The communication among the controller modules is performed using shared variables and is coordinated by the consumer structure.