Chapter 10

Safety Reconfiguration of Embedded Control Systems

Atef Gharbi
University of Carthage, Tunisia

Hamza Gharsellaoui
University of Carthage, Tunisia

Antonio Valentini
O3neida Europe, Belgium

Mohamed Khalgui
University of Carthage, Tunisia & CNR Research Council, Italy & Xidian University, China

ABSTRACT

The authors study the safety reconfiguration of embedded control systems following component-based approaches from the functional level to the operational level. At the functional level, a Control Component is defined as an event-triggered software unit characterized by an interface that supports interactions with the environment (the plant or other Control Components). They define the architecture of the Reconfiguration Agent, which is modelled by nested state machines to apply local reconfigurations. The authors propose technical solutions to implement the agent-based architecture by defining UML meta-models for both Control Components and also agents. At the operational level, a task is assumed to be a set of components having some properties independently from any real-time operating system. To guarantee safety reconfigurations of tasks at run-time, the authors define service and reconfiguration processes for tasks and use the semaphore concept to ensure safety mutual exclusions. They apply the priority ceiling protocol as a method to ensure the scheduling between periodic tasks with precedence and mutual exclusion constraints.

INTRODUCTION

Embedded Control Systems have become useful in our daily lives such as automotive application, industrial management, control avionics, … To reduce their cost of development, these systems must be reusable. The component-based programming seems the best solution for the development of such systems.

Several component technologies are proposed such as JavaBeans (related to Sun society) (Jubin, 2000), Component Object Model (related to Microsoft society) (COM, 2010), Corba Component

DOI: 10.4018/978-1-4666-3922-5.ch010

Model (provided by the Object Management Group [OMG]) (Pérez, 2002).

However, there are few kinds of component technologies (such as Koala [Jonge, 2009], PBO [Stewart, 1997], PECOS [Wuyts, 2005], …) used in the development of embedded system due to extra-functional properties to be verified (for example quality of service, timeliness, …) (Artist, 2003).

Anyway, each component technology has its benefits and its drawbacks.

As in our work, we want to be independent of any component technology, we propose a new concept of component named "Control Component" which is considered as a software part having interaction with other Control Components and ensuring control of the plant through data provided from (resp. to) sensors (resp. actuator).

A Control System is assumed to be a composition of Control Components with precedence constraints to control the plant according to well-defined execution orders.

The proposed method to ensure Functional Safety of the interconnected Control Component is an agent-oriented software. On the one hand, we study the Functional Safety in a central system i.e. a single agent supervising the whole system.

This agent reacts as soon as an error occurs in the plant. The decision taken may vary from changing the set of Control Components that constitute the system, modifying the connection between different Control Components, substituting the behavior of some Control Component by another behavior or even modifying data. According to these functionalities, it is possible to define the architecture of the agent as based on four levels.

We propose useful meta-models for Control Components and also for intelligent agents. These meta-models are used to implement adaptive embedded control systems.

As we choose to apply dynamic scenarios, the system should run even during automatic reconfigurations, while preserving correct executions of functional tasks.

Given that Control Components are defined in general to run sequentially, this feature is inconvenient for real-time applications which typically handle several inputs and outputs in a too short time constraint.

To meet performance and timing requirements, a real-time must be designed for concurrency.

To do so, we define at the operational level some sequential program units called real-time tasks.

Thus, we define a real-time task as a set of Control Components having some real-time constraints. We characterize a task by a set of properties independently from any Real Time Operating System (RTOS).

We define service processes as software processes for tasks to provide system's functionalities, and define reconfiguration processes as tasks to apply reconfiguration scenarios at run-time. In fact, service processes are functional tasks of components to be reconfigured by reconfiguration processes.

To guarantee a correct and safety behavior of the system, we use semaphores to ensure the synchronization between processes. We apply the famous algorithm of synchronization between reader and writer processes such that executing a service is considered as a reader and reconfiguring a component is assumed to be a writer process. The proposed algorithm ensures that many service processes can be simultaneously executed, whereas reconfiguration processes must have exclusive access.

We study in particular the scheduling of tasks through a Real Time Operating System. We apply the priority ceiling protocol proposed by (Sha, 1990) to avoid the problem of priority inversion as well as the deadlock between the different tasks. The priority ceiling protocol supposes that each semaphore is assigned a priority ceiling which is equal to the highest priority task using this