Real-Time Reconfigurations of Embedded Control Systems

Atef Gharbi, National Higher Engineering School of Tunis (ENSIT), Tunis University, Tunisia & National Institute of Applied Sciences and Technology (INSAT), Carthage University, Tunisia
Hamza Gharsellaoui, National Institute of Applied Sciences and Technology (INSAT), Carthage University, Tunisia & National Engineering School of Carthage (ENIC), Carthage University, Tunisia
Mohamed Khalgui, National Institute of Applied Sciences and Technology (INSAT), Carthage University, Tunisia & Systems Control Lab, Xidian University, China

ABSTRACT

This paper deals with the study of the reconfiguration of embedded control systems with safety following component-based approaches from the functional level to the operational level. The authors define the architecture of the Reconfiguration Agent which is modelled by nested state machines to apply local reconfigurations. They propose in this journal paper technical solutions to implement the whole agent-based architecture, by defining UML meta-models for both Control Components and also agents. To guarantee safety reconfigurations of tasks at run-time, they define service and reconfiguration processes for tasks and use the semaphore concept to ensure safety mutual exclusions. As a method to ensure the scheduling between periodic tasks with precedence and mutual exclusion constraints, the authors apply the priority ceiling protocol.

KEYWORDS

Agent-Based Architecture, Control Component, Real-Time Scheduling, Reconfiguration, Reconfiguration Agent, Safety Nested State Machines

1. INTRODUCTION

Real-Time systems are playing a crucial role in our society, and in the last two decades, there has been an explosive growth in the number of real-time systems being used in our daily lives and in industry production. Systems such as chemical and nuclear plant control, space missions, flight control systems, military systems, telecommunications, multimedia systems, and so on all make use of real-time technologies (Gharsellaoui, 2013). To reduce their cost of development, these systems must be reusable. The component-based programming seems the best solution for the development of such real-time systems. For example, between 250 and 500 process steps on 50–120 different types of equipment are required to produce a chip of average complexity. Since the 1990s, the market of semiconductor fabrication has become increasingly global, dynamic and customer driven. An organization’s competitive advantage depends more and more on its responsiveness in meeting market changes and opportunities, and in coping with unforeseen circumstances (i.e., machine breakdowns, rush orders, etc.). Thus, it is important to reduce inventories, decrease cycle time, and improve resource utilization (Zhang, 2014).
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 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 systems 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 (Azar & Vaidyanathan, 2015a, b,c; Zhu & Azar, 2015; Azar & Zhu, 2015). The proposed method to ensure Functional Safety of the interconnected Control Component is 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 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). On one hand, 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. The task parameters must be adapted on-line to cope with the overload. The idea is to adapt the periods of the tasks when needed to reduce the processor utilization (Gharsellaoui, 2012).

On the other hand, 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 semaphore. Any task is only allowed to enter its critical section if its assigned priority is higher than the priority ceilings of all semaphores currently locked by other tasks. Finally, we implement a real-time task through the use of RTLinux as an example of RTOS.

The contributions of this research work have been applied on the benchmarking production system: FESTO system (used as running example) allowing us to validate our results. We present in the next
Principles of Classification

[www.igi-global.com/chapter/principles-classification/69672?camid=4v1a](www.igi-global.com/chapter/principles-classification/69672?camid=4v1a)