An Architecture-Based Adaptation Framework for Soft Real-Time Applications

Ning Gui, University of Antwerp, Belgium & Central South University, China
Hong Sun, University of Antwerp, Belgium
Chris Blondia, University of Antwerp, Belgium

ABSTRACT

Real-time systems are increasingly used in dynamic changing environments with variable user needs hosting real-time applications ranging in number and nature over time. However, to the authors’ knowledge, no unified framework exists that is able to cope with those competing real-time concerns across multiple real-time application domains. This paper proposes an architecture-based framework for managing real-time components’ dependence and lifecycle in an effective and uniform way. A real-time component runtime service is proposed here to maintain the global view, control the whole lifecycle of the components, and keep existing real-time components’ promised contracts in the face of run-time changes. This framework is designed to be easily extended with other constraint resolving policies as well as dependence descriptions languages. At end of this paper, the framework is tested by a simulated control application via adaptation and performance aspects.

Keywords: Application Domains, Architecture-Based Adaptation, Lifecycle, OSGi, Soft Real-Time

INTRODUCTION

Traditional real-time systems such as process control systems typically work in closed and highly predictable environments. Those systems were meant for predefined and fixed services that were not supposed to change. As long as those assumptions and conditions hold, this approach allows hard real-time requirements to be met with very high assurance (Nilsen & Klein, 2003; Wall, Larsson, & Norstrom, 2002).

However, some soft real-time systems, such as smart devices, are increasingly operating in changing environments and with diverse user needs (Kelvin, 2007). In coping with dynamic and diverse requirements, such devices normally are equipped with open software system that able to host applications ranging in number and nature over time. In such system, the traditional workflow approach for real-time application development: design, verify, map, and deploy could not work effectively as the system con-
configuration will evolve during the whole system lifecycle (Musliner, Goldman, Pelican, & Krebsbach, 1999; Sangiovanni-Vincentelli & Di Natale, 2007).

This open and evolving execution environment requires new approaches in building and manages real-time applications. Thorough off-line system analysis is forbidden by such dynamicity. However, due to scarcity of system resources, these installed applications normally are, to a certain extent, inter-dependent with one another, as one application may competing with other applications for the CPU, memory, networks or other system resources. Failed to deal with such dependence may lead to breaking applications’ real-time contracts which the system designer has supported to satisfy.

Traditional RTOS and real-time schedulers based approach could not cope with that complex dependence, as the scheduler itself has no way to get and could not understand that high level application dependence which can only be interpreted according to current environment. The RTOS is in a sense too “far away” from the application-level concerns to be able to take the right decisions on which components to select and which resources to dispatch.

In order to support the dynamic changing set of real-time applications, from our experience in designing adaptive real-time systems, any effective solution should have the following three important aspects. First, those resource requirements across the currently hosted applications must be obtained and managed during system run-time. Second, those component dependences, including the explicit functional dependencies and the implicit dependencies due to competing for the limited available resources, need to be explicitly modeled into architectural model across application domains. Third, the architectural model should be used to drive the lower-level managements of the corresponding real-time tasks.

Our framework tackles these problems from different perspectives. In this paper, we introduce an architecture-based management framework - the Declarative Real-time Control Runtime (DRCR) Framework, spanning through installed application domains. By eliciting each real-time component’ distinguishes attributes from its meta-data and its component management interface, an accurate global architecture model for current real-time system can be built. In order to guarantee certain real-time application performance, this model then is mapped to adjust low-level real-time task’s attributes and states according to different adaptation policies. Benefited from the Service oriented architecture, this framework can be easily extended with other dependence resolving policies to deal with diverse system requirement. The effectiveness of our architecture is demonstrated both from qualitative and a quantitative point of view. This work is based our previous work on Declarative Real-time component model (DRCom) (Gui, De Florio, Sun, & Blondia, 2008).

The rest of this paper is structured as follows: A scenario shows the importance for inter real-time application dependence managements. After this, a general overview of our architecture is given. Next, we describe the component’s declarative real-time meta-data format design and the component’s dependence resolving process is discussed. Then, our framework is evaluated from two aspects while providing a comparison with related works. The future work and our conclusions are introduced at the end of this paper.

**SCENARIO DESCRIPTION**

To better illustrate all the complexities in introducing the context knowledge into the application composition process, we introduce an example scenario that will be revisited several times through the course of this paper.

Now, the Set-top devices are equipped with open system can be harnessed to provide services in addition to its core application for temporal or long-term base. As a Set-top device, its basic function should be TV application. In brief, this application will received Stream Video data from remote server, decode this data and output to the TV port. As an open system,
Deferred System's Design: Countering the Primacy of Reflective IS Development with Action-Based Information Systems
[www.igi-global.com/chapter/deferred-system-design/4212?camid=4v1a](www.igi-global.com/chapter/deferred-system-design/4212?camid=4v1a)