The Journal of Systems and Software 91 (2014) 147–162

Contents lists available at ScienceDirect

The Journal of Systems and Software

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

Predictable integration and reuse of executable real-time components

Rafia Inam*, Jan Carlson, Mikael Sjödin, Jiří Kunčar

Mälardalen Real-Time Research Centre, Mälardalen University, Box 883, Västerås, Sweden

ARTICLE INFO

Article history:
Received 15 May 2013
Received in revised form
19 November 2013
Accepted 23 December 2013
Available online 4 January 2014

Keywords:
Real-time components' integration
Component reuse
Hierarchical scheduling

ABSTRACT

We present the concept of runnable virtual node (RVN) as a means to achieve predictable integration and reuse of executable real-time components in embedded systems. A runnable virtual node is a coarse-grained software component that provides functional and temporal isolation with respect to its environment. Its interaction with the environment is bounded both by a functional and a temporal interface, and the validity of its internal temporal behaviour is preserved when integrated with other components or when reused in a new environment. Our realization of RVN exploits the latest techniques for hierarchical scheduling to achieve temporal isolation, and the principles from component-based software-engineering to achieve functional isolation. It uses a two-level deployment process, i.e. deploying functional entities to RVNs and then deploying RVNs to physical nodes, and thus also gives development benefits with respect to composability, system integration, testing, and validation. In addition, we have implemented a server-based inter-RVN communication strategy to not only support the predictable integration and reuse properties of RVNs by keeping the communication code in a separate server, but also increasing the maintainability and flexibility to change the communication code without affecting the timing properties of RVNs. We have applied our approach to a case study, implemented in the ProCom component technology executing on top of a FreeRTOS-based hierarchical scheduling framework and present the results as a proof-of-concept.

1. Introduction

In this paper we target development of the large class of embedded systems which is required to perform multiple simultaneous control-functions with real-time requirements. From the development point of view, it often makes sense to develop the different control-functions as separate software-components (Crnkovic and Larsson, 2002). Typically, these components are first developed and tested in isolation, and later integrated to form the final software for the system. Furthermore, many industrial systems are developed in an evolutionary fashion, reusing components from previous versions or from related products. It means that the reused components are re-integrated in new environments.

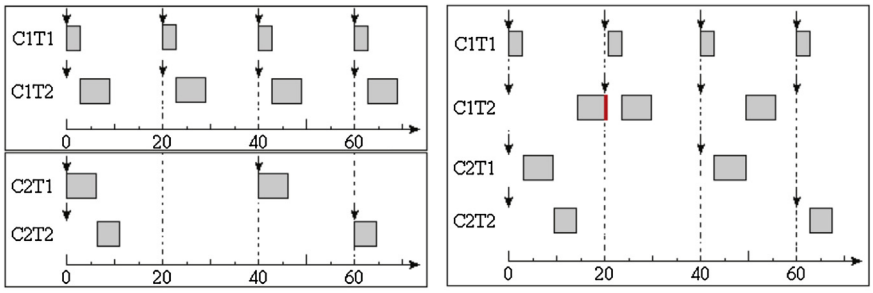
Temporal behaviour of real-time software components poses difficulties in their integration. When multiple software components are deployed on the same hardware node, the emerging timing behaviour of each of the components is typically unpredictable. For example, the temporal behaviour of two components *C*1 and *C*2 and their tasks execution is depicted in Fig. 1, where the horizontal axis represents time, an arrow represents task arrival and a filled rectangle shows task execution. The temporal behaviour of both components is tested to be correct during unit testing and all tasks of both components meet their deadlines when executed separately before integration as obvious from Fig. 1(a). However, upon their integration, tasks of one component affect the scheduling of tasks of other components and as a result task *C*1*T*2 misses its deadline at time 20 in Fig. 1(b). This means that for an embedded system with real-time constraints; a component that is found correct during unit testing may fail due to a change in temporal behaviour when integrated in a system. Even if a new component is still operating correctly in the system, the integration could cause a previously integrated (and correctly operating) component to fail. Similarly, the temporal behaviour of a component is altered if the component is reused in a new system. Since this alteration is unpredictable as well, a previously correct component may fail when reused.

In this paper we focus on the schedulability of tasks, i.e. meeting their deadlines, as the main timing property. An RVN's timing behaviour is *predictable* during its integration and reuse, as long as the schedulability of tasks that have been validated during its development within a component is guaranteed when components are integrated together.

In the real-time community, *Hierarchical Scheduling Framework (HSF)* (Deng and Liu, 1997) is known as a technique for solving this

* Corresponding author. Tel.: +46 21 103196.
E-mail addresses: rafia.inam@mdh.se, rafia.inaam@gmail.com (R. Inam), jan.carlson@mdh.se (J. Carlson), mikael.sjodin@mdh.se (M. Sjödin), jiri.kuncar@gmail.com (J. Kunčar).

0164-1212/$ – see front matter © 2014 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jss.2013.12.040



(a) Tasks' execution within separate components

(b) Tasks' execution after components' integration

Fig. 1. Schedulability problem during components' integration: (a) tasks' execution within separate components and (b) tasks' execution after components' integration.

predictability problem by providing temporal isolation between components. It supports CPU time sharing among components or applications (means leveraging the CPU-time partitioning from task-level to the component-level by executing each component in a separate server), hence isolating components' functionality from each other for, e.g., temporal fault containment, compositional verification, and unit testing. HSF has been proposed to develop complex real-time systems by enabling temporal isolation and predictable integration of software-functions (Nolte, 2011).

We address the challenges of preserving the timing properties within components and to apply these properties during components' integration. We propose the concept of a *runnable virtual node (RVN)*, in which we integrate HSF within a component technology for embedded real-time systems; to realize our ideas of guaranteeing temporal properties of real-time components, their predictable integrations and reusability. An RVN represents the functionality of software-component (or a set of integrated components) combined with allocated timing resources and a real-time scheduler to be executed as a server in the HSF. It introduces an intermediate level between the functional entities and the physical nodes. Thereby it leads to a *two-level deployment process* instead of a single big-stepped deployment; i.e. deploying functional entities to the virtual nodes in *a first-step*, and then, deploying multiple virtual nodes to the physical node (target hardware) in *a second-step*.

An important feature during component integration is to provide communication among various components of a target software system. This communication should also be predictable in case of real-time components and do not affect the schedulability of tasks. We implement a communication strategy that enables to execute the communication code independently from RVNs hence making the RVNs integration predictable, since communication time will not affect the schedulability of RVNs' tasks.

The main contributions of this paper are:

- We *realize the concept of runnable virtual nodes* for the ProCom component technology (Sentilles et al., 2008) by exploiting the HSF implementation (Inam et al., 2011). The purpose is to make the integration of real-time components predictable, and to ease the component's reuse in the new systems.
- We *introduce a two-level deployment process* instead of a single big deployment. The two-level process gives development benefits with respect to composability, system integration, testing, validation and certification. Further it leverages the hierarchical scheduling to preserve the validity of an RVN's internal temporal behaviour when integrated with other components or when reused in a new environment.
- We *implement a communication strategy* that supports the predictable integration and reuse of runnable entities. We *evaluate this strategy* against a direct strategy for efficiency and reusability aspects of RVN. We *develop an analysis tool End-to-End Latency Analyzer for ProCom (EELAP)* (Kunčar et al., 2013; Kunčar, 2013) to compute the end-to-end latencies of both communication strategies/or to evaluate both communication strategies.
- We *provide a case study* as a proof-of-concept of our approach: we implement it using the ProCom component technology and execute it on a real hardware an AVR 32-bit microcontroller (EVK, 2013). We *demonstrate the runnable virtual node's properties* with respect to temporal isolation and reusability.

Once the RVN is assigned for timing properties, it will preserve these properties without regard of other RVNs it is integrated with on the same physical node. Our realization allows predictable coexistence of virtual nodes that have been either constructed with different development methodologies or constructed using the same development technology but having different timing properties. E.g., a ProCom-RVN can co-exist with an RVN with legacy FreeRTOS-tasks, or an RVN with hard real-time components that has been verified with formal methods can co-exist with an RVN with components without real-time requirements and that has not undergone extensive validation.

The RVN discussed in this paper is the extension of our previous work on the concept of virtual node in (Inam et al., 2012a) and the initial implementation in (Inam et al., 2012b): both papers are based on the idea of a real-time component that preserves its timing properties when integrated with other components on a physical platform. The work described in this paper is the extension of the initial implementation of the concept of the RVN component, and the synthesis of the final executables with the emphasis on using a two-step deployment process. The previous work of (Inam et al., 2012a), on the other hand, focused on just the presentation of the general idea of virtual node as a real-time component at a high level of abstraction and described inclusion of virtual nodes within different components technologies like AUTOSAR, AADL, and ProCom. It did not address the synthesis process and lacked a practical implementation.

The previous work of (Inam et al., 2012b) only presented the initial RVN implementation for components integration, and the evaluation of predictable integration of real-time components using a case study on cruise controller system and its execution in ProCom component model on the AVR-based board EVK1100 (EVK, 2013). In this paper we extend the implementation to incorporate *the reuse of real-time components along with their timing properties* and discuss how the two-level deployment helps accomplishing the predictable coexistence of real-time components together and facilitate their reuse. We also evaluate the reuse property of RVN by extending the previous case study with the new functionality of the adaptive cruise controller system and