Journal of Systems Architecture 61 (2015) 472–485

Contents lists available at ScienceDirect

Journal of Systems Architecture

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

A composable real-time architecture for replicated railway applications

Stefan Resch [a,*], Andreas Steininger [b], Christoph Scherrer [a]

[a] Thales Austria GmbH, Handelskai 92, A-1200 Vienna, Austria
[b] Vienna University of Technology, Embedded Computing Systems Group E182-2, Treitlstr. 3, A-1040 Vienna, Austria

A R T I C L E I N F O

Article history:
Received 30 June 2014
Received in revised form 8 January 2015
Accepted 2 April 2015
Available online 16 April 2015

Keywords:
Composability
Mixed-criticality
Virtualization
Triple modular redundancy
Scheduling
Certification

A B S T R A C T

Triple-modular-redundant applications are widely used for fault-tolerant safety–critical computation. They have strict timing requirements for correct operation. We present an architecture which provides composability and mixed-criticality to support integration and to ease certification of such safety–critical applications. In this architecture, an additional layer is required for the sharing/partitioning of resources. This potentially jeopardizes the synchronization necessary for the triple-modular-redundant applications.

We investigate the effects of different (unsynchronized) scheduling methods for the resource-sharing layer in this architecture and conclude that an out-of-the-box solution, which guarantees the technical separation between applications with fast reaction time requirements is only feasible when executing at most one instance of a triple-modular-redundant application per CPU-core for single and multi-core CPUs. Only when accepting changes in the applications or the applications' synchronization mechanisms, are more flexible solutions with good performance and resource utilization available.

1. Introduction

Increasing computational power and multi-core CPUs enable integration of numerous functions within a single system, which opens the potential for better utilization of resources through appropriate sharing. Such a sharing of resources has many implications for safety–critical systems. One major aspect is that for safety–critical systems, integration and changing parts of the system requires (re-) certification of the whole system. It is the aim of architectures providing *composability* and *mixed criticality* to achieve this integration without affecting safety and availability of the individual (pre-certified) applications, while considering only the new setup or changes for certification. This is a well known approach in the avionics domain with the Integrated Modular Avionics defined in DO-297 [1] and the corresponding ARINC 653 standards [2].

In this paper, we present an architecture supporting the composable integration of fail-safe triple modular redundant (TMR) applications for the railway domain. These applications have strong requirements on the properties of the internal communication channels and reactivity of individually synchronized replicas. Together with their lower-level software, the applications assume full access to the hardware resources. With the introduction of a layer for sharing these resources, this full access is inevitably removed. The arising problems are now twofold. First, the applications' timely behaviour is affected due to the (now) restricted access to these resources. Secondly, this restriction changes the basis of the applications' safety concepts for certification.

According to Kopetz [3, pp. 102–103], there are four principles of composability:

- Independent development of components,
- stability of prior services,
- non-interfering interactions, and
- preservation of the component abstraction in case of failures.

In our approach, which is based on the analysis from [4], the independent development and certification of sub-services, as well as the integration environment is enabled through the use of contracts, i.e. an element is certified only with respect to its contract. Also, the stability of prior services is addressed with the contract concept. Instead of non-interfering interactions, we require *predictability* for determining whether the individual required reaction times of the sub-systems prior to integration can be satisfied.

In the following, we use the term *function-set* (FS) for a sub-service and *FS nodes* for the entities, which in compound make its functionality fault tolerant, e.g. replica of a process. These FSs are then deployed in an *integration environment* (IE), providing computation on the *IE nodes* and communication resources,

* Corresponding author. Tel.: +43 1 27711 3123; fax: +43 1 27711 3171.
E-mail address: stefan.resch@thalesgroup.com (S. Resch).

http://dx.doi.org/10.1016/j.sysarc.2015.04.003
1383-7621/© 2015 Elsevier B.V. All rights reserved.

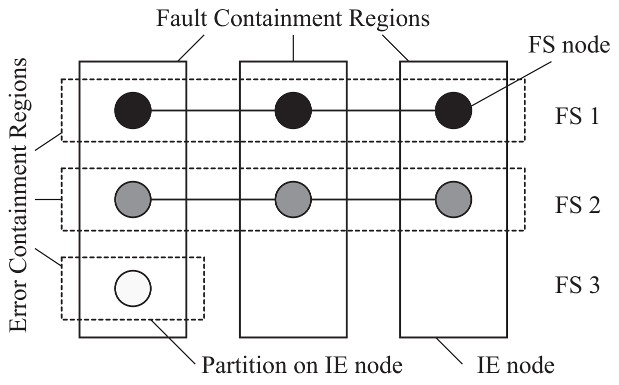

Fig. 1. Elements of the composable architecture.

together with error containment regions as illustrated in Fig. 1. The local instantiation of the error containment regions on the IE nodes are partitions, which ensure safe operation when individual FSs or FS nodes fail, i.e. provide the preservation of the component abstraction in case of failures. The fault containment regions for the FS nodes are established through the use of physically disjoint *IE nodes* in the IE. Certification is then performed in three steps:

(1) Each safety–critical FS is certified together with its FS requirements contract, which specifies the required resources and features for safe FS operation.
(2) An IE is certified with its IE contract stating the provided resources and features for the FSs, and the IE's feasibility evaluation method used for FS integration.
(3) The overall system is certified by matching the FSs contracts to the IE contract, together with performing the IE contract's feasibility evaluation method in a deployment contract.

Since the error containment regions provide sufficient isolation, this approach is also suitable for *mixed criticality*, i.e. deploying FSs with different levels of criticality.

When mapping these entities to the CENELEC E.N. 50129 [5] standard, one additional common element is necessary. The IE is a generic product and each FS in itself a generic application. Both are based on another common generic product defining the contract concept and architectural elements. An overall system is then the specific application consisting of the integrated FSs and used IE.

It is important to note that the error containment regions for composability are merely concerned with the separation issue, i.e. protecting one FS from the failure of another, with whom it shares resources. They are not used for tolerating faults in these resources.

The TMR approach used by our targeted applications is a widely spread technique for attaining fault tolerance in safety–critical systems. That is why combining its use with composability is of interest. With TMR, a system is separated into three fault containment regions. In each of these, tasks compute the same workload as in the other ones. In our model, fault containment is orthogonal to error containment as illustrated in Fig. 1. Fault containment is ensured by using different IE nodes when deploying FS nodes belonging to the same FS, while error containment is provided between FSs through separation of the FS nodes' resources on the IE nodes and interconnect. The FS nodes are synchronized via output-data exchange and voting. Additionally, a membership service is necessary for isolation and recovery of faulty FS nodes. Various synchronization methods exist for TMR, from hardware implemented to purely software-based approaches. The latter ensures the highest flexibility with respect to the integration environment, since no assumptions on the underlying hardware features are made for replication. With software-based TMR single random hardware faults and single quasi-random software faults [6] can be tolerated. Software-based TMR strongly relies on timeouts for supervising progress and synchronization of replicas. As will be discussed later on, the degree of synchronization that can be attained among the replicated FS nodes is a key property of the TMR system. As already mentioned, with the introduction of resource sharing, the timing behaviour of communication and computation ultimately gets obfuscated. This compromises the synchronization quality for TMR, potentially even to the point where proper operation becomes jeopardized, also for fail-safe systems. It is exactly this issue that has to be solved in order to provide an integration environment for TMR applications.

Consequently, the task is to find a strategy that balances the following three different objectives:

- Provide a technical foundation for independent certification,
- Fulfil the reaction time requirements of the applications, and
- Efficiently use the available hardware resources to take advantage of the integration.

Based on these objectives we evaluate different IEs providing static-cyclic, fixed-priority and EDF scheduling strategies for integrating TMR-based FSs. We show that without changing the TMR synchronization mechanism or applications within the FSs only IEs using fixed-priority scheduling with at most one safety critical FS node running per CPU core provide a good balance. Tighter integration of safety–critical applications with good performance is only possible with changes. Here, we propose two solutions. The first one uses fixed-priority scheduling, but weakens the separation property of the IE, whereas the second one with EDF requires extensive changes in the applications themselves.

Following a survey of related work in the next section, we describe the targeted safety–critical applications and their requirements towards the IE in Section 3. The architecture and model used for analysis are presented in Section 4, followed by a description of the synchronization mechanisms in Section 5. In Section 6 we present the scheduling analysis and possible improvements for the IE with a classic TMR architecture and in Section 7 for an IE providing dynamic deployment. Results obtained with a prototype are discussed in Section 8, and we finally conclude the paper in Section 9.

2. Related work

The already mentioned ARINC 653 platform standard [2] and the AUTOSAR platform standard [7] provide methods for achieving composability and mixed-criticality in the avionics and automotive domain. These two platform architectures are based on software partitioning, a concept introduced together with the separation kernel by Rushby [8]. The MILS (Multiple Independent Levels of Security) separation kernels also have the same origin [9]. Such kernels provide strong space and time partitioning. Server class hypervisors like Xen [10] provide space partitioning for virtual machines with the use of virtualization technology, while hypervisors designed for real-time systems may also guarantee time partitioning, e.g. XtratuM [11]. Gu and Zhao [12] provide a good overview on the state-of-the-art of virtualization for embedded systems. Effects on partitioning with the use of multi-core have been studied by Nowotsch and Paulitsch [13]. Time and space partitioning are also core principles of the time-triggered architecture (TTA) [14]. Time-triggering has furthermore been used to provide a composable environment within a system-on-chip [15]. Annex F of the IEC 61508–3 standard [16] presents techniques for establishing