# EVENT-TRIGGERED AND TIME-TRIGGERED CONTROL PARADIGMS

By
Roman Obermaisser

Foreword by
Hermann Kopetz

Springer

# Event-Triggered and Time-Triggered Control Paradigms

# *REAL-TIME SYSTEMS SERIES*

*Consulting Editor:* **John A. Stankovic,** *University of Virginia*

**Other books in this series:**

**DEADLINE SCHEDULING FOR REAL-TIME SYSTEMS:** ***EDF and Related Algorithms,*** John A. Stankovic, Marco Spuri, Krithi Ramamritham, Giorgio C. Buttazzo, ISBN: 0-7923-8269-2

**HARD REAL-TIME COMPUTING SYSTEMS:** ***Predictable Scheduling Algorithms and Applications,*** by Giorgio C. Buttazzo, ISBN: 0-7923-9994-3

**REAL-TIME DATABASE AND INFORMATION RESEARCH ADVANCES,** by Azer Bestavros, Victor Wolfe, ISBN: 0-7923-8011-8

**REAL-TIME SYSTEMS:** ***Design Principles for Distributed Embedded Applications,*** by Hermann Kopetz, ISBN: 0-7923-9894-7

**REAL-TIME DATABASE SYSTEMS:** ***Issues and Applications,*** edited by Azer Bestavros, Kwei-Jay Lin and Sang Hyuk Son, ISBN: 0-7923-9897-1

**FAULT-TOLERANT REAL-TIME SYSTEMS:** ***The Problem of Replica Determinism,*** by Stefan Poledna, ISBN: 0-7923-9657-X

**RESPONSIVE COMPUTER SYSTEMS:** ***Steps Toward Fault-Tolerant Real-Time Systems,*** by Donald Fussell and Miroslaw Malek, ISBN: 0-7923-9563-8

**IMPRECISE AND APPROXIMATE COMPUTATION,** by Swaminathan Natarajan, ISBN: 0-7923-9579-4

**FOUNDATIONS OF DEPENDABLE COMPUTING:** ***System Implementation,*** edited by Gary M. Koob and Clifford G. Lau, ISBN: 0-7923-9486-0

**FOUNDATIONS OF DEPENDABLE COMPUTING:** ***Paradigms for Dependable Applications,*** edited by Gary M. Koob and Clifford G. Lau, ISBN: 0-7923-9485-2

**FOUNDATIONS OF DEPENDABLE COMPUTING:** ***Models and Frameworks for Dependable Systems,*** edited by G. M. Koob and C. G. Lau, ISBN: 0-7923-9484-4

**THE TESTABILITY OF DISTRIBUTED REAL-TIME SYSTEMS,** Werner Schütz; ISBN: 0-7923-9386-4

**A PRACTITIONER'S HANDBOOK FOR REAL-TIME ANALYSIS:** ***Guide to Rate Monotonic Analysis for Real-Time Systems,*** Carnegie Mellon University (M. Klein, T. Ralya, B. Pollak, R. Obenza, M. G. Harbour); ISBN: 0-7923-9361-9

**FORMAL TECHNIQUES IN REAL-TIME FAULT-TOLERANT SYSTEMS,** J. Vytopil; ISBN: 0-7923-9332-5

**SYNCHRONOUS PROGRAMMING OF REACTIVE SYSTEMS,** N. Halbwachs; ISBN: 0-7923-9311-2

**REAL-TIME SYSTEMS ENGINEERING AND APPLICATIONS,** M. Schiebe, S. Pferrer; ISBN: 0-7923-9196-9

**SYNCHRONIZATION IN REAL-TIME SYSTEMS:** ***A Priority Inheritance Approach,*** R. Rajkumar; ISBN: 0-7923-9211-6

**CONSTRUCTING PREDICTABLE REAL TIME SYSTEMS,** W. A. Halang, A. D. Stoyenko; ISBN: 0-7923-9202-7

**FOUNDATIONS OF REAL-TIME COMPUTING:** ***Formal Specifications and Methods,*** A. M. van Tilborg, G. M. Koob; ISBN: 0-7923-9167-5

**FOUNDATIONS OF REAL-TIME COMPUTING:** ***Scheduling and Resource Management,*** A. M. van Tilborg, G. M. Koob; ISBN: 0-7923-9166-7

**REAL-TIME UNIX SYSTEMS:** ***Design and Application Guide,*** B. Furht, D. Grostick, D. Gluch, G. Rabbat, J. Parker, M. McRoberts, ISBN: 0-7923-9099-7

# Event-Triggered and Time-Triggered Control Paradigms

**Roman Obermaisser**
*Vienna University of Technology*
*Vienna, Austria*

Foreword by Hermann Kopetz

**Springer**

ISBN: 0-387-23044-0 (eBook)
Print ISBN: 0-387-23043-2



Visit Springer's eBookstore at: http://www.ebooks.kluweronline.com
and the Springer Global Website Online at: http://www.springeronline.com

# Contents

# Foreword

More than twenty-five generations of semiconductor chips have followed Moore's law (which states that the number of devices that can be placed on a single die doubles about every two years). Up to today, this has led to the development of Systems-on-a-Chip (SoCs) with more than one hundred million devices. According to industry forecasts, this increase in device density will continue at least until the end of this decade, resulting in SoCs with about a billion devices. Despite this tremendous increase in the number of functions per chip, the reliability of a chip with respect to permanent hardware faults has remained more or less the same over the years – a mean-time-to-fail (MTTF) of about one thousand years. This implies that in the past decades the reliability per function has increased as dramatically as the increase in the number of devices per chip.

It is an architectural challenge to exploit these technological advances in function dependability in order to increase the dependability of services at the system level, particularly in the field of safety-critical control applications. In these application a service reliability at the system level of better than 100 000 years must be achieved. Despite the observed remarkable level of reliability of state-of-the-art semiconductor chips, such a high system reliability can only be achieved by the implementation of fault-tolerance. Fault tolerance can be realized by replicating functions at independent fault-containment units (FCU), i.e. on independent SoCs and matching the independently computed outputs by a (replicated) voter, outvoting the result of a faulty component. In order to achieve the necessary level of independence of the FCUs – also with respect to spatial proximity faults – a physically distributed architecture is an absolute requirement. The physical distribution of nodes that is needed in order to assure independent failures of FCUs is thus dictated by dependability concerns and will not be affected by a further increase of the functional capabilities of the ever more powerful SoCs.

Today's state of the art distributed systems, e.g., in the automotive industry, are federated. In a federated architecture every function is hosted at a dedicated node computer. The communication to the other nodes of a given distributed application subsystem is realized by a dedicated network. This leads to a high number of system nodes – close to 100 in current premium cars – and a large number of cables and connector points. The main advantages of a federated architecture are

the strong fault isolation properties and the physically constrained possibilities of error propagation. Other advantages relate to the clear management responsibility for a distributed application subsystem and the possibility to protect the intellectual property of the software provider.

A significant reduction in the number of node computers and the number of cables could be achieved if a single SoC could host several functions of different criticality and if many different virtual communication channels with known temporal properties could share a single physical wire. This reduction in the number of nodes and cabling points leads already to a marked improvement in the hardware dependability. A further dependability improvement can be realized if the integrated architecture provides structural support for the implementation of transparent fault tolerance by replicating functions on different SoCs and voting on the results.

However, even a slight interference between the diverse software functions implemented within a multi-criticality node could cancel out all those stipulated advantages. An integrated architecture must thus contain effective mechanisms for fault isolation, error containment and diagnosis in order not to lose the advantages of the federated approach.

The first part of this very readable book by Roman Obermaisser gives an excellent survey about existing architectures for safety-critical applications and discusses the issues that must be considered when moving from a federated to an integrated architecture. It then focuses on one key topic, the amalgamation of the event-triggered and the time-triggered control paradigm into a coherent integrated architecture. The architecture provides for the integration of independent distributed application subsystems by introducing multi-criticality nodes and virtual networks of known temporal properties. In order to reduce the complexity of application software development, a virtual network with a single name-space can span multiple physical networks and hide the physical gateways between these networks. The subjects of fault isolation, error containment and diagnosis are treated with utmost care in order to maintain the advantages of the federated approach and to avoid any increase in application software complexity when moving to an integrated environment. The architecture supports the migration of legacy software by emulating widely used communication interfaces, such as the CAN interface in both, the value and the temporal domain. The architecture has been implemented on a distributed TTP/C/LINUX prototype. The feasibility and the tangible advantages of this new architecture have been demonstrated on practical examples taken from the automotive industry. I am sure that the interested reader will gain deep insights into the architecture and design of integrated embedded systems, both at the conceptual and at the practical level.

Hermann Kopetz
Vienna University of Technology

# Preface

The shift from federated to integrated systems is a hot topic in many industries, such as the automotive and the avionic domain. In a federated system, each major function (e.g., autopilot in avionic system or brake-by-wire in automotive system) has its own dedicated computer system with internal redundancy, while an integrated system is characterized by the integration of multiple functions within a single distributed computer system. Federated systems have been preferred for ultra-dependable applications due to the natural separation of application functions, thus minimizing interactions and dependencies between the various autonomous computer systems. The ability to reason about the behaviour of an application function in isolation helps in controlling overall complexity. However, integrated systems promise massive cost savings through the reduction of resource duplication. In addition, integrated systems permit an optimal interplay of application functions, reliability improvements with respect to wiring and connectors, and overcome limitations for spare components and redundancy management. An ideal future system architecture combines the complexity management advantages of the federated approach, but also realizes the functional integration and hardware efficiency benefits of an integrated system.

The major contribution of this work is the introduction of a generic system architecture for integrated systems that preserves the advantages of the federated system approach. This goal is reached by the provision of error containment mechanisms through generic architectural services. In addition, the integrated architecture supports different paradigms of temporal control. It has been recognized that communication protocols fall into two general categories with corresponding strengths and deficiencies: event-triggered and time-triggered control. Event-triggered protocols (e.g., TCP/IP, CAN, Ethernet, ARINC629) offer flexibility and resource efficiency. Time-triggered protocols (e.g., TTP, SafeBus, FlexRay, Spider) excel with respect to predictability, composability, error detection and error containment. Through the support for both time-triggered and event-triggered communication activities, the integrated architecture is suitable for mixed-criticality and legacy integration. Safety-critical time-triggered applications coexist with event-triggered legacy applications and newly developed, non-critical event-triggered applications.

Our starting-point is a time-triggered architecture, for which we assume a set of four fundamental, basic services: a predictable, fault-tolerant time-triggered trans-

port service, clock synchronization, error containment, and a membership service. On top of these basic services, we construct higher-level services for handling imprecise temporal specifications in the event-triggered subsystem: an event-triggered transport service, gateway services, membership information for event-triggered application tasks, and additional error containment mechanisms for preventing propagation of software faults. Furthermore, this book devises a solution for the reuse of event-triggered legacy applications in the proposed system architecture. We describe a generic model for the establishment of existing event-triggered communication protocols on top of basic, time-triggered architectural services. This approach supports a gradual evolution of systems and improves event-triggered communication services through a reliable underlying time-triggered subsystem.

The CAN emulation in the Time-Triggered Architecture is an application of the generic system architecture for the integration of event-triggered and time-triggered services. The CAN emulation is of high industrial relevance for the automotive domain, since the CAN protocol is widely used in present day automotive networks for powertrain and body/comfort functions. Despite the use of time-triggered architectures in future by-wire cars, CAN is likely to remain as a communication protocol for non-safety critical functions due to the higher flexibility and average performance. Even for safety-related functions, CAN-based legacy applications will not be replaced instantly.

This work is a revised version of my dissertation, which was carried out at the Vienna University of Technology during 2002–2003. This work has been supported, in part, by the NextTTA Research Project 'High-Confidence Architecture for Distributed Control Applications' (IST-2001-32111).

The completion of this book would not have been possible without the continuous support from various people and institutions. Above all, I would like to thank my thesis advisor Prof. Hermann Kopetz for his valuable support and numerous fruitful discussions. I have been honored to share his long experience in the field of distributed real-time systems. I would also like to thank him for providing the foreword to this book. Furthermore, my gratitude goes to my secondary advisor Prof. Wolfgang Kastner for interesting discussions and helpful suggestions. I would also like to thank Jack Stankovic, who provided suggestions for my thesis and established the contacts with Kluwer Academic Publishers.

Thanks also to all my colleagues from the department of Technische Informatik at the Vienna University of Technology, who have given me their support by proofreading as well as a very pleasant working environment. I would particularly like to express my thanks and appreciation to Philipp Peti for his close cooperation and friendship and for his valuable inputs.

Comments and suggestions concerning this book will be welcomed and can be sent to me by e-mail at `ro@vmars.tuwien.ac.at`.

ROMAN OBERMAISSER

Chapter 1

# INTRODUCTION

Advances in computer and communication technologies have made it feasible to extend the application of embedded computer systems to more and more critical applications, such as automotive and avionic systems. Due to the many different and, partially, contradicting requirements, there exists no single model for building systems that interact with a physical environment. Well-known trade-offs are predictability versus flexibility, and resource adequacy versus best-effort strategies [Kopetz, 1997]. Thus, the chosen system model depends heavily on the requirements of the application.

For example, in safety-critical real-time control applications, such as X-by-wire systems in the automotive or avionic domain, a system's inability to provide its specified services can result in a catastrophe involving endangerment of lives and/or financial loss an order of magnitude higher than the overall cost of the system. These hard real-time systems must be designed according to the resource adequacy policy by providing sufficient computing resources to handle the specified worst-case load and fault scenarios.

At present, two different paradigms are prevalent in the design of real-time architectures. In *event-triggered architectures* the system activities, such as sending a message or starting computational activities, are triggered by the occurrence of events in the environment or the computer system. In *time-triggered architectures,* activities are triggered by the progression of global time. The major contrast between event-triggered and time-triggered approaches lies in the location of control. Time-triggered systems exhibit autonomous control and interact with the environment according to an internal predefined schedule, whereas event-triggered systems are under the control of the environment and must respond to stimuli as they occur.

The time-triggered approach is generally preferred for safety-critical systems [Kopetz, 1995b; Rushby, 2001a]. For example, in the automotive industry a time-triggered architecture will provide the ability to handle the communication

R. Obermaisser, *Event-Triggered and Time-Triggered Control Paradigms*,
DOI 10.1007/978-0-387-23044-3_1, © 2005 Springer Science + Business Media, Inc.

needs of by-wire cars [Bretz, 2001]. In addition to hard real-time performance, time-triggered architectures help in managing the complexity of fault-tolerance and corresponding formal dependability models, as required for the establishment of ultra-high reliability [Suri et al., 1995] (failure rates in the order of $10^{-9}$ failures/hour). The predetermined points in time of the periodic message transmissions allow error detection and establishing of membership information. Redundancy can be established transparently to applications [Bauer and Kopetz, 2000], i.e. without any modification of the function and timing of application systems. A time-triggered system also supports replica determinism [Poledna, 1995], which is essential for establishing fault-tolerance through active redundancy. Furthermore, time-triggered systems support temporal composability [Kopetz and Obermaisser, 2002] via a precise specification of the interfaces between subsystems, both in the value and time domain. The communication controller in a time-triggered system decides autonomously when a message is transmitted. The communication network interface is a temporal firewall which isolates the temporal behavior of the host and the rest of the system.

In non safety-critical (soft real-time) applications, however, the event-triggered control paradigm may be preferred due to higher flexibility and resource efficiency. Event-triggered architectures support dynamic resource allocation strategies and resource sharing. In event-triggered systems the provision of resources can be biased towards average demands, thus allowing timing failures to occur during worst-case scenarios in favor of more cost-effective solutions. If the correlation between the resource usages of different applications is known, resources can be multiplexed between different applications while providing probabilistic guarantees for communication latencies and sufficiency of buffering capacities.

In addition to the classification of distributed real-time systems into event-triggered and time-triggered systems according to the employed paradigm of control, one can perform a differentiation of computer systems according to the level of integration into *federated* and *integrated systems.* These two classes of systems differ in the allocation of functions to the available computer systems. In a federated system, every major subsystem is implemented on its own dedicated distributed computer system. Subsystems are interconnected by gateways in order to realize a limited level of coordination in the interaction with the controlled object. An integrated system, on the other hand, is characterized by the integration of multiple application subsystems within a single distributed computer system. An *integrated architecture* provides a framework for the construction of such an integrated system. By restoring the level of fault isolation and error containment of a federated system, an integrated architecture promises a better coordination of control functions, a significant reduction of hardware resources and an overall improvement in the dependability.

## 1.1 Goal of this book

Many computer systems in the avionics and automotive domain are mixed-criticality systems. For example, in-vehicle electronics in the automotive industry involve numerous functions with different criticality levels. The spectrum starts at control applications for comfort electronics like seat and window movement controls that are non safety-critical. Modern cars also contain safety-related functions such as engine management and anti-lock braking. X-by-wire systems, which use electronics for control without mechanical or hydraulic backup systems impose the highest reliability requirements in automotive and avionic applications.

The objective of this book is the development of an integrated system architecture for the coexistence of subsystems with different degrees of synchrony and criticality. A single distributed computing platform serves as a shared resource for different functions and supports both the event-triggered and time-triggered control paradigms.

The primary goal of the integrated system architecture is the support for ultra-dependable hard real-time systems through a stable set of architectural services. In particular, error containment mechanisms must ensure the protection of safety-critical functions from the effects of misbehaving functions of lower levels of criticality. Otherwise, the potential for error propagation from a non-critical function to a function of higher criticality would elevate the criticality of the first one to the level of the second one.

A secondary goal of the integrated architecture is the establishment of generic higher-level services tailored to non safety-critical applications. In these applications, emphasis lies on low-cost, flexibility and resource efficiency. For economic reasons, non safety-critical applications can be designed non resource-adequately. Furthermore, incomplete knowledge about computational latencies and input load can lead to imprecise temporal specifications. We aim at providing higher-level services in an integrated architecture for handling these imprecise temporal specifications and the resulting occasional timing failures. These services include a best-effort communication service, gateway services, membership information, and additional error containment mechanisms for preventing propagation of software faults.

This book devises solutions for the integration of the event-triggered and time-triggered communication paradigms in a proposed system architecture by layering event-triggered communication on top of a time-triggered communication service. The layering of event-triggered on top of time-triggered services promises to minimize the effects of the best-effort communication service on the basic transport service. Furthermore, the event-triggered protocol can benefit from the fault-tolerance mechanisms of the underlying time-triggered basic transport protocol. The event-triggered and time-triggered communication services, as well as the corresponding applications employ a single fault-tolerant distributed computer system. Event-

triggered and time-triggered functions share common computing resources, instead of using dedicated physical networks and node computers.

The proposed integrated system architecture will result in quantifiable cost reductions in the development and deployment of embedded systems in the areas of system hardware cost and maintenance. The integration of event-triggered and time-triggered communication services on top of a single physical network reduces physical wiring. In addition, the sharing of components via encapsulated execution environments for applications of different levels of criticality leads to a reduction of the overall number of components. In order to quantify these savings, we will estimate the implications of the sharing of physical networks and node computers onto the required number of hardware units on-board a high-end car.

Let us assume that the multiplexing of component and network resources will result in a reduction of 20% of the hardware units and a corresponding reduction in the number of wiring points of a car. On the other side, the new hardware units will be more powerful and may thus cost more. If we consider a typical high-end distributed system on board a car with 50 components, each node costing on average about € 35 and 1000 wires, each wire costing about € 0.5, then the total hardware cost of such a system is about € 2250. If an integrated system is deployed, the number of components will be reduced to 40 components of € 40 each (increase of the node cost by € 5), and the number of wires to 800. The total hardware cost will thus be reduced to € 2000 or € 250 per system. A hardware cost reduction of about 10% can thus be realized. The induced savings in the hardware domain during the production of 100 000 cars amount then to about € 25 Mio.

For estimating the implications with respect to maintenance cost, let us assume that the cost of maintenance of an electronic system onboard a car is about € 300 per car over the lifetime of a car. By reducing the number of components by 20% and the number of wiring points by 20%, a reduction of the maintenance cost by 20% can be expected. The induced savings in the maintenance domain in 100 000 cars amount then to about € 6 Mio, not considering the image gain of the manufacturer due to the improved dependability of its product.

A further economically relevant goal of this book is the reuse of event-triggered legacy applications. At present, the CAN protocol is widely used in automotive networks. In modern cars CAN is employed for powertrain, and body/comfort networks [Leen et al., 1999]. Body and comfort networks control seat and window movement and other non-critical user interfaces. Powertrain networks interconnect electronic control units (ECUs) for engine management, anti-locking braking (ABS), electronic stability programs (ESP) [Bosch, 1998], transmission control, and cruise control.

Driven by the introduction of X-by-wire functionality, future generations of cars will contain time-triggered networks [Bretz, 2001]. However, due to the higher flexibility and average performance, an event-triggered CAN communication service is likely to remain as the communication infrastructure for non safety-critical

applications in cars. Even for safety-related functions, time-triggered solutions will not replace CAN-based solutions instantly. In this context, the CAN emulation in a time-triggered environment enables the integration of applications for which a CAN communication service is preferable, as well as the integration of CAN legacy applications into a time-triggered computing platform. By providing CAN execution environments within nodes of a time-triggered system in combination with CAN overlay networks, there is the potential for a significant reduction in the number of ECUs and wiring. In addition to lower cost, this strategy increases reliability as wiring and connectors are currently a prevalent source of faults in the automotive area [Swingler and McBride, 1999].

We will demonstrate the utility of the concepts for the integration of event-triggered and time-triggered services by instantiating the proposed integrated system architecture for employing a CAN communication service on top of the Time-Triggered Architecture [Kopetz and Bauer, 2003]. This instantiation provides an authentic CAN communication service, as required for the integration of CAN-based legacy applications. We identify and establish significant properties of a conventional CAN network (message ordering, message permanence, message cancelability, transmission latencies). We further show that this CAN emulation solves prevalent problems of the CAN protocol [Bosch, 1991] (e.g., fault-tolerant atomic broadcast, inaccessibility). An implementation of this CAN emulation model serves as a proof-of-concept of our integrated system architecture.

## 1.2 Overview

The book is organized as follows: Chapter 2 introduces the basic terms and concepts that are used throughout this book. The first part of this chapter describes distributed real-time systems and concepts of dependability. Afterwards, we give a brief overview of different synchrony models and present the event-triggered and time-triggered communication system paradigms. In addition, we map these control paradigms to prevalent models of computation.

Chapter 3 focuses on the requirement of an integrated system architecture for ultra-dependable real-time systems. We discuss four integration directions, with emphasis on the necessary architectural services. We summarize the basic services of an integrated architecture for ultra-dependable systems and outline higher-level services for easing application development.

Chapter 4 presents the integrated system architecture for the event-triggered and time-triggered control paradigms. We relate this architecture to the well-studied models of synchronous and asynchronous systems. Subsequently, we specify the basic and higher-level services of the integrated system architecture. The second part of this chapter defines the underlying fault hypothesis and provides information about the construction of error detection and error containment mechanisms.

Chapter 5 performs an instantiation of the integrated system architecture for providing CAN communication services on top of the Time-Triggered Architecture.

We first give a brief introduction about the Time-Triggered Architecture and the CAN protocol. We identify significant properties of CAN and present a solution for emulated CAN communication services. The chapter ends with an overview of a prototype implementation.

Chapter 6 demonstrates the CAN emulation's ability to handle the communication needs of newly developed CAN applications and CAN-based legacy applications. The analysis occurs through a comparison of measurement results from the prototype implementation with simulations of a conventional CAN network. In particular, this chapter discusses the improvements of the CAN emulation in relation to a physical CAN network.

Finally, the book ends with a conclusion in chapter 7 summarizing the key results of the presented work.