Chapter 13
The Formal Design Model of a Real-Time Operating System (RTOS+): Conceptual and Architectural Frameworks

Yingxu Wang
University of Calgary, Canada

Cyprian F. Ngolah
Sentinel Trending & Diagnostics Ltd., Canada

Guangping Zeng
University of Science and Technology, China and University of California, Berkeley,USA

Phillip C.-Y. Sheu
Wuhan University, China and University of California, Irvine, USA

C. Philip Choy
University of Calgary, Canada

Yousheng Tian
University of Calgary, Canada

ABSTRACT

A real-time operating system (RTOS) provides a platform for the design and implementation of a wide range of applications in real-time systems, embedded systems, and mission-critical systems. This paper presents a formal design model for a general RTOS known as RTOS+ that enables a specific target RTOS to be rigorously and efficiently derived in real-world applications. The methodology of a denotational mathematics, Real-Time Process Algebra (RTPA), is described for formally modeling and refining architectures, static behaviors, and dynamic behaviors of RTOS+. The conceptual model of the RTOS+ system is introduced as the initial requirements for the system. The architectural model of RTOS+ is created using RTPA architectural modeling methodologies and refined by a set of Unified Data Models (UDMs). The static behaviors of RTOS+ are specified and refined by a set of Unified Process Models (UPMs). The dynamic behaviors of the RTOS+ system are specified and refined by the real-time process scheduler and system dispatcher. This work is presented in two papers; the conceptual and architectural models of RTOS+ is described in this paper, while the static and dynamic behavioral models of RTOS+ will be elaborated in a forthcoming paper.

DOI: 10.4018/978-1-4666-0264-9.ch013

INTRODUCTION

An *operating system* is a set of integrated system software that organizes, manages, and controls the resources and computing power of a computer or a computer network. It also provides users a logical interface for accessing the physical machine in order to run applications. A general-purpose operating system may be perceived as an agent between the hardware and resources of a computer and the applications and users. An operating system may be divided into three subsystems known as those of the *kernel* or *system management*, the *resource management*, and the *process management* (Dijkstra, 1968; Brinch-Hansen, 1971; Liu & Layland, 1973; Peterson & Silberschantz, 1985; Anderson et al., 1989; McDermid, 1991; Deitel & Kogan, 1992; Tanenbaum, 1994; Viscarola & Mason, 2001; Silberschatz et al., 2003; Wang, 2004, 2007). The kernel of an operating system is a set of central components for computing, including CPU scheduling and process management. The resource management subsystem is a set of supporting functions for various system resources and user interfaces. The process management subsystem is a set of transition manipulation mechanisms for processes and threads interacting with the system kernel and resources.

A *real-time operating system* (RTOS) is an operating system that supports and guarantees timely responses to external and internal events of real-time systems (Laplante, 1977; Sha et al., 1990; ISO/IEC, 1996; Ford, 1997; Bollella et al., 2002; Kreuzinger et al., 2002; Ngolah, Wang, & Tan, 2004). An RTOS monitors, responds to, and controls an external environment, which is connected to the computer system through sensors, actuators, or other input-output (I/O) devices. In a real-time system in general and an RTOS in particular, the correctness of system behaviors depends not only on the logical results of computation but also on the time point at which the results are obtained. Real-time systems can be divided into hard and soft real-time systems. In the former, a failure to meet timing constraints will be of serious consequences, while in the latter, a timing failure may not significantly affect the functioning of the system.

A great variety of RTOS's have been developed in the last decades (Lewis & Berg, 1998; Labrosse, 1999; Yodaiken, 1999; Rivas & Harbour, 2001; Lamie, 2008; ETTX, 2009). The existing RTOS's are characterized as target-machine-specific, implementation-dependent, and not formally modeled. Therefore, they are usually not portable as a generic real-time operating system to be seamlessly incorporated into real-time or embedded system implementations. Problems often faced by RTOS's are CPU and tasks scheduling, time/event management, and resource management. Efficient and precise methodologies and notations for describing solutions to these problems are critical in RTOS design and implementation. Generally, RTOS's require multitasking, process threads, and a sufficient number of interrupt levels to deal with the random interrupt mechanisms in real-time systems. In modern RTOS's, *multitasking* is a technique used for enabling multiple tasks to share a single processor. A *thread* is individual execution of a process in order to handle concurrent tasks. In addition, an *interrupt* is a request of an external device or internal process that causes the operating system to suspend the execution of a current low priority task, serve the interrupt, and hand control back to the interrupted process.

This paper develops a comprehensive design paradigm of the formal real-time operating system (RTOS+) in a top-down approach on the basis of the RTPA methodology. The conceptual model, architectural model, and the static/dynamic behavioral models of RTOS+ are systematically presented. In the remainder of this paper, the conceptual model of RTOS+ is described as the initial requirements for the system. The architectural model of RTOS+ is created based on the conceptual model using the RTPA architectural modeling methodologies and refined by a set of unified data models (UDMs). Then, the static