Chapter 6
Model-Driven Configuration of Distributed Real-Time and Embedded Systems

Brian Dougherty
Vanderbilt University, USA

Jules White
Virginia Tech, USA

Douglas C. Schmidt
Vanderbilt University, USA

ABSTRACT

Distributed real-time and embedded (DRE) systems are increasingly being constructed with commercial-off-the-shelf (COTS) components to reduce development time and effort. The configuration of these components must ensure that real-time quality-of-service (QoS) and resource constraints are satisfied. Due to the numerous QoS constraints that must be met, manual system configuration is hard. Model-Driven Architecture (MDA) is a design paradigm that incorporates models to provide visual representations of design entities. MDAs show promise for addressing many of these challenges by allowing the definition and automated enforcement of design constraints. This chapter presents MDA techniques and tools that simplify and automate the configuration of COTS components for DRE systems. First, the challenges that make manual DRE system configuration infeasible are presented. Second, the authors provide an incremental methodology for constructing modeling tools to alleviate these difficulties. Finally, the authors provide a case study describing the construction of the Ascent Modeling Platform (AMP), which is a modeling tool capable of producing near-optimal DRE system configurations.

INTRODUCTION

Emerging trends and challenges. Distributed real-time embedded (DRE) systems (such as avionics systems, satellite imaging systems, smart cars, and intelligent transportation systems) are subject to stringent requirements and quality of service (QoS) constraints. For example, timing constraints require that tasks be completed by real-time deadlines. Likewise, rigorous QoS de-

DOI: 10.4018/978-1-61692-874-2.ch006

mands (such as dependability and security), may require a system to recover and remain active in the face of multiple failures (Wang, 2003). In addition, DRE systems must satisfy domain-specific constraints, such as the need for power management in embedded systems. To cope with these complex issues, applications for DRE systems have traditionally been built from scratch using specialized, project-specific software components that are tightly coupled with specialized hardware components (Schmidt, 2002).

New DRE systems are increasingly being developed by *configuring* applications from multiple layers of commercial-off-the-shelf (COTS) hardware, operating systems, and middleware components resulting in reduced development cycle-time and cost (Voas, 1998). These types of DRE systems require the integration of 100's-1,000's of software components that provide distinct functionality, such as I/O, data manipulation, and data transfer. This functionality must work in concert with other software and hardware components to accomplish mission-critical tasks, such as self-stabilization, error notification, and power management. The software configuration of a DRE system thus directly impacts its performance, cost, and quality.

Traditionally, DRE systems have been built completely in-house from scratch. These design techniques are based on in-house proprietary construction techniques and are not designed to handle the complexities of configuring systems from existing components (Gokhale, 2002). The new generation of configuration-based approaches construct DRE systems by determining which combination of hardware/software components provide the requisite QoS (Alves, 2001; Chung, 2004; Morisio; 2002), In addition, the combined purchase cost of the components cannot exceed a predefined amount, referred to as the project budget.

A DRE system can be split into a software configuration and a hardware configuration. Valid software configuration must meet all real-time constraints, such as minimum latency and maximum throughput, provide required functionality, meet software architecture constraints, such as interface compatibility, and also satisfy all domain-specific design constraints, such as minimum power consumption. Moreover, the cost of the software configuration must not exceed the available budget for purchasing software components. Similarly, the hardware configuration must meet all constraints without exceeding the available hardware component budget. At the same time, the hardware and software configuration must be aligned so that the hardware configuration provides sufficient resources, such as RAM, for the chosen software configuration. Additional constraints may also be present based on the type and application of the DRE system being configured.

Often, there are multiple COTS components that can meet each functional requirement for a DRE system. Each individual COTS component differs in QoS provided, the amounts/types of computational resources required, and the purchase cost. Creating and maintaining error-free COTS configurations is hard due to the large number of complex configuration rules and QoS requirements. The complexity associated with examining the tradeoffs of choosing between 100's to 1,000's of COTS components makes it hard to determine a configuration that satisfies all constraints *and* is not needlessly expensive or resource intensive.

Solution approach → Model-driven automated configuration techniques. This chapter presents techniques and tools that leverage the *Model Driven Architecture* (MDA) paradigm (Mellor, 2004), which is a design approach for specifying system configuration constraints with platform-independent models (PIMs). Each PIM can be used as a blueprint for constructing platform-specific models (PSM)s (Poole, 2001). In this chapter, MDA is utilized to construct modeling tools that can be used to create model instances of potential DRE system configurations. These tools are then applied in a motivating example to