Chapter 4
Concern Separation for Adaptive QoS Modeling in Distributed Real-time Embedded Systems

Jeff Gray
University of Alabama at Birmingham, USA

Sandeep Neema
Vanderbilt University, USA

Jing Zhang
Motorola Research, USA

Yuehua Lin
Honda Manufacturing of Alabama, USA

Ted Bapty
Vanderbilt University, USA

Aniruddha Gokhale
Vanderbilt University, USA

Douglas C. Schmidt
Vanderbilt University, USA

ABSTRACT

The development of distributed real-time and embedded (DRE) systems is often challenging due to conflicting quality-of-service (QoS) constraints that must be explored as trade-offs among a series of alternative design decisions. The ability to model a set of possible design alternatives—and to analyze and simulate the execution of the representative model—helps derive the correct set of QoS parameters needed to satisfy DRE system requirements. QoS adaptation is accomplished via rules that specify how to modify application or middleware behavior in response to changes in resource availability. This chapter
presents a model-driven approach for generating QoS adaptation rules in DRE systems. This approach creates high-level graphical models representing QoS adaptation policies. The models are constructed using a domain-specific modeling language—the adaptive quality modeling language (AQML)—which assists in separating common concerns of a DRE system via different modeling views. The chapter motivates the need for model transformations to address crosscutting and scalability concerns within models. In addition, a case study is presented based on bandwidth adaptation in video streaming of unmanned aerial vehicles.

INTRODUCTION

The ability to adapt is an essential trait of distributed real-time and embedded (DRE) systems, where quality-of-service (QoS) requirements demand a system adjust to external environment changes in a timely manner. A DRE system typically consists of a stack of software layers that are coupled to a physical system (e.g., a control system within an automotive factory or avionics subsystem). Several capabilities are needed to provide adaptability within DRE systems, including (1) the ability to express QoS requirements in some manipulatable form, (2) a mechanism to monitor important conditions associated with the environment and physical system, and (3) a causal relation between the monitoring of the environment and the specification of the QoS requirements (Karr et al., 2001). Addressing QoS concerns via software adaptation has a concomitant effect on the physical system, such that a feedback loop is established between the physical parts of the system and the corresponding software.

Recent advances in middleware technologies have enabled a new generation of object-oriented DRE systems based on platforms such as Real-time CORBA and the Real-time Specification for Java (Dibble, 2008). Although middleware solutions promote software reuse resulting in higher development productivity (Schantz & Schmidt, 2001), provisioning and satisfying QoS requirements, such as predictable end-to-end latencies, remains a fundamental challenge for DRE systems, due to their distributed nature, unpredictable resource loads, and resource sharing. Moreover, specifying and satisfying QoS requirements in DRE systems often uses ad hoc problem-specific code optimizations, which impede the principles of reuse and portability.

To satisfy QoS requirements in the presence of variable resource availability, programmable QoS adaptation layers atop the middleware infrastructure have been proposed (Schantz et al., 2002). These QoS adaptation layers implement QoS adaptation policies, which include rules for modifying application or middleware behavior in response to a change in resource availability. The key idea behind such adaptation layers is to separate concerns with respect to QoS requirements. This separation provides improved modularization for separating QoS requirements from the functional parts of the software.

For example, the Quality Objects (QuO) project, developed by BBN, is an adaptation layer (Sharma et al., 2004) that extends the Object Management Group’s (OMG) Interface Definition Language (IDL). This extension, known as the Contract Definition Language (CDL), is a textual language that supports the specification of QoS requirements and adaptation policies in the style of a state-machine. QuO contracts written in CDL are compiled to create stubs that are integrated into the QuO kernel and are used to monitor and adapt the QoS parameters when the system is operational.