The Viable System Architecture

Charles Herring and Simon Kaplan

Department of Computer Science and Electrical Engineering
The University of Queensland
Brisbane, Queensland QLD 4072
{herring kaplan}@dstc.edu.au

Abstract

This paper presents the Viable System Architecture as a high-level reference architecture. It is component system architecture motivated by an emerging class of applications we classify as “complex systems.” The architecture is based on the Viable System Model: a cybernetic model of organizations. The concept of Viability is introduced as the overall quality desired of software for complex systems. We explain how viability is achieved by the interaction of a number of principles: autonomy and adaptation; recursion and hierarchy; and invariants and self-reference. The special structure of a component in this architecture is described in detail. The nature of an interface is also described. This unique component interface mechanism defines the component framework and provides for dynamic assembly of systems of sub-systems. We present an outline of a business-to-business e-commerce application to illustrate the qualities and principles expected from software systems developed based on the architecture. We are currently building a prototype of this system to verify and validate the architecture.

1. Introduction

We identify a class of software systems as being "complex systems" [1] [2]. Examples of complex systems include Smart Environments, Ambient Computing, Multi-Agent Systems, Adaptive/Intelligent User Interfaces and Business-to-Business e-Commerce. Building this type of system raises new software engineering challenges, and new software architectures are required to provide design and implementation guidance. Complex systems are characterized by large numbers of heterogeneous components with a high degree of interconnections, relationships and dependences. They exist in a dynamically changing environment that demands dynamically responding behavior. In other words, these systems must adapt to their environment.

We have developed the Viable System Architecture to address the requirements of this class of software. The architecture is based on a Cybernetic control theory model called the Viable System Model [3]. Our architecture defines a unique set of component interfaces that in turn defines the framework itself. We believe the special nature of the framework will permit development of protocols for dynamic assembly of systems from sub-system frameworks. We describe the architecture in detail and present an example application in the business-to-business e-commerce domain. We are currently implementing this application.

2. Beyond Objects, Beyond Components

In the paper “Beyond objects: A software design paradigm based on process control” [4], Shaw analyzes the “canonical” object-oriented design problem – the automobile cruise control. Her fundamental insight is that it is a control system and there is a body of engineering knowledge associated with such systems. She develops a software architecture, the “control paradigm”, based on classical feedback control theory (as opposed to a generic software development methodology.) A diagram of the cruise control system is shown in Figure 1.

Figure 1. Control Architecture for Cruise Control

She notes the control paradigm separates the plant (the Engine in this case) of the main process from the compensation for external disturbances, the control. This separation of concern yields appropriate abstractions that