Available online at www.sciencedirect.com
SCIENCE DIRECT
Mathematics and Computers in Simulation 70 (2006) 314–329
MATHEMATICS AND COMPUTERS IN SIMULATION
www.elsevier.com/locate/matcom

Architectural concepts and Design Patterns for behavior modeling and integration

Jean-Marc Perronne *, Laurent Thiry, Bernard Thirion

MIPS, Université de Haute Alsace, 12 rue des frères Lumière, 68093 Mulhouse, France

Available online 20 December 2005

Abstract

The design of the control software for complex systems is a difficult task. It requires the modeling, the simulation, the integration and the adaptation of a multitude of interconnected entities and behaviors. To tackle this complexity, the approach proposed consists in combining architectural concepts, Design Patterns and object-oriented modeling with unified modeling language (UML). In this context, the present paper describes a modeling framework to take greater advantage of these concepts and to design flexible, intelligible control software. It proposes to objectify the behaviors, which leads to a two-level architecture based on three concepts: resources software images of the controlled system-behaviors applied to these resources, and meta-behaviors, i.e. means for behavior integration and adaptation. Two Design Patterns are proposed to describe how to specify behaviors and define the means to combine and adapt them. The first pattern, Polymorphic Behavior, provides the means to define new behaviors for a system and to plug them dynamically. The second one, Structured Behavior, provides the means to use finite state machines for behavior switching. The originality of the framework is that it defines concepts, a UML-based notation and heuristics which specifies how to apply these concepts. To illustrate the elements mentioned, this paper uses the control software of a walking robot as a running example.

Keywords: Software architecture; Object-oriented modeling; Control software; Design Patterns; Complex behaviors

1. Introduction

The design of the control software for complex systems is a difficult task [29]. In particular, it requires means – i.e. concepts, notations and guides – for the integration and adaptation of a number of local behaviors within the framework of global control [36]. To tackle this complexity, one of the current approaches takes advantage of the know-how acquired from object-oriented software. In this context, the present paper proposes a modeling framework which explains how to capitalize this know-how in order to find a new way to design complex software systems which are controllers. The basic concept proposed by this paper is that of behavioral objects, which consists in reifying the behaviors of a subsystem. This founding principle opens an important field of investigation of complex systems. In particular, it helps to model all the elements considered (subsystems, control laws and interactions) in a uniform way with objects. The well-known principles of the object-oriented approach – classification, composition and delegation – can then be applied to the behavioral aspects. The notion of behavioral objects leads to an analysis guided by a two-level architecture that sets up three kinds of entities: resources, behaviors and meta-behaviors. These two levels

* Corresponding author. Tel.: +33 3 89 33 69 67; fax: +33 3 89 42 32 82.
E-mail address: jm.perronne@uha.fr (J.-M. Perronne).

0378-4754/$32.00 © 2005 IMACS. Published by Elsevier B.V. All rights reserved.
doi:10.1016/j.matcom.2005.11.004

must not be mistaken for the traditional notion of hierarchy. The first conceptual level includes entities which model resources, i.e. software images of the physical components. The resources help to model the structure of the controlled system and to specify the available services to make this structure evolve. The second conceptual level includes entities which model behaviors and allow the control of the previous elements. A behavioral object can then be considered as a resource for behavioral objects of a higher order. These behavioral objects, called meta-behaviors, help to integrate, adapt and coordinate other behaviors; they represent the third concept of the architecture. The heuristics associated with the present architecture matches a modeling step with each of the above-mentioned concept. The first step consists in modeling the controlled system with the different objects it is composed of and their relations. The second step determines the behaviors and the local laws which apply to each of the entities found. The last step uses meta-behaviors to coordinate the specified behaviors until the desired global control strategy is obtained.

The present paper is divided into three parts. The first part describes the main problems of modeling controllers which are particular software system. It explains how the complexity of the controlled systems is also to be found in the control software and presents the current ways to approach this complexity. The second part describes the modeling framework proposed. This framework includes concepts, notation and heuristics used to reduce the modeling efforts by describing the software components necessary for the global control of a complex system, the way to represent them and to organize them. The third part shows how the behaviors can be synthesized from Design Patterns adapted to control software. The control of a hexapod robot will be the example used throughout the whole study.

2. Software control of complex systems

2.1. Software and control

The control field includes the necessary know-how for the synthesizes of an algorithm dedicated to the control of a particular subsystem. For example, Astrom and Wittenmark [4] explain: (1) how to synthesize a control law with optimality and robustness constraints and (2) how to implement this law with an algorithm. So far, however, the control field has no general framework which would explain how to integrate the multitude of controllers necessary for the global control of a complex system, into a single software, in a flexible way. So, the software control of complex system leads to a software system which is also complex. To tackle this complexity, Van Bremen et al. [38] suggest to take advantage of the concepts from the field of multi-agent systems. Each controller is modeled by an agent which is a kind of active object performing a control law. The global behavior is then modeled as a society of agents which collaborate or are coordinated by agents of a higher level. Van Bremen and de Vries [37] present an implementation of these concepts for servo-controlled room temperature.

The present paper follows a similar approach. It shows how to take advantage of the object-oriented concepts to allow the easier software synthesis for complex control systems and it uses the example of the control of a legged robot [33] to illustrate the concepts.

This system, shown in Fig. 1, consists of a multitude of interdependent variables or entities which must be organized. To control it, it is necessary to integrate several types of controllers which perform each a particular behavior. In the present case, the control software must integrate local controllers to servo-control each leg and a global supervisor to synchronize the local motions and to set the walking speed. Each local controller can be decomposed into three simpler controllers: a retraction controller which allows the platform to move, a protraction controller which determines where and how to reposition a leg and a controller which coordinates the previous two controllers. The proposed modeling framework helps to tackle the complexity of such a system.

2.2. Object-oriented approach to control

One of the current ways to master the increasing complexity of control software consists in reusing concepts issued from software engineering. Sanz et al. [29] present the advantages and difficulties of such an approach. They explain that software engineering contains the necessary concepts to tackle the complexity and that using them helps to reduce the design efforts. However, their application to the control field requires new knowledge [18].

The elements from the software field which prove most promising in the control field are the object-oriented concepts – with the UML language [13] – and the architectural elements – with the Design Patterns [16]. Booch [6] and