MODELING PHYSICAL SYSTEMS BY COMPLEX STRUCTURAL OBJECTS AND COMPLEX FUNCTIONAL OBJECTS

Shamkant B. Navathe and A. Cornelio
Database Systems Research and Development Center
University of Florida, Gainesville, FL 32611, USA
e-mail: sham@ufl.edu.csnet

ABSTRACT

This paper describes the general properties of complex objects in engineering designs. There are two types of complex objects: (i) the complex structural objects which describe the physical composition of the design, and (ii) the complex functional objects which describe the behavior of the design and its components. Data manipulation operations on complex structural objects are governed by a set of structural invariants. Similarly, the validation of functional abstraction is governed by a set of functional invariants. The structure-function interactions are represented by interaction objects that describe a set of mappings. These three object types constitute the Structure-Function paradigm. The S-F paradigm can be used to represent engineering designs and active environments, monitor manufacturing operations and industrial processes, and carry out simulations.

1 INTRODUCTION

Object-oriented systems are rapidly becoming acceptable for modeling many non business applications, such as office information systems, [BAN87], engineering CAD/CAM, [KET85, KEM87], robotic workcell design [JAY88], spatial information, [DAY87], etc. The power of these modeling systems lies in their ability to represent the semantics of structures by including the operations along with the data. Inheritance of data and operations reduces the time spent in developing new applications by sharing previously developed code.

We have adapted the object-oriented paradigm (for engineering design applications) by modeling the physical configuration of a design by structural objects, the behavioral aspect of a design by functional objects and the interface between the structural and functional objects by interaction objects. We call this modeling principle as the Structure-Function paradigm, or **S-F paradigm.** The S-F paradigm preserves the structural and behavioral schemas of an application by modeling and abstracting the structures and functions independently and relating these schemas by a well defined interface. This paradigm provides a platform for (a) modeling and analyzing engineering designs, (b) representing manufacturing tasks, and (c) monitoring industrial processes. The reasons for proposing the S-F paradigm are briefly described below, [COR89b] provides an indepth analysis of the S-F paradigm. Our approach applies to any environment with physical objects having visible interfaces. This includes examples from software engineering, biological systems, and computer hardware.

In the engineering design domain, there are complex interactions between a system's structure and behavior. This interaction is further complicated because a structure can serve many functions (either independently or together) and a function can have many alternate structural implementations. For example, a resistor (structure) can function as a load, voltage to current converter,

current to voltage converter, voltage bias, etc.; on the other hand, the function of lifting an object can be done by many alternate structures--robot, crane, fork-lift, etc. There is some work in artificial intelligence, where, for the sake of better reasoning, knowledge is modeled more precisely by separating functions from structures [DAV84]. In our approach, we explicitly represent the associative knowledge between structures and functions by interaction objects [COR89b].

Abstractions in the structural and functional domains for most realistic engineering applications are not isomorphic, i.e., a set of structures are aggregated according to their physical configuration and spatial locality, whereas functions are aggregated according to system behavior. Therefore, in the S-F paradigm, we extend the object oriented principle by autonomously abstracting the structural objects and the functional objects and then relating these domains by a well defined interface (made of interaction objects). We have shown with a real life robotic workcell design example in [COR90a] that the system naturally supports engineering design knowledge and simulation data. The other benefit that accrues from the S-F paradigm is the straight-forward integration of structural information with domain-specific application programs. This capability alleviates the most common bottleneck (of incompatible systems) in design automation. The details of this integration are the subject of ongoing research [COR90b].

An important characteristic of design information is that the structures and functions are aggregated to form complex structural objects [BAT84, BAT85] and complex functional objects. [KIM87a, KIM87b, KIM89] describes an object-oriented system for complex objects. In this paper, we show that the S-F paradigm extends the complex object definition of [KIM87b] by (a) including Assembly Relations (which are interconnection relationships among structural components) in the complex structural object definition along with the PART-OF relationship; (b) specifying the abstraction of complex structural objects in terms of the external features of its sub-structural objects; (c) defining complex functional objects and complex functional object hierarchies to model active data; and (d) validating the correctness of functional equivalence between two levels of a functional hierarchy.

Complex structural objects and complex functional objects are the primary constructs to model a design's structure and behavior in the S-F paradigm. In this paper, we concentrate on the properties of a complex structural object and propose data manipulation rules that are derived from a set of structural invariants. These invariants form the basis for inserting, deleting, and updating complex objects in a structural hierarchy. Similarly, the abstraction of complex functional objects is described by a set of functional invariants. Functional invariants formally state the correctness conditions between an abstract functional object and a set of sub-functional objects.

The focus of this paper is the definition and abstraction of complex structural and complex functional objects in the S-F paradigm. The next section briefly introduces the S-F paradigm. Section 3 describes complex structural objects. Section 4 describes complex functional objects. Section 5 is the conclusion.

2 THE S-F PARADIGM

The underlying principle of the S-F paradigm is to have three distinct types of objects: Structural, Functional, and Interaction. In this paper, our focus will be on the structural and functional objects.

A **Structural Object,** S, is a two-place tuple of features, S: <Es, Is> where Es are the *external features* of S and Is are the *internal features* of S. The external features are the interface