

# A Data Model for Engineering Design Objects

Alexandros Biliris  
Computer Science Department  
Boston University  
Boston, MA 02215  
Email: biliris@bu-cs.bu.edu

ACM, IEEE, International  
Conference on Data and Knowledge  
Systems for Manufacturing  
and Engineering,  
Gaithersburg, Maryland,  
October 1989, pp.49-58

## Abstract

*Design objects in CAD applications have versions and participate in the construction of other more complex design objects. In this paper we describe data model aspects of an experimental database system for CAD applications, called Pegasus. Our model is based on previously published work on extensible and object-oriented database systems. The novel idea of Pegasus that is presented in this paper is the reconciliation of two subtyping (inheritance) mechanisms: the first, called refinement, is based on the usual semantics of schema copying and the second, called extension, is based on the inheritance semantics between prototypes and their extensions. We use these modeling elements to show how generic and version objects as well as component occurrences of (generic or version) components can be modeled.*

## 1. Introduction

A CAD product is an aggregation of design objects and associated documents. For example, a product in a mechanical application is associated with a variety of drawings such as detail and assembly drawings, exploded assembly drawings, etc. These design objects are frequently called *representations*, [Katz87]. A design object may have many *versions*. Each version represents a particular description of the design object as it has been defined by a user at some point in time. The object that represents all versions of the (semantically) same object is called *generic*, [Ditt88]; it keeps data about its versions, their relationships, and their common properties. A version object is associated with exactly one generic object. Versions of a single generic object  $x$  form what is frequently called the *version set* of  $x$ .

A design object may reference (consist of) a number of other objects that in turn may be the constituents of other objects. Such objects that are composed of other objects are called *complex*, [Lori83], or *composite*, [Bane87], the hierarchical composition of a complex object is referred to as the *configuration hierarchy* of the object, and the constituent objects are called *components*. For example, the design item Car (a composite object) consists of a frame, wheels, doors, seats, etc., which are the component objects of

Car. Of course, a component object (generic or version) is by itself another design object and as such it may have its own components.

Each reference by a complex object to one of its components corresponds to a *component occurrence*<sup>1</sup> of the referenced object. For example, each of the four wheels that are components of the design item Car, correspond to four different component occurrences of the same single design object Wheel. Figure 1 shows two generic design objects, Wheel and Frame; currently, there are three versions of Wheel and two versions of Frame available to engineers to design a car. The figure shows four component occurrences of Wheel and one of Frame. Each component occurrence describes how the component object participates in the composite object. In this example, attributes of component occurrences of wheels may include a name (e.g., left front wheel), and a transformation matrix that describes where the component is located relative to the composite object. As we may see from the figure, the designer of the car has chosen version v1 of Frame to be used in his design. On the other hand, component occurrences of Wheel reference the generic object. A *generic reference* leaves the exact version unspecified. Generic references provide the means to an engineer to postpone specific decisions about non essential details for a later time. They also provide the means to reference an object  $x$  that is itself under development, perhaps by another group of engineers, without prematurely binding this reference to a specific version of  $x$  that happens to be currently available (if there is one available at all). This is important because in large design projects different users edit different components concurrently.

Briefly, the minimum requirements of a database management system for design applications include the following:

---

<sup>1</sup> In [Bato85] a component occurrence is called *component instance*. We avoided the word *instance* since it is used in a different context in object-oriented languages.