Modelling of Mechatronic Systems by an Object–Oriented Data Model

U. Neerpasch, W. Schiehlen

Institute B of Mechanics, University of Stuttgart, Pfaffenwaldring 9, D–70550 Stuttgart

Abstract

An object oriented data model is defined to describe multibody systems. Extensions for modelling mechatronic elements like sensors and actuators within the multibody system as well as interfaces to other dynamic systems have been developed. An implementation in a neutral modelling kernel and a format to store the description of a multibody system on a data exchange file are directly derived from this data model. Data converters transfer these data to several multibody formalisms.

1 Introduction

In this paper an object–oriented data model for multibody systems with extensions to mechatronic elements is described. A multibody system defined by this data model consists of rigid bodies connected by ideal joints and force elements. Measurement elements called sensors deliver internal, time dependent quantities like distances, accelerations as well as forces. Position actuators are defined to model drives within the multibody system.
The data model is independent of a specific multibody program and can therefore be used as a neutral format for the exchange of multibody system descriptions. The datamodel is given as a block with input/output interfaces for the connection to control units or other elements.

2 Object–Oriented Data Model

Engineering applications, such as multibody systems may be described by an object oriented data model in a natural and efficient way, following the discussion of a data model by Otter, Hocke, Daberkow, Leister [1]. The data model for the description of multibody systems is based on the simple, neutral, object–oriented data model due to Ullman [2].

In an object–oriented data model, the structure of the objects and their behaviour are described by classes. A class description consists of two parts: the scheme description of the object type and the specification of the available methods. Both aspects of a class are discussed in more detail in the following sections.

Object types:
The first part of a class description consists of the definition of the object type. At a basic level the data model supports a set of elementary data types, like integer values, real values or character strings. Furthermore, multi–dimensional arrays of the elementary data types are supported. New object types are defined by building composed or complex object types out of already defined object types (**recordof**) or by building collections of a number of objects of the same class (**setof**) and by deriving class descriptions by inheritance from superclasses, according to Ullman [2]. Applying these rules, arbitrary complex object types can be defined based on a small set of elementary data types.

Methods:
The second part of a class description consists of the specification of the available methods.

The data model distinguishes between administrative methods like creating, deleting or manipulating objects and class specific methods which can only be applied to objects of a specific class.

3 Description of Multibody Systems with Mechatronic Elements

Multibody Systems consist of material bodies (parts) connected by constraint elements (joints) and coupling elements (forces, torques), see Schiehlen [3]. They are well qualified for the dynamical analysis of machines, mechanisms, robots, and vehicles.

A multibody system is defined by an object of class *mbs* (multibody system), which is derived form the class *block*. Class *block* describes a general dynamical system and is characterized by input and output—signals, parameters, and internal signals which depend on the mathmatical model of the block.

A multibody system is essentially composed of the two basic elements: class *part* and class *interact*, see figure 1.

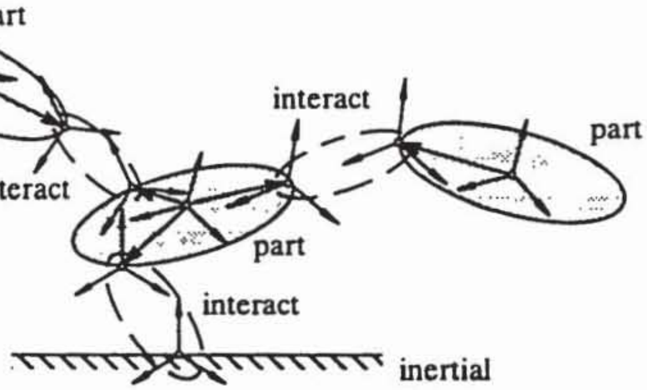

Figure 1: Elements of a multibody system

Class *part* defines a rigid body as a collection of coordinate systems, or frames, respectively. An object of class *frame* is described with respect to a reference frame on the same part and provides operations to evaluate the position vector and the rotation matrix from the reference frame to the frame. Class *rigid* is a subclass of class *part*. It has all the characteristics of the superclass and additionally the component **body** of class *body*. Class *body* is used to characterize the mass and the inertia tensor of the rigid body.

An object of class *interact* describes the interaction between one frame on a first part and one frame on a second part. Class *interact* has the components **connect** and **member** which form the class *connect* and *member*, respectively. The names of the two parts and two frames of the interaction element are stored in the object of class *connect*. The object of class *member* consists of the components **joint**, **force** and **sensor**.The object of class *joint* defines the restrictions of the relative motion between the two frames imposed by an interaction element. Component **force** is a set of objects of class *force* and defines the forces and torques exerted by the interaction element. Finally component **sensor** is a set of objects of class *sensor* which serves as a superclass. The derivations of this class will be discussed later.

Due to inheritance, this class description of a multibody system represents a decomposition into basic elements. Therefore the class *mbs* consists of the components of class *block* and additionally of the components **global**, **part** and **interact**. Component **global** contains all the data needed for the overall multibody system like the definition of gravity. Component **part** is a set of objects of class *part*. Similary, component **interact** is a set of objects of class interact. The class hierarchy of this decomposition is show in figure 2.

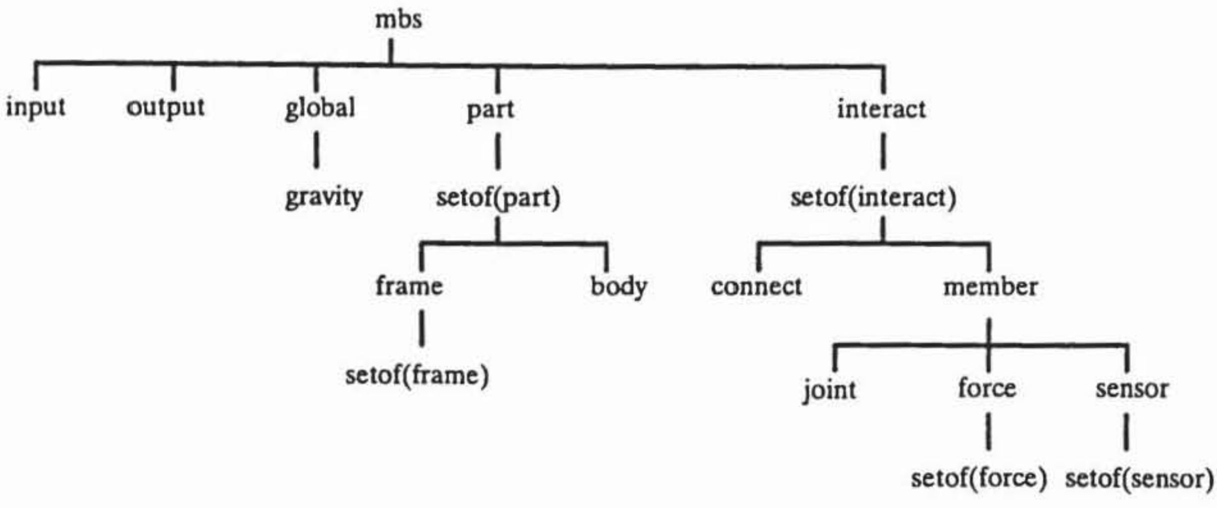

Figure 2: Object hierarchy of the multibody system data model

The data model for multibody systems has been extended in the direction to mechatronics. Classes are added to describe elements of mechatronic systems like sensors and actuators within the multibody system. Sensor elements are used to determine quantities that occur between two frames, e.g. kinematic quantities, applied forces, and reaction forces. These quantities can be used as input signals for other dynamical systems. Class descriptions for rheonomic joints are available for the modelling of position actuators. The connection of these elements which are derived from the description of class *joint* with other dynamical systems like controllers is realized via strictly defined interfaces.

Description of Sensor Elements

An object of class *sensor* defines quantities that are not explicitly defined in the datamodel to be computed and resolved in a desired frame.

Three classes of sensor elements *srel*, *sab* and *slin* are derived from the basic class *sensor* to specify a frame to which the results have to be transformed. An object of class *srel* consists of the components **inpart** and **inframe** to specify an arbitrary frame. Class *sab* is developed to refer to the inertial frame or one of the two frames specified in the object *connect* for the output of the desired quantity. A class *slin* is defined to compute the amount of a certain quantity. To specify the observed quantity between two frames, several classes are derived by inheritance from these three basic classes. Figure 3 shows the hierarchy of the objects of class *sensor*.

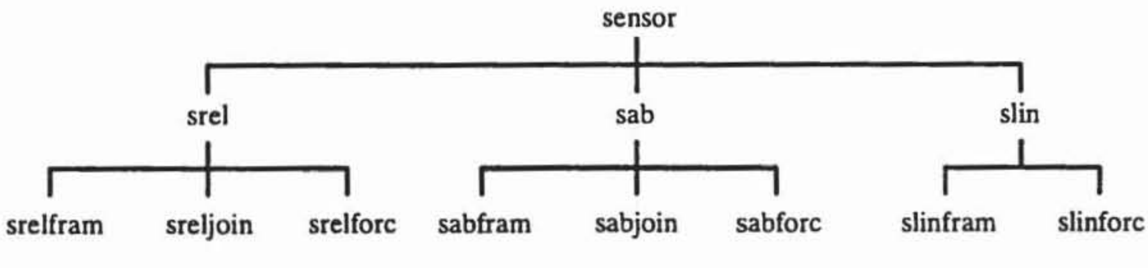

Figure 3: Object hierarchy of class *sensor*

The classes *srelfram*, *sabfram* and *slinfram* are defined to observe kinematic quantitites between two frames. Objects of class *sreljoin* and *sabjoin* are used to analyze reaction forces or joint coordinates. In a similar way, the classes *srelforc*, *sabforc* and *slinforc* are defined to obtain actual exerting forces of coupling elements. For a detailled description refer to Seybold and Neerpasch [4].

Description of Actuators

Classes to describe position actuators are derived by inheritance from objects of class *joint* for modelling the behaviour of multibody systems containing rheonomic constraints. Based on the description of class *joint*, new components **pos**, **vel** and **acc** are added to enable the definition of relative position, velocity, and acceleration between the two connected frames. If more than one component is used, the integrity will be checked. Furthermore, components are added to define the initial conditions of the actuator with respect to position and velocity. Using this scheme of inheritance the classes *revrh* and *transrh* describing a revolute and translational rheonomic joint, respectively, are defined. The class *jgenrh* (joint general rheonomic) allows the description of complex actuators with any combination of free, blocked or driven directions of movement.

A class to describe force actuators is to be defined. All actuators are driven by external signals e.g. they may be functions of time. The signals are transmitted via the object of class *input* which realizes the input interface for signals of other dynamical systems and the connection to the elements defined within the class *mbs*.

4 Implementation and Data Exchange

The object oriented data model for multibody systems and its extensions has been implemented using RSYST, a software environment for scientific and engineering applications. Another implementation, called DAMOS−C has been realized by Daberkow [5]. DAMOS−C represents a spezialized modelling kernel for multibody systems consisting of a data base and methods acting on this data base. Using this methods to access the data means a complete data encapsulation, since the structure of the data on the data base is hidden by the method. The user only has to know the interface of the method but not the structure of the data base.

DAMOS−C is suitable to be used as a neutral data exchange interface between several programm packages for the analysis of mechanical systems.

5 Summary

New classes with respect to mechatronic systems have been derived by inheritance from an existing object−oriented data model for multibody systems.

Class *sensor* and all of its subclasses enables to measure internal, time dependent quantities of the multibody system. The development of class descriptions for rheonomic constraints enables the integration of position actuators in multibody systems.

6 References

[1] Otter, M.: Hocke, M.; Daberkow, A.; Leister, G.: Ein objektorientiertes Datenmodell zur Beschreibung von Mehrkörpersystemen unter Verwendung von RSYST. Stuttgart: Universität, Institut B für Mechanik, Institutsbericht IB−16

[2] Ullman, J. D.: Principles of Database and Knowledge−Base Systems, Volume 1. Computer Science Press, 1988.

[3] Schiehlen, W: Technische Dynamik. Stuttgart: Teubner, 1986.

[4] Seybold, J.; Neerpasch, U.: Erweiterung des objektorientierten Datenmodells zur Beschreibung von Mehrkörpersystemen. Stuttgart: Universität, Institut B für Mechanik, Institutsbericht IB−24, Mai 1993.

[5] Daberkow, A.: DAMOS−C, Beschreibung der Programmschnittstelle der Klassen− und Methodenbibliothek für die Modellierung von Mehrkörpersystemen. Stuttgart: Universität, Institut B für Mechanik, Institutsbericht IB−23, 1992.