NOVI SAD J. MATH.
VOL. 30, NO. 2, 2000, 125-132

Proceedings of the TARA 2000 Conference
Novi Sad, Yugoslavia, September 6-7, 2000

SPECIFICATION OF THE CLASSES FOR GENERATING ROBOTIC MECHANISM MODELS

Miloš Racković[1]

Abstract. Automatic generating of the symbolic mathematical models of robotic mechanisms belongs to the class of problems of combinational optimisation. To cope with the mathematical complexity in solving this problem one requires developing of special algorithms and data structures. One way of dealing with this problem is the application of object-oriented approach to design of the system for mathematical modelling of robotic mechanisms. This paper presents formal specification of the process for forming the robotic mechanism model using the Unified Modelling Language.

AMS Mathematics Subject Classification (1991): 68Q40, 70B15

Key words and phrases: object-oriented specification, robotic mechanism models

1. Introduction

Considerable progress in modelling robotic mechanisms has been achieved by introducing the numeric-symbolic [1] and symbolic methods which develop special data structures for representing analytic expressions of the model and enable reduction of numerical complexity of the generated model. In [2, 3] a database management system was introduced into the process of modelling robotic mechanisms and the basic Newton-Euler method for forming the model of simple kinematic chain dynamics in closed form [1] has been expanded in such a way as to enable modelling of both complex and closed kinematic chains using the notations introduced in [4]. Now the model of the robotic mechanism in closed form [2, 3] can be expressed as:

$$P = H(q,\Theta)\ddot{q} + \dot{q}^{\top}C(q,\Theta)\dot{q} + +h^{G}(q,\Theta) + B(q,\Theta)\sigma \tag{1}$$

where:

$H = H(q,\Theta) : R^n \times R^m \to R^{n\times n}$ - inertial matrix of the mechanism;

[1]Faculty of Sciences, Institute of Mathematics, Trg Dositeja Obradovića 4, 21000 Novi Sad, Yugoslavia e-mail:rackovic@unsim.ns.ac.yu

$C = C(q, \Theta) : R^n \times R^m \to R^{n \times n \times n}$ - matrix of the Coriolis and centrifugal effects;

$h^G = h^G(q, \Theta) : R^n \times R^m \to R^n$ - gravity vector;

$B = B(q, \Theta) : R^n \times R^m \to R^{n \times 6}$ - matrix of chain closure;

$\sigma \in R^6$ - reaction vector of chain closure.

A detailed mathematical analysis shows that both kind of model expressions, which are basic mathematical expressions for forming the dynamic mechanism model and the fully-developed analytic expressions of the model, are of the same polynomial form:

$$Y = \sum_{i=1}^{N} k_i \cdot \prod_{l=1}^{L[i]} x_{il}^{e_{il}} \tag{2}$$

where:

Y - robotic quantity to be calculated;

k_i - constant coefficient related to the i-th addend;

x_{il} - one of the variables of the robotic mechanism model represented by its name.

e_{il} - exponent of the l-th variable of the i-th addend.

In [5, 6] is introduced the object-oriented approach to design a system for symbolic modelling of robotic mechanisms. The formal specification of the system is given in Unified Modelling Language (UML) [7] by use case and class diagrams. The main task of this paper is to specify in more details the first process of the complete system, which is the process for forming the starting model of the mechanism. Methods concerned with this process are specified using the sequence diagrams.

2. Class diagram

The system statics is represented by the class diagram in Fig. 1, [6]. This diagram represents the classes which store information about the model expressions.

The basic classes of the system are **Calculating graph** and **Analytic expressions**. The class **Calculating graph** represents mathematical expressions which form the mathematical model of the mechanism dynamics, and the class **Analytic expressions** represents the fully-developed analytic expressions of the model. Both classes have the attribute *phase* which contains the integer value indicating the phase in the process of modelling.

In order to represent the polynomial expressions (Eq. 2) we introduced the abstract classes **Variable** and **Addends**. The class **Variable** is used to represent all the variables in polynomial expressions which stand for the robotic quantities. Each variable has the attributes *code* and *value*. The attribute *code*

uniquely determines the variable and the attribute *value* stores the numerical value of the robotic quantity which is represented by that variable. The class **Addends** serves for connecting the expression with its addends. This class contains the attributes *code*, *sn* and *value*. The attribute *code* stores the code of the variable from the left-hand side of the equation sign in the expression, the attribute *sn* stores the serial number of the addend and the attribute *value* stores the value of the constant coefficient of the addend.

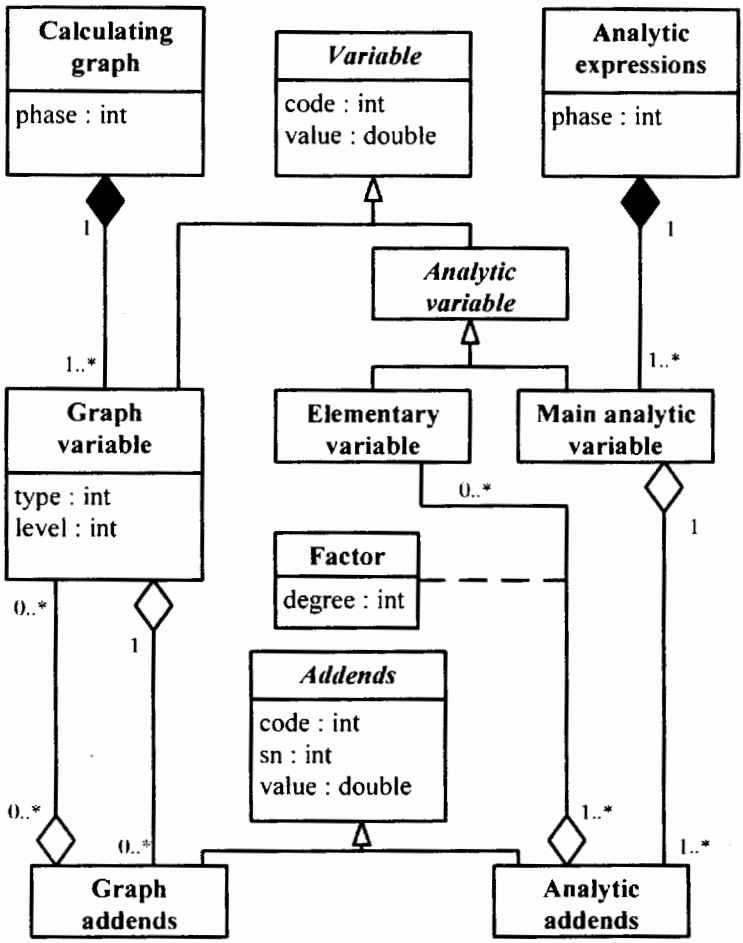

Figure 1: Class diagram

Although both expressions are of the same type we must introduce different classes for their representation in order to distinguish them in the process of forming the model and reducing its calculating complexity. We specialise the class **Variable** on the classes **Graph variable** and **Analytic variable**. The class **Addends** is specialised on the classes **Graph addends** and **Analytic**

addends for the same reason of distinguishing the two kinds of expressions.

The class **Analytic variable** does not contain any additional attributes apart from the inherited ones. On the contrary, the class **Graph variable** in addition to the inherited attributes have two additional attributes *type* and *level*. The attribute *type* takes values from the set {'0','1','2','3'}, where '0' denotes that the variable has a constant numerical value stored in the attribute *value*, '1' denotes that this variable is one of the robotic mechanism parameters (i.e. mass, moment of inertia,...), while '2' denotes that this variable represents some of the mechanism's generalised coordinates, or the sine or cosine of some of the generalised coordinates. The value '3' denotes that calculation of this variable is described by an expression. The attribute *level* is introduced to enable calculating of the robotic quantities by bottom-up navigating through the database.

The class **Analytic variable** is an abstract class and it is also specialised on the classes **Elementary variable** and **Main analytic variable** because the fully-developed analytic expressions contain only two kinds of variables. In the expressions for generating the mechanism model there are variables which are on the left-hand side of the equality sign in one expression and on the right-hand side of the equality sign in other expressions, so only one class **Graph variable** is used for representing all the variables from the calculating graph.

Let us describe storing of the starting model expressions. The class **Graph variable** contains all the variables (each variable takes a new instance of the class) and represents the both sides of the expressions. At the highest calculation level there are variables which are only on the left-hand side of the equality sign in expressions. These variables are connected to their addends by aggregation to the class **Graph addends**, where the constant coefficients of every addend are stored in the attribute *value*, so that each addend takes a new instance of the class (different serial number). Connection between the addends and their factors is represented by aggregation from the class **Graph addends** to the class **Graph variable**. Each factor is an instance of the class **Graph variable** because if it is not the basic robotic quantity, then there is an expression in which it appears on the left-hand side of the equality sign. This expression is again represented by the analogous connections.

In this way all the expressions forming the mathematical model of the mechanism are represented recursively in the form of calculating graph starting from the variables on the highest calculation level to the basic variables. We use the term *graph* instead of *tree* because some quantities may participate in more than one expression of the quantities from the higher calculation level. The fully-developed analytic expressions are stored in a way analogous to the previous case, but we will not describe this here because the process for forming the starting model does not use these expressions.

3. Forming of the robotic mechanism model

In [5, 6] the use case diagram *Modelling of the robotic mechanisms* is given, which describes the interaction between the system, user and the outer systems. These out systems (*Method for model generation*, *Topology and parameters of the mechanism*, *Generalised coordinates* and *Mechanism model*) supply the input and take output from the system for modelling of robotic mechanisms.

Here we describe only in brief the first use case which forms the mathematical model of the robotic mechanism. The actor *User* starts the use case *Model forming*, which on the basis of the method for forming the mathematical model of the robotic mechanism dynamics and the data on the mechanism's topology and parameters, generates in the database a complete model of the dynamics in the form of a calculating graph. The system's inputs are one of the existing methods for modelling of robotic mechanisms, containing the mathematical expressions to form the mechanism's model taken over from the *Method for model generation*, and data about topology of the concrete robotic mechanism taken over from *Topology and parameters of the mechanism*.

In this paper we adopted the approach of forming the robotic mechanism's mathematical model in closed form, according to Newton-Euler's method (Eq. 1). On the basis of the mathematical expressions taken from the *Method for model generation* and the concrete mechanism topology given in the *Topology and mechanism parameters*, a calculating graph is formed as an instance of the class **Calculating graph**.

The graph nodes represent the quantities participating in the mathematical expressions of the model forming method. Each mathematical expression that has been taken from the modelling method is stored in the object **Calculating graph** so that the quantities participating in the expression are interconnected in a way enabling the later calculation of the expression which is described in the previous section.

The sequence diagram specifying the process of forming the robotic mechanism model is given in Fig. 2.

The actor *User* initialises the object of the **Calculating graph** class and then its method *graph_form* is called to generate the model of the robotic mechanism in the form of calculating graph. The object of the class **Model expressions** contains all model expressions (Eq. 2) for chosen forming method and robot configuration. These expressions are one by one taken by the method *get_exp* and then stored in the form of calculating graph by the method *mem_exp* which belongs to the class **Utility**. This class contains utility methods and does not serve for storing the polynomial expressions. For this reason this class and the class **Model expressions** are not included in the class diagram in Fig. 1.

In order to store the expressions, we first have to initialise the level of the variable from the left-hand side of the equality sign (method *init_level*). Then we have to check all variables from the right-hand side of the equality sign, which is achieved by the double loop (one for the addends and one for the factors of

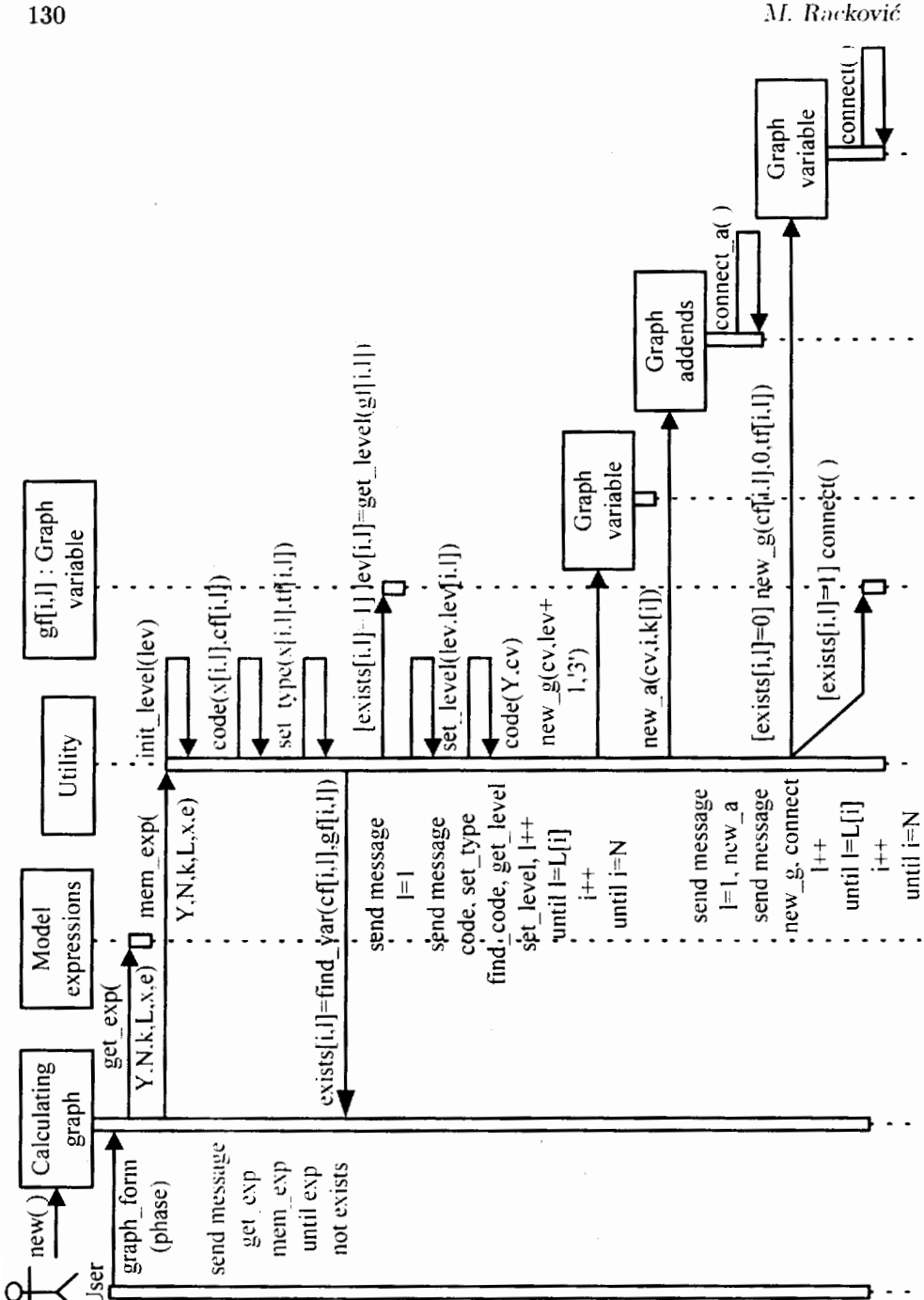

Figure 2: Sequence diagram

the expression). For each factor, the code (method *code*) and the type (method *set_type*) are determined. We also have to check if the factor has already been stored in the model (method *find_var* which belongs to the class **Calculating graph**). If it was we must read its level (method *get_level*), and if it was not, this means that this variable represents the parameter of robot mechanism or the constant quantity, and that its level has zero value. We need the levels of all the factors in order to determine the proper level of the variable from the left-hand side of the equality sign, as the maximum of all the levels plus one (method *set_level* finds the maximum).

When we checked all factors we can store all variables from the expression and connect them in the form of a calculating graph. First we code (method *code*) and memorise the variable from the left-hand side of the equality sign (method *new_g* which belongs to the class **Graph variable**). Then we have to store all addends and factors in a double loop analogous to the previous one. The addend is memorised by the method *new_a* and connected to the variable from the left-hand side of the equality sign by the method *connect_a*. Both methods belong to the class **Graph addends**. The factor is stored by the method *new_g* again because it is a variable graph too. Method *connect* connects the current factor with the current addend. If the factor is already stored in the model then we call the method *connect* only. When all expressions from the class **Model expressions** are processed, the complete model is stored in the form of calculating graph.

Now we can use this model for calculating the numeric values of corresponding robotic quantities, but in order to obtain more efficient model in a sense of smaller number of calculating operations we first have to reduce its calculating complexity. Processes for reducing the calculating complexity of generated mechanism model and for numeric and symbolic calculations over the generated model have to be specified too, but this specification is not the subject of this paper.

4. Conclusion

By means of a detailed analysis of the Newton-Euler method it was established that the mathematical relations forming the robotic mechanism model are in the polynomial form. To form mathematical models of robotic mechanisms in symbolic form, it is necessary to develop special data structures, suitable for storing polynomial expressions and reducing of their calculating complexity.

In this paper is used the object-oriented approach in design of the system for mathematical modelling of robotic mechanisms in symbolic form. Using UML, the complete formal specification of the informational requests of complex technical systems can be performed. In the class diagram, consisting part of this modelling language, the statics of the given system is described. The system dynamics can be specified too, using the use case and the other diagrams of UML. The first process of system modelling, which is generation of the starting

robotic mechanism model, is specified here by the sequence diagram. By adding the specifications for the other system's processes we can obtain a complete formal specification of the given system, which is the basis for its implementation.

Acknowledgement: The work was supported by the Ministry for Science and Technology of Serbia through the Institute of Mathematics, Novi Sad.

References

[1] Vukobratović, M., Kirćanski, N., Real-Time Dynamics of Manipulation Robots. Springer-Verlag, (1985)

[2] Racković, M., Generation of the Mathematical Models of Complex Robotic Mechanisms in Symbolic Form. Ph. D. thesis, University of Novi Sad, (1996)

[3] Racković, M., Vukobratović, M., Surla, D., Generation of Dynamic Models of Complex Robotic Mechanisms in Symbolic Form, Robotica (1998) volume 16, pp. 23-36

[4] Vukobratović, M., Borovac, B., Surla, D., Stokić, D., Biped Locomotion, Dynamics, Stability, Control and Application, (Springer-Verlag 1990)

[5] Surla, D., Konjović, Z., Racković, M., UML Specification of the System for Generating Symbolic Models of Robotic Mechanisms Dynamics, 13th ISPE/IEE International Conference on CAD/CAM, Robotics & Factories of the Future - Pereira'97, pp. 58-63

[6] Racković, M., Surla, D., Object-Oriented Specification of the System for Generating Mathematical Models of Robotic Mechanisms Dynamics, Fourth ECPD International Conference on Advanced Robotics, Intelligent Automation and Active Systems, Moscow, August 24-26, 1998, pp. 270-275

[7] Booch, G., Rumbaugh J., Jacobson, I., Unified Modelling Language. Version 1.3, Rational Software Corporation, March, 1999, http://www.rational.com