Proceedings of the 11th World Congress in Mechanism and Machine Science
April 1-4, 2004, Tianjin, China
China Machine Press, edited by Tian Huang

A New Software Environment for MBS Simulation Based on XML and Integrated with CAD/CAE Packages

M. González, D. Dopico, J. Cuadrado
Mechanical Engineering Laboratory, University of La Coruña, 15403 Ferrol, Spain
e-mail: lolo@cdf.udc.es

Abstract: This paper presents a new software environment for MBS simulation based on XML (eXtensible Markup Language), the universal format for structured documents and data on the World Wide Web. The core of this environment is MechML, a work-in-progress language to describe MBS data and related information based on XML. MechML takes full advantage of the latest developments in data modeling and offers important advantages over existing data formats. MechML has been integrated with I-DEAS, a commercial CAD/CAE package, so users can build models within I-DEAS and automatically generate MechML data files. This working environment reduces significantly the time needed to generate a MBS model.

Keywords: Multi-body systems, XML, Data formats, CAD/CAE

1 Introduction

During the last years our group has been working on multibody dynamics. We focus on developing new formulations for real-time dynamics, consideration of flexible bodies, component stress calculation during the motion, control integration and graphic visualization. An example of this kind of work can be found in [1]. Therefore, we often need to choose different dynamic formulations (e.g. global vs. topological methods), combine them with various integration algorithms, and establish comparison among all the resultant alternatives for a certain number of benchmark problems. In order to do that, custom computer programs should be developed to implement each particular method for each particular example.

The current working method is slow and error-prone. Three models must be developed for a mechanism: 1) A mathematical model of the multi-body system. The corresponding equations are hand-calculated using natural coordinates. 2) A CAD model for each body. CAD models are created in I-DEAS, a high-end CAD/CAE package. 3) FE models for flexible bodies. These models and the corresponding analyses are performed in COSMOS, a commercial FEA package. Once the three models are ready, all the information is gathered, combined and hard-coded in a Fortran 77 routine, which is linked with Fortran 77 integration routines and a C++ graphic user interface to build the simulation program.

The resulting code is a custom solver for a particular mechanism. The hard-coding of the system equations reverts in a very fast code, but has disadvantages: reutilization of code is difficult and users developing new models must have advanced programming skills. Moreover, as the equations are written by hand using the information of the three models (MBS, CAD, FE), a single change in one of these models (e.g. a change of dimension in a body) forces the user to update the other two models, check model coherency, rewrite the Fortran 77 routine and build the program again.

1.1 Proposed approach

In order to overcome the abovementioned drawbacks of our current working methodology, the new one should have the following characteristics:

(1) Data file and solver should be separated.
(2) Model generation and actualization has to be streamlined.
(3) Different modeling techniques must be accepted (e.g. relative coordinates, reference point coordinates, natural coordinates).
(4) The solver has to be structured in such a way that new dynamic formulations and/or numerical integrators can be easily tested.
(5) Efficiency cannot be significantly reduced.
(6) Cooperation with other research groups should be favoured.

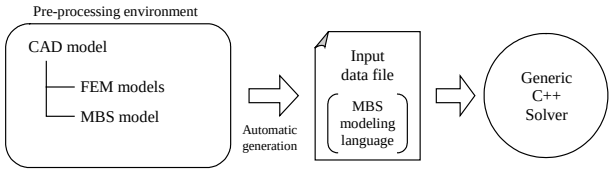

Figure 1 Proposed working environment

To accomplish the requirements listed above, a new working environment is proposed. It is composed of the three elements shown in figure 1:

(1) An input data file written in a neutral multi-body system modeling language: this file contains all the information about the system (rigid and flexible body description, joints, constraints, forces, etc.) expressed in a simple, easy-to-read data format.
(2) A pre-processing environment with a friendly graphic user interface. The user can use this environment to generate CAD models of bodies, FE models of flexible bodies, and build the multi-body system model by assembling the CAD models, adding joints, forces and constraints. The environment shall be capable of generating the input data file in an automatic way.
(3) A generic multi-body system solver framework, implemented in C++, which reads the input data file

and dynamically builds the system equations to perform a simulation.

This is the usual configuration of CAD/CAE packages which integrate external dynamic solvers like ADAMS. These commercial solutions don't fit our requirements: the data format used is proprietary, complex and don't support important features, such generic flexible bodies, control systems, etc.

This paper presents the first two components of our new working environment: the modeling language and the pre-processing system. The solver framework will be the subject of future communications.

1.1.1 Multi-body systems modeling language

At the present moment there is a lack of data standardization in the Multibody Systems Dynamics research and industrial activities: both commercial software and research groups developing simulation codes use proprietary, non-compatible data formats. This situation is a serious obstacle for collaboration and model sharing between groups around the world [2].

Several efforts to provide a neutral, universal mechanism modeling language have been made in the recent years (STEP Part 105 [3], Dymola [4], Modelica [5]) but none of them had success, in part due to the lack of software for reading and processing the proposed data formats.

This paper presents MechML, a work-in-progress language to describe MBS data and related information based on XML. Despite of being in an initial stage, MechML takes full advantage of the latest developments in data modelling and offers important advantages over existing data formats.

1.1.2 Pre-processing environment

All major CAD/CAE packages implement client-server communication using distributed object computing technologies like CORBA or DCOM. Users can exploit this feature to develop software that customizes, automates and extends the standard capabilities of the CAD/CAE package. Usually these packages have mechanism design modules, which make them ideal for our purposes: the user can build mechanism models using this module and write custom code for accessing in-memory data structures and retrieve information about them. This is the approach taken in this work: I-DEAS (a high-end CAD/CAE tool) is used as a pre-processing environment, and a custom Java application developed by the authors writes a data file using the MechML modeling language.

2 The MechML Modeling Language

MechML is based on XML (extensible Markup Language [6]), the universal format for structured documents and data on the World Wide Web. XML was designed in 1998 by the World Wide Web Consortium to provide a simple, versatile and extensible markup language for the Internet industry.

2.1 Benefits of using XML

XML is not a language itself, it is a meta-language: a language that can be used to create and describe other languages. These XML-derived languages are called XML vocabularies. XML has many advantages over other data formats: its simple, easy to use, license-free, platform-independent, well-supported (on-line resources, documentation and software available) and follows international standards (Unicode character set, URL references, IEEE floating-point numbers, ISO date-time codes …). XML has been successfully used to create vocabularies in several domains of science: Mathematics (MathML [7]), Chemistry (CML [8]), Geography (GML [9]), etc. XML technology will be briefly described with examples.

The structure and syntax of an XML vocabulary is rather simple. This code fragment shows how to codify the inertial data (mass and center of gravity) of a rigid body using MechML:

```
<body name="aBody">
  <inertia
    mass="3.2537E-5"
    cogx="0.05185"
    cogy="5.27520"
    cogz="0.0"/>
</body>
```

All XML vocabularies have a hierarchical structure composed of elements (like body or inertia in this example). Elements are the basic building blocks of XML markup and may have attributes (like name or mass) or nested elements. Each XML vocabulary defines the valid element names and structure, and the data type of each attribute.

One of the strong points of XML is the availability of parsers or processors. These software libraries greatly simplify the task of reading and processing the information contained in a XML file, saving many hours of work to the software developers. Currently there are several good-quality XML parsers, free or commercial, implemented in Java, C, C++, and Visual Basic. This Java code fragment shows how to read the previous XML example using the Apache Xerces-J parser [10]:

```
import org.apache.xerces.parsers.*;
import org.w3c.dom.*;
...
// Parse file
DOMParser parser = new DOMParser();
parser.parse(filename);
Document myDocument = parser.getDocument();
// walk hierarchy and extract data
Element body = myDocument.getDocumentElement();
String name = body.getAttribute("name").getValue();
Element inertia = body.getFirstChild();
...
```

Whenever a data file is read, the syntax and structure must be checked to ensure integrity and data type validity. This is usually a hard task for the programmer, and the resulting code can be very lengthy and obfuscated. XML provides simple and reliable ways to perform this job. One of them is using the XML Schema language [11] to describe the structure of the XML-derived language. This XML Schema code shows how to describe the structure of the previous XML example:

```
<element name="body">
  <sequence>
    <element name="inertia" >
      <attribute name="mass" value="double" />
      <attribute name="cogx" value="double" />
      <attribute name="cogy" value="double" />
```

```
      <attribute name="cogz" value="double" />
    </element>
  </sequence>
  <attribute name="name" value="string" />
</element>
```

XML Schema can describe element content (attributes and nested elements) and attribute data types of an XML vocabulary. MechML has been described in this way, and the resulting file has been published on the WWW (URL: http://lim.ii.udc.es/mechml/mechml.xsd). Multi-body system models written in MechML make reference to this URL in their header, so when the model is parsed, the XML parser can validate the MBS data file against the standard language syntax contained in the XML Schema file. If the MBS data file content doesn't fit the Schema description, the parser shows descriptive messages that allow easy detection of the error source.

Document processing and validating capabilities are not the only advantages of MechML. Due to the great success of XML on Internet a variety of software tools are available: editors, viewers, developing environments, etc. Moreover, there are some technologies that could be useful for a multi-body system modeling language like MechML:

(1) XSLT, Extensible Stylesheet Language Transformation [12], is a language for specifying transformations of XML documents. It is a declarative, interpreted language, simple yet powerful, than can be used to update MechML documents from older versions or to develop translators between MechML and other mechanism modeling languages.

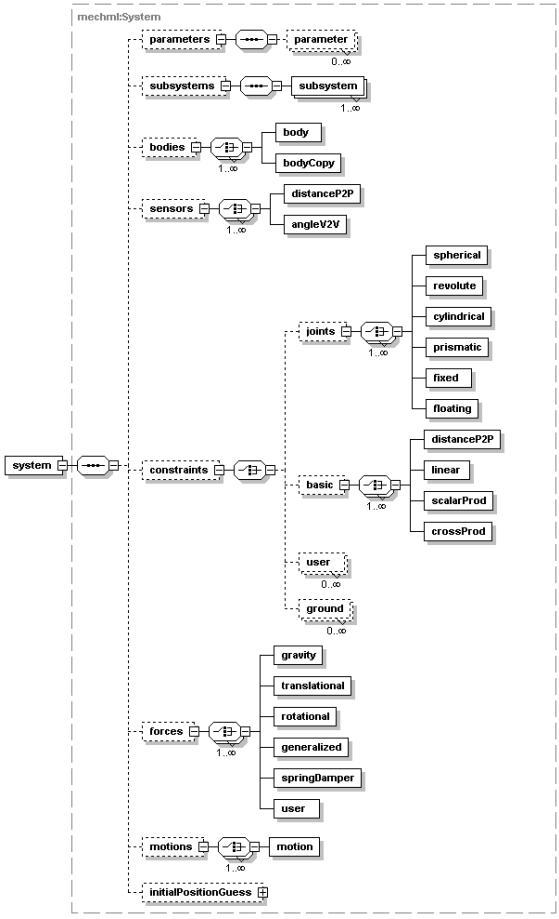

Figure 2 MechML data model: *system* element

(2) All of the major Relational Database Management Systems have become XML-enabled. Multi-body system models written in MechML can be handled by these databases, allowing the easy creation of on-line MBS model repositories with indexing and searching facilities.

(3) RDF, Resource Description Framework [13] is a recent technique for describing resources on the web: meta-data about the resource (author, subject, keywords, etc.) is expressed in an XML vocabulary and RDF-enabled search engines can read and use this information to classify and rate the resource. MechML models can take advantage of this technique.

2.2 MechML data model

The root element of a MechML document is the MBS element. Its attributes hold information about the URL of the MechML Schema, used by the parsers to check document validity. The content is divided in two sections: the system element, describing the multi-body system itself, and the analysis element, describing the analysis that shall be performed. Figures 2 and 3 show the data model of these elements. The following examples will show how to model a slider-crank mechanism (figure 4) using this data model. This is the basic structure of a MechML document:

```
<?xml version="1.0" encoding="UTF-8"?>
<MechML:MBS
 xmlns:MechML="http://lim.ii.udc.es/MechML"
 schemaLocation="http://lim.ii.udc.es/mechml.xsd
>
  <system>
  ...
  </system>
  <analysis outputFile="slider-crank.out.xml">
  ...
  </analysis>
</MBS>
```

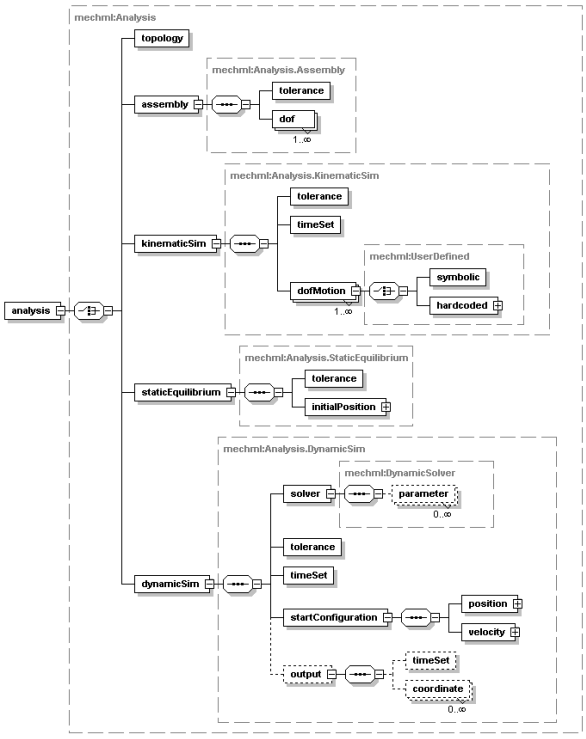

Figure 3 MechML data model: *analysis* element

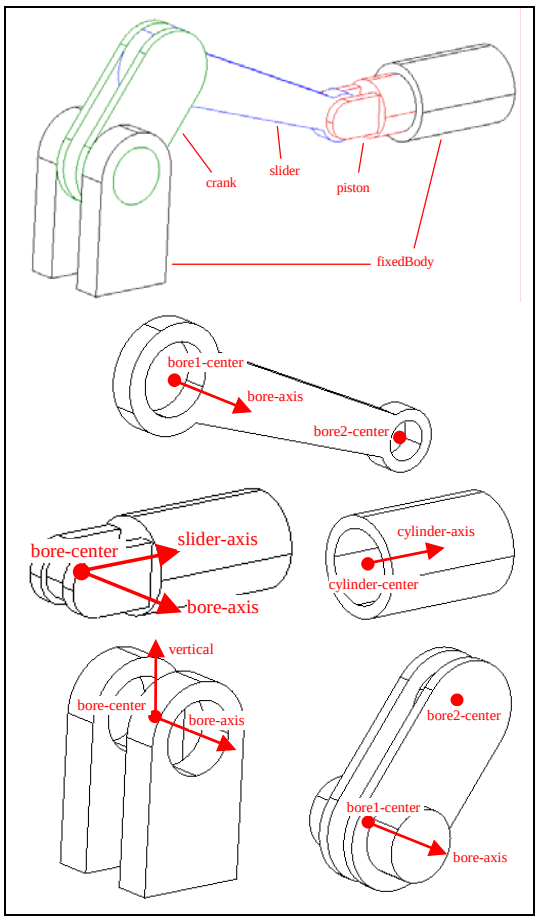

Figure 4 Slider-crank mechanism modeled in natural coordinates

2.2.1 Multi-body system definition

The system element holds all the information about the multi-body system. The data is structured in nested sub-elements. Body elements describe bodies using natural coordinates [14]: name, geometry, inertial properties, points and vectors are provided:

```
<body name="slider">
  <graphics format="VRML2" source="slider.wrl"/>
  <point name="bore1-center" x="0.0" y="0" z="0" />
  <point name="bore2-center" x="0.1" y="0" z="0" />
  <vector name="bore-axis" x="0" y="0" z="1.0" />
  <inertia
    mass="3.2537E-5"
    cogx="0.051858" cogy="0.0" cogz="0.0"
    Ixx="2.99396E-9"
    Iyy="5.88586E-8"
    Izz="6.13103E-8"/>
</body>
```

Sometimes angles or distances need to be measured in order to define constraints; they can be defined in the sensors element using points and vectors of bodies:

```
<angleV2V name="crank_angle">
  <fromVector ref="fixedBody:vertical"/>
  <toVector
    fromPoint="crank:bore1-center"
    toPoint="crank:bore2-center"
  />
</angleV2V>

<distanceP2P name="piston_distance"
  fromPoint="fixedBody:cylinder-center"
  toPoint="piston:bore-center"
/>
```

Joints (spherical, revolute, cylindrical, prismatic, fixed and floating) are defined in the *constraints* element. Other joints can be imposed using *basic* constraints (linear equations, scalar and cross vector products …). The *ground* element is used to fix a body to the global reference frame. The data model also supports user-defined forces and constraints.

```
<joints>
  <revolution name="slider-crank">
    <body ref="slider">
      <axisPoint ref="bore1-center"/>
      <axisVector ref="bore-axis"/>
    </body>
    <body ref="crank">
      <axisPoint ref="bore2-center"/>
      <axisVector ref="bore-axis"/>
    </body>
  </revolution>
</joints>
```

MechML also supports parametric models. For example, if a car suspension is modeled, dimensions, inertial properties of the bodies and forces can be defined with symbolic expressions depending on key parameters. The subsystem element allows the reutilization of the suspension model as a component in a bigger model, giving values to the parameters:

```
<subsystem
 name="front_left_suspension"
 source="McPherson.xml">
  <parameter name="length"    value="0.450"/>
  <parameter name="stiffness" value="0.140"/>
  <parameter name="damping"   value="0.0023"/>
</subsystem>
```

2.2.2 Analysis definition

MechML also provides a way of specifying the analysis that shall be performed on the multi-body system: topology, assembly, kinematic, dynamic and static equilibrium analysis are supported. The authors are working on extending MechML to provide a data model for the results of the simulation:

```
<analysis outputFile="slider-crank.out.xml">
  <kinematicSim>
    <tolerance rel="1E-3" abs="1E-4"/>
    <timeSet start="0" end="2.0" step="0.01"/>
    <dofMotion coordinate="crank-angle">
      <symbolic expr="3.14*TIME"/>
    </dofMotion>
  </kinematicSim>
</analysis>
```

3 Preprocessing Environment: Integration with I-Deas

Writing the MechML model of a multi-body system with a high number of bodies and joints can be a time-consuming and error-prone task. Using a graphic user interface to build the model would revert in great time savings. All major CAD/CAE packages implement client-server communication using distributed object computing technologies like CORBA or DCOM, so users can develop software that customizes, automates and extends the standard capabilities of the CAD/CAE package. The authors have taken advantage of this feature to implement a plug-in for I-DEAS (a high-end CAD/CAE tool) capable of generating a MechML data file. This plug-in, called Ideas2MechML, allows to perform almost all the

modelization within I-DEAS. The working methodology is described:

(1) A CAD model for each body is created. The user can place points and vectors on the model, which will be used for mechanism modeling (modeling guidelines have been developed).
(2) A FE model for each flexible body is created. The FE model is geometry-based, so future changes in body geometry will be automatically transferred to the FE. Analyses are also performed within I-DEAS.
(3) A multi-body system model is created by assembling CAD models of bodies, adding joints and constraints.
(4) The Ideas2MechML command is executed within I-DEAS and the MechML data file for the multi-body system is generated.
(5) The MechML file is completed adding code by hand (not all MechML features can be modeled within I-DEAS).

Ideas2MechML is written in Java and communicates with I-DEAS using CORBA [15]. The plug-in accesses the in-memory data structures of CAD and MBS models, retrieves information about them and writes a MechML document. Future versions will allow extracting FE information from the CAE tool and witting it in MechML format. The plug-in can be easily configured and extended.

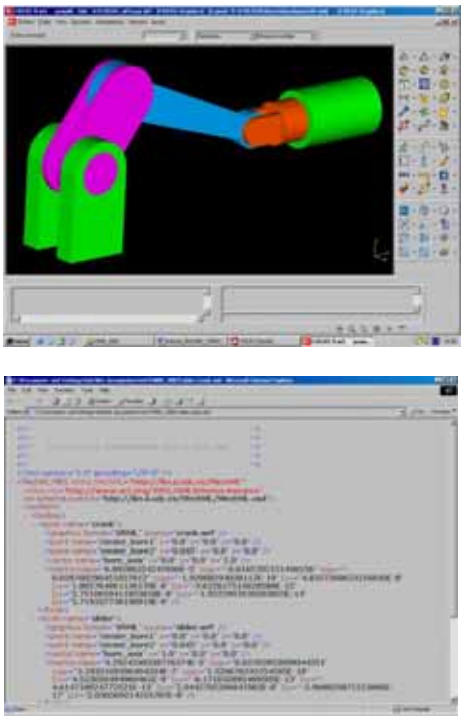

Figure 5 Slider-crank mechanism: I-DEAS model (upper) and automatically generated MechML data file (bottom)

Plug-ins similar to Ideas2MechML can be developed for other CAD/CAE systems like CATIA, Unigraphics or Pro/Engineer. This would significantly reduce the time needed for generating MBS models using MechML as input file to the solvers.

4 Conclusions

Research on multi-body system dynamics is a software-intensive activity. Testing new formulations for real-time dynamics, consideration of flexible bodies, control integration and graphic visualization requires the development of custom applications, and several tools need to be used to generate a MBS model (CAD modelers, FEA packages, etc). This makes the modeling process slow and error-prone. In addition, there is a lack of data standardization in the MBS dynamics research and industrial activities: both commercial software and research groups use proprietary, non-compatible data formats. This situation is a serious obstacle for collaboration and model sharing between groups.

This paper presents a new software environment for MBS simulation based on XML (eXtensible Markup Language), the universal format for structured documents and data on the World Wide Web. The core of this environment is MechML, a work-in-progress language to describe MBS data and related information based on XML. MechML takes full advantage of the latest developments in data modeling and offers important advantages over existing data formats. MechML has been integrated with I-DEAS, a commercial CAD/CAE package, so users can build models within I-DEAS and automatically generate MechML data files. This working environment reduces the time needed to generate MBS models and simplifies model sharing and distribution.

References

1. Cuadrado, J., Gutiérrez, R., Naya, M. A., Morer, P. A comparison in terms of accuracy and efficiency between a MBS dynamic formulation with stress analysis and a non-linear FEA code, *International Journal for Numerical Methods in Engineering*, 51(9): 1033-1052, 2001
2. Keil, A., Hermsdorf, H., Enderlein, V., On the description of multibody system models, In: *Proc. Euromech Colloquium 404, IDMEC/IST*, Lisbon, Portugal, 1999
3. ISO, Industrial Automation Systems and Integration, Product Data Representation and Exchange, STEP Part 105: Kinematics, *ISO 10303*, http://www.iso.ch/cate/cat.html, 1994
4. Otter, M., Elmqvist, H., Cellier, F., Modeling of multibody systems with the object-oriented language dymola, *Nonlinear Dynamics*, 9: 91-112, 1996
5. Elmqvist, H., Mattsson, S. E., Otter, M., Modelica: The new object-oriented modeling language, In: *The 12th European Simulation Multi-conference, ESM'98*, Manchester, U.K, 1998
6. World Wide Web Consortium (W3C), eXtensible Markup Language (XML), <http://www.w3.org/XML>, 1998
7. World Wide Web Consortium (W3C), Mathematical Markup Language (MathML), <http://www.w3.org/Math>, 2001
8. Murray-Rust, P., Rzepa, H., Chemical Markup Language (CML), <http://www.xml-cml.org>, 2002
9. Open GIS Consortium Inc., Geography Markup Language (GML), <http://opengis.net/gml/>, 2000
10. Apache Software Foundation, Xerces2 Java Parser, <http://xml.apache.org/xerces2-j/>, 2000
11. World Wide Web Consortium (W3C), XML Schema, <http://www.w3.org/XML/schema/>, 2001
12. World Wide Web Consortium (W3C), Extensible Stylesheet Language Transformation, <http://www.w3.org/Style/XSL>, 2001
13. World Wide Web Consortium (W3C), Resource Description Framework (RDF) Model and Syntax Specification, <http://www.w3.org/RDF/>, 2001
14. García de Jalón, J., Bayo, E., Kinematic and dynamic simulation of multibody systems, Springer-Verlang, 1994
15. The Object Management Group, Common Object Request Broker Architecture Specification, <http://www.omg.org/technology/documents/formal/corba_iiop.htm>, 1998