MBSLAB: A NEW COLLABORATIVE SIMULATION ENVIRONMENT FOR MULTIBODY SYSTEM ANALYSIS

Manuel González
Laboratory of Mechanical Engineering
University of La Coruña
Ferrol 15403, SPAIN
Email: lolo@cdf.udc.es

Javier Cuadrado
Laboratory of Mechanical Engineering
University of La Coruña
Ferrol 15403, SPAIN
Email: javicuad@cdf.udc.es

ABSTRACT

This article describes a new collaborative simulation environment for multibody system analysis designed to streamline the development, testing and evaluation of new methods for multibody system simulation. The environment, named MbsLab, is made up of 3 components: (1) MbsML, a neutral and extensible XML-based language for describing multibody system models and related information, (2) MbsEngine, an open-source modular simulation tool that uses MbsML as native data format, and (3) MbsBenchmark, a benchmarking suite for multibody system analysis software. The three components can be used together to develop and compare the performance of new simulation methods in a short time. Currently, the environment only supports rigid bodies, but it is expected to grow to support flexible bodies, contact, impact, etc.

1. INTRODUCTION

Multibody system simulation is a complex process involving several factors: modeling, dynamic formulation and integration procedures. Cuadrado (1997) concluded that the adequate combination of these factors depends heavily on the properties of the system to be analyzed, and therefore there is not a universal solution: different alternatives (modeling techniques, formulations and integration schemes) must be combined, evaluated and compared in order to find the optimal combination for a particular system. Commercial tools do not allow this level of flexibility, since they usually implement just one kind of dynamic formulation and a few integrators, and the capabilities of the solver kernel cannot be modified or replaced with user code. Therefore, custom tools must be developed by the researcher.

The authors have developed a simulation environment designed to streamline the development and testing of new methods for multibody system simulation. The system is made up of three components: (1) MbsML, an neutral XML-based language for describing multibody system models and related information, (2) MbsEngine, an open-source modular simulation tool that uses MbsML as native data format, and (3) MbsBenchmark, a benchmarking suite for multibody system analysis software. The three components are described in the following sections.

2. MBSML

Multibody system data standardization is still an open research field (Schiehlen, 1997). Available simulation tools, commercial and from academic research groups, differ in model description and data formats. Since there is no common standard language, bidirectional translations and adapting interfaces are required in order to exchange models within a simulation environment made up by different simulation tools. This is a major obstacle for model sharing and cooperation.

Previous approaches to multibody system information modeling fall in two main categories: the *data* approach and the *document* approach. In the data approach, information is managed by relational or object-oriented databases. This solution is quite efficient and uses standardized techniques for information storage, search and retrieval (Tisell, 2000). However, databases do not offer god facilities for structured text and the user needs special tools to access to the information in an intuitive way. On the other hand, the document approach has all the advantages that databases lack of: documents can be copied, e-mailed and printed easily, and structured formats, like Dymola (Otter, 1996), can represent very well the structure of the real system. But this approach has also disadvantages: information processing is difficult, since document syntax must be parsed. If the document syntax is very complex, writing compliant parsers will be cumbersome, very few tools will be developed to support that syntax and the language will not became a standard.

Recently, XML (Bray, 1998) has emerged as an increasingly popular format to encode information. A key feature of XML is that it integrates the benefits of the data and document approaches in one single format (Murray-Rust, 1999), so we have decided to design an XML-based

language for modeling the information used and generated in our simulation environment. The following sections give a brief introduction of XML and MbsML.

2.1 Brief introduction to XML

XML, Extensible Markup Language (Bray, 1998) is a simple, very flexible text format derived from Standard Generalized Markup Language (SGML, ISO 8879) which was developed in the 1970s for the large-scale storage of structured text documents. The XML standard was published in February 1998 by the World Wide Web Consortium, the organization responsible for defining many internet-related standards, most notably HTML. Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere.

2.1.1 Structure. XML documents have a hierarchical structure composed of tag elements. Figure 1 shows an MbsML document that models a pendulum. Elements (like <model> in the example) are the basic building blocks of XML markup and may have attributes (like "id") or nested elements (like <ground> or <body>). XML only defines the syntax of the document; each XML vocabulary defines the valid structure: names for elements and attributes, content type (numeric, text, nested elements, etc.) and cardinality.

2.1.2 Parsing. One of the strong points of XML is the availability of parsers. Parsers read the information contained in a XML file using standardized techniques, thus saving many hours of work to software developers. Currently there are good-quality XML parsers, free or commercial, implemented in Java, C, C++, Visual Basic, Python, etc.

2.1.3 Validation. Before processing the information contained in a document, the structure and content 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. The syntax of an XML-based language can be described with a DTD (Document Type Definition), or one of the several schema languages available. A *validating* parser can use this language definition to validate the content of an XML document.

2.1.4 Other tools. A wide range of technologies are available to work with XML: XSLT (Clark, 1999) specifies transformations of XML into other versions or formats, XPath (Clark, 1999) selects fragments of XML documents, XInclude (Marsh, 2003) is a general inclusion mechanism, RDF (Manola, 2004) embeds meta-data inside XML, etc. In addition, all major Relational Database Management Systems have become XML-enabled.

2.1.5 Applications. XML has been applied successfully to define languages in several fields: MathML in Mathematics (Ion, 1999), CML in Chemistry (Murray-Rust, 2001), CellML in biology (Hedley, 2001), GML in Geography (Lake, 2000) etc. In the field of engineering, several organizations are also working with XML: the NIST (2001) is working on MatML for material properties modeling; ISTOS (2001) is developing femML, for encoding finite element models in XML format; and PDES (2001) is developing STEPml, an XML version of some parts of STEP.

2.2 MbsML Goals

MbsML has been designed to be used in a collaborative environment for research in multibody system simulation; therefore the design was driven by the following requirements:

2.2.1 Foster data exchange providing a neutral modeling language *independent on the formalism* used to perform multibody system simulations. MbsML provides only a physical description of the system. Modeling concepts which are meaningful only to a particular formalism are avoided. Therefore, MbsML models can be used as input data for different simulation tools, and different research groups can exchange models and results easily.

2.2.2 Foster data reutilization *splitting the description of a simulation* into logical sections: model, analysis and method. This organization provides maximum decoupling and allows easy combination of different methods to perform different analysis on different models.

2.2.3 Foster extensibility providing a *modular* language, which users can adapt, configure or extend to fit their particular needs.

```
<model id="Pendulum">
  <ground id="ground">
    <geometry>
      <point id="Origin">
        <x>0.0</x>
        <y>0.0</y>
        <z>0.0</z>
      </point>
    </geometry>
  </ground>
  <body id="rod">
    <geometry>
      <point id="P1">
        <x>0.0</x>
        <y>0.0</y>
        <z>0.0</z>
      </point>
      <point id="P2">
        <x>1.0</x>
        <y>0.0</y>
        <z>0.0</z>
      </point>
    </geometry>
    <inertia>
      <rod>
        <mass>1.0</mass>
        <fromPoint idref="P1"/>
        <toPoint idref="P2"/>
      </rod>
    </inertia>
  </body>
  <joint.spherical id="theJoint">
    <body idref="ground">
      <centerPoint idref="Origin"/>
    </body>
    <body idref="rod">
      <centerPoint idref="P1"/>
    </body>
  </joint.spherical>
  <load.gravity>
    <direction bodyRef="ground" id="-Z">
      <x>0.0</x>
      <y>0.0</y>
      <z>-1.0</z>
    </direction>
    <magnitude>9.81</magnitude>
  </load.gravity>
</model>
```

Figure 1. EXAMPLE OF MBSML DOCUMENT.

2.3 MbsML document structure

Several *schema languages* are available to define the formal syntax of an XML vocabulary. The most used are DTD (Document Type Definition, a part of the first XML specification) and XML Schema (Thompson, 2001), designed to overcome the limitations of DTD. Recently, a new schema language called RelaxNG (Clark, 2001) has emerged. Although RelaxNG is not widely supported by XML validating and authoring tools, its simplicity and powerful modeling capabilities make it a good choice for rapid development of XML-based languages. The authors use RelaxNG to develop MbsML, and final versions are also released in XML Schema format to take advantage of its wide support. The MbsML schema allows four types of documents: *model*, *analysis*, *method* and *job*.

2.3.1 Model. A <model> element describes the multibody system: bodies, joints, forces, actuators, constraints, nested sub-models, etc. Angles and distances between components can be defined. Figure 1 shows an example of a model. Every component is identified by a unique "id" attribute value; component can refer to other components using this "id". Currently only rigid bodies described with natural coordinates are supported.

2.3.2 Analysis. An <analysis> element describes the analysis to be performed on a particular multibody system model: type of analysis (kinematic simulation, forward or inverse dynamics, static equilibrium), conditions (for example, initial and final time), etc. The initial conditions of the system and the desired output are also described in this element. This element only describes the kind of analysis, not how to perform it.

2.3.3 Method. A <method> element describes the method to be used to perform a particular analysis: formulation, integrator type, algorithm parameters (like the integration time step), etc.

2.3.4 Job. Once the model, the analysis and the method are defined, they are combined in a job, which serves as input file to the simulation tool. The <job> element is made up by one <model> and one or more <task> element; each <task> is made up by an <analysis> and a <method>, and tasks are performed sequentially (the final state of the system after a task is the initial state for the next task). With this decoupling between model, analysis and method, it is easy to build and test combinations: different analysis for the same model, same analysis on different models, different methods for the same analysis, etc.

2.4.4 Results. The results from a simulation are also encoded in MbsML: time-history of coordinates, reaction

forces, final state of the system, etc.

2.5 Features

Other interesting features of MbsML are:

2.5.1 Parametric models. They are essential in optimization, and they are also very useful to build libraries of components that will serve as building blocks for bigger components. In MbsML, every numeric field (coordinates, masses, force magnitudes, etc) can be given by a symbolic expression involving parameters. These parameters must be declared at the beginning of the model element, and its value can be defined in the XML document or by the simulation software.

2.5.2 XInclude support. The XInclude technology (Marsh, 2003) can be used to build XML documents from small parts using an <include> element that works like a link:

<include href=”someXMLfile.xml”/>

<include> elements inserted in MbsML files are a powerful method to reuse MbsML components (bodies, forces) and combine them in high-level models.

2.5.3 Units. By default all quantities are given in SI units. When other units are used in an element, the user must specify the unit in a “uom” (units of measure) attribute:

<mass uom=”pound”/> 23 </mass>

An XML dictionary of units is available, which provides conversion factors to the corresponding SI unit. This dictionary can be extended and used by the simulation software to perform unit conversion.

2.5.4 Metadata. Medatada can be embedded in MbsML documents using RDF. Resource Description Framework (Manola, 2004) is a XML language for representing information about resources in the World Wide Web. It is particularly intended for representing metadata of a document, such as title, author, modification date, copyright and licensing information. Models encoded in MbsML can carry this information, and when they are shared in a web environment, RDF-enabled search engines can read and use it to classify the resource.

2.6 Modular design

MbsML has been designed with a modular philosophy inspired by XHTML (Altheim, 2000): the schema language definition is split into modules; each module defines a particular construction (a body, a joint) or feature (parameters, unit support). These modules are assembled together to build the language. This approach has many advantages: it is very easy to customize the language by removing, substituting or adding modules. For example, if a potential user needs support for flexible bodies (not supported in the current MbsML specification), he can develop a new module and start to use it immediately. Later, the module can be included in the next official version of the language and other researchers can use it. In this way, the language can evolve driven by the needs of the community.

3. MBSENGINE

MbsEngine is a C++ library for multibody system simulation with a highly modular design that provides different solver components (dynamic formulations, integrators, etc.) which can be easily combined to obtain a particular solver configuration. The main features of the library are the native support for MechML and the collaborative development environment used to implement it.

3.1 Support for MbsML

Reading MbsML files involves using a XML parser and one of the several existing interfaces to access XML data. The main interfaces are DOM (Wood, 1998) and SAX (Megginson, 1998). DOM (Document Object Model) presents an XML document as an in-memory tree structure composed by elements and attributes that replicate the structure of the XML document. The developer can navigate, access or modify this structure easily. SAX has a different approach for reading XML: the parsed reads data sequentially, and signals events to the application with information about the elements and attributes. This method consumes few resources (memory and CPU cycles) than DOM, but it is more complex from the programming point of view.

Both methods (DOM and SAX) present information to the application in the form of element and attributes. When reading MbsML documents, the application must interpret these items and convert them in meaningful objects: bodies, forces, etc. The authors have developed a library that performs this task: each MbsML component has been mapped into a C++ class with methods for reading from XML to memory and writing from memory to XML. Similar to the DOM approach, the library has methods to

navigate and modify the resulting object structure. The library is independent from MbsEngine, so it can be used by other researchers to add MbsML support to other simulation tools.

3.2 Collaborative development

MbsEngine is developed using a Free/Open Source Project Hosting Site (FOSPHost). The site provides a centralized source code repository managed by CVS, mailing lists for users and developers, bug tracking system, task management tools, etc. Since the C++ library has a modular design, different research groups can work on different modules and communicate with others easily using the facilities provided by the site. The resulting software is released under the GPL license.

4. MBS BENCHMARK

Benchmarking is widely used in computer science to measure the performance of hardware and software (Weiker, 2002). This phenomenon is not restricted to vendors: benchmarks are also used in academic research to prove the qualitative advantage of new ideas, and by manufacturers to drive product development. Standard, well-known benchmarks exist for testing hardware (CPU, multi-CPU systems, memory, network, graphics, disk IO ...) and software (primary servers: e-mail, databases, e-Commerce platforms ...). These benchmarks use to be administered by industry associations (like SPEC) or major software vendors (like Oracle or SAP), and results are made public periodically.

In the field of multibody systems, benchmarking is no a standardized practice: academic papers measure performance of new formalisms with different, non-standard problems, making impossible to compare results. MBS Benchmark (Multi Body Systems Benchmark) is a collaborative project dedicated to develop and maintain a standardized set of problems and procedures which enable easy and objective performance evaluation of multibody systems simulation software.

4.1 Problem collection

The collection is divided in groups of problems; currently there are only 2 groups: group A, composed by simple, academic problems, and group B, composed by real-life problems. Both groups involve only rigid bodies. It is expected than new groups will be added in the future to cover flexible bodies, contact, impacts and collisions, etc. Each problem is documented with a detailed description, a model in MbsML format and a reference solution. This reference solution is usually the state of the system at the end of the simulation, and has been obtained using reliable solvers with low integration steps and tolerances (at the cost of a high computational time).

Table 1. PROBLEM COLLECTION.

	Problem	Characteristic
Group A: academic problems with rigid bodies		
A01	Double pendulum	High accelerations
A02	Doble cuadrilatero	Singular positions
A03	Andrew's mechanism	Small time scale
A04	Bricard's mechanism	Redundant equations
A05	Bicicleta	Stiff system
Group B: real-life problems with rigid bodies		
B01	Iltis vehicle	Automotive
B02	Dornier antenna	Aerospatial
B03	Human body	Biomechanics
B04	PUMA robot	Robotics (serial)
B05	Stewart platform	Robotics (parallel)

4.2 Performance evaluation

Every problem has a precision constraint: when a problem is solved using a particular software, the relative error between the obtained solution and the reference solution must be below a certain limit (which is given in the problem documentation) in order to consider the obtained solution as valid. Currently the only performance indicator is the CPU time spent in solving the problem, which varies in inverse proportion to the relative error. Therefore, the researcher must tune the software (integration step, tolerances, etc.) to achieve the required precision with a minimum computation time.

Once the problem is solved within the given precision, the results can be submitted to a central database using a web browser. This relational database stores all information about the environment used to solve the problem. This information is categorized into seven sections: table 2 shows the database structure and the main fields in each table. The first two tables hold contact information of the submitter. Table *computers* holds information about hardware and operational system. In order to compare performance results produced in different computers, this table stores a performance index which is computed by a program distributed from the MbsBenchmark website. Information about the simulation tool is split into 3 sections: *softwares* (contact information), *builds* (technical information about the build process: code version, compiler, optimization flags, etc.) and *methods* (description of the

formalism, formulation and integrator implemented by the software). Table *problems* holds links to the problem documentation and the last table holds the performance results. All the information can be submitted to the central database manually (filling HTML forms in a web browser) or automatically (the information can be encoded in XML files and submitted to the server using scripts available from the benchmark website).

Table 2. STRUCTURE OF THE RELATIONAL DATABASE.

Table	Fields
Users	UserID, *OrganizationID*, FirstName, LastName, Email, Password
Organizations	OrganizationID, Name, Adress, City, Country, URL
Computers	ComputerID, *OrganizationID*, Nickname, Brand, Model, Motherboard, CPUmodel, CPUnumber, Memory, OSname, PerformanceRatio
Softwares	SoftwareID, Name, Author, URL
Builds	BuildID, *SoftwareID*, Version, BuildSystem, BuildOptions, Libraries
Methods	MethodID, *SoftwareID*, Name, Coordinates, Formulation, Integrator, Comments
Problems	ProblemID, Name, URL
Results	ResultID, *ProblemID, UserID, ComputerID, SoftwareID, BuildID, MethodID*, Tags, IntegrationStep, CPUtime, RelativeError, Comments

4.3 Performance comparison

The information stored in the database can be queried and retrieved from the benchmark website using a simple HTML interface. The visitor specifies search criteria and a report is generated. Currently, only two types of reports can be generated: (a) comparison of several methods to solve a particular problem, (b) comparison between two methods in solving a range of problems. Reports are in HTML format with embedded SVG (XML format) graphs; figure 2 shows and example. In the future new features will be added: more kinds of reports (relative error versus CPU-time curves for a particular method, obtained with different integration steps or tolerance settings), performance comparisons between hardware configurations and compilers, facilities to export the performance data in XML or plain text format, etc. The web application is implemented using a MySQL database, Java Server Pages (JSP) and XSLT.

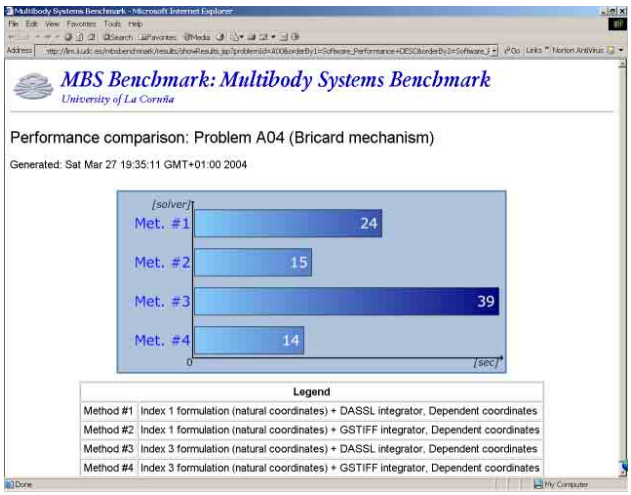

Figure 2. EXAMPLE OF PERFORMANCE REPORT.

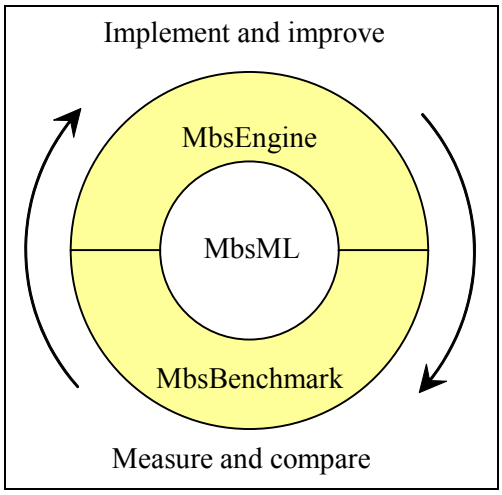

Figure 3. INTEGRATED DEVELOPMENT WITH MBSML, MBSENGINE AND MBSBENCHMARK.

5. CONCLUSIONS

The three components (MbsML, MbsEngine and MbsBenchmark) are used together to develop and test new methods for multibody system simulation: different researchers implement new formulations and algorithms within the MbsEngine framework as new library modules; their performance is measured with any of the problems from the benchmark collection, and results are stored in the central database. Then, the alternatives developed by different researchers can be compared easily, and results guide future developments, as showed in figure 3. The

evolution of the system is driven by the needs of the community. If researchers want to compare methods to solve a particular kind problem (for example, systems involving contact), MbsML and MbsEngine are extended to model and solve that problem, and new problems are added to the MbsBenchmark collection.

The resulting system streamlines the development, testing and evaluation of new methods for multibody system simulation and facilitates the cooperation between different research groups by providing a neutral modeling language.

REFERENCES

Altheim, M., and McCarron, S., *Building XHTML Modules*, W3C, 2000. <http://www.w3.org/TR/2000/WD-xhtml-building-20000105>

Bray, T., Paoli, J., and Sperberg-McQueen, C.M., *Extensible Markup Language (XML) 1.0: W3C Recommendation*, W3C, 1998. <http://www.w3.org/TR/REC-xml>

Clark, J., *XSL Transformations (XSLT) Version 1.0*, W3C, 1999. <http://www.w3.org/TR/xslt>

Clark, J., and DeRose, S., *XML Path Language (XPath) Version 1.0*, W3C Recommendation, 1999. <http://www.w3.org/TR/1999/REC-xpath-19991116>

Clark, J., and Makoto, M., *RELAX NG Specification*, OASIS, 2001. <http://www.relaxng.org/spec-20011203.html>

Cuadrado, J., Cardenal, J., and Morer, P., "Modeling and Solution Methods for Efficient Real-Time Simulation of Multibody Dynamics", *Multibody System Dynamics*, Vol. 1, No. 3, pp. 259-280, 1997.

Hedley, W., Nelson, M., Bullivant, D., and Nielsen, P., "A short introduction to CellML", *Philosophical Transactions: Mathematical, Physical & Engineering Sciences*, The Royal Society, Vol. 359, No. 1783, pp. 1073-1089, 2001.

Ion, P., Miner, R., et al, *Mathematical Markup Language (MathML) 1.01 Specification: W3C Recommendation*, W3C, 1999. <http://www.w3.org/TR/REC-MathML>

ISTOS, *Finite Element Modeling Markup Language (FEMML)*, 2001. <http://www.istos.org/femML>

Lake, R., *Geography Markup Language (GML) v1.0*, OpenGIS Consortium, Inc (OGC), 2000. <http://www.opengis.org/docs/00-029.pdf>

Marsh, J., and Orchard, D., *XML Inclusions (XInclude) Version 1.0*, W3C, W3C Working Draft, 2003. <http://www.w3.org/TR/2003/WD-xinclude-20031110>

Megginson, D., "SAX 1.0 (the Simple API for XML)", 1998. <http://www.saxproject.org>

Murray-Rust, P. and Rzepa, H.S., *CML (Chemical Markup Language)*, 2001. <www.xml-cml.org>

National Institute of Standards and Technology, *MatML*, 2001. <http://www.matml.org>

Otter, M., Elmqvist, H., and Cellier, F.E., "Modeling of multibody systems with the object-oriented modeling language Dymola", *Nonlinear Dynamics*, Vol. 9, No. 1-2, pp. 91-112, 1996.

Otter, M., Hocke, M., Daberkow, A., and Leister, G., "An Object-Oriented Data Model for Multibody Systems", *Advanced Multibody System Dynamics - Simulation and Software Tools*, Ed. Schiehlen, W., Kluwer Academic Publisher, 1993.

PDES Inc., *STEPml*, 2001. <http://www.stepml.org>

Shiehlen, W., "Multibody System Dynamics: Roots and Perspectives", *Multibody System Dynamics*, Vol. 1, No. 2, pp. 149-188, 1997.

Thompson, H.S., Beech, D., Maloney, M. and Mendelsohn, N., *XML Schema Part 1: Structures*, W3C, 2001. < http://www.w3.org/TR/2001/PR-xmlschema-1-20010330>

Tisell, C. and Orsborn, K., "A system for multibody analysis based on object-relational database tecnology", *Advances in Engineering Software*, Vol. 31, No. 12, pp. 971-984, 2000.

Wood, L., *Document Object Model (DOM) Level 1 Specification* , W3C Recommendation, 1998. <http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001>