P. Bunus, V. Engelson, P. Fritzson:
Mechanical Models Translation, Simulation and Visualization in Modelica.
Modelica Workshop 2000 Proceedings, pp. 199-208.

Paper presented at the Modelica Workshop 2000, Oct. 23.-24., 2000, Lund, Sweden.

All papers of this workshop can be downloaded from
http://www.Modelica.org/modelica2000/proceedings.html

Workshop Program Committee:
- Peter Fritzson, PELAB, Department of Computer and Information Science, Linköping University, Sweden (chairman of the program committee).
- Martin Otter, German Aerospace Center, Institute of Robotics and Mechatronics, Oberpfaffenhofen, Germany.
- Hilding Elmqvist, Dynasim AB, Lund, Sweden.
- Hubertus Tummescheit, Department of Automatic Control, Lund University, Sweden.

Workshop Organizing Committee:
- Hubertus Tummescheit, Department of Automatic Control, Lund University, Sweden.
- Vadim Engelson, Department of Computer and Information Science, Linköping University, Sweden.

Mechanical Models Translation, Simulation and Visualization in Modelica

Peter Bunus, Vadim Engelson, Peter Fritzson
PELAB, Programming Environment Laboratory
Department of Computer and Information Science,
Linköping University, SE-581 83, Linköping, Sweden
{petbu,vaden,petfr}@ida.liu.se

ABSTRACT

Modeling and simulation have become central to all disciplines of engineering and science. In a comprehensive modeling and simulation environment, it is desirable to integrate models specified in different modeling formalisms and to extend modeling language constructs to support multi-domain and multi-formalism modeling integrated with powerful visualization capabilities. This paper is concerned with the design of a complete integrated environment that combines the powerful mechanical model design of various CAD packages with the structuring mechanisms of object oriented modeling languages including a mathematical and logical behavior representation. We present an integrated environment for simulation of multi-domain models which has been implemented using Modelica as a standard model representation. The user can work with mechanical models designed with AutoDesk's Mechanical Desktop, extend the corresponding Modelica model in various ways, and analyze the simulation results in a high performance interactive visualization environment.

Introduction

Modelica is a new language for hierarchical object-oriented physical modeling, which is developed through an international effort. The language unifies and generalizes previous object-oriented modeling languages. Modelica is intended to become a *de facto* standard. The language has been designed to allow existing and future compilers to generate efficient simulation code automatically with the main objective to facilitate exchange of models, model libraries, and simulation specifications. It allows defining simulation models modularly and hierarchically by combining various formalisms expressible in the more general Modelica formalism. The multi-domain capability of Modelica gives the user the possibility to combine electrical, mechanical, hydraulic, thermodynamic, etc. model components within the same application model. Interaction between system components, which are often complex and difficult to analyze, can be easily studied.

In order to automate the design of mechanical models, CAD tools can be utilized. We have focused on adding the ability to easily interface the Modelica simulation environment with a wider variety of CAD systems. At the first stage we have focused on achieving full integration with widely accepted mechanical CAD solutions like SolidWorks and Mechanical Desktop. In this paper we present a translator implementation from AutoDesk's Mechanical Desktop to Modelica, which extracts geometric and parametric information from an existing designed mechanical model and produces a corresponding set of Modelica class instances with connections between them.

AutoDesk's Mechanical Desktop 4 is an integrated package of advanced 3D modeling tools and 2D drafting and drawing capabilities, that help the modeler to conceptualize, design, and document a mechanical product. Our comprehensive integrated environment consists of a CAD tool, a simulation environment, like Dymola or MathModelica, and a visualizer that provides online dynamic display of the assembly (during simulation) or offline (based on saved state information for each time step).

The developed integrated environment allows designers and engineers to build very quickly a virtual prototype which enables them to identify design flaws that would previously only have been found after building costly and time consuming physical prototypes. The combined mechanical design environment enhanced with simulation capabilities given by the Modelica simulation environment emphasizes the concept of functional design. In Modelica it is possible to specify arbitrary control algorithms for mechanical and mechatronic models. In that way it is possible to model and simulate both control an mechanical aspects of the desired mechanical application. Through simulation of the complete mechatronic system, the designer can predict how his mechanical subsystem will interact with the other subsystems. The dynamics of mechanical and/or control systems is also documented by plots of system variables completing in that way the realistic animation of the visualizer. Given an initial mechanical design, the user can modify this design (based on the simulated results) in order to improve the functional performance of the mechanical assembly.

Figure 1 represents a general view of our integrated environment for design, simulation and visualisation of mechanical models. In order to create such an environment we have combined a typical CAD environment with an equation based simulation environment with enhanced visualization capabilities. First the user will define the mechanical model in his/her favourite CAD package using, maybe standard, predefined component library models. The result of this operation usually is a static wire-frame model without any dynamic capabilities. A plug-in to the CAD package extracts from the drawing the geometry, mass, inertia and constraints information, translating them to a simulation language source code. This code is combined with other code fragments (e.g. control systems), simulated, and the output can be visualised as a data plot of the system variables and/or as a 3D or 2D dynamic model animation. The 3D visualisations are scenes that display the geometry of the parts in motions prescribed by the simulation results. The graphical user interface of the CAD model and the output visualisation capabilities of the simulation environment make it easy to describe and modify model geometry as well as examine analysis results at the same time. More implementation details of the integrated environment will be given in the remainder of the paper.

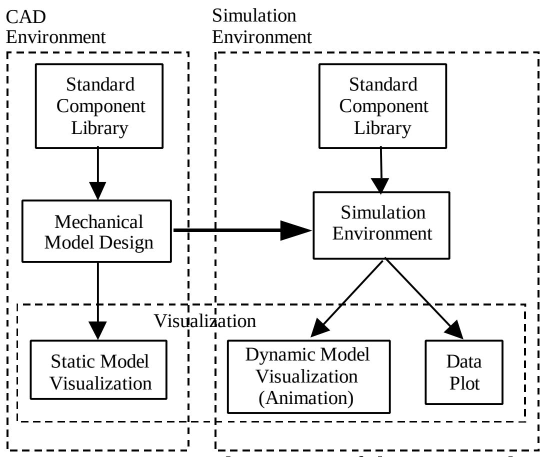
CAD Environment
Simulation Environment
Standard Component Library
Standard Component Library
Mechanical Model Design
Simulation Environment
Visualization
Static Model Visualization
Dynamic Model Visualization (Animation)
Data Plot

Figure 1. Functional structure of the integrated environment

In this paper, we first give a brief introduction to the overall design of the developed simulation environment. At the same time, we examine the implementation details of the developed translator. A brief overview of the Modelica language is also given with an emphasis on the modular and hierarchical facilities of the language. The Modelica Multi Body System Library (MBS) is briefly presented together with a simple modeling and simulation example. We will also present some principles of the developed translator implementation. The use of the translator is demonstrated on several examples. We conclude with an overview concerning further development based on the integrated design and simulation environment.

The Modelica Modeling Language

Modelica is a new language for hierarchical object-oriented physical modeling which is developed through an international effort [Fritzson and Engelson 1998; Elmqvist et al. 1999].

Compared to other modeling languages available today, Modelica offers four important advantages from the simulation practitioner point of view:

- Acausal modeling based on ordinary differential equations (ODE) and differential algebraic equations (DAE). There is also, ongoing research to include partial differential equations (PDE) in the language syntax and semantics [Saldamli and Fritzson 2000].
- Multi-domain modeling capability, which provides the user with the possibility to combine electrical, mechanical, thermodynamic, hydraulic etc., model components within the same application model.
- A general type system that unifies object-orientation, multiple inheritance, and templates within a single class construct. This facilitates reuse of components and evolution of models.
- A strong software component model, with constructs for creating and connecting components. Thus the language is ideally suited as an architectural description language for complex physical systems, and to some extent for software systems.

The reader of the paper is referred to [Modelica Association 1999a] and [Modelica Association 1999b] for a complete description of the language and its functionality from the perspective of the motivations and design goals of the researchers who developed it. Those interested in shorter overviews of the language may wish to consult [Fritzson and Engelson 1998] or [Elmqvist et al. 1999].

Two environments employing the declarative equation based programming paradigm and Modelica language will be extensively discussed in this paper: MathModelica [Jirstrand et al. 1999] and Dymola with Modelica support.

The dynamic simulation capabilities of the language has been demonstrated many times in the literature by modeling and simulating heat exchangers [Mattsson 1997], automatic gear boxes [Otter et al 1997] or hydraulic systems [Ferreira et al 199], [Tummescheit and Eborn 1998]. The advantage of such a modeling language is that the user can concentrate on the logic of the problem rather than on a detailed algorithmic implementation of the simulation model. In order for declarative equation based modeling languages to achieve widespread acceptance, associated programming environments and development tools must become more accessible to the user.

Related Work

In this part of the paper, we briefly survey some of the commercial virtual prototyping packages available which are most closely related to our developed environment.

VisualNastran 4D from MSC Working Knowledge provides an integrated environment for motion and FEA (Finite Element Analysis) simulation and complete suite of tools for the development and communication of physics-based virtual prototypes. Constraints and drivers can be defined by numeric or equation input in the formula editor, or with tabular data.

ADAMS, standing for Automatic Dynamic Analysis of Mechanical Systems developed by Mechanical Dynamics Inc., provides a fully integrated virtual prototyping environment. In addition to the powerful modeling and visualization capabilities includes an analysis engine called ADAMS/Solver which converts an ADAMS model to equations of motion, and then solves the equations, typically in the time domain. ADAMS/Solver can resolve redundant constraints, handle unlimited degrees of freedom, and perform static equilibrium, kinematic, and dynamic analyses.

Dynamic Designer/Motion and Simply Motion, two other products from Mechanical Dynamics Inc., provides a full integration with Mechanical Desktop. Simply Motion written also in AutoDesk's ARX development language extends the design automation capabilities of Mechanical Desktop to include realistic 3D dynamic motion simulation. Simply Motion anticipates the mechanical designer needs and automatically updates the motion data. Through a browser called IntelliMotion Browser, the user can add joints, springs and input motion to the mechanical model.

DADS, standing for Dynamic Analysis and Design System available from LMS International Inc. (CADSI), performs assembly, kinematic, dynamic, inverse dynamic and preload analysis. It incorporates advanced numerical methods to solve Differential Algebraic Equations (DAE) using both implicit and explicit solvers.

The primary limitation of these environments is the difficulty of integrating multi-domain simulation in the same environment. Usually an interface to other popular simulation tools, like MATLAB and Simulink, is provided, but this solution does not offer too much flexibility. We have identified two major needs for a virtual prototyping system:

- The need to integrate multi-domain simulation in the same environment.
- The generation of quality documentation coupled to the design and code.

In the following pages, we detail our proposed procedure for avoiding the current limitations of the software in this area.

MBS (Multi Body System) Library in Modelica

The equation-based foundation of the Modelica language enables simulation in an extremely broad range of scientific and engineering areas. Some of the model libraries include application areas such as mechanics, electronics, hydraulics and pneumatics. The MBS (Multi Body System) library has been developed in [Otter 1995], and an overview can be found in [Otter et al. 1996]. We briefly present the multi-body system library together with a simple modeling examples.

A distinguishing feature of mechanical multi-body systems is the presence of joints, which impose different type of kinematic constrains between the various bodies of the system. Kinematic constraints are enforced between the kinematic variables of two bodies. These constraints express the conditions for relative translation or rotation of the two bodies along or around a body fixed axis, and imply the relative sliding of the two bodies which remain in constant contact with each other. However, in Mechanical Desktop it is possible to define a relative distance when specifying the mates between two bodies.

In order to correctly simulate a mechanism, it is necessary to have a closed kinematic chain with one link fixed. When we say that one link is fixed, we mean that it is chosen as a frame of reference for all other links, i.e., that the motion of all other points on the linkage will be measured with respect to this link, thought of as being fixed. The CAD environment has the possibility of specifying which part of the kinematic chain will be fixed. Later, in the translation phase, this fixed part will be connected with an instance of the `InertialSystem` class.

An instance of the `Inertial` class defines the global coordinate system and gravitational forces (the inertial frame). All parameter vectors and tensors are given in the home position of the multi-body system with respect to the inertial frame. One instance of class `Inertial` must always be present for every multi-body model. All other objects are in some way connected to the inertia system, either directly or through other objects.

Every basic mechanical component from the MBS library has at least one or two interfaces to connect the element rigidly to another mechanical elements.

An example of the MBS library usage is shown by the following modeling examples.

Our first modeling example consists of a mass hanging on a spring in a gravity field. When the spring-mounted body is disturbed from its equilibrium position, its ensuing motion in the absence of any imposed external forces is termed free vibration. However, in the real world, every mechanical system posses some inherent degree of friction which will act as a consumer of mechanical energy. Therefore, we should add to our system a viscous damper for the purpose of limiting or retarding the vibration.

A schematic diagram of our system under consideration is shown in Figure2.

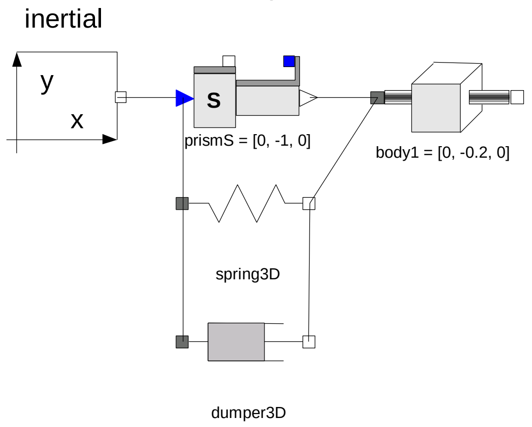

Figure 2. Schematic diagram of the damped free vibration mass system

The Modelica code for the model considered above is shown below:

```
import "library/mbs/mbscom.mo";
import "library/mbs/mbs1.mo";
import "library/drive/drivep1.mo";

model DFVmass

  parameter Real c = 300;
  Inertial inertial;
  PrismaticS prismS1 (n=[0,-1,0]);
  BoxBody body1(r=[0,-0.2,0],Width=0.2,
                             Height=0.2);
  Spring spring3D (c=c);
  Damper damper3D(d=2);

  equation
       connect(inertial.b, prismS1.a);
       connect(prismS1.b,  body1.a);
       connect(prismS1.a,  spring3D.a);
       connect(spring3D.a, damper3D.a);
       connect(spring3D.b, damper3D.b);
       connect(body1.a,    spring3D.b);

end DFVmass;
```

As we have seen from the previous example, a simulation model that uses the MBS library consists of an inertial system (an instance of `Inertial` class) and different mechanical components connected together with the `connect` statement. The statement `connect(v1,v2)` expresses coupling between variables. These variables are called connectors and belong to the connected objects. Each connection specifies interaction between components. A connector should contain all quantities needed to describe the interaction. This gives a flexible way of specifying topology of physical systems described in an object-oriented way using Modelica.

A distinguishing feature of multi-body systems is the presence of joints, which impose different types of kinematic constrains between the various bodies of the kinematic chain. The motions between links of the mechanism must to be constrained to produce the proper relative motion, those chosen by the designer for the particular task to be performed.

A Prismatic joint has been introduced in order to produce the relative motion in the Y-direction. The relative motion direction is specified by the parameter vector n=`[0,-1,0]` which is the axis of translation resolved in frame a.

Mechanical Desktop

In the above we have explored how a simple simulation is expressed with the help of the Modelica language and the Multi-Body library. Now we can take a look how typical design of a multi-body system is done in our target CAD environment and what facilities are offered by this environment.

AutoDesk's Mechanical Desktop, like other typical CAD/CAM systems supports modeling the geometry of parts and static assemblies of parts. The assemblies can be built combining two or more parts, or parts grouped in subassemblies. Like part features, parts and subassemblies act like building blocks. Each solid component (a rigid body) is modeled as a separate part which can be saved in a separate document and later externally referenced to the assembly. In an assembly model, these parts are put together in order to form a complete model. Mechanical Desktop builds individual parts and subassemblies into an assembly. Using externally referenced parts into an assembly creates a truly parametric assembly design. Changes to an external reference can be made from within the assembly or in the original file.

Mechanical Desktop has the possibility to automate the design and revision process by using parametric geometry, which controls relationships among design elements and automatically adjusts models and drawings as they are refined.

The assembly document defines the mobility between the parts of an assembly. After parts or subassemblies have been created, constraints are applied to position them relative to one another. Each time when a constraint is applied to a part, some degrees of freedom are eliminated. After the parts have been assembled and constraints have been applied an interference checking can be performed and mass calculations can be performed on parts to insure that they are structurally sound.

The Mechanical Desktop offers the following mate types:

- AMMATE - Mate constraint. Causes a plane or axis on one part to be coincident with a plane, point, or axis on another part in a specified direction. Removes a translational or rotational degree of freedom.

- AMFLUSH - Flush constraint. Make two planes coplanar with their faces aligned in the same direction.
- AMINSERT - Insert constraint. Aligns center points and planes of two circles in a specified direction. Removes translational degrees of freedom. Used to constrain a bolt in a hole, for example.
- AMMANGLE – Angular Constraint. Specifies an angle between two planes, two vectors, or a combination of a plane and a vector.

The links of the designed mechanism have been connected together in some manner in order to transmit motion from the *driver* (input link) to the *follower* (output link). The joints between the links are also called (kinematic) pairs, because each joint consist of a pair of mating surfaces, two elements, one matting surface or element being part of each of the joined links as it is shown in Figure 3.

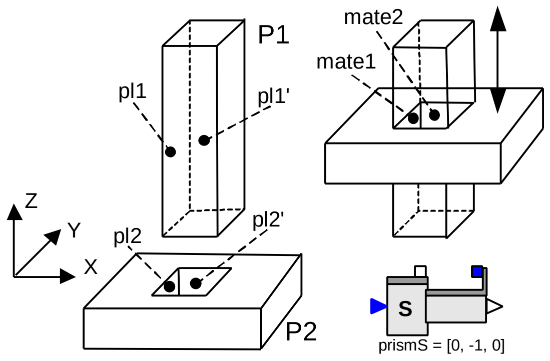

Figure 3. Two links connected together by two coincident plane/plane mates and the corresponding joint from the MBS library

Each time when we apply a constraint to a part, some degrees of freedom are eliminated. The number of degrees of freedom determines the movement of a part in various directions; the more constraints applied, the less the part can move. At the beginning, we have applied a mate constraint **`mate1`** which cause plane **`pl1`** from part **`P1`** to be coincident with plane **`pl2`** from part **`P2`**. The mate constraint **`mate1`** has eliminated a translational degree of freedom on axis X and two rotational degrees of freedom: one around axis Y and the other one around axis Z. Applying the second constraint **`mate2`** by constraining plane **`pl2'`** to be coincided with plane **`pl1'`** we have eliminated a translational degree of freedom of axis Y and the rotational degree of freedom around axis X. The rotational degree of freedom around axis Z have been already eliminated by the first applied mate constraint. In conclusion, our assembly will have only one degree of freedom, namely a translational degree of freedom on axis Z.

The Mate constraint has the following available options:

- Two planes coplanar with their normals aligned in opposite directions (facing each other).
- An axis *planar* with a plane.
- Two axes that share the same direction and slope (collinear).
- A point that lies on an axis.
- Two coincident points.
- A sphere, cylinder, or cone tangent to a plane or to other spheres, cylinders, and cones.

The translator will gather the information about the mate constraints applied to the two parts and will be translated to a corresponding joint object from the MBS library. In our example the translated joint will correspond to a prismatic joint which only permits a relative sliding motion and therefore is often called a sliding joint. This type of joint only has a single degree of freedom.

Any invalid combinations of mates are automatically rejected by the Mechanical Desktop so the possibility of translating to an incorrect joint is eliminated already during the design phase. Each valid combination of mates will be translated to a combination of joints, which resides in the multi-body library. Table 1 lists the names of the lower joints, together with the number of translational and rotational degrees of freedom [Shigley 1995], and their correspondents from the MBS library. All other joint types are called higher pairs. They are combinations of the basic joints and are not listed in Table 1.

Pair	Tr. and Rot. DOF	Nr. of DOF	Relative motion	MBS name
Revolute	1rot	1	Circular	RevoluteS
Prismatic	1tr	1	Linear	PrismaticS
Cylindric	1rot; 1tr	2	Cylindric	CylindicalS
Sphere	3rot	3	Spheric	SphereCardanS
Flat	3tr	3	Planar	PlanarS

Table 1. Lower joints

Translator Implementation

The translator was implemented as a plug-in to AutoDesk's Mechanical Desktop by using a provided application development tool, AutoDesk Mechanical Application Programming Interface (MCAD API) [AutoDesk 1999b]. The MCAD API provide a direct and unified access mechanism for AutoCAD and Mechanical Desktop geometry making possible in that way to translate the geometrical, mass, inertia and constraint information to source code in Modelica. From the AutoCAD point of view, our translator is an ObjectARX application. An ObjectARX application is a dynamic link library (DLL) that shares the address space of AutoCAD and makes direct function calls to AutoCAD. It is possible to add new classes to the ObjectARX. The ObjectARX entities created are virtually indistinguishable from the built-in AutoCAD entities. The ObjectARX protocol can be extended by adding functions at runtime to existing AutoCAD classes [AutoDesk 1999a].

.

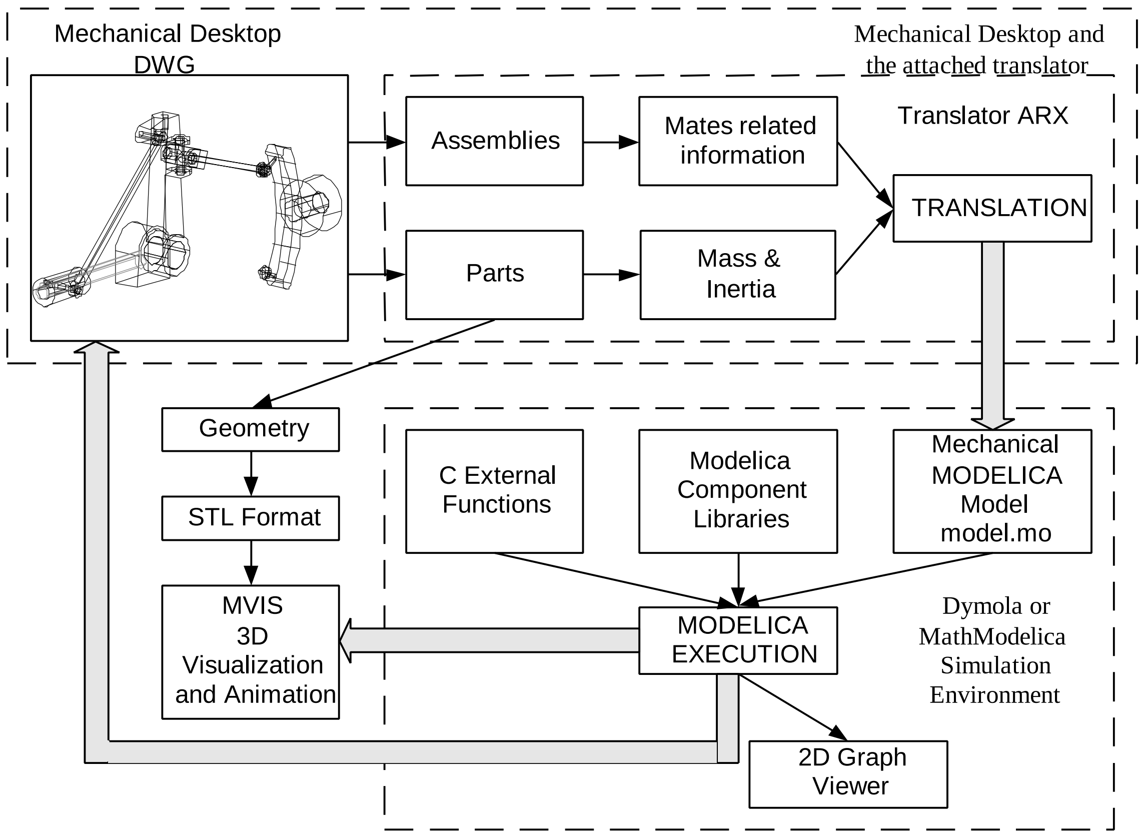
Mechanical Desktop
DWG
Mechanical Desktop and the attached translator
Translator ARX
Assemblies
Mates related information
Parts
Mass & Inertia
TRANSLATION
Geometry
STL Format
MVIS 3D Visualization and Animation
C External Functions
Modelica Component Libraries
Mechanical MODELICA Model model.mo
MODELICA EXECUTION
Dymola or MathModelica Simulation Environment
2D Graph Viewer

Figure 4. The path from a Mechanical Desktop static model to a dynamic system visualization

The overall architecture of our virtual prototyping environment and how the developed translator is integrated in this environment is shown in Figure 4.

A mechanical model designed and fully constrained in the Mechanical Desktop Environment serves as a starting point in the virtual prototyping. The models are saved in the DWG format, which contains all the information related to the geometrical properties of the parts and information related to the mechanical assembly like mates and constraints.

The geometry of each part is exported to the STL file format [3Dsystems, 2000]. At the same time, mass and inertia of the parts are extracted together with mates information from the mechanical assembly. The translator will use this information to generate a corresponding set of Modelica class instances with connections between them. This automatically generated Modelica file is processed by a simulation environment like Dymola or MathModelica. In contrast to other virtual prototyping environments presented in the related work chapter, our environment creates readable Modelica code so the programmer can combine it with other code fragments and modify it if necessary. For instance, the simulation code can be enhanced by adding other components from other Modelica libraries or by adding externally defined C code. In that phase electrical, control or hydraulics components can be added to the generated mechanical model, providing in that way a multi-domain simulation.

By default, only the gravity force is applied to the translated model. The translated CAD model models a set of dynamic equations of motion. Therefore, the simulation can predict the mechanism response to a given set of initial conditions or force (or torque) load, which might be function of time. External functions can be specified by adding an instance of `ExtForce` class from the MBS library.

The results of the simulation can be visualized as 2D plotting of the simulation variables or as a 3D dynamic animation of the mechanical assembly with the MVIS (Modelica VISualizer) and OpenGL based interactive visualization tool [Engelson 2000]. Compared to the similar translator developed for SolidWorks [Larson 1999], the main improvement of the Mechanical Desktop to Modelica translator is the possibility of visualizing the simulation result inside the CAD editor (marked in Figure. 5 as the arrow which connects the Modelica execution with the Mechanical Desktop assembly drawing). This is usually combined with visualization of variables in the 2D Graph viewer incorporated in the simulation environment.

The overall system will produce a dynamic multi-body system simulation in time domain for the evaluation of the dynamic interaction between several parts of the mechanical system or between the mechanical assembly and the attached controller. This is very useful in optimizing the mechanical components geometry, and leads to a good deal of confidence when it comes the time to go from drawings to fabricating the equipment. Simulating a couple of scenarios with mechanical rigid body models and inspecting animations and numerical results has validated our approach.

More details will be given next about the MVIS – Modelica Interactive Visualization Tool and about the geometric data translation between the CAD system and the simulation environment together with a brief description of the STL geometry export format.

MVIS
Modelica Interactive Visualization Tool

The integrated environment includes a 3D viewer that provides online dynamic display of the assembly (during simulation) or offline (based on the saved state information for each step).

This tool loads the corresponding STL file for each part and optimizes is for rendering. After that, rendering is performed by OpenGL library functions. During optimization, the vertices positioned very close are merged together. Optimized STL code is stored in a binary file for future use.

The user can alternatively use the pop-up menu system, keyboard shortcuts, or a command string in order to control various options. We found that the following options (that can be turned on and off) should be available, and we have implemented them:

- Rotating the camera in 2 degrees of freedom (DOF), moving the camera in three DOF, zooming in and out.
- Using perspective and orthographic projections.
- Parts are displayed as a wire-frame, lighted or hidden, with or without a shadow.
- Vectors (forces, velocities, etc.).
- Application specific environment (road for car simulation, or an airport for flight simulation).
- Planned and actual trajectory (mission) of some parts.
- Synchronization of animation with machine clock.
- Starting, stopping, continuing animation, steeping forward and backward.
- Targeting camera center of view on a particular part, so that camera follows the part all the time
- Rotating camera together with the target part

Geometric Data Translation. The STL File Format.

Geometry information is saved in a separate STL file for each mechanical part using the export capabilities of the Mechanical Desktop environment. STL files are used for representation of 3D surfaces. The surface is tessellated or broken down logically into a series of small triangles (facets). Each facet is described by its normal vector and three points representing the vertices (corners) of the triangle.[3DSystems 2000]. The resolution of the mesh created by these triangles can be controlled by the AutoCAD system variable FACETRES. Our translator uses the STL ASCII format for translating the geometry information. The syntax for an ASCII STL file is as follows:

```
solid
...
facet normal 0.00 0.00 1.00
  outer loop
       vertex  2.00  2.00  0.00
       vertex -1.00  1.00  0.00
       vertex  0.00 -1.00  0.00
  endloop
endfacet
...
endsolid
```

In future, the environment will be able to support other geometry formats, for instance VRML.

A Double Pendulum Model Translation and Simulation

The next part of the paper reports direct modeling experience with the implemented translator by showing a simple modeling and simulation example.

In the following, we analyze the simulation of a simple mechanical double pendulum with the purpose of validating our environment and showing its basic capabilities. The model of the pendulum (Figure 5) was designed and fully constrained in Mechanical Desktop.

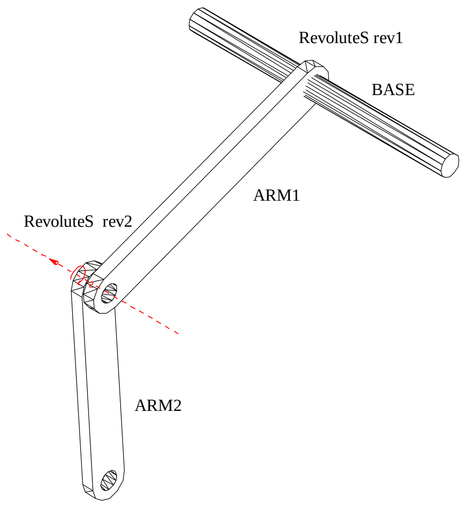

Figure 5. Double Pendulum model, designed and constrained in Mechanical Desktop

At the start, the first link of the pendulum has an angle (w.r.t the vertical axis) and the initial velocity is 0.

First, based on the extracted mass and inertia related information, three instances of the BodyME class will be created: BASE, ARM1, ARM2. In addition to the normal Body class, this class includes information how to position the object for the purpose of visualization with the MVIS tool. A complete description of the BodyME class can be found in [Larsson 1999]. Then, based on the assembly constraint information the type of joints will be identified.

The translator will automatically create a revolute joint when two components share a line/line and a plane/plane assembly constraint and the line/line is oriented along the normal of the plane/plane. In the design of the presented pendulum we have used the AMINSERT command in order to create the assembly constraint between the two circular faces of the connecting part of the pendulum. AMINSERT will constrain the parts by making the selected edges or faces share the same axis, while forcing their faces to be coplanar. The same type of mate was used to constrain the first part of the pendulum with the support cylinder. The automatically translated revolute joint will allow the corresponding two parts of the pendulum to rotate relative to each other about a common axis. It will remove all three translational degrees of freedom and two rotational degrees of freedom from the part is attached to.

Then coordinate translation components (class `Bar`) are added between the connection points. The kinematic diagram of the pendulum under consideration is shown in Figure 6.

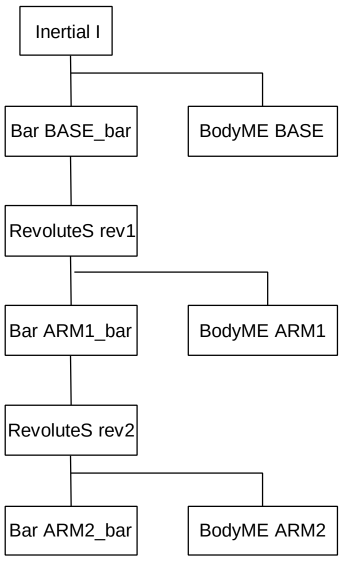

Figure 6. Kinematic diagram of the pendulum under consideration

The translator will automatically generate the connections between the class instances. The following connections will be generated as a result of the assembly mechanical design structure:

```
connect(I.b, BASE.a)
connect(I.b, BASE_bar.a);
connect(BASE_bar.a, rev1.a);
connect(rev1.b, ARM1.a);
connect(rev1.b, ARM1_base.a);
connect(ARM1_base.s, rev2.a);
connect(rev2.b, ARM2.a);
connect(rev2.b, ARM2_bar.a);
```

Up to this point, we have automatically generated the Modelica simulation code for the given mechanical design. After the translation phase the user can easily add the first revolute joint a rotary motion generator attached to its available rotational degree of freedom by editing the generated code. At this phase, multi-domain simulation can be integrated with the mechanical model by connecting it with other domain simulation models. The simulation environment allows the user to couple various physical domains in one simulation.

The added visualization capabilities are very important both for engineering interpretation and for design reviews. Data variable plots and animation of the mechanical system make it even easier to spot trends and region of interest.

Integration with the MathModelica Simulation Environment

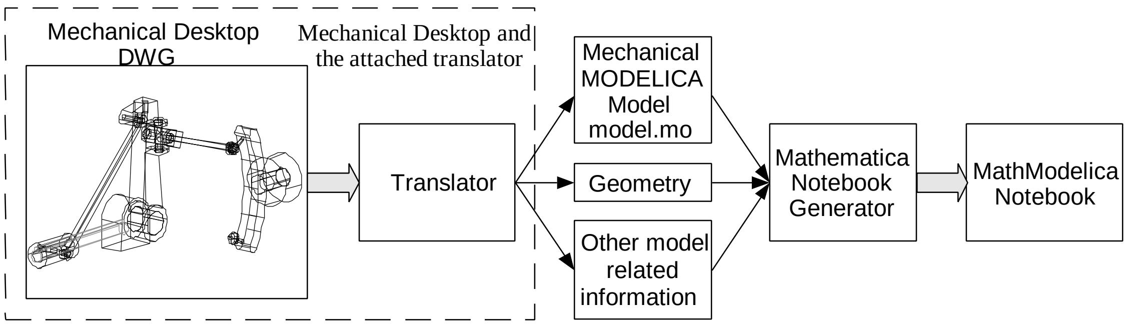
Mechanical Desktop DWG
Mechanical Desktop and the attached translator
Translator
Mechanical MODELICA Model model.mo
Geometry
Other model related information
Mathematica Notebook Generator
MathModelica Notebook

Figure.7 CAD integration with the MathModelica environment

Our environment will, in the future, be fully integrated with the MathModelica environment. MathModelica is an integrated problem-solving environment (PSE) for full system modeling and simulation [Jirstrand et al. 1999; Jirstrand 2000]. The environment integrates Modelica-based modeling and simulation with graphic design, advanced scripting facilities, integration of code and documentation, and symbolic formula manipulation provided via Mathematica. Import and export of Modelica code between internal structured and external textual representation is supported by MathModelica. The environment use extensively the principles of literate programming [Knuth 1984] and integrates most activities needed in simulation design: modeling, symbolic processing, transformation and formula manipulation, storage of simulation models, version control, input and output data visualization, storage and generation of documentation. Mathematica [Wolfram 1996] is an interpreted language and integrates several features into a unified integrated environment: numerical and symbolic calculations, functional, procedural, rule-based and graphical programming. Also the language incorporates many features of traditional mathematical notation and the goal of the language is to provide a precise and consistent way to specify computations.

Mathematica is divided into two distinct parts: the computer algebra engine ("kernel") that receives and evaluates all expressions sent to it and the user interface ("front-end"). The front-end provides the program interface to the user and is concerned with such issues as how input is entered and how computation results are displayed to the user. Mathematica's front-end documents are called notebooks [Wolfram 1996]. They combine text, executable commands, numerical results, graphics, and sound in a single document. A notebook provides the users with a medium in which they can document their solution along side the computation itself.

A functional structure diagram of the CAD integration with the MathModelica environment is given in Figure 7. A translator will extract all the necessary information from the CAD system generating a mechanical Modelica model, geometry related information and other related information of the mechanical model. This information will be processed by a Mathematica notebook generator which outputs a MathModelica notebook. The generated notebook is processed by the MathModelica simulation environment using the incorporated simulation engine, 2D plotting of the system variables are generated together with a realistic animation of the mechanical system under consideration.

The advantages of such a development chain for virtual prototyping are:

- Early detection of mechanical design flaws.
- High quality generated code.
- High quality documentation generation which is coupled to the design and code.
- Reduced time for design validation and implicitly reduced development time.
- Multi-domain modeling is made possible.
- The possibility to handle information about a product's entire life cycle, from the design phase to the manufacturing phase.

SUMMARY AND FUTURE WORK.

The objective of the work presented herein was to demonstrate that the integration of a typical CAD/CAM system and an equation based simulation environment can produce a feasible virtual prototyping environment with an enhanced flexibility compared to other traditional commercially available environments. The main advantage of our environment is that the multi-domain simulation is made possible in the same environment.

The improved CAD integration provides the users with a more intuitive and sound way of constructing and verifying large, moving assemblies. In that way designers can take into account the dynamic nature of the problem and simulate the entire mechanical assembly, rather than visualize a static part or a small subassembly, resulting in more accurate modeling and design solutions

Commercially available MBS simulation packages like ADAMS or Working Model 3D cannot be directly used for modeling the motion of mechanical systems with attached controllers. To solve this problem we propose a methodology based on the utilization of possibilities available from both Mechanical Desktop and Modelica. The efficiency of the proposed methodology has been illustrated by the modeling and simulation of the motion of a simple pendulum. The integration of Mechanical Desktop and Modelica enabled us to prove that our design concept of a mechanical system can fulfill the functional specification. The control algorithms can be tested in parallel with the mechanical design of the systems.

The combination of two environments, the CAD modeling environment and the simulation environment, in effect, collapse the phase of coding.

In the future, we shall concentrate on the following tasks:

- A complete integration of the CAD translators to the MathModelica environment
- Better collision detection handling and visualization of forces and effects of collisions.
- Automatic output of the force data to finite element analysis (FEA) packages for structural analysis and other applications.

ACKNOWLEDGMENTS

The work has been supported by the European Commission Research and Development Program as part of the RealSim project and by the ECSEL Graduate School supported by the Swedish Strategic Research Foundation.

REFERENCES

3Dsystems Inc, 2000. “Stereo Lithography Interface Specification.”

AutoDesk Inc. 1999a. “AutoCAD 2000 ObjectARX Developer’s Guide.

AutoDesk Inc. 1999b. “Mechanical Application Programming Interface [API] – Developers Guide.

Elmqvist, H.; S. E. Mattsson and M. Otter. 1999. “Modelica - A Language for Physical System Modeling, Visualization and Interaction.” In Proceedings of the 1999 IEEE Symposium on Computer-Aided Control System Design (Hawaii, Aug. 22-27).

Elmqvist, H.; D. Bruck; M. Otter. 1996. *Dymola – User’s Manual*. Dynasim AB, Research Park Ideon, Lund

Engelson, V. 2000. “Tools for Design, Interactive Simulation, and Visualization of Object-Oriented Models in Scientific Computing”. Ph.D. Thesis, IDA, Linköping University, Sweden.

Engelson, V.; H. Larsson; P. Fritzson. 1999. “A Design, Simulation and Visualization Environment for Object-Oriented Mechanical and Multi-Domain Models in Modelica. “ In *Proceedings of 1999 IEEE International Conference on Information Visualization* (London, July 14-16), 188-193.

Fritzson P. and V Engelson. 1998. “Modelica - A Unified Object-Oriented Language for System Modeling and Simulation.” In *Proceedings of the 12th European Conference on Object-Oriented Programming* (ECOOP'98 , Brussels, Belgium, Jul. 20-24).

Ferreira, J.A.; J.E de Oliveira; V.A. Costa; 1999. “Modeling of Hydraulic Systems for Harware-in-the-loop Simulation: a Methodology Proposal.” In Proceedings of *The International Mechanical Engineering Congress and Exposition.* (Nashville, USA, November 14-19).

Jirstrand, M.; 2000. “MathModelica – A Full System Simulation Tool”. In Proceedings of *Modelica Workshop 2000* (Lund, Sweden, Oct. 23-24)

Jirstrand, M.; J. Gunnarsson; and P. Fritzson. 1999. “MathModelica – a new modeling and simulation environment for Modelica. ” *In Proceedings of the Third International Mathematica Symposium* (IMS’99, Linz, Austria, Aug).

Knuth, D.E. 1984. “Literate Programming” *The Computer Journal*, NO27(2) (May): 97-111.

Larson, H.,1999. Translation of 3D CAD models to Modelica. Master thesis. IDA, Linköping University, Sweden, April.

Mattsson, S. E. 1997. “On Modeling of Heat Exchangers in Modelica. “ In *Proceedings of European Simulation Symposium* (Passau, Germany, October 19-22)

Modelica Association 1999. *Modelica – A Unified Object-Oriented Language for Physical Systems Modeling - Tutorial and Design Rationale Version 1.2*(Dec).

Modelica Association 1999. *Modelica – A Unified Object-Oriented Language for Physical Systems Modeling – Language Specification Version 1.3.* (Dec).

Otter, M.; C. Schlegel; H. Elmqvist. 1997 “Modeling and Realtime Simulation of Automatic Gearbox using Modelica.” In *Proceedings of European Simulation Symposium* (Passau, Germany, October 19-22).

Otter, M. ; Elmqvist H.; F. E. Cellier. 1996. “Modeling of Multibody Systems with the Object-oriented Modeling Language Dymola, Nonlinear Dynamics, 9:91-112, Kluwer Academic Publishers.

Otter, M. 1995 Objektorientierte Modellierung mechatronischer Systeme am Beispiel geregelter Roboter, Dissertation, Fortshrittberichte VDI, Reihe 20, Nr 147.

Saldamli, L.; Fritzson, P.;2000 “Object-oriented Modeling With Partial Differential Equations”. In Proceedings of *Modelica Workshop 2000* (Lund, Sweden, Oct. 23-24)

Shigley, J. E. 1995. *Theory of Machines and Mechanisms.* McGraw-Hill, New York series in mechanical engineering.

Tummescheit H.; J.Eborn. 1998. “Design of Thermo-Hydraulic Library in Modelica.” In Proceedings of *The 12th European Simulation Multiconference* (Machester, UK, June 16-19)

Wolfram S.. 1996. *The Mathematica Book*. Wolfram Media Inc. (February).