ABSTRACT
A model import tool referred to as the MutliTranslator (MT), which is used by the Paragon modeling environment, is described in this paper. The MultiTranslator is based on a standard XML format and also provides an interactive wizard to add and/or edit any information to the imported model. The approach described here also demonstrates how Paragon’s XML schema can act as a centroid for third party tools and make the models available in widely used languages like VHDL-AMS, Verilog-AMS and MAST. This mechanism is illustrated with importing a Modelica model into the Virtual Test Bed simulation environment.

Keywords
Modelica, model import, XML, tools, HDLs

1. INTRODUCTION
The need for modeling different electrical and mechanical devices in the same simulations is gaining increasing focus as more sensor/actuator systems are deployed. The SPICE-based simulators lack a convenient way to develop either mixed-signal or mixed-technology models. Mixed-signal Hardware Description Languages (HDLs) [1, 20] are providing the flexibility of letting the designer write his/her own models. Knowing the fact that developing and maintaining behavioral models even in these HDLs is a time-consuming and error-prone process, making these models available in multiple simulation environments has become another major requirement in the contemporary design world [23]. The language technology alone is not sufficient to meet the challenges of the complexity of the design of real systems. This is due to the fact that complexity of such systems is now involving people from a variety of scientific backgrounds. It is very unlikely that each of these disciplines will have taught the use of the same languages or the same simulator.

With the aid of advanced modeling tools, models can be efficiently shared among designers in their convenient environments for verification purposes. The objective of this paper is to illustrate the usefulness of tools like Paragon [2-4] and MultiTranslator [5], [6] where both use the same language-independent format to represent model information. The MultiTranslator is a software tool developed at the Taganrog State University of Radio Engineering (TSURE) intended to translate models written in Advanced Continuous Simulation Language (ASCL) [7], Modelica [8] and other modeling languages. Each translator is equipped with an individual grammar module, which can translate the model information into other formats. Among the existing MultiTranslator modules, the Modelica importer is described in this paper. The imported Modelica model is embedded and simulated in the Virtual Test Bed simulation environment.

2. MOTIVATION
The VTB is an efficient mixed-technology simulation and visualization tool. As in normal simulators, there is always a growing need for new models to be added to the existing library. The simulation kernel supports a C++ based interface for models rather than popular HDLs. Handling simulator specific information in models also complicates the procedure of bringing new models into VTB. To fulfill this need, Paragon was chosen as a companion tool to bring new models into the VTB environment. The model importer extracts all the model information and removes all language specific constructs to save in Paragon’s internal format.
Once a model is rendered in this internal format, it can be viewed pictorially and better understood with the help of graphical editors. The biggest advantages of the model importing mechanism are: a) harness the code generation capability of Paragon to generate code for the same model in multiple forms, b) act as a teaching/training aid to others needing to understand the internal workings of a previously coded model, c) the creation of a more object-oriented model such as that found in Paragon, and d) automatic generation of model documentation (analogous to model data sheets).

3. Modelica Importer
Paragon was designed to create a user-friendly modeling environment, which would alleviate the need of mastering various HDLs [16]. The present design utilizes XML and MathML [17], [18] as the internal format. As XML is a widely accepted standard and available open-source, it is well suited for expressing model descriptions. The extensible nature of XML easily facilitates addition of new features and phenomena, for example, multi-physical systems (thermal, optical, mechanical, etc). Paragon’s XML template can be described via the diagram shown in Fig. 1 below.

![Paragon’s internal XML template](image1)

Fig. 1. Paragon’s internal XML template.

The entirety of the model information is encapsulated in the Model parent tag. The Model tag in turn is sub-divided into two tags Interface and Body, where the former represents the model parameters and connection points information and the latter incorporates the model implementation details. Parameter processing and ranges of validity information are saved in the parameter tag whereas the name, type and nature of connection points are saved in the port tags. The model can be viewed pictorially by a set of branches where each branch is associated with through and across variables. The Topology tag gives the branch connections and wiring information of the model.

Modelica is an object-oriented modeling language designed to allow convenient component-oriented modeling of complex physical systems (e.g., systems containing mechanical, electrical, electronic, hydraulic, thermal, control, electric power or process-oriented subcomponents). The general template of a Modelica model is shown in Fig. 2. The structure of a Modelica model can be divided into three sections. When compared against Paragon’s XML format, section 1 contains the information of model parameters, ports and internal variables. Both the parameters and ports information are saved in the Interface tag, whereas the internal variables are saved in the Body -> Internals section. In a similar way, section 2 carries all the model topology information, which is saved in the Topology and Branch tags. Section 3 represents the characteristic equations governing the model behavior and this information is saved in the equation tags of the Paragon format. All of the model expressions and equations are expressed in MathML in the XML representation, which is an XML application itself for describing mathematical notations.

![General Modelica template structure](image2)

Fig. 2. General Modelica template structure.

![Block diagram of the model importer in Paragon](image3)

Fig. 3. Block diagram of the model importer in Paragon.
3.1. Implementation

The MultiTranslator utilizes a grammar module for translating Modelica models into the target XML description. The block diagram in Fig. 3 shows the overall process of the model import mechanism. The MultiTranslator acts as a plug-in tool for importing Modelica models into Paragon, whereas the native VTB C++ code is generated by an automatic code generation module called UDD (User Defined Device) [19]. Further, the generated C++ is compiled to a Dynamic Linked Library (dll) and then loaded into the VTB model library. The integration of Paragon and the MultiTranslator is implemented using the client-server Component Object Model (COM) interface. The MultiTranslator acts as a server and responds to the function calls made by Paragon to import the Modelica model.

The MultiTranslator’s grammar module contains the description of the Modelica language grammar constructs and a set of rules. These rules define the actions to take place based on the input to generate the correct XML output. A section of the Modelica grammar module defining two rules is shown in Fig. 4. Every rule is followed by a keyword rule and actions to be performed are defined within the body of the rule section. Multiple actions can be performed based on the input in the variant section. The example given below demonstrates how mathematical operators are converted to equivalent XML format. The <mo> tag is the presentation tag used in MathML to represent mathematical operators. After parsing the Modelica model, the grammar module converts the model to Paragon’s format on a query-driven basis.

3.2. MultiTranslator Model Wizard

The model wizard is a part of the MultiTranslator tool, which allows the user to interact with the tool while importing a Modelica model. This wizard allows the user to make changes when the model is being imported into Paragon, such as augmenting the model with additional information, while still keeping the Modelica model unaltered. The screenshot of the model wizard is shown in Fig. 5 illustrating with a DC motor model example given below. In the example shown, the model consists of a set of variables and two equations describing the model behavior. Though this is legal in Modelica, in HDLs connection points are needed to instantiate the model in a netlist. In this case, this wizard will give the user the flexibility of entering connection points, their natures and branches between connection points.

```model DCMotor "DC Motor"
Real Tq=10.0 "Torque applied at the shaft"
Real V=20.0 "Voltage across terminals A and B"
Real i "Current through terminal A"
Real w "Angular rotor speed"
Real J=0.5 "The rotor equivalent moment of inertia"
Real R=0.4 "Resistance"
Real L = 0.0025 "Self-inductance"
Real w0=125.664 "Rated angular speed"
Real V0=115.0 "Rated voltage"
Real b=0.196 "Constant of tough friction"
equation
V=L*der(i)+R*i+ V0/w0*w;
Tq=-J*der(w)+ V0/w0*i-b*w;
end DC Motor;
```

Once the model is imported into the Paragon modeling environment, native VTB C++ models can be generated through the UDD code generation module. This UDD code generation module expects the model information to be in a specific format before generating native VTB models. For example, UDD always expects equations having through variables as a function of across variables. Paragon’s internal code generation and analysis tools are capable of converting the model information into UDD compatible format. The UDD file for the DC motor model is shown in Fig. 6.
4. RESULTS

The last step before exporting the models into the VTB simulation environment is to compile the generated C++ code to a .dll and saving it into the model library. The VTB symbol editor can be used to create a symbol for the new model for further use in simulations. The DC motor model was imported into Paragon and the equivalent VTB model was generated utilizing the UDD tool. A drive system consisting of a petroleum engine, flexible shaft, single-phase generator, DC-motor and ventilator was created in the VTB environment. Two DC motors were used while one of them served as a generator. To verify the results, two drive systems were created, one with the components available in the VTB model library, whereas the other with models imported and generated using MultiTranslator, Paragon and UDD tools. Both the test setups are shown as a screenshot in Fig. 7. Both mechanical and electrical characteristics such as voltage values for the generators and angular speed for DC motors were plotted and compared as shown in Fig. 8. Many other models like those of a gyrator, propeller, shaft and transformer were imported and verified against built-in VTB models. The model import mechanism was also very successfully implemented with MAST models [16].

5. CONCLUSIONS

This paper describes an example of how a standard, open-source interface such as the XML format used in Paragon can be leveraged to deploy models of one language into multiple other possibilities. Consequently, this prevents models from becoming obsolete when designers switch to different design environments. Modelica models were imported into Paragon and then exported into C++ (via the UDD format) for the Virtual Test Bed mixed-technology system simulation environment.
6. REFERENCES


[18] Mathematical Markup Language (MathML), http://www.w3.org/Math


