Modeling Vehicle Dynamics Based on Modelica

Shuguang Feng[1], Jifeng He[2], and Lichen Zhang[3]

[1,2,3] *Shanghai Key Laboratory of Trustworthy Computing,
East China Normal University,
Shanghai 200062, China,
E-mail:shuguang826@gmail.com*

Abstract

Modelica is a language for modeling the physical world. This language is developed by researchers from different organizations. Modelica is an object-oriented language, and based on equations, which belongs to non-causal modeling. There are tools developed for Modelica. Modelica can model vehicle motion in a lively and easily way. With Modelica, many complex systems can be modeled and analysed.

Keywords: *Modeling physical world, Object-oriented language, equation-based language, vehicle dynamics*

1 Introduction

Modelica is designed for physical modeling. Modeling and simulation are becoming more important since engineers need to analyse increasingly complex systems composed of components from different domains. Examples are mechatronic systems within automotive, aerospace and robotics applications. Such systems are composed of components from domains like electrical, mechanical, hydraulical, control, etc.[1] Directly designing and developing in these domains may lead to high cost and low efficience, for the non-thought up and uncertain factors. Therefore, modeling is important for a mature product.

In October 1996, a group of researchers from universities and industry started work towards standardization for the several existing object-oriented languages at the present.[2] Therefore, Modelica came out with aiming at modeling dynamic behaviour of engineering systems and meta-modeling extensions have been developed. Then, Modelica has become a standard in this domain.

Modelica has some features comparing with other modeling language: object-oriented, non-causal modeling, allow tools to generate component. These kinds of art in Modelica make it more suitable for modeling complex system.[3]

Modelica is object-oriented. Every thing in Modelica can be an object, such as a Class, Function, Model, Connector, Record. Every object has its attribute and equation. Object-oriented language is suitable for large systems for reusability, which does a lot help for the efficience and can reduce system complexity.

Modelica is non-causal modeling. Non-causal modeling is based on equations and can lead to lower system complexity comparing with the value-assigned modeling style. The solution of equations of a model can be assigned according to the data flow automatically.

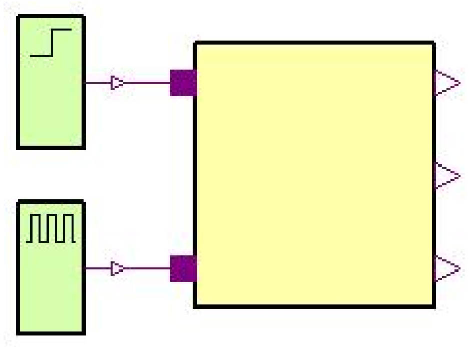
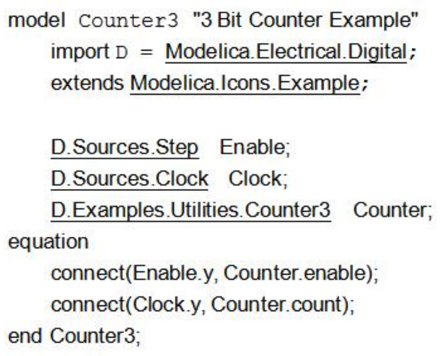


Figure 1. 3 Bit Counter



```
model Counter3 "3 Bit Counter Example"
      import D = Modelica.Electrical.Digital;
      extends Modelica.Icons.Example;

      D.Sources.Step    Enable;
      D.Sources.Clock    Clock;
      D.Examples.Utilities.Counter3    Counter;
equation
      connect(Enable.y, Counter.enable);
      connect(Clock.y, Counter.count);
end Counter3;
```

Figure 2. 3 Bit Counter‘s code in Modelica

Modelica has tool-support to generate component. There are several tools can support drag-component style with Modelica, such as, OpenModelica and Dymola.

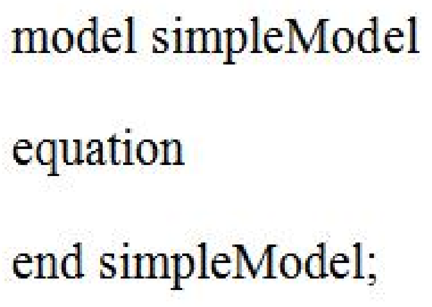
Modelica has a standard library. This library is useful for elementary modeling mechanical, electrical, electronic, and hydraulic elements.[4] Modelica is open-source so that users can develop libraries for specific needs. In this paper, we use a library Vehicle Dynamics to model automobile steering.

In this paper, Section 2 will introduce the modelica language‘s features and standard library. Section 3 will outline the Vehicle Dynamics library. Section 4 will give an example with library Vehicle Dynamics library. Finally, Section 5 will conclude the experiences with Vehicle Dynamics Library.

2 Modelica and Standard Library

First introduce with an example. Figure 1 shows a 3 Bit Counter Example. This simple system is composed of three components. The system supports 3 Bit counter.

From Figure 2, variable Enable is connected to Counter with the interfaces and variable Clock is connected to Counter with the interfaces.The standard library is basic for modeling and has been used to implement heavy modeling works with satisfying results.[5]



```
model simpleModel

equation

end simpleModel;
```

Figure 3. Modelica model structure

2.1 Model program structure

Modelica program structure is such like Figure 2. The simplest model is Figure 3. Within the blocks model simpleModel and equation, variables can be declared. Within the block between equation and end simpleModel, equations and connection statements can be put into. In Modelica not only Model can be declared, but Record, Class, Connector, Function. Everything can also be treated as an object.

2.2 Object-Oriented Modelica

Modelica is Object-Oriented. There are several objects in Modelica: Class, Model, Connector, Record, Block, Function. Each type of class can be inherited. Modelica has encapsulation. The models of system components are connected through rigorously defined interfaces or connectors(e.g. Fluid connectors with pressure, flow rate, and enthalpy, or heat connectors with temperatures and heat fluxes). Any two components with compatible connectors can be bound together, regardless of their internal details.[6] In each class, there can be multi-type classed be declared. These can help multi-physics modeling.

2.3 Modelica equation

Modelica is equation-based language. From Figure 2 this can be seen that the connection statements are in equation section. The equations in equation section can help increase reusability of model components, since components adapt to the data flow context in which they are used.[2] Without value assignment statements but the equations, the system allows acausal modeling high level specification and increased correctness. For complex systems, equation-based statements can not only be easily implemented but clearly.[7]

Some work can be more convenient in Modelica, for exmaple, PDE(Partial Different Equations) Models, since Modelica is equation-baed.[8]

3 Vehicle Dynamics library

Modelica can import libraries to model some specific model.[9, 10]Vehicle Dynamics library is designed to model physical state of vehicle motion. This way to extend the Modelica environment is using by many people who wants to model more.[11]

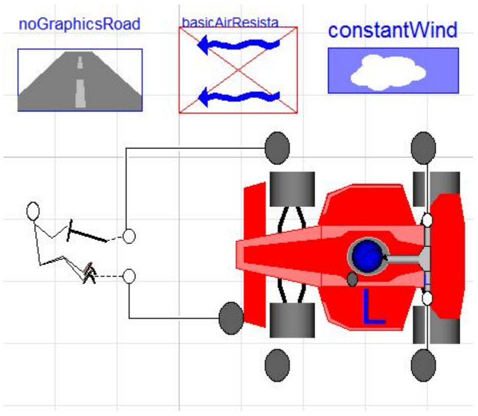


Figure 4. Vehicle Dynamics model

Systems for control of Vehicle Dynamics went to series production for the first time in 1978 with the limitation of brake pressure to avoid locking the wheels to ensure cornering under all braking conditions.[12]

These models are directly modeled with Modelica and its tool. However, models from other environments can be transformed into Modelica, which is a attractive for working direction.[13, 14]

Vehicle Dynamics library is to model vehicle chassis. This library has seven main components: Chassis component, Wheels component, Drivers component, Utilities component, Environments component, PowerTrain component, Aerodynamics component.[15]

Figure 4 shows the basic library structure. The noGraphicsRoad is the model of driving road of this car, basicAirResista is air resistance model and constantWind model the wind. The chassis is the main body. A driver controls the chassis. The chassis have connectors to connect the drivers and wheels.

There are four levels in a model. They are vehicle level, chassis level, suspension level and component level. The highest level is the vehicle level and can be seen in Figure 4.

Within chassis level, a complete chassis is built up using suspensions, wheels and a main automobile body. Within suspension level, components are used to build up new suspensions. Within component level the foundation for efficient reuse of vehicle models is laid. Components like a-arms, trailing arms, multi-links, anti roll linkages, rack steerings etc. are available.

4 Case Study

The vehicle dynamics library is to model vehicle. In Figure 5,

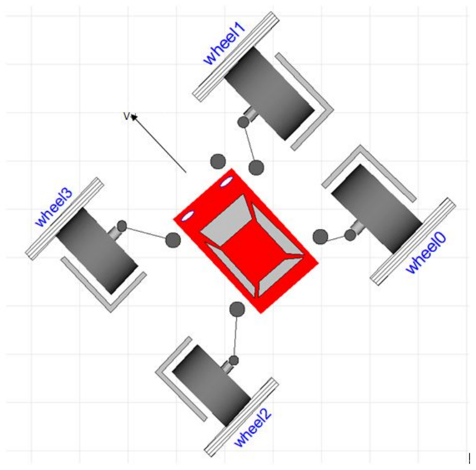


Figure 5. Vehicle

the Car has four wheels. There are two three-dimensional coordinate systems. The coordinate system (Xa, Ya, Ra)(Figure 7) marks the coordinate position where the car is running. Another coordinate system (Xb, Yb, Rb)(FIgure 8) marks the car‘s inner components‘ coordinate position.

The car is composed by one chassis and four wheels. Wheels connect the chassis with connectors. In (Xa, Ya, Ra), the three axises Xa, Ya, Ra have same origin of coordinates Oa. V represents car‘s moving direction and speed.

Figure 5 is the vehicle level. The coding station of vehicle level is in Figure 6. This figure makes it clear that this vehicle model is composed of five main components. These components are connected by the connectors.

```
model Vehicle

   extends VehicleDynamics;

   VehicleDynamics.Chassis.StandardChassis standardChassis;
   VehicleDynamics.Wheels.RillTyre.Wheel wheel0;
   VehicleDynamics.Wheels.RillTyre.Wheel wheel1;
   VehicleDynamics.Wheels.RillTyre.Wheel wheel2;
   VehicleDynamics.Wheels.RillTyre.Wheel wheel3;
 equation

   connect(wheel3.driveShaft1D, standardChassis.flange_2);
   connect(wheel2.driveShaft1D, standardChassis.flange_4);
   connect(wheel0.driveShaft1D, standardChassis.flange_3);
   connect(wheel1.driveShaft1D, standardChassis.flange_1);
 end Vehicle;
```

Figure 6. Vehicle code

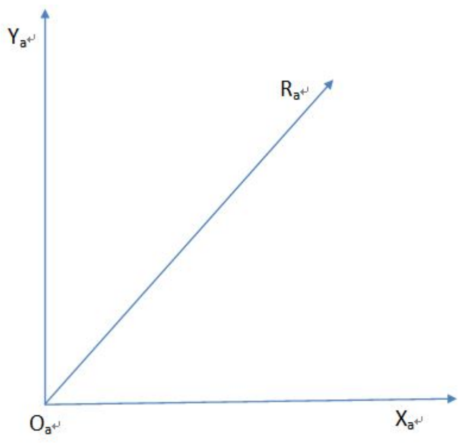


Figure 7. Universal Coordinate System

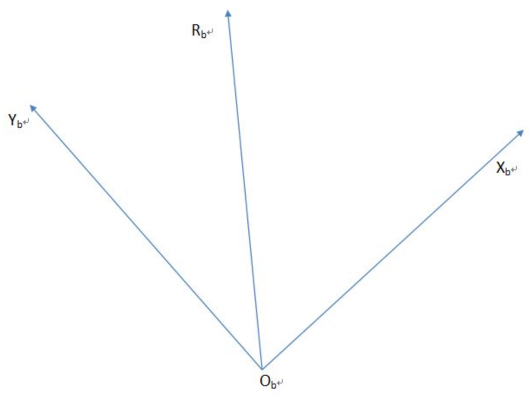


Figure 8. Automobile Coordinate System

Figure 9 is the chassis level. The car's structure is showed. Main components are four wheels, two suspensions and one body. Wheels are connected with the suspensions. The two suspensions are connected with each other. Figure 10 is the code of this level. Components are declared to add onto this vehicle.

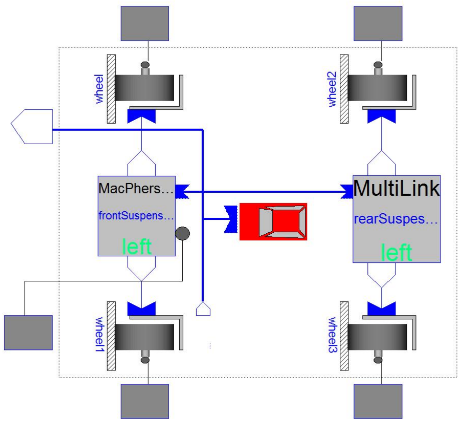


Figure 9. Chassis Level

```
model Chassis

   VehicleDynamics.Chassis.Components.SedanBody sedanBody;
   VehicleDynamics.Wheels.RillTyre.Wheel wheel;
   VehicleDynamics.Wheels.RillTyre.Wheel wheel1;
   VehicleDynamics.Wheels.RillTyre.Wheel wheel2;
   VehicleDynamics.Wheels.RillTyre.Wheel wheel3;
   VehicleDynamics.Chassis.Suspensions.MacPhersonSuspension5 frontSuspension;
   VehicleDynamics.Chassis.Suspensions.MultiLink4Suspension3 rearSuspesion;
 end Chassis;
```

Figure 10. Chassis Level code

Figure 11 is the Suspension level. This level shows the suspension‘s structure. In this picture, the car is subdivided once more. We can see how the suspension is composed in Figure 12, the code picture of Suspension level

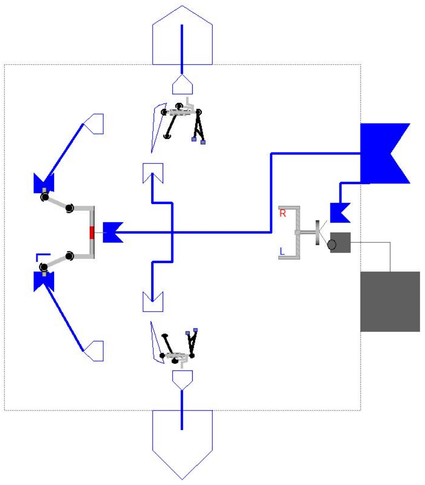


Figure 11. Suspension Level

```
model Suspension

  VehicleDynamics.Chassis.Components.BushingAntiRollBar bushingAntiRollBar;
  VehicleDynamics.Chassis.Components.MacPherson4 macPherson4_1;
  VehicleDynamics.Chassis.Components.MacPherson4 macPherson4_2;
  VehicleDynamics.Chassis.Components.RackSteering4 rackSteering4_1;
end Suspension;
```

Figure 12. Suspension Level

Figure 13 is the Component level. This figure can show the the component BushingAntiRollBar of Figure 11. Figure 14 demonstrates the Component Level of BushingAntiRollBar. Comparing with Figure 12, it can be seen that BushingAntiRollBar is the part of Suspension.

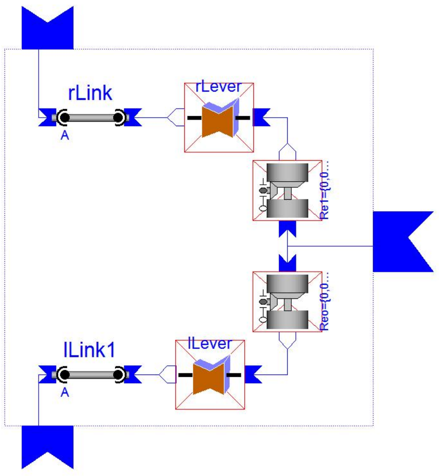


Figure 13. Component Level

```
model Component

  VehicleDynamics.Chassis.Components.Link rLink;
  VehicleDynamics.Utilities.Parts.Body rLever;
  VehicleDynamics.Utilities.Joints.DependentRevolute Re1;
  VehicleDynamics.Utilities.Joints.DependentRevolute Reo;
  VehicleDynamics.Utilities.Parts.Body lLever;
  VehicleDynamics.Chassis.Components.Link lLink1;
end Component;
```

Figure 14. Component Level

The design patterns can be put into use. For Real-Time applications, reusable design patterns can be used.[16] For some strict Real-Time system[17], Modelica can model this system by more variables and equations. For real-time system, it is strict on the modeling language.[18] Modelica can be also used in this system. Another language UML can model some dynamic scenes.[19] But

Modelica can be more suitable than UML for its equation based style.

5 Conclusion

This paper shows the basis of Modelica. Modelica is an object-oriented and non-causal equation language. The property of object-orienting increases the reusability of a system. Non-causal equations can be can reduce system complexity, since this can regardless flow of data comparing with value assignment method. Vehicle Dynamics library is designed to help model dynamic vehicle scenes.

Since Modelica can be useful for modeling physical world. In the future work, We can try to extend the standard library and develop more libraries to meet the needs for modeling more about the physical world. Meanwhile, another branch is modeling transformation. Since UML and other modeling language is easily to build up a model but difficultly to verify this model. We can transform these UML models to Modelica models.

Acknowledgment

This work is supported by Shanghai Knowledge Service Platform Project (No. ZF1213), national high technology research and development program of China (No.2011AA010101), national basic research program of China (No.2011CB302904), the national science foundation of China under grant (No.61173046, No.61021004, No.61061130541, No.91118008), doctoral program foundation of institutions of higher education of China (No. 20120076130003), national science foundation of Guangdong province under grant (No.S2011010004905).

References

[1] Hilding Elmqvist, Sven Erik Mattsson and Martin Otter. Modelica- A Language for Physical System Modeling, Visualization and Interaction. Compter Aided Control System Design, (1999), August 22-27; Kohala Coast, HI

[2] Peter Frizson and Senior Member. Modelica-A Cyber-Pysical Modeling Language and the OpenModelica Envirionment. Wireless Communications and Mobile Computing Conference (IWCMC), (2011), July 4-8;Istanbul

[3] Hilding Elmqvist and Sven Erik Mattsson. An Introduction To The Physical Modelica Language Modelica. European Simulation Symposium, (1997) October 19-22; Passau, Germany

[4] Tobolar J, Otter M, Bunte T. Modelling of Vehicle Powertrains with the Modelica PowerTrain Library. Systemanalyse in der Kfz-Antriebstechnik IV. 79: 204-216(2007);Augsburg

[5] Marwan Chamoun, Romuald Rulliere, Philippe Haberschill and Jean Francois Berail. Dynamic model of an industrial heat pump using water as refrigerant. International Journal of Refrigeration. 35, 4: 1080-1091(2012)

[6] Jiasheng Guo, Chaokui Qin and Gerhard Schmitz. Object-oriented modeling and simulation of a natural gas-fuelled spark ignition engine with Modelica. Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference, (2010) July 9-11; Chengdu, China

[7] Stefan Heller and Tilman Bünte. MODELICA vehicle dynamics library: Implementation of driving maneuvers and a controller for active car steering. 3rd International Modelica Conference, (2003) November 3-4; Linköping, Sweden

[8] Zhihua Li, Ling Zheng and Huili Zhang. Solving PDE Models in Modelica. Information Science and Engineering, (2008) December 20-22. Hangzhou, China

[9] Michael Tiller, Paul Bowles and Mike Dempsey. Development of a Vehicle Model Architecture in Modelica. 3rd International Modelica Conference, (2003) November 3-4; Linköping, Sweden

[10] Martin Otter, Hilding Elmqvist and Sven Erik Mattsson. The New Modelica MultiBody Library. 3rd International Modelica Conference, (2003) November 3-4; Linköping, Sweden

[11] Wuzhu Chen, Michaela Huhn and Peter Fritzson. A Generic FMU Interface for Modelica.

[12] Christian Knobel, Gabriel Janin and Andrew Woodruff. Development and Verification of a Series Car Modelica/Dymola Multi-Body Model to Investigate Vehicle Dynamics Systems. 5th International Modelica Conference, (2006) September 4-5; Vienna, Austria

[13] Vadim Engelson, Hakan Larsson and Peter Fritzson. A Design, Simulation and Visualization Environment for Object-Oriented Mechanical and Multi-Domain Models in Modelica. Information Visualization, (1999) July 14-16; London, Britain

[14] Carla Martin-Villalba, Alfonso Urquia and Sebastian Dormido. Visualization and interactive simulation of Modelica models for control education. Control and Decision Conference, (2009) June 17-19; Guilin, China

[15] Johan Andreasson. Vehicle Dynamics library. 3rd International Modelica, (2003) November 3-4; Linköping, Sweden

[16] Saoussen Rekhis, Nadia Bouassida and Rafik Bouaziz. Modeling Real-Time applications with Reusable Design Patterns. International Journal of Advanced Science and Technology (IJAST). 9 (2010)

[17] Labib Francis Gergis. Space-Time Hyper Phase Shift Keying Over Fading Channels. International Journal of Advanced Science and Technology (IJAST). 8, 33 (2011)

[18] Xinxiu Wen and Huiqun Yu. Real-Time Systems Modeling and Verification with Aspect-Oriented Timed Statecharts. International Journal of Hybrid Information Technology (IJHIT). 5, 2 (2012)

[19] Dr. Vipin Saxena and Manish Shrivastava. UML Modeling and Performance Evaluation of Multithreaded Programs on Dual Core Processor. International Journal of Hybrid Information Technology (IJHIT). 2, 3 (2009)

Authors

Shuguang Feng
Shuguang Feng was born in Hebei, China. Now he has been studying in Software Engineering Institute, East China Normal University In Shanghai, since 2010.