APPENDIX

This tool uses the idea of MDE to build a collaborative platform based on our meta-model for a graph drawing tool. In the modeling process, we first analyze the problem diagram, state machine diagram and block diagram to derive the elements contained in these graphs, and then design the meta-model according to these elements. In the meta-model, our main frames is the problem diagram, which mainly contains Domain, Requirement, Machine, Phenomenon, and so on. In the meta-model, Phenomenon maps to the connection from machine to domain, domain to domain and requirement to domain.The attribute type in Domain is used to determine the type of domain, and there are three types of domain, Biddable,causal and lexical.Detailed design of the PF meta-model is shown in Figure 1.

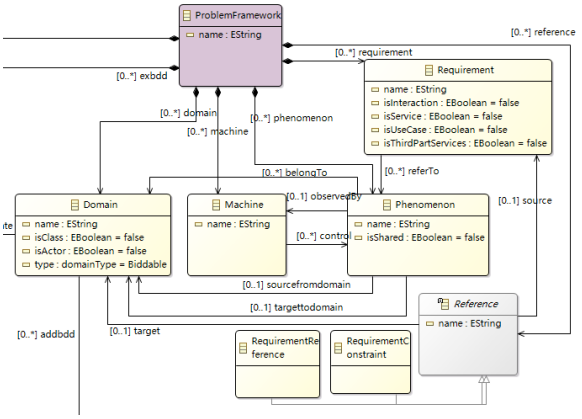

Figure 1. The meta-model of Problem Frames

Since Domain is the environment part of a real-world computer system, we propose to embed a state machine diagram and a block diagram into Domain. The state machine diagram contains state, region, pseudostate, transition, etc. The attributes isComposite, isSimple, isOrthogonal, etc. States determine the type of state, and pseudostate has different representations depending on the kind, such as Initial, deepHistory, etc. The state machine diagram meta-model is shown in Figure 2.

Block diagram is one of the most common diagrams in the system modeling process. BDD is a kind of structure diagram, which mainly describes the structural composition of the system and the relationship between the constituent elements. In a block diagram, a block contains two subclasses operation and properties. Generalization, and SharedAssociation are different types of associations between blocks. The block diagram meta-model is shown in Figure 3.

In our meta-model, statemachine and block are subclasses of problem frames, so they can be associated from Domain to other diagrams. Statemachine.ecore and block.ecore are as Add-in diagram to problemframework.ecore, and then add the association addstate, addbdd, from Domain to statemachine and block respectively, so that these three parts are independent of each other and can be related. The associations are shown in Figure 4.

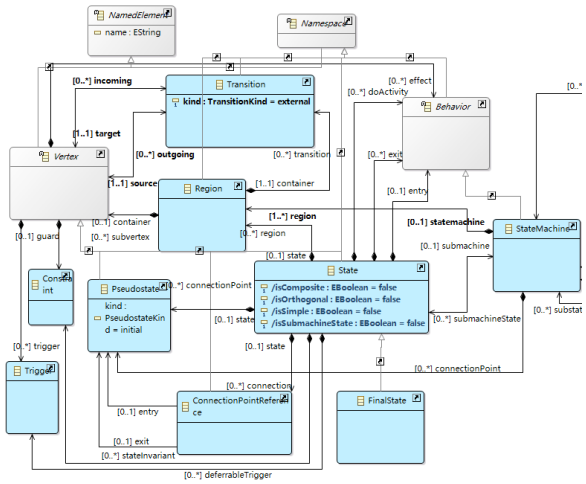

Figure 2. The meta-model of StateMachine

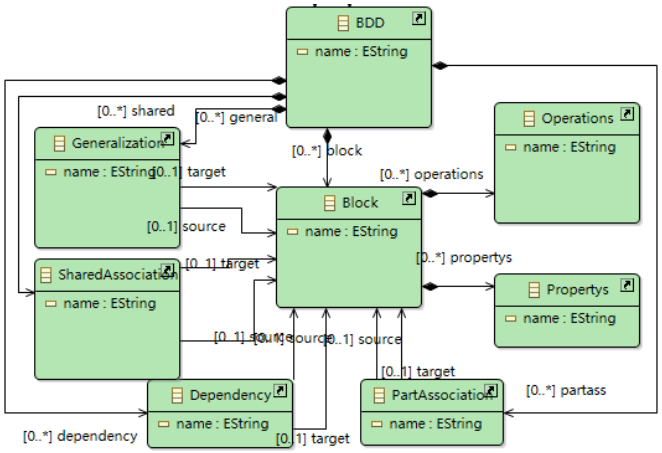

Figure 3. The meta-model of Block

After the modeling is finished, the .genmodel file is generated based on the meta-model file. Then start the project as an Eclipse Application. The final step is to run the generated model in the Runtime workspace and build the graph editor with Sirius according to the problem diagram mapping standard. In our tool, the combination of dynamic StateMachine diagrams and static block diagrams allows those with domain expertise to analyze the system in greater depth. For example, in the case of the HybridSUV, which is currently a very popular vehicle on the market, we can analyze itself and its environment and simplify it to produce a problem diagram as shown in Figure 5.

We can find that the problem diagram is a macroscopic description of the system, which has the advantage of facilitating the domain involved to have an abstract concept of the system, but when analyzing and studying some systems, we need not only a macroscopic description, but also some microscopic analysis of the system. In this HSUV example, we see in

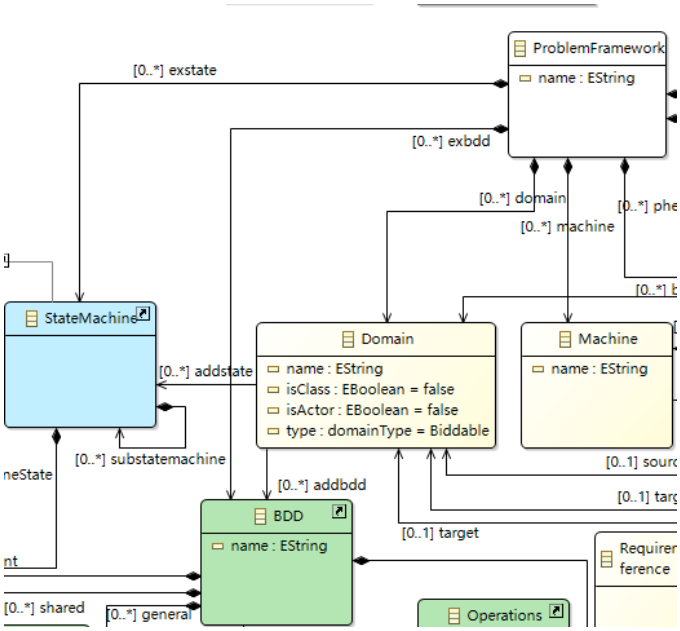

Figure 4. Association between metamodels

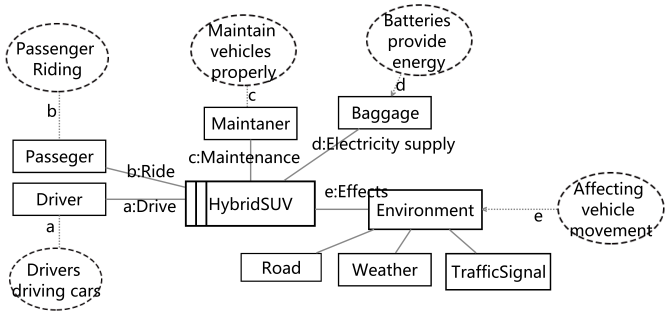

Figure 5. Problem diagram of HSUV

the problem diagram that the whole system consists of the battery, the driver, and the environment. But we know from life experience that the battery module in a hybrid vehicle is very complex and contains many physical devices. Therefore, we can double click on the domain to create its state machine diagrams and block diagrams. Figure 6 shows the dialogue box which allows the modeller to choose to draw the block diagram or the state machine diagram.

The first thing is to open BlockDiagram, if it is the first time to open this diagram, there will be a prompt to create one. Once created, it will automatically jump to the block diagram editing interface. In this interface we can draw the internal components of Baggage. As shown in Figure 7, there are also properties and actions that can be created in each block in the block diagram. The diagram of the powersystem contains elements such as BatteryPack, PowerControlUnit, etc. The diagram describes the composition and structure of the system.

UML state machine diagrams help to represent the life cycle of a single object in a clear and intuitive way. Similarly on the

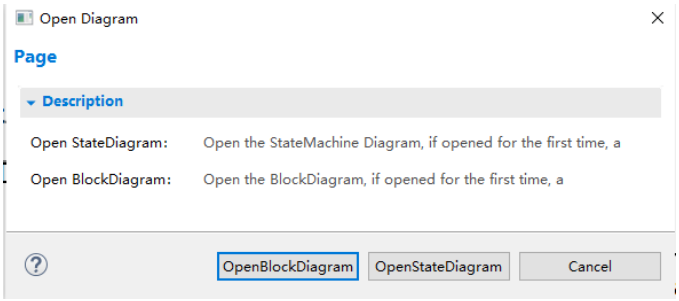

Figure 6. Open Diagram

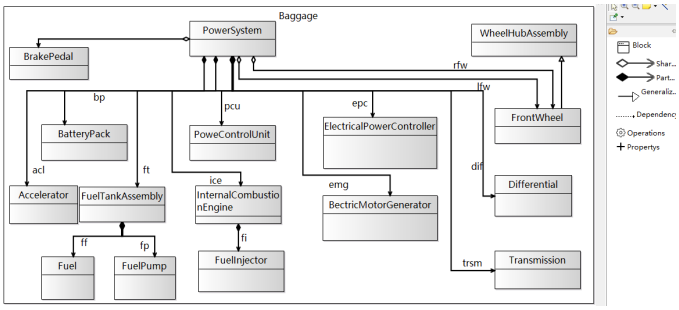

Figure 7. Block diagram of HSUV

tool we can also create a state machine diagram. As shown in Figure 6, the modeller can select Open StateDiagram and in the state machine diagram we can describe the various states of Baggage at work, as shown in Figure 8. Our tool not only provides complete state machine diagram components, but also the ability to draw sub-state machines and nesting of state machines. For example, in Figure 8, the BatteryPack also contains its own sub-states.

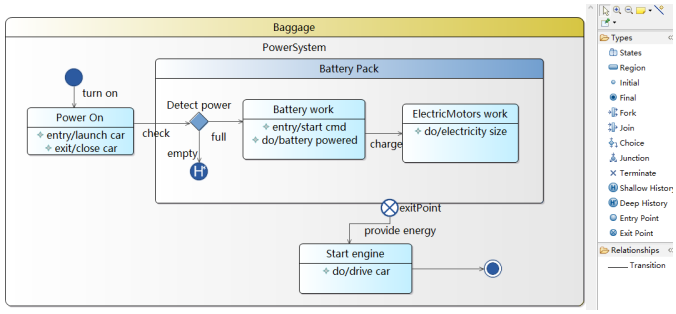

Figure 8. StateMachine diagram of HSUV