DEVS Modeling and Simulation Methodology with MS4 Me Software Tool

Chungman Seo, Bernard P. Zeigler, Robert Coop, and Doohwan Kim

RTSync Corp.
530 Bartow Drive Suite A
Sierra Vista, AZ, 85635, USA
cseo, zeigler, robert.coop, dhkim@rtsync.com

Keywords: Sequence Diagram, State Diagram, DEVS Natural Language, MS4 Me Software Tool

Abstract

There are many implementations of DEVS Modeling and Simulation in various computer languages and software tools. Most of them focus on modeler-friendly approaches which mean a user should have knowledge of modeling and computer languages. In this paper, we introduce high and low level design methodology to help domain experts (who might not have an in-depth understanding of DEVS modeling theory) solve their domain problems with DEVS modeling and simulation software called MS4 Modeling Environment (MS4 M)e. The tool enables high level model design through a sequence diagram from which template DEVS models are automatically created. The sequence design is converted to a System Entity Structure (SES) document representing a coupled model. The template DEVS models are written to constrained natural language called DEVS Natural Language (DNL) to express DEVS atomic modesl which MS4 Me displays to a state diagram as a low level model design. The state diagram contains detailed information from the domain experts who provide logic, variables, and message types for each atomic model. More in-depth technical implementation details are provided by DEVS modelers working together with domain experts. In this manner, MS4 Me provides a collaborative DEVS modeling and simulation environment for domain experts and DEVS modelers.

1. INTRODUCTION

It is hard for domain experts to create a simulation model in their domain of interest with only a technical guide or manual. Although such users have knowledge of specific domains, they need to understand how to make models with modeling and simulation theory. Most simulation tools help users easily implement their domain models by hiding complex modeling theory and knowledge of computer language implementing the modeling theory. They provide graphic interfaces to gather information required to generate models. In this case, the simulation tools stick to specific domain areas such as OPNET [1], and Arena [2].

One of the broadly applicable modeling and simulation theories is DEVS (Discrete Event System Specification) modeling and simulation [3]. There are many implementations of DEVS modeling and simulation in various computer languages and software tools. CD++ [4], PowerDEVS [5] and ADEVS [6] are implemented using C++. DEVS-Suite [7] is implemented using Java. Most of them focus on modeler-friendly approaches which mean a user should have knowledge of both modeling theory and the underlying computer languages to create models. To reduce time to create models, a DEVS software tool needs to provide a collaboration environment to inject domain knowledge into generic frameworks such as UML [8].

The aim of the MS4 Modeling Environment software tool is to help users develop DEVS models and simulate them. It provides a sequence diagram to allow users to express the overall system structure and each component's behavior without having to have knowledge of DEVS theory or a computer language. Moreover, the sequence diagram generates template DEVS models described in restricted natural language from the system design. The template DEVS model is expressed in DEVS Natural Language (DNL) [9] which enables sharing information between domain experts and modelers. MS4 Me generates Java language models from the DNL files automatically. To implement more realistic models, a DNL document can include tag blocks for handling logic codes, and specification of variables, message types, and input/output ports. The DNL document is represented to a state diagram which manipulates states and message transitions to alter model's behaviors. The sequence diagram depicts a coupled model in a restricted natural language for expressing System Entity Structures (SES) [10]. An SES file is converted to a coupled model through a pruning and transformation process.

One of the powerful capabilities of the MS4 Me tool is the ability to couple multiple models into a larger and more complete system. The SES language is used to describe how a system is decomposed into subsystems when viewed from a certain perspective, different specializations of a system that might occur, messages sent from one system to another, and variables that a system might have [9].

In this paper, we introduce a model development methodology in three stages to create domain models using the collaboration environment supported by MS4 Me. Through collaborative work between domain experts and modelers, the time for domain modeling can be reduced and efficiency for modeling can be increased.

In the rest of the paper, section 2 addresses features of MS4 Me software tool and shows a sequence diagram to explain how to use MS4 Me. Section 3 addresses modeling and simulation methods with MS4 Me. A military example using the model development methodology in three stages explained in section 3 is shown in section 4. The paper's summary and future work are in the section 5.

2. MS4 MODELING ENVIRONMENT SOFTWARE

MS4 Modeling Environment (MS4 Me) is a software tool for DEVS Modeling and Simulation based on Java computer language and Eclipse RCP environment [11] developed by MS4Systems (ms4systems.com). It provides a top down modeling methodology with a sequence diagram which represents the overall system structure as well as bottom up modeling method with a state diagram which constructs behaviors of an atomic model. The sequence diagram generates System Entity Structure (SES) documents expressed in restricted natural language [9], and the state diagram is expressed in a DEVS Natural Language (DNL) document and represents a DEVS atomic model with restricted natural language [10]. MS4 Me automatically generates atomic models implemented in the Java language from the DNL files, and a coupled model through running a PES file generated by a pruning process for the SES document. For advanced users who are familiar with restricted natural languages for SES and DNL documents, MS4 Me provides editing capability with highlighted keywords used in SES documents and DNL documents.

2.1. MS4 Me Launch Page

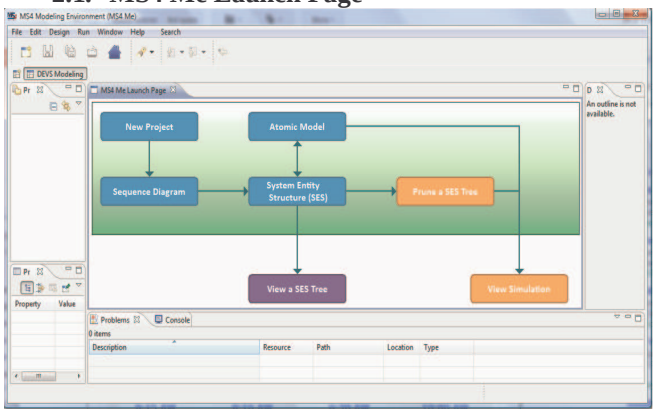

Figure 1. Launch Page of MS4 Me

Figure 1 shows the MS4 Me Launch Page which outlines the work flow of how to use MS4 Me, and enables users to reach ses files, pes files, and dnl files in project folders directly. The Launch Page starts with a New Project button which guides to create a project for DEVS modeling and Simulation. Also, a project is created by a New Project menu in a File menu. The project has a src folder containing seven sub folders which are Models.animation, Models.dnl, Models.java, Models.pes, Models.ses, Models.txt, and Models.xml. After creating a project, users can generate any domain models with the sequence diagram, the state diagram, ses editor, and dnl editor. As an easy way to generate a system, MS4 Me provides a sequence diagram which describes DEVS model entities and message relations among them, and generates a SES document for the system when saving the sequence design. After finishing constructing the system, the sequence diagram generates dnl files from the SES document. In the launch page, the Sequence Diagram icon links to System Entity Structure (SES), and the System Entity Structure (SES) icon links to Atomic Model represented by dnl files. The SES document is viewed as a SES Tree, and the Prune SES provides a pruning interface that generates Pruned Entity Structure (PES) documents. To simulate the designed system, users can use a pes file or a Java file to display the system in the simulation viewer using View Simulation button in the launch page.

2.2. Sequence Diagram and SES Document

The sequence diagram generates a simple scenario of a system with entities and message arrows. An Entity in the sequence diagram GUI represents an atomic or coupled model and has properties describing variables, specialization, decomposition, and multiple aspects for SES document [10]. An entity with a decomposition property is shown as a coupled model. This paper focuses on DEVS modeling not SES. For more information about SES handling in MS4 Me, refer to [9]. With a decomposition property, the sequence diagram can express the hierarchical structures of DEVS models.

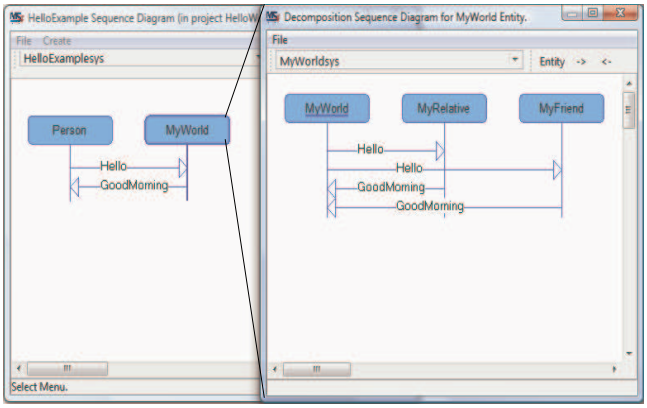

Figure 2. Sequence Diagram for HelloExample

Figure 2 shows the HelloExample sequence diagram where MyWorld entity with thicker line is a coupled model having two atomic models. A top model will be viewed with Person and MyWorld models. The Person model sends a Hello message to MyWorld model and the MyWorld model sends a GoodMorning message to the Person model. In the sequence diagram for a MyWorld, there is a MyWorld entity with underscore to show ports for the coupled MyWorld model. The ports connect messages from outside models to the models in the MyWorld coupled model.

```
From the HelloExamplesys perspective, HelloExample is made of  Person and MyWorld!
From the HelloExamplesys perspective, Person sends Hello to MyWorld!
From the HelloExamplesys perspective, MyWorld sends GoodMorning to Person!
From the MyWorldsys perspective, MyWorld is made of  MyRelative and MyFriend!
From the MyWorldsys perspective, MyWorld sends Hello to MyRelative!
From the MyWorldsys perspective, MyWorld sends Hello to MyFriend!
From the MyWorldsys perspective, MyRelative sends GoodMorning to MyWorld!
From the MyWorldsys perspective, MyFriend sends GoodMorning to MyWorld!
```

Figure 3. SES Document of HelloExample

Figure 3 shows a SES Document from a HelloExample sequence diagram. MS4 Me editor highlights reserved keywords which are in red for the SES document. Keyword highlighting and syntactic error checking make it easy for advanced users to generate SES documents. For more detailed information on SES natural language keywords, refer to [9]. Words in blue are entities generated to SES or DNL files. This feature allows users to easily navigate to files related to entities.

The system in the sequence diagram expresses conceptual design which means there are no concrete implementations for logics, message types, and variables used in each model. To implement models with such detailed information, appropriate atomic models must be created. The sequence diagram automatically creates atomic models for all entities from a ses file except entities with decomposition

2.3. State Diagram and DNL Document

DNL document represents behaviors, input ports, and output ports of an atomic model with restricted natural language. In case of the atomic models generated from a sequence diagram, the behaviors of the atomic models are determined by interactions of messages between entities. For example, if Person entity sends Hello message to MyWorld entity, the Person model starts with internal transition after passing certain time and generates an output event (message) to Hello port. After sending the message, the Person model waits for an external event from GoodMorning port. After receiving a message from the port, the Person model waits forever. This is a basic rule to generate DNL documents from the SES document.

Figure 4 shows a DNL document for the Person entity in the above sequence diagram. The DNL editor has syntactic checking function through highlighting reserved keywords for the DNL document. The DNL document in figure 4 contains definitions of input and output ports, an internal event, an output message, and an external event. The DNL file is automatically converted to a Java file implemented for an atomic model in MS4 Me environment. The Java file is more complicated than the DNL file. So, users only construct the DNL files and MS4 Me takes care of the others.

```
accepts input on GoodMorning !
generates output on Hello !

to start, hold in sendHello for time 1!
after sendHello output Hello!
from sendHello go to waitforGoodMorning!
passivate in waitforGoodMorning !
when in waitforGoodMorning and receive GoodMorning go to passive!
passivate in passive!
```

Figure 4. DNL Document for Person Entity

The advanced users code a DNL document in the DNL editor directly. To make it easy to construct a DNL document, MS4 Me provides a state diagram seen in figure 5. The double rectangles signify an initial state, here sendHello, with 1 unit time advance value, and the arrow means a state transition with either internal or external event. The single rectangle indicates a state containing a state name and time advance. There are two characters (!,?) under the transition. The exclamation mark (!) means that an internal event occurs and an output port generates a message. The question mark (?) means that a message comes through an input port and an external event is triggered. Figures 4 and 5 represent the same atomic model.

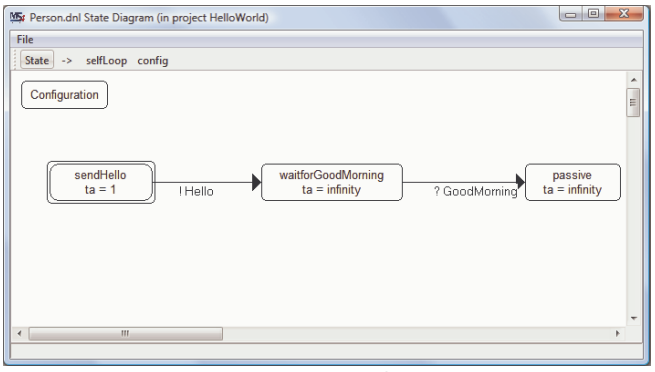

Figure 5. State Diagram for Person.dnl

In the state diagram there is a rectangle labeling configuration in to provide additional information not obtained from the sequence diagram. The configuration can contain definition of variables, and additional information required to generate Java codes such as initializing variables, adding libraries, and adding Java codes in the model. The DNL document can contain actual piece of Java code

between <% and %> called tag blocks. The tag blocks enable the DNL document to cover all aspects of Java atomic models. The model implemented in Java language can import any existing libraries and add any methods which are out of scope of the DNL document made with the restricted natural language. The internal and external events, and output message need to be implemented with actual Java codes. The DNL document contains the Java codes in the tag blocks for internal event, external event, and output message. For more details for restricted DEVS Natural Language, refer to [9].

2.4. Simulation Viewer

The simulation viewer shows a coupled model and its sub models. To open the simulation viewer, a PES file should be created using Prune SES in the PES menu that appears after opening the SES file. The simulation viewer opens when clicking Run PES file in SimViewer menu on a PES editor. After pruning, a Java coupled model is created in Models.java folder. The simulation can be controlled using Restart, Pause, Run, Step, and Run to buttons in the bottom of the simulation viewer. Once the Java coupled model is created, users can run the coupled model to open a simulation viewer out of MS4 Me.

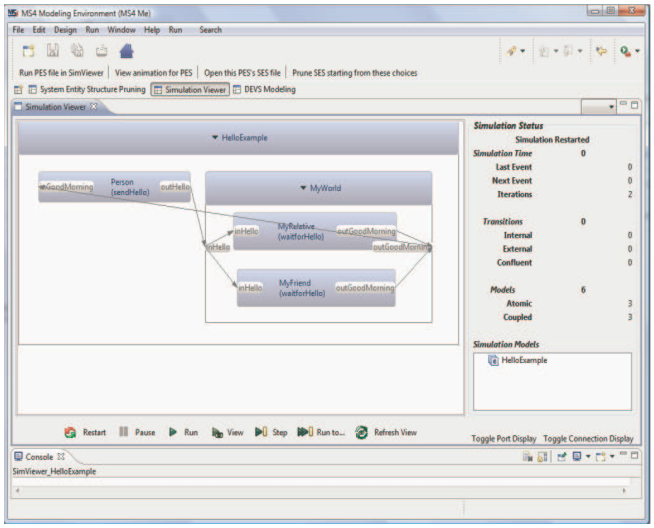

Figure 6. Simulation Viewer for HelloExample

Figure 6 shows the whole coupled model of HelloExample consisting of one atomic model (Person) and one coupled model (MyWorld). Each model shows its own ports and connections between the ports. Simulation status shows in right hand side of the simulation viewer. Step button makes the model be simulated for one cycle. “Run to” button opens a GUI to set simulation time or cycles and runs the model until reaching the simulation time or cycles. The Run button simulates the model until finishing the simulation. The Pause button is used to stop simulation. The Restart button resets the model to a beginning state.

2.5. Other capabilities

MS4 Me supports dynamic structure modeling which creates and delete models and couplings during simulation time. This capability can reduce simulation time and construct more realistic models. For more detail information of how to create dynamic structure models, refer to MS4 Me user guide [12].

MS4 Me provides graphing capability which enables plotting of the values of variables during runs of the model. A ‘graph’ command is used in a DNL file to create a graph instance with a label of a variable, a title of a graph, and a variable which comes during simulation. For examples of the graphing capability, refer to MS4 Me user guide [12].

3. MODELING AND SIMULATION METHODOLOGY WITH MS4 ME

Many modeling software tools are implemented to create simulation models and execute them. Beginners who want to learn a simulation tool take some time to understand how to create a model and how to use the tool, and require computer programing knowledge for implementing detail logics in their models. Those tools usually do not provide two level designs such as high level design and low level design. High level design is considered as an abstract level design or conceptual design with which domian experts create systems with their point of view. Low level design is an implementation of specific modeling and simulation based on the high level design. Modeling experts produce models requiring modeling skills learned from many years.

MS4 Me provides two level designs for domain experts and modelers seen in figure 7. The domain experts’ design starts with making scenarioes for systems which they want to simulate. The scenarioes could be abstracted to simple scenarioes expressed in the sequence diagram. With the sequence diagram, the domain experts generate template models from the SES document and validate system behaviors through a simulation viewer. For detailed implementation of each atomic model, they may define message types for each event and variables used in the models, and provide event handling logic. Domain experts should know basic DEVS atomic model behaviors. For example, if a model gets a message on an input port, an external event handler can be given logic to process the message. The external event could affect model’s variables or output messages. In the case of an internal event, the domain expert expresses how output messages are generated with information which comes from the model’s variables or ramdomly generated values. Also, an internal event handler can be given logic to process the state transition in a

manner similar to that of the external event handler. To refine the model, the modeler translates the logic into Java code that is stored in a tag block. If there are no errors in the Java folder, the domain expert can run the simple scenario which contains all required logic in the atomic models, and validate the simple scenario, checking instances of input/output message and model variables. The first development stage is basic implementation of the scenario with a sequence diagram and a state diagram. In the first stage, a modeler can help a domain expert convert logics into Java codes.

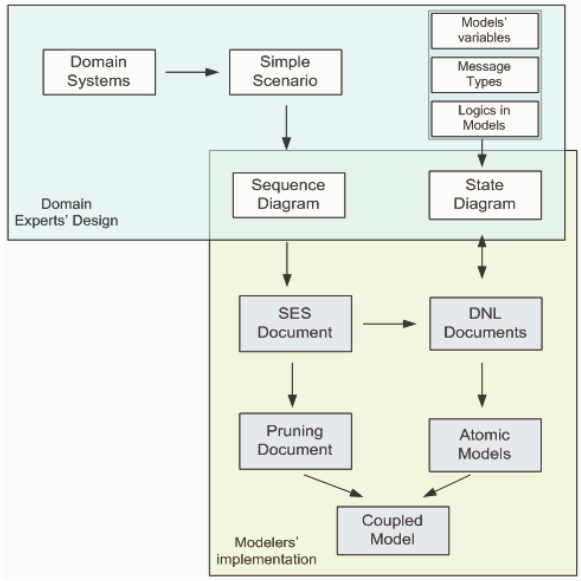

Figure 7. Two level designs for collaboration environment in MS4 Me

The sequence diagram shows one cycle of behaviors of the system which the domain expert designs. The atomic models' behavoirs are described out of the sequence diagram. For example, a model receives an input message containing a value and compares the received value and a variable to decide a next state. The next state could be the initial state not a different next state decided in the sequence diagram. Given that the sequence diagram does not express this situation, a modeler can alter the behavior of an atomic using its state diagram. The modeler can add/delete a state symbol (rectangle shape) and transition symbol (arrow) in the state diagram to change model's behaviors. Otherwise, he/she can put code in the tag blocks to alter the model's behaviors. When implementing a change in an atomic model's behaviors the modeler could modify the sequence diagram to more easily implement the change. This is the second development stage which implements detailed behaviors of atomic models. The domain expert designs complex behaviors of atomic models and the modeler implements atomic models with the complicated behaviors.

Through the second stage, the detailed scenario model is created. The next stage is to test the model with an experiment frame model which consists of a generator and a transducer models. The generator model generates system conditions and decides how many the system model is executed (simulated). The transducer model collects messages from the system model and displays statistics in a chart. A domain expert and a modeler decide which values are generated in the generator model and which messages are observed in the transducer model. After that, the squence diagram is modified to add an experiment frame model to the system. The generator model sends a configuration message to the system and the system sends a observing message to the transducer model. The new sequence diagram generates new SES document. The modeler can generate a full coupled model which consists of the system model and the experiment frame model based on the previous implementation from the second stage.

The domain expert simulates the final model coming from the third stage and can see the simulation results on the various charts.

4. MILITARY APPLICATION EXAMPLE

We will illustrate a battle model implemented by our collaboration environment using MS4 Me. The example is a battle between two combat units with different weapon systems. Simple behaviors of the example are the following.

There are NT and ST combat units in a simple scenario. NT consists of NCommand, NCo1, and NCo2. ST is made of SCommand, SCo1, and SCo2. NCommand sends an attack message to NCo1 which attacks SCo2. SCo2 reports an attacked status to SCommand. SCommand initiates an attack order to SCo1 and SCo2. SCo1 fires its weapons to an assigned target. SCommand checks total damage from SCo1 and SCo2, and sends attack messages to SCo1 and SCo2. After attacking NCo1 and NCo2 from SCo1 and SCo2, NCo1 and NCo2 fire their weapons to SCo1 and SCo2. The simple scenario ends with two firings in NT and ST.

To define message types, variables, and logics, we assign a hitting rate and a property value to each weapon. ST uses K9, M130, SPIKE, and M150 as weapons and NT uses M240, M120, M160 and M100. The damage status is calculated by number of attacked ammunition. The condition of finishing the simulation is that any weapon's property value is less than 30% of the initial value. The first stage implements a sequence diagram from the simple scenario and inserts required basic information into template models. The second stage implements looping behaviors in the models to get the result of the combat. The third stage implements an overall system with an experimental frame model and the battle model. To start simulation, the generator in the experimental frame sends an

AssignWeapon message to the NT and ST models. After calculating property values for component models in NT and ST, the transducer model stops the simulation if there is a model with less than 30% property value allowing it to continue otherwise. In this example, the generator generates four AssignWeapon messages containing combination of weapon systems for NT and ST models.

		Hitting (%)	Property($)
ST	K9	65	3000000
	M130	50	2000000
	SPIKE	70	2500000
	M150	55	2500000
NT	M240	55	2500000
	M120	60	3000000
	M160	70	2800000
	M100	65	2000000

Table 1. Initial values of each weapon

	ST(SCo1, SCo2)	NT(NCo1, NCo2)
1 round	K9	M240
	M130	M120
2 round	SPIKE	M160
	M150	M100
3 round	K9	M160
	M130	M100
4 round	SPIKE	M240
	M150	M120

Table 2. Assigning weapons to ST and NT

Table 1 shows initial hitting rate and property value for each weapon system. Each model having one of the weapons has a hitting rate and a property value in its variables. The generator generates four messages with information in table 2.

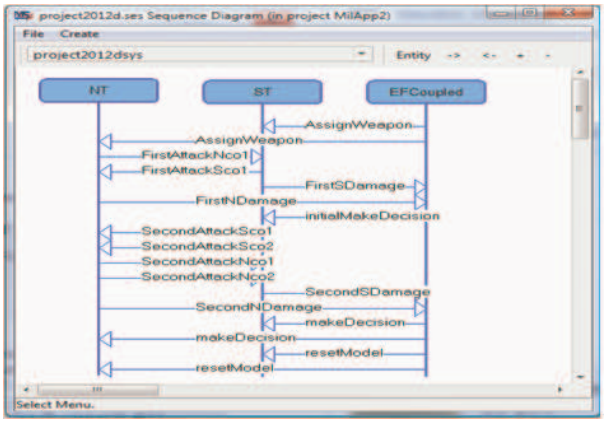

Figure 8. Overall Sequence Diagram for the final stage on the example

Figure 8 displays overall sequence diagram for a battle between two combat units. The EFCoupled entity sends a message containing weapons' information addressed in table 2. The NCo1, NCo2, SCo1, and SCo2 models use assigned weapons when they fight each other. When they are attacked, NT and ST report their damage to the EFCoupled model to check a simulation stop condition. After a winner is decided, the EFCoupled model sends AssignWeapon messages to NT and ST model for another round.

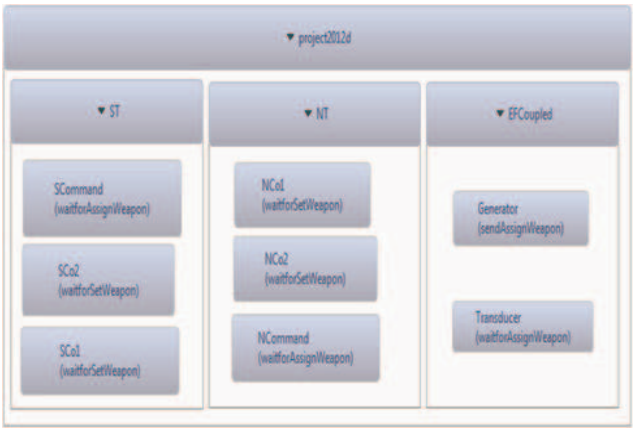

Figure 9. Simulation viewer of the example

Figure 9 shows a simulation viewer for a battle between combat units. ST, NT, and EFCoupled models have their component models. Each component model has a model name and its state.

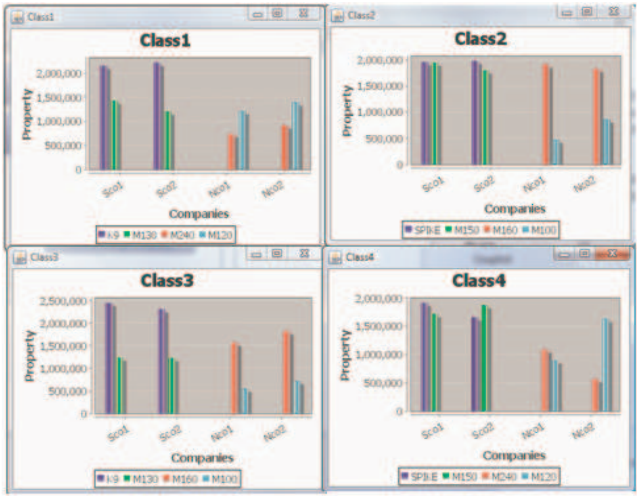

Figure 10. Bar charts displaying results of each round

Figure 10 shows bar charts containing property values of four models in NT and ST. The combat unit with the lowest property value loses the battle. In the result of the simulation, ST model wins for four rounds.

5. CONCLUSIONS

In this paper, we introduced a MS4 Me software tool which helps users easily implement DEVS modeling and simulation in any domain. DEVS modeling and simulation

used in the MS4 Me tool is implemented in Java and based on the Eclipse RCP environment with plug-in extensibility. It provides SES, PES, and DNL editors to provide content assistance and check for syntactic errors in SES, PES, and DNL documents. The editor capability is for advanced users who have learned DEVS modeling and simulation. For novice users, MS4 Me provides sequence and state designers for generating SES and DNL documents from graphic input. The sequence diagram helps domain users easily design their systems with entities and message passing. One of unique features in the environment is automatic generation of template DNL documents from SES documents. Domain experts save time to make atomic models with this MS4 Me feature. The template DNL document contains behaviors, input ports, and output ports derived from the sequence diagram. Detailed information on message types, event handling logics, and variables is required to implement realistic models from template models. The information can be inserted into DNL documents through a DNL editor or a state designer. The logics are described in Java codes placed in tag blocks in the DNL document. A SES document generates through a pruning process a PES document which creates a coupled model and opens a simulation viewer.

We proposed three model development stages to develop a domain model which a domain expert designs and a modeler refines the design. The domain expert and the modeler can share model design and implementation through MS4 Me software. The first stage includes creating a basic system with a simple scenario. The basic system is made of atomic models with event handling logics and variables. The domain user validates the system through a simulation viewer. The second stage elaborates models' behaviors in the simple system using a state designer. This allows expressing non-sequential behavior patterns which are not covered by the sequence designer. After the second stage, the domain system has been completely described. The final stage is to add an experiment frame model to the domain system. The experiment frame model adds simulation control to the domain system and observes simulation outputs during execution. After finishing the simulation, the experiment frame model provides statistical results in charts. We used a military example to apply three model development stages with MS4 Me.

For continued development of its capabilities, work on MS4 Me will seek to remove current limitations and enhance its user assistance features. For example, in the current version there is a limitation on automatic generation of atomic models from SES documents that contain advanced SES constructs. In enhancing its assistance features, the goal is to enable domain experts to create domain systems with minimum intervention of modelers while still hiding the complexities of the underlying DEVS formalism. In this way, MS4 Me recognizes that each type of user is indispensible. MS4 Me strongly differs from efforts to support domain-driven modeling environments for specific applications (e.g. [13,14]) in which the DEVS modeler is in principle eliminated. Rather its goal is to provide a generic, DEVS modeling and simulation environment enabling domain experts and DEVS modelers to truly collaborate, each contributing his/her unique knowledge and skills.

References

[1] www.opnet.com/solutions/network_rd/modeler.html

[2] W. David Kelton, Randall P. Sadowski, Nancy B. Swets, Simulation with Arena, McGraw-Hill, 2010

[3] Zeigler, B.P., Kim, T.G., and Praehofer, H., Theory of Modeling and Simulation, 2nd ed., Academic Press, New York, 2000.

[4] Gabriel A. Wainer, Discrete-Event Modeling and Simulation: A Practitioner's Approach, CRC Press, 2009.

[5] Federico Bergero, Ernesto Kofman, PowerDEVS: atool for hybrid system modeling and real-time simulation, Simulation, vol.87 no. 1-2 113-132 Jan 2011.

[6] James J. Nutaro, Building software for Simulation, wiley, 2011,

[7] http://acims.asu.edu/software/devs-suite

[8]Shaikh, R., H. Vangheluwe, 2011, "Transforming UML2.0 class diagrams and statecharts to atomic DEVS", Symposium on Theory of Modeling & Simulation, 205—212, Boston, MA, USA.

[9] Zeigler, Bernard P., Sarjoughian, Hessam S. Guide To Modeling And Simulation Of Systems Of Systems Series: Simulation Foundations, Methods And Applications Springer Pub. Co., pp. 330, 2013.

[10] Zeigler, B.P and Phillip Hammonds (2007), "Modeling&Simulation-Based Data Engineering: Introducing Pragmatics into Ontologies for Net-Centric Information Exchange", ", Academic Press, Boston,. 448 pages

[11] http://www.eclipse.org/home/categories/rcp.php

[12] MS4 Me user guide (www.ms4system.com)

[13] Ferayorni, A., H. S. Sarjoughian, 2007, "Domain driven modeling for simulation of software architectures", Summer Computer Simulation Conference, 1-8, San Diego, CA, USA.

[14]Mittal, S., Douglass, S.: From domain specific languages to DEVS components: application to cognitive M&S. In: Proceedings of the 2011 Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium, pp. 256–265. Society for Computer Simulation International (2011)