A Conceptual Framework for Modeling and Simulation of Variable Structure Hybrid Systems

Liming Chen, S. Xia, Paul Luker

Department of Computer and Information Sciences
De Montfort University
Hammerwood Gate Kents Hill
Milton Keynes, MK7 6HP
U.K.
Email: lchen@dmu.ac.uk

Abstract

This paper introduces a conceptual framework for modeling and simulation of variable structure hybrid systems. The framework proposed is based on and further extends the DEVS-formalism, providing a novel definition for modular-hierarchical systems with discrete structural changes. An interactive, graphical model development approach and Object Oriented implementation are also presented. Finally, an application of the framework to variable structure systems is demonstrated.

1. Introduction

Formal study of discrete event dynamical systems is receiving ever more attention (Ho 1989). Work on a mathematical foundation of discrete event dynamic modeling and simulation began in the 70s (Zeigler 1976, 1984 and 1990) when DEVS (discrete event specification scheme) was introduced as an abstract formalism for discrete event modeling. Because of its system theoretic basis, DEVS is a universal formalism for discrete event dynamical systems (DEDS). Indeed, DEVS is properly viewed a short-hand to specify systems whose input, state and output trajectories are piecewise constant. The step-like transitions in the trajectories are identified as discrete events.

DEVS-formalism is extensively accepted and applied to a wide range of disciplines. Many extensions to the conventional DEVS have been made to ease the modeling and simulation of discrete event systems. They include DEVS with conditional events, combined discrete-continuous multiformalism (DEV & DESS) and DEVS for activity scanning.

More recently, interest in variable structure hybrid systems has been growing and attracting ever-increasing attentions. A variable structure hybrid system is a system with a hierarchical variable structure and a hybrid behavior functioning in continuous time. A hybrid behavior is a behavior that has both continuous and discrete phases and sequentially exchange one for another over time. A variable structure means that the components that form a system or the relations among components or between components and the system vary over time. Such systems are prominent in such areas as intelligent control, decision-making and reactive system design. Though combined discrete-continuous multiformalism provides a general common ground for modeling such systems, more investigation into their unique characteristics and new methodologies for dealing with those features are still needed.

In addition, object-oriented paradigm has long been regarded as a robust methodology that can be used for modeling and simulation. However there are - compared to other fields like database or computer graphics - only a few modeling and simulation tools that adopt the most important concepts like object, data encapsulation, generic class, inheritance or polymorphism operation from the object-oriented paradigm methodology.

In this paper some of the main characteristics of modular hierarchical systems with discrete structural changes are discussed and a novel knowledge representation scheme for specifying complex structural changes by direct extension of DEVS-formalism is proposed. A graphical editor is introduced to undertake the modeling visually and interactively. Then an approach to object-oriented modeling of modular hierarchical systems is presented. This approach makes use of the important concepts in object-oriented paradigm, especially the inheritance and polymorphism operations on the system modeling level. At the end of this paper, the framework is applied to a time variable system --- financial portfolio risk management.

2. Knowledge Representation for Variable Structure Hybrid Systems

Variable structure systems have been introduced as a new system by Oeren (Oeren 1975). Such systems have not only trajectory but also structural behavior, i.e. input-output relations between system and its components as

well as system components themselves vary in time. A variable structure system can be represented by atomic models and coupled models with the same notations as in traditional DEVS from the system theory point of view. However, the specification and implication of the models are different from the conventional DEVS due to the characteristics of variable structure systems. The systems studied in this paper have the following dynamics.

- Variable structure atomic models are self-organized with input, output, states, and transition functions for both discrete behavioral and structural events.
- No interconnections exist between components.
- Interactions take place between atomic models and their high level models. Interactions between models at the same hierarchical level is undertaken through their high level models.
- High level models are formed by connecting inputs (or outputs) of components to inputs (or outputs) of its high level models as shown in Figure1. We shall refer to such relations as aggregated models hereafter.
- A variable structure system can be specified with variable structure atomic models and variable structure aggregated models as shown in Figure 2.
- Any external or internal behavioral or structural events on atomic model level can cause changes to itself or further to aggregated models if some conditions are met. Behavioral and structural events in aggregated models will definitely lead to behavioral and/or structural changes to some or all of the models they contain.

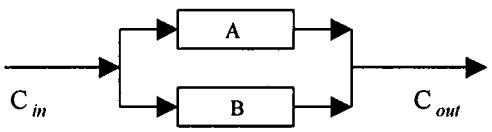

Figure 1: Aggregated models

Where:

A, B --- Variable structure atomic models.

C --- Variable structure aggregated models.

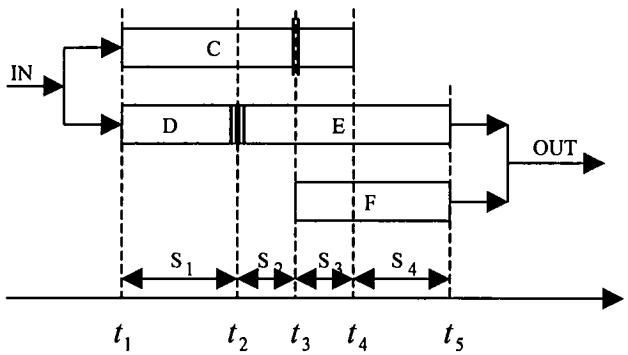

Figure2: Dynamic description of variable structure systems

Where C, D, E, F --- Variable structure atomic or aggregated models.

S_1, S_2, S_3, S_4 --- Different variable structure states

t_2, t_3, t_4 --- Times when structural events took place.

Currently, there are two popular modeling methodologies for variable structure systems. One is system-oriented approach, which was first specified in details by Zeigler (Zeigler 1986). In addition to his well-defined DEVS-formalism and multi-facetted modeling methodology (Zeigler 1984), he specifies "non-trivial" variable structure systems. "non-trivial" variable structure systems are characterized by phases of ordinary trajectory behavior and phases where the structure of the system is changed and the trajectory behavior state of the former structure is linked to the trajectory behavior state of the novel system structure. Each system structure represents a structure state. Another methodology for modeling variable structure systems, i.e. individual-oriented approach, focus on the autonomy of system components as self organizing units (Hogweg and Hesper 1989). System components are modeled as autonomous entities (individuals) with a clear boundary to their environment and they can change their internal structure or their environment according to internal rules.

The system representation proposed in this paper is mainly based on Zeigler's theory for modular hierarchical systems, but in contrast to other investigation for modeling and simulation of variable structure systems based on this theory, we further develop the specification of conventional coupled models. Aggregated models are introduced to describe the hierarchical embedded structure of variable structure systems. The relations between atomic model dynamics and aggregated model dynamics are also different from conventional DEVS formalism.

In conventional DEVS modeling, the component and coupling structure of a coupled model is fixed by model creation, i.e. a set of several atomic or coupling subsystems are specified and the coupling relations between them are defined. Atomic models exclusively define the behavioral and structural dynamics of the coupled models. In variable structure systems described above, the component and structure of an aggregated model vary in time. The dynamics of atomic models not only determines the dynamics of an aggregated model but also changes as the aggregated model undergoes behavioral or structural events. The structural events on atomic model level are defined to realize structural changes like switching between different sets of equations or different events. They do not definitely lead to structural or behavioral changes on aggregated model level. On the other hand, behavioral or structural events on aggregated model level are bound to result in behavioral or structural changes on atomic model level. Possible structural events are addition, deletion, adjustment and modification of atomic or aggregated components as well as their aggregated relations.

Additionally, we assume the system studied in this paper allows value aggregation. This means that input and output variables are introduced and can be connected to transfer variable value between a component and its aggregated model. These input and output variables can be aggregated in the same way as input and output ports. Furthermore, we allow the internal event to be conditional, i.e., the occurrence of an internal event depends on an internal event condition. Only when the time given by the time advance function has elapsed and/or the internal event condition evaluated to be true, the event is executed, otherwise the event is due.

Taking into account all those considerations, we formally introduce the mathematical representation for atomic models as follows:

$$Ato-Var-Model = (X, X_c, Y, Y_c, S, \delta_{ext}, \delta_{\text{int}}, C_{\text{int}}, \lambda, t_a, \lambda_c)$$

Where $X = \{(x_1, x_2, ..., x_m) \mid x_1 \in X_1, x_2 \in X_2, ..., x_m \in X_m\}$ is the structured set of inputs that includes behavioral and structural events with m input ports x_i, $X_c = \{(x_{c1}, x_{c2}, ...) \mid x_{c1} \in X_{c1}, x_{c2} \in X_{c2}, ...\}$ is the structured set of value inputs with input variables x_{ci}, $Y = \{(y_1, y_2, ..., y_p) \mid y_1 \in Y_1, y_2 \in Y_2, ..., y_p \in Y_p\}$ is the structured set of outputs with p output ports y_i, $Y_c = \{(y_{c1}, y_{c2}, ...) \mid y_{c1} \in Y_{c1}, y_{c2} \in Y_{c2}, ...\}$ is the structured set of value outputs with output variables y_{ci}, $S = (s_1, s_2, ..., s_n) \mid s_1 \in S_1, s_2 \in S_2, ..., s_n \in S_n\}$ is the structured set of sequential states, $\delta_{ext} : Q \times X_c \times X \to S$ is the external state transition function, with $Q = \{(s, e) : s \in S, 0 \le e \le t_{a(s)}\}$ the set of total states, $\delta_{\text{int}} : S \times X_c \to S$ is the internal state transition function, $\lambda : S \times X_c \to Y$ is the output event function, $\lambda_c : S \times X_c \to Y_c$ is the value output function to define the output variable values, $t_a : S \times X_c \to R_0^+ \cup \{\infty\}$ is the time advance function, and $C_{\text{int}} : S \times X_c \to \beta$ is the internal condition function for conditioning the execution and output of internal events.

We specify aggregated models in the following way:

$$Aggr-Var-Model = (X_A, X_{Ac}, Y_A, Y_{Ac}, S_A, C_A, C_{\text{int}}, \lambda, \lambda_c, t_a)$$

Where $C_{\text{int}}, \lambda, \lambda_c$ and t_a are the same as the ones defined in atomic models, $X_A = \{(x_{A1}, x_{A2}, ...x_{Ap}) \mid x_{A1} \in X_{A1}, x_{A2} \in X_{A2} ...\}$ is the set of inputs of the aggregated model with p input ports x_{Ai}, $Y_A = \{(y_{A1}, y_{A2}, ...y_{Aq}) \mid y_{A1} \in Y_{A1}, y_{A2} \in Y_{A2} ...\}$ is the set of outputs of the aggregated model with q output ports y_{Ai}, $X_{Ac} = \{(x_{Ac1}, x_{Ac2}, ...) \mid x_{Ac1} \in X_{Ac1}, x_{Ac2} \in X_{Ac2} ...\}$ is the set of value inputs of the aggregated model, $Y_{Ac} = \{(y_{Ac1}, y_{Ac}, ...) \mid y_{Ac1} \in Y_{Ac1}, y_{Ac2} \in Y_{Ac2} ...\}$ is the set of value outputs of the aggregated model, $S_A = \{(s_{A1}, s_{A2}, ...s_{An}) \mid s_{A1} \in S_{A1}, s_{A2} \in S_{A2}, ...\}$ is the set of sequential system states, each $s_{Ai} = \{C_{Ai}, S_i, IAR_i, OAR_i, IAR_{ci}, OAR_{ci}\}$ with $(i = 1,2,...n)$, $C_A = \{Ato-Var-Model \quad d \mid d \in (1,2,....m)\}$ is the set of component atomic models contained in the system, $S = \{(s_1, s_2, ..., s_n) \mid s_1 \in S_1, s_2 \in S_2, ..., s_n \in S_n, ...\}$ is the set of behavioral and structural states, $IAR \subset \{(x_{Ai}, x_{dj}) \mid i \in (1,2,...p), d \in (1,2,...m), j \in (1,2,...m_d)$ is the input aggregation relation between atomic models and aggregated models, $OAR \subset \{(y_{dj}, y_{Ai}) \mid i \in (1,2,...q), d \in (1,2,...m), j \in (1,2,...m_d)\}$ is the output aggregated relation between atomic models and aggregated models, IAR_c and OAR_c are input and output value aggregation relations between atomic models and aggregated models respectively, with the nearly same notations as the IAR and OAR. It should be noted that the input and output aggregation relations are more complicated than that in conventional DEVS-formalism. They could be as simple as several fixed percentages or as complex as a set of derivative or algebraic equations.

In the variable structure hybrid systems specified above, atomic models and aggregated models will directly interact with each other. External events of aggregated models will be used as input events of atomic models. Output of atomic models can mean nothing to aggregated models or be used as an internal event of aggregated models, depending on the nature, magnitude and time of the output. An internal event of aggregated models can do nothing to aggregated or/and atomic models or results in behavioral or/and structural changes to aggregated and atomic models, depending on the evaluation of event condition functions.

A structure event condition of a variable structure hybrid system can be defined as time condition, trajectory state condition or structure state condition or as a combination of them. An internal (or external) event on atomic model level can pass on to aggregated model level through its output and finally leads to structural changes on aggregated model level if all the conditions are met. An internal (or external) event of aggregated models can modify not only the behavior or structure of the aggregated model but also by means of addition, deletion or modification of component models the structure of the lower subsystems.

As shown in Figure 2 the structural event at time t_2 causes the change from structural state S_1 to structural state S_2, i.e., model D is deleted from the system model and model E is added into the system model. The internal event at time t_3 in model C eventually results in a structural change on system model, i.e., a new model F is added. The system input IN and output OUT are connected to the inputs and outputs of its component models through aggregation relations.

3. Interactive, Graphical Modeling with Hierarchical State Transition Diagram

In the above framework on variable structure hybrid system, atomic model specification is organized around various phases that form a larger state space for an aggregated model and further a global state space for the system studied. The different phases of a model represent a partitioning of the state space into a set of mutual exclusive blocks where the different blocks identify qualitatively different system behaviors. For dynamically continuous systems, the phase can be used to associate

different derivatives with different phases and the phase transitions mean a change from one derivative to another.

The state space phase concept and the dynamic behavior specification organized around phases can serve as a basis for a graphical model representation and construction. The phase and phase transitions are naturally represented by a state transition diagram. A state transition diagram is used to show the state space of a given model class, the events that cause a transition from one state to another, and the actions that result from a state change. In the directed graph, the nodes depict the phases and the arcs the event transitions. In contrast to DEVS-diagram (Praehofer and Pree 1993), we distinguish not only the external and internal event arcs but also behavioral and structural event arcs. Each phase consists of constructional parameters, state parameters and behavioral functions. Each transition arcs consists of conditions that will invoke the transitions and transition functions. The conditions include external and internal behavioral and structural events and value condition expressions that are semantically tested continuously while the model is in the transition's source state. Transition functions are executed each time the transition is taken and a new phase is reached afterward.

A graphical editor is developed to facilitate the graphical design of an atomic or aggregated model. It contains a toolbox consisting of different functional building blocks in different shape icons. Each type of icon represents a phase state or transition with different implications. We use the following notations. Circular shape represents a phase that has only one way to next phase. Diamond shape represents a phase that has several possible next states depending on the conditions. Rectangular shape represents an end phase. Single solid line represents a transition caused by an external event, single dash line a transition resulted from a structural event, a single thick black line a transition caused by behavioral event and double solid line a transition resulted from an internal event. An arrow denotes the causal relation between two phases, i.e., the direction of the change of phase. Each icon has an interactive dialog that can be used to specify the correspondent phase state variables or behavior functions in the form of class member variables or functions. The graphical representation of an atomic model is shown in Figure 3.

Based on the above graphical representation, we propose an interactive visual modeling method by means of a graphical editor. The method consists of the following steps:

- The input and output interface of atomic models are specified in the form of input and output ports and variables.
- Each phase of an atomic model and its relevant transition originating from this phase are specified in the form of constructional parameters, state parameters, behavioral Functions, conditions and transition functions.

Sometimes the above steps are already completed when a model base is available. So we only need to start from next step.

- The system studied is decomposed into constitutional atomic models according to domain knowledge. Input and output interface of systems are also specified in the form of input and output ports and variables.
- All phases of all atomic models contained in high level models are specified in the form of message sequence charts, i.e., diagrams that show the relations, interaction and action sequence among atomic models or components over time.
- Further specify the message sequence charts with states, state behavior and transition information until all the dynamic relations and behaviors are clear.
- Implementation and validation.

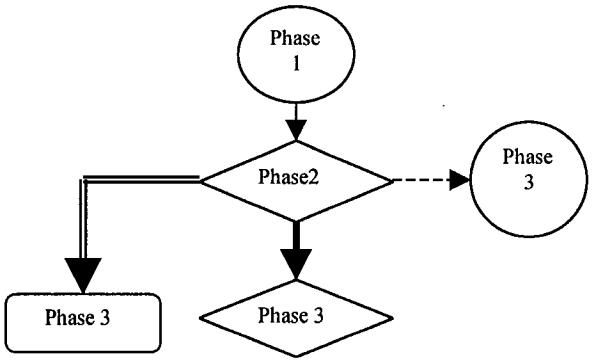

Figure 3 Graphical Representation of an Atomic Model

4. Object-Oriented Implementation with C++

Zeigler's System Entity Structure (SES) representation scheme combines knowledge of decomposition, component taxonomies and coupling relationships to present all possible structure of a complex system. The SES distinguishes between entities, aspects and specialization, in which entities are atomic or coupled systems, aspects describe decomposition of coupled system and specialization represents entity classification. Every entity in a specialization inherits the characteristics of the parent entity. Associated with SES is a model base.

We apply object-oriented paradigm to our specification for variable structure hybrid system based on DEVS formalism by mapping some important object oriented concepts to our specification. In our implementation, all atomic and aggregated models are defined by C++ classes. All elements, namely phase state parameters, behavior functions and local structures are declared as member variables or member functions in a model class declaration. For example, inputs and outputs (and constructional parameters in a sense)) are defined as public elements of a model class, all other elements of the model class are encapsulated into its body, called private elements. Transition functions and aggregated definitions are realized by polymorphic methods. For the

implementation of external transition function, a virtual method must be implemented. Internal condition function, internal transition function and output events have to be specified in such a way that they return a Boolean type value to indicate if the event is due.

A specific model component (instance) is declared as a variable of its root model class or, preferably, as a pointer to a model class. A static model instance is created or destroyed automatically when an instance of an appropriate compound model is created or destroyed. A static instance of a project is created when model execution begins and destroyed when it stops. A dynamic instance of a model can be created and destroyed as the results of "new" and "delete" statements in transition actions.

All model classes are derived from predefined correspondent root classes. All subclass attributes such as local structures, phase state variables, etc, will be inherited from a root class. You can add to subclass definition new elements such as new phase variables, parameters, local behavior as well as you may overload or override local behaviors, transition functions, etc,. Model polymorphism is supported, that means any decedent may be actually used in place of an ancestor.

A variable structure system is time variant, therefore a "pruning" process at modeling and simulation runtime is necessary. The "pruning" process can be realized by introducing structural events, which are able to modify the system structure by the following methods: CreateSystem(...), DeleteSystem(...), ExchangeSystem(...), CreateConnection(...), DeleteConnection(...), etc.

Interactions between atomic models and aggregated models or a system is through message passing. Recall that sometimes one output of an atomic model may not cause any change to its high level model. However, as time goes, several outputs from different models may combine together to lead to a behavioral or structural change to its high level model. So the conceptual framework supports asynchronous message passing or so-called later binding. Asynchronous means that the sender can send a message at any time, irrespective of the receiver status, and the control return to the sender immediately when the message gets to an input port at another model. The message transmitted may be dealt with immediately or stored in a queue waiting for later processing. Messages are sent and received through ports. An object sends a message to a port rather than to an object directly. The set of destination objects is defined by the port connection topology.

To summarize, the modeling and simulation of variable structure hybrid system can be realized by combining object-oriented paradigm and DEVS-based formalism in the following way:

- A number of template classes that are correspondent to atomic model classes of a domain are designed and standardized. All the classes form a model base.
- The system studied is decomposed and fully analyzed according to domain knowledge and application requirements.
- System model is constructed by using the specification framework described in Section 2 and the interactive graphical editor.
- Each model component is defined as pointers to its root template class. Modification is made through virtual functions and override to tailor the generic class to fit the specific model specification.

5. Application: Modeling Framework for Portfolio Risk Management

Exchanges, investment banks and other financial institutions have large portfolios of derivative securities that must be priced and hedged or risk managed. Each large investment portfolio contains a number of derivative securities and covers a fixed or varied period of time. The type and size of the derivative securities contained in a portfolio may vary according to the changes of political, economical or financial situation all over the world. Some derivative securities may be excluded from the portfolio and others may be added into it in terms of the properties of each derivative security. Sometimes the percentage distribution taken by derivative securities of a portfolio can be redistributed during the portfolio lifecycle. How and when these operations are undertaken depends on the level of risk exposure, the level of payoff rate portfolio managers will accept and the change of the financial market. Each time a derivative security is deleted from or is added into the portfolio or changes its size, a discrete event takes place, then portfolio becomes a continuous process before it expires. Portfolio risk management with derivative securities is a typical discrete and continuous process with the characteristics of variable structure.

Using the knowledge representation scheme for variable structure systems described in section 2, portfolio risk management with derivative securities can be modeled by a combined discrete and continuous model approach. Each derivative security such as options, futures, swap and forward contracts is represented as an atomic model and specified with the input, output, state and other relevant parameters. Assumptions and specification that underlie a financial instrument are made explicitly. All derivative securities models will form a knowledge base. Portfolio is represented as an aggregated model that consists of several derivative securities. Derivative security addition and deletion are modeled as structural events. Return requirement for a portfolio and its risk exposure level are used as the general modeling principles. In C++ realization, each type of derivative security is represented as a foundation class. The application of a derivative security is implemented as an instance (or a pointer) of the foundation class or its derivative class. Interaction between models, i.e. between classes, is through message passing.

The intelligent integrated framework for modeling and simulation in banking and risk management will contain the following functions:

- As close as possible to instantaneous pricing of individual financial instrument in support of daily trading activity.
- Automatic dynamic model construction and performance analysis of a given portfolio
- Fast reporting of the value and sensitivity of the portfolio to the market.
- Accurate end-of-day reporting of the detailed performance such as payoff, risk exposure of the portfolio.

6. Conclusion

In our research, we further extend the DEVS formalism to support variable structure system modeling and simulation. The modified formalism allows defining models modularly and hierarchically and combining various formalisms. The C++ implementation of these concepts has proposed a powerful modeling and simulation environment. The graphical editor provides building blocks that can be used to construct system models visually and interactively. The object-oriented features can be exploited to realize reusable model libraries.

Next investigations are going to automate the modeling process that involves in introducing reasoning mechanism into the formalism. Some model satisfaction criteria also need to be developed. Furthermore how to express and implement these mechanism to achieve automated modeling with C++ is still an open problem.

References

Zeigler, B. P.; Song, H. S.; and Praehofer, H. 1996. DEVS Framework for Modeling, Simulation, Analysis and Design of Hybrid Systems.

Kolesov, Y. B.; and Senichenkov, Y. B. 1995. Model Vision : a tool for hybrid systems simulation and its ODE solver. RSCC'95 –2nd Russian-Sweden control conference.

Ho, Y. C. 1989. Special issue on discrete event dynamic systems. Proceedings of the IEEE, 77(1).

Zeigler, B. P. 1976. Theory of Modeling and Simulation. John Wiley, New York.

Zeigler, B. P. 1984. Multifacetted Modeling and Discrete Event Simulation. Academic Press, London.

Zeigler, B. P. 1990. Object-Oriented Simulation with Hierarchical, Modular Models. Academic Press, London.

Oeren, T. I. 1975. Simulation of time-varying systems. Advances in Cybernetics and Systems.

Zeigler, B. P. 1986. Toward a simulation methodology for variable structure modeling. Modeling and Simulation Methodology in the Artificial Intelligence.

Hogeweg, P.; and Hesper, B. 1989. An adaptive, selfmodifying, non goal modeling methodology. Modeling and Simulation Methodology. Elsevier Science Pub.

Praehofer, H.; and Pree, D. 1993. Visual modelling of dev-based systems based on higraphs. In Winter Simulation Conference 1993, Los Angeles CA.

Pawletta, T.; Lampe, B.; Pawletta, S.; and Drewelow, W. An Object Oriented Framework for Modeling and Simulation of Variable Structure Systems. Univ. of Wismar and Univ. of Rostock.

Hull, J. Options, Futures and Other Derivative Securities. Second edition, Printice-Hall International, Inc.

Praehofer, H. 1991. System Theoretic Foundations for Combined Discrete-Continuous System Simulation. PhD thesis, Johannes Kepler Univ., Linz, Austria.