International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:2, No:5, 2008

Modeling Approaches for Large-Scale Reconfigurable Engineering Systems

Kwa-Sur Tam

***Abstract*—**This paper reviews various approaches that have been used for the modeling and simulation of large-scale engineering systems and determines their appropriateness in the development of a RICS modeling and simulation tool. Bond graphs, linear graphs, block diagrams, differential and difference equations, modeling languages, cellular automata and agents are reviewed. This tool should be based on linear graph representation and supports symbolic programming, functional programming, the development of non-causal models and the incorporation of decentralized approaches.

***Keywords*—**Interdisciplinary, dynamic, functional programming, object-oriented.

I. Introduction

WITH the advent of technology, engineering systems are becoming more capable but also more complex. In recent years, some systems have become complex due to the integration of large-scale interdisciplinary systems that have interdependencies on one another. They are sometimes referred to as systems of systems (SoS). Although there is no universally accepted definition of a SoS, it generally conveys the notion that it depends on all of its constituent systems to work cooperatively and continuously to accomplish the common mission.

Reconfigurable interdisciplinary complex systems (RICS) are interdisciplinary systems that support reconfiguration between the constituent systems and within some or all of the constituent systems. A RICS is a reconfigurable SoS. To optimize the utilization of available features, a SoS could be designed or retrofitted so that it can be reconfigured for different situations. At normal steady state, a RICS can be configured to optimize system performance such as reduction of losses or costs, improved reliability or operation flexibility. The same system can then be re-configured to accommodate systematic maintenance for different portions of the system or for installation of new components. When an event causes damage to a portion of the RICS, the system can be reconfigured for survival or to maintain as much of its critical missions as possible.

Without a modeling and simulation tool, the constituent systems of a RICS can only be analyzed separately. The total system response is then obtained by the superposition of responses of the constituent systems rather than the result of coordinated efforts. A RICS modeling and simulation tool needs to be developed to solve this problem and to enhance the design and analysis of RICS. Such a tool can play an important role in implementing system engineering principles to the entire life cycle of a RICS, resulting in higher quality of service provided by the RICS and lower total lifetime costs.

Currently, computer-based tools that can be used for reconfiguration analysis are discipline-specific. For example, there is software that reconfigures electric network and there is software that reconfigures fluid flow, but they cannot be integrated to simulate interdisciplinary systems. A few tools that can be used for the simulation of interdisciplinary engineering systems are not designed to support reconfiguration analysis. There is a need to develop a computer-based tool for the modeling and simulation of RICS. Since the costs of computers have decreased so much in recent years, computing resources required to simulate the reconfiguration of large-scale interdisciplinary engineering systems are no longer unattainable. Rather than looking for ways to patch together software from different disciplines, it is time to examine how a RICS modeling and simulation tool should be developed from scratch.

The objectives of this paper are to review various approaches that have been used for the modeling and simulation of engineering systems and to determine their appropriateness in the development of a RICS modeling and simulation tool. Approaches such as bond graphs, linear graphs, block diagrams, differential and difference equations, modeling languages, cellular automata and agents are reviewed in Section II. Section III presents an analysis of what features are needed for reconfiguration analysis and the advantages and disadvantages of the different approaches. Factors such as topology manipulation, model structure for reconfiguration analysis, causality, decentralized approaches, symbolic programming, functional programming, and modeling formalism are discussed from the viewpoint of engineering applications. Conclusions are presented in Section IV.

Kwa-Sur Tam is with the Virginia Polytechnic Institute and State University, Blacksburg, VA 24061 USA (e-mail: ktam@ vt.edu).

II. Modeling Engineering Systems

A. Equations

A mathematical model describes the behavior of an actual system by using equations. Table I shows the different types of equations used in different types of models that are formed by using different criteria [1]. When a system obeys the principle of superposition, it can be represented by a linear model using linear differential equations. When the parameters of a system depend on their spatial coordinates, a distributed model needs to be used instead of a lumped model. Difference equations are used in discrete-time model in which events only happen at distinct values of time. A discrete model describes a system in which both time and magnitudes only exist at distinct values. A hybrid model describes a system that has both continuous and discrete components. To facilitate analysis, the model of a system can be transformed from the time domain into another domain such as the s domain or the z domain. Algebraic equations are used to describe behavior of models formulated in these domains.

The types of models shown in Table I are not necessarily exclusive of one another. A model commonly used in engineering is the linear time-invariant (LTI) model which could be in continuous or discrete time, or in the s domain. The types of equations shown in Table 1 can be summarized by the acronym DAE (differential algebraic equations). A model that simulates dynamic behavior has to have DAE at its core. Models created by the approaches discussed below are converted into DAE before computation is performed.

B. Block Diagrams

A block diagram describes the relationship between input variables and output variables [2]. A common modeling approach in engineering is to use block diagrams to represent the models of basic components. These block diagram models are then interconnected to form subsystem models which in turn are interconnected to form an overall system model. Compared to DAE, block diagrams has the benefits of model testability and modularity.

C. Bond Graphs

Bond graphs have been used to represent a variety of engineering systems [2-4]. Fig. 1 shows the bond graph representation of a resistor R. The direction of the half arrow indicates the direction of power flow. The power flow through each component of a bond graph is the product of the effort variable and the flow variable. Table II shows how the effort and flow variables map into system variables of different types of engineering systems. To form a graph, the components are joined together by the 0-junction where the effort variables are made the same and the flow variables are summed to zero, and the 1-junction where the flow variables are made the same and the effort variables are added. Bond graphs are based on conservation laws.

TABLE I
Equations for Different Types of Models

Type of Model	Type of Equation
Nonlinear	• Nonlinear differential equations • Linearized differential equations
Linear	Linear differential equations
Distributed	Partial differential equations
Lumped	Ordinary differential equations
Time-varying	Differential equations with time-varying parameters
Stationary	Differential equations with constant parameters
Continuous	Differential equations
Discrete-time	Difference equations
Discrete	Difference equations
Hybrid	Differential and difference equations
s domain	Algebraic equations
z domain	Algebraic equations

TABLE II
Comparison of Bond Graph and Linear Graph Variables

System	System Variables	Bond Graph Variables	Linear Graph Variables
Electric	Voltage	Effort	Across
	Current	Flow	Through
Fluid	Pressure	Effort	Across
	Volume flow rate	Flow	Through
Translational	Force	Effort	Through
	Velocity	Flow	Across
Rotational	Torque	Effort	Through
	Angular velocity	Flow	Across

Bond graphs show clearly how power flows between components of the systems they represent. The effort and flow variables enable bidirectional information flow in the bond graph model. When the power level of a system is low, bond graphs degenerate into signal flow graphs in which information is mainly one dimensional and power is minimal. Signal flow graphs can be used to model monitoring circuits in which information flows from the sensors to the controllers and control circuits in which information flows in the opposite direction.

The topology of a bond graph can be quite different from the topology of the system that it represents [2]. Bond graphs are also not convenient for modeling hybrid continuous-discrete systems [2]. Since the direction of the arrow needs to be assigned when the model is created, bond graph is not a good candidate for modeling non-causal situations in which the direction of input-output relationship can change.

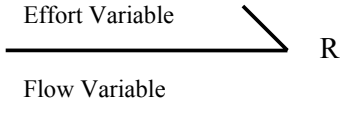


Fig. 1 Bond graph representation of a resistor

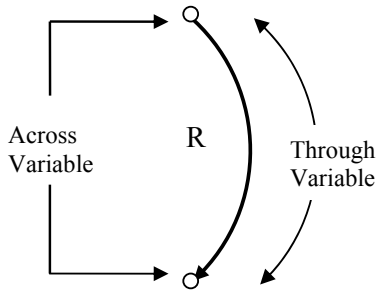


Fig. 2 Linear graph representation of a resistor

D. Linear Graphs

Physical networks such as the electric power systems, the water pipeline networks or the communication networks can be modeled by using linear graphs. A linear graph consists of arcs, which represent system components, and nodes which represents where the arcs are joined together. Fig. 2 shows the linear graph representation of a resistor.

Linear networks can be described by using two sets of variables, the through variables and the across variables [2][5]. Table II shows how the through and across variables map into system variables of different types of engineering systems. The through variables observe the law of continuity which is based on either the law of conservation of mass or the law of conservation of electric charge. The across variable observes the law of conservation of energy in various forms such as the Kirchhoff's voltage law in electric networks and the Bernoulli equation in fluid flow networks.

For linear graph representation, a system component is modeled by using the equation that describes the relationship between the through variable and across variable of that component. Based on how the components are interconnected, component equations are combined to form a set of system equations. Dynamic behavior of the system is simulated by solving the system equations.

Table II compares the variables used in bond graphs and linear graphs. In some cases, effort variables in bond graphs correspond to across variables in linear graphs. In other cases, effort variables in bond graphs correspond to through variables in linear graphs. Bond graphs cannot be transformed into linear graphs by substituting effort variables for across variables and flow variables for through variables. Furthermore, unlike bond graph models, the topology of a linear graph model closely resembles that of the actual system. It is also easier to create non-causal models by using linear graphs than bond graphs.

E. Modeling Languages

In electronics, hardware description languages (HDL) have been used to specify temporal behavior as well as structures of electronic systems. While net-list languages only express circuit connectivity, HDL expresses structure as well as time and concurrency. VHDL-AMS (Very-high-speed-integrated circuits Hardware Description Language-Analog Mixed Signal) has been developed in recent years to model multi-disciplinary systems in both continuous and discrete domains [6]. VHDL-AMS and another multi-disciplinary modeling language, Modelica [7], support the development of non-causal models as well as causal models. For non-causal models, the mathematical causality of the equations is not specified by the modeler. The simulation engine converts the equations into procedures that can be evaluated by the computer. Non-causal models enhance flexibility in information flows and enable model reuse.

F. Decentralized Approaches

Many large-scale engineering systems have been designed based on centralized control and hierarchical organization. In recent years, it has been observed that many behaviors of large-scale complex systems stem from the combined effect of many localized reactions to local events. Furthermore, a local event and the impact of the ensuring local reactions can ripple through a system and cause more widespread impact to the overall system. It is envisioned that with the advent of various enabling technologies, intelligent decentralized control will play an increasingly important role in real-time auto-reconfiguration of engineering systems. Decentralized control needs to be incorporated into the modeling tool for RICS.

Autonomous decentralized control can be implemented and modeled by using cellular automata (CA) techniques or agent-based systems [8][9]. The key to modeling the impact of CA is to represent the effect of the CA rules. Although CA rules are designed for a cell, CA-based model still requires a model for the overall system, preferable in the form of a matrix, for the CA rules to be applied.

Agent-based modeling and simulation (ABMS) has been applied to a variety of applications [9][10]. An intelligent agent determines the state of its surrounding and autonomously takes action according to the plan that has been programmed into it. Decentralized real-time control provides fast response to local events but such actions may have implications for other parts of the system. Before they are implemented, decentralized control schemes using agent or CA should be tested on a model of the overall system. A RICS model is still needed even when decentralized control is used. A RICS model also supports HIL (hardware-in-the-loop) approach in the design and testing of new component.

III. Modeling Reconfigurable Systems

System configuration is a fundamental attribute that affects all aspects of a system. The RICS modeling and simulating tool needs to capture all major effects of reconfiguring a system for various purposes and based on different criteria.

A. Structure Manipulation

It is essential that the modeling tool has an effective way to manipulate system topology to explore different possibilities. System structures can be represented by graphs [11][12].

Since linear graphs provides a closer resemblance of the topology of the actual system than bond graphs, models for reconfigurable systems should be based on linear graphs. Furthermore, graph-based algorithms can be used to formulate optimal structures such as that for maximum connectivity (and thus reliability) [12] or optimal operation strategy such as maximum flow based on minimum cut. Graph theory can also be applied to analyze various characteristics or properties of a RICS represented by a linear graph.

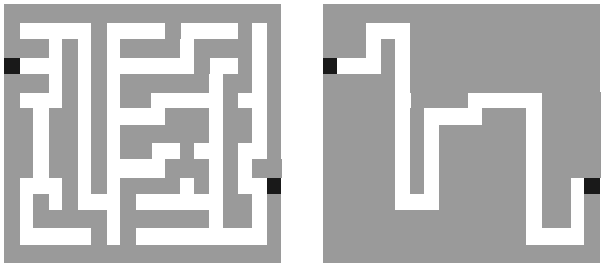
Linear graphs can be represented in various forms such as adjacency matrices, incidence matrices, edge lists and ordered pair lists. A linear graph can be transformed from one form to another. Depending on the type of analysis and synthesis, applying the right form can make the job a lot easier.

A key enabling tool for structure manipulation is symbolic programming. The topology of many engineering system structures can be represented in the form of lists (one of the forms of linear graph). A programming language that can process lists symbolically can be applied to manipulate the structures that the lists represent. Since list and matrix are transformable from one another, a list-generated structure can be transformed into its matrix equivalent for other purposes.

Matrix representation of system structure has the benefit that decentralized technique such as cellular automata needs a matrix-like structure to work on [8]. Being able to convert topology information from other forms to matrix form opens the door for many new possibilities because centralized techniques can now be combined with decentralized techniques.

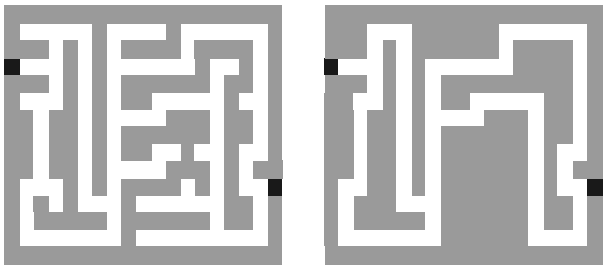
The example shown by Fig. 3 and Fig. 4 provides a glimpse of the benefits that a system reconfiguration tool can offer. Depending on what entity needs to be moved from one place to another, routing is a problem that appears in different forms in different disciplines of engineering. Fig. 3a shows the graphical representation of a network that enables traffic to go between various locations of the network. Suppose that among other requirements, it is important to have a route between one location on the left (indicated by a black square) and another location on the right (indicated by another black square). Either CA technique or a search algorithm can be implemented to find a route between these designated sites. Fig. 3b shows that only one route can be found. Traffic between the two designated sites depends on this one route.

The goal of a reconfigurable model is to determine what can be changed so that more routes can be established between the two designated sites without major rebuild. Fig. 4a shows that two minor structural changes can result in more possible routes as shown in Fig. 4b. The two changes are joining the two paths near the upper right corner and joining a north-south path with an east-west path near the middle of the lower boundary. Fig. 4b shows that between the two designated sites there are now two adjoining loops that provide 4 possible routes. The probability that traffic between the two designated sites be totally shut down is reduced dramatically for the reconfigured traffic network.



(a) (b)

Fig. 3 (a) A traffic network (b) A route between designated sites



(a) (b)

Fig. 4 (a) A reconfigured traffic network (b) Multiple routes between designated sites

The situation gets more complicated for large-scale systems and for interdisciplinary systems that have interdependence on one another [13]. Fig. 5 shows the linear graph representation of two engineering systems from different disciplines. The dependency relationship is represented by the double arrows, the direction of which shows the dependency. For example, the two interdependent systems can be the electric power system (system 1) and the natural-gas system (system 2). The double arrow that goes from system 2 to system 1 represents the dependency of a gas-fired power station (node in system 1) on the supply of natural gas from a delivery point (node in system 2) in the natural-gas pipeline system. The double arrow that goes from system 1 to system 2 represents the dependency of an electric-driven gas compressor that depends on the electric power from a delivery point (a load node in system 1) in the electric power system. Interdependency relationships between other basic systems can be represented similarly. Reconfiguration of the electric power system becomes complicated when interdependency relationships with other systems need to be taken into consideration. If both the electric power system and the natural gas system need to be reconfigured simultaneously (for example, after a natural disaster), the problem becomes even more challenging.

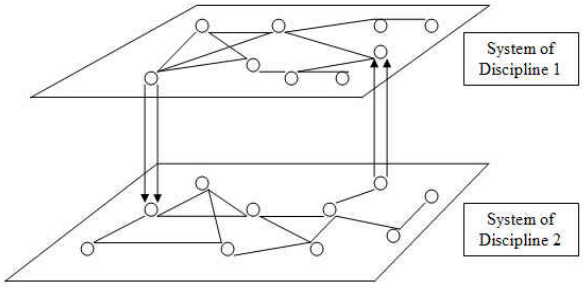


Fig. 5 Linear graph representation of two systems

B. Model Structure

To support reconfiguration, the component models need to include component equations but not the predetermined causal relationships. For non-causal models, equations should be in declarative form rather than in assignment form. Since block diagrams require predetermined input-output relationships, this approach is not suitable for the modeling and simulation of RICS.

Using the same procedure that sets up system equations in the linear graph approach, other types of analysis can be performed on candidate systems of different structures if the component models contain other information in addition to dynamic equations. When the arcs of a graph represent capacities of the network components, the problem can be structured to analyze flow patterns to maximize flow [12][14]. When the arcs of the same network represent costs, the problem can be re-structured to analyze least-cost operation. The same idea can be applied to the analysis of reconfigurable systems if the component models contain information such as component failure rate (for reliability analysis), cost (for cost analysis), and mathematical representation in other domains (for analysis in those domains) in addition to the time-domain dynamic equation.

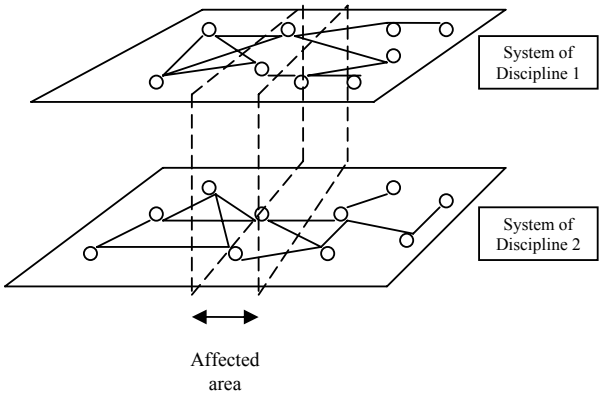


Fig. 6 Common cause failure of two systems

Object-oriented approach is the natural choice for formulating system components which are represented as arcs in the linear graph representation. Engineering components in different domains have been generalized into a generic hierarchy framework [5]. This framework can be implemented by a hierarchy of classes. For example, there is a class of two-terminal elements. Under this class, there are subclasses of D-type, T-type and A-type elements. Domain-specific subclasses of elements such as capacitor, torsional spring and damper can be created from these generic subclasses.

As part of the linear graph representation, component models are interconnected through the joining of terminals. The conventional linear graph representation only provides interconnection information but no information on spatial coordinates. One kind of interdependency relationship is environmental dependency [13], an example of which is shown in Fig. 6. It is quite common that portions of two interdependent systems are physically close to each other or co-locate within the same facility such as an underground conduit. When an event such as a natural disaster strikes a location, it damages all systems that reside in that general area. This will trigger a reconfiguration of all engineering systems involved. To model this kind of common cause failure, environmental dependency needs to be modeled. One way to do this is to incorporate spatial coordinate information as instance variables in the component object model.

C. Hierarchy of Functions

The major characteristics of object-oriented approach are encapsulation, inheritance and polymorphism. These object-oriented ideas can be applied to the organization of functions. Engineering systems share similar approaches in performing steady-state analysis, transient analysis, reliability analysis, etc. Numerical techniques such as Newton-Raphson method have been applied to analyze power flows in electric grid [15] as well as fluid flows in pipeline systems [16].

Some procedures, with minor modification, can be applied to perform different kinds of analysis. For example, in electric systems, a network of impedances can be combined to form an equivalent impedance by applying the rules for series and parallel connection and in some cases wye-delta transformation. For the same network, if the component model contains data on failure rate (as discussed in the previous section), an equivalent failure rate for the entire network can be obtained by applying the rules of series and parallel connection and in some cases transformation into minimum cut-set [17]. Although the rules of reduction are different, the overall procedure to reduce a network to a representative number is similar. This procedure can be generalized into a generic function class called NetEquivalent. Two functions, Equivalent_Impedance and Equivalent_Failure-Rate, can be created from this class by combining the features inherent from being an instantiation of NetEquivalent and the specific rules for combining impedances and failure rates, respectively.

To facilitate reconfiguration analysis of large-scale interdisciplinary systems, it will be useful to have various analytical functions organized into a hierarchy using the class concept as much as possible. Discipline-specific information can be added when a discipline-specific analysis function is

created from a generic class. A modeling tool that supports functional programming will be very helpful in handling the various functions of functions.

Functional programming allows functions to be used as arguments of other functions. Since a computer program may be viewed as a function, functional programming languages such as Mathematica treat a computer program like data. It enables "function components" to be assembled in different ways to create various kinds of more powerful "function blocks". For example, functions that compute admittance matrix of an electric network and functions that compute power flows are used inside the functions that determine transient stability [15]. Similar to objects, functions can be encapsulated, inherited and be made polymorphic.

In the theory of modeling and simulation, different modeling formalisms have been used to specify different kinds of system models [18]. Formally, a system is defined by $S = (T, X, \Omega, Y, Q, \Delta, \Lambda)$, where T is a time base, X is the input values set, Ω is the set of allowable input segments, Y is the output values set, Q is the set of states, Δ is the state transition function and Λ is the output function. This formalism assumes that the system has a time invariant structure.

Changes in structure, either in terms of addition/deletion or the modification of relationships among components, introduce γ, the structure function, and $\sum^*$, the set of network structures, to the system specification [18] [19]. A dynamic structure system is then defined by $S_\chi = (T, X_\chi, \Omega_\chi, Y_\chi, Q_\chi, \Delta_\chi, \Lambda_\chi, \gamma, \sum^*)$, where χ is the name of the network executive which is dependent on the system structure. S_χ can represent RICS with the inputs, outputs and transition functions etc., expanded to include pertinent multi-disciplinary entities.

IV. Conclusion

This paper reviews various approaches that have been used for the modeling and simulation of large-scale engineering systems and determines their appropriateness in the development of a RICS modeling and simulation tool. This tool should be based on linear graph representation and supports symbolic programming, functional programming, the development of non-causal models and the incorporation of decentralized approaches. Using object-oriented approach, both the component models and the analytical functions can be organized into a hierarchy of generic classes from which domain-specific models and function classes can be created.

References

[1] J. L. Shearer, B. T. Kulakowski, *Dynamic Modeling and Control of Engineering Systems*, New York, Macmillan Publishing Company, 1990, ch. 1.

[2] R. Sinha, C.J.J. Paredis, V-C Liang, P.K. Khosla, "Modeling and Simulation Methods for Design of Engineering Systems," *ASME Journal of Computing and Information Science in Engineering*, vol. 1, March 2001, pp. 84-91.

[3] D. C. Karnopp, D. L. Margolis, R. C. Rosenberg, *System Dynamics Modeling and Simulation of Mechatronic Systems,* Hoboken, New Jersey, John Wiley & Sons, 2006.

[4] A. J. Blundell, *Bond Graphs for Modeling Engineering Systems,* West Sussex, Ellis Horwood Limited, 1982.

[5] D. Rowell, D. N. Wormley, *System Dynamics An Introduction*, Upper Saddle River, New Jersey, Prentice Hall, 1997.

[6] S. A. Huss, *Model Engineering in Mixed-Signal Circuit Design*, Boston, Kluwer Academic Publishers, 2001.

[7] P. Fritzson, *Principles of Object-Oriented Modeling and Simulation with Modelica,* New York, IEEE Press, 2004.

[8] R. J. Gaylord, K. Nishidate, *Modeling Nature*, New York, Springer-Verlag, 1996.

[9] C. M. Macal, M. J. North, "Tutorial on Agent-based Modeling and Simulation, *Proceedings, Winter Simulation Conference,* 2005.

[10] H. V. D. Parunak, "Practical and Industrial Applications of Agent-Based Systems, http://agents.umbc.edu/papers/apps98.pdf, 1998.

[11] J. L. Gross, J. Yellen, *Graph Theory and Its Applications*, Boca Raton, Chapman & Hall/CRC, 2006.

[12] S. Pemmaraju, S. Skiena, *Computational Discrete Mathematics,* Cambridge, Cambridge University Press, 2003.

[13] K. S. Tam, R. B. Broadwater, "A Framework to Model Interdependent Engineering Systems, *IASTED International Conference on Modeling Identification and Control,* Innsbruck, Austria, February 2005.

[14] W-K, Chen, *Theory of Nets: Flows in Networks,* New York, John Wiley & Sons, 1990.

[15] J. D. Glover, M. S. Sarma, *Power System Analysis and Design,* Brooks/Cole Thomson Learning, 2002.

[16] J. Saleh, *Fluid Flow Handbook,* New York, McGraw-Hill, 2002.

[17] R. E. Barlow, *Engineering Reliability,* Philadelphia, Society for Industrial and Applied Mathematics, 1998.

[18] B. P. Zeigler, H. Praehofer, T. G. Kim, *Theory of Modeling and Simulation*, San Diego, Academic Press, 2000.

[19] F. J. Barros, "Modeling Formalisms for Dynamic Structure Systems", *ACM Transactions on Modeling and Computer Simulation,* vol. 7, No. 4, October 1997, pp. 501-515.