Proceedings of the 1996 Winter Simulation Conference
ed. J. M. Charnes, D. J. Morrice, D. T. Brunner, and J. J. Swain

DYNAMIC MODEL ABSTRACTION

Kangsun Lee
Paul A. Fishwick

Department of Computer and Information Science and Engineering
University of Florida
Bldg. CSE, Room 301
Gainesville, FL 32611

ABSTRACT

While complex behavior can be generated through simple systems, as in chaotic and nonlinear systems, complex systems are found where a systems study contains multiple physical objects and interactions. Through the use of *hierarchy*, we are able to simplify and organize the complex system. Every level within the hierarchy may be refined into another level. System abstraction involves simplification through structural system representation as well as through behavioral approximations of executed model structure. There has been little work on creating a unified taxonomy for model abstraction. We present such a taxonomy and define two major sub-fields of model abstraction, while illustrating both sub-fields through detailed examples. The introduction of this taxonomy provides system and simulation researchers with a way in which to view and manage complex systems.

1 INTRODUCTION

Real world dynamic systems involve a large number of variables and interconnections. Abstraction is a technique of suppressing details and dealing instead with the generalized, idealized model of a system. The need of abstract models and traversing levels of abstractions are essential as complex models are used in practice. Computational efficiency and representational economy are main reasons of using abstract models in simulation (Fishwick 1987; Fishwick 1989; Zeigler 1972) and well as in programming languages (Berzins et al. 1986; Booch 1991).

Although many diverse areas employ abstraction methods, no agreed-upon taxonomy has been developed to categorize and structure them with underlying characterization of a general approach. Our goal is to clarify how abstraction methods relate to each other under a uniform taxonomy. We define system abstraction to be one of two types: *behavioral* or *structural*. In most cases, one should explore both of these methods when constructing systems. For instance, when a system is first being designed, one should construct it hierarchically, with simple system types at first, graduating to more complex model types later. Structural abstraction corresponds to this iterative procedure (Fishwick and Lee 1996; Fishwick 1996a; Fishwick 1996c). After creating the hierarchy, we may want to isolate abstraction levels, so a level can be executed apart from the rest of hierarchy with no detailed internal structure. This is where the behavioral approaches are employed. In depth discussions of each abstraction technique follow in the subsequent sections.

Our contribution is the formulation of a taxonomy capturing two types of abstraction, which have generally been overviewed in separate disciplines. Structural abstraction is found mostly in information on *design*, whereas behavioral abstraction is strewn across many fields of computer science and simulation. Through a unification in terminology, we demonstrate that structural and behavioral methods are complementary aspects of system abstraction. Structural abstraction is common in programming language development within computer science as well as in simulation. Behavioral abstraction is common in statistical analysis and automatic control where system abstractions are used in lieu of more complicated model-based transfer functions. Along with our discuss of the taxonomy, we present examples of each approach to complete the discussion.

The paper is organized as follows : we present the new system abstraction taxonomy with specific methods of each category in Section 2. Then we illustrate the abstraction types using two scenarios and show how abstraction methods perform in both linear and nonlinear system abstraction, in Sections 3 and 4. We close with a summary of the taxonomy and its advantages, with future goals to be achieved in Section 5.

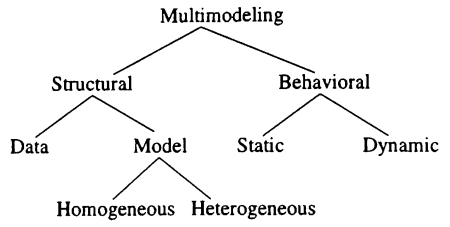

Figure 1: Proposed Taxonomy for Abstraction

2 ABSTRACTION TAXONOMY

Figure 1 illustrates our abstraction taxonomy. A system consists of data and model components. Data refers to values obtained either by observation or arbitrary assignment of values to model components. Model components, which serve as fundamental building blocks for models, take on the data values. Sample model components include state and event (Fishwick 1995). We sub-define structural abstraction of a system into *data abstraction* and *model abstraction.*

- Data Abstraction : abstraction of input, output, time, parameter system values or time-dependent trajectories.
- Model Abstraction : abstraction of dynamical models.

Examples of data abstraction are symbolic value, statistic mean and variance, interval, ratio and fuzzy numbers. Data abstraction represents a way of compressing time-dependent information. In constructing a model, we further refine model abstraction to *homogeneous* and *heterogeneous* abstraction (Miller and Fishwick 1992; Fishwick 1991). For homogeneous-structural abstraction, dynamical systems can be abstracted with only one model type, depending on the level of information that one expects to receive from analysis. Specific model types are required at different abstraction levels. For example, one would not choose to model low-level physical behavior with a Petri net since a Petri net is an appropriate model type for a particular sort of condition within a system, where there is contention for resources by discretely-defined moving entities. Examples of homogeneous-structural abstraction are conceptual, declarative, functional, constraint and spatial modeling. Detailed discussion on each model type is shown in (Fishwick 1995). Models must be multi-layered so that different abstraction levels of the model respond to different needs of the analyst.

In heterogeneous-structural abstraction, different abstraction levels of a system are provided by allowing either homogeneous or heterogeneous model types together under one structure. To incorporate different levels together, we have constructed a multimodeling methodology (Fishwick 1991; Fishwick and Zeigler 1992; Fishwick 1993; Fishwick et al. 1994), which provides a way of structuring a heterogeneous and homogeneous set of model types together so that each type performs its part, and the behavior is preserved as levels are mapped (Fishwick 1988; Zeigler 1972; Zeigler 1990). Heterogeneous-structural abstraction is equivalent to multimodeling in the sense that we abstract a system structurally using homomorphic relationships of one level to another, providing multiple level abstractions. While the multimodel approach is sound for well-structured models defined in terms of state space functions and set-theoretic components, selecting system components in each level are dependent on the next-lowest level. This implies that we are unable to run each level *independently.* It is possible, to obtain *output* for any abstraction level but, nevertheless, the system model must be executed at the lowest levels of the hierarchy. A new definition and methodology are needed to better handle abstraction of systems and components.

Behavioral abstraction is where a system is abstracted by its behavior. We replace a system component with something more generic that approximates, to some degree of accuracy, the behavior of the system component at its refined levels. Therefore, discarding the refined levels that define a system component will still result in a complete behavioral description of a system (Fishwick and Lee 1996). By incorporating behavioral abstraction approaches into multimodeling allows each level to be understood independetly of the others. This is why we put multimodeling on the top of our taxonomy.

We have two approaches of specifying system behavior:

- Static approach : one takes a system and captures only the steady state output value instead of a complete output trajectory. The input value is defined to be the integral of time value over the simulation trajectory.
- Dynamic approach : one needs to associate time-dependent input and output trajectories.

System identification (Ljung and Soderstrom 1983; Johansson 1993) is to abstract a system by *mathematical models.* Modeling the system consists of selecting a general, parameterized mathematical representation and then tuning the parameters, so that behavior predicted by the model coincides with measurements from the real system. Parameter estimation procedure provides a search through parameter space, effectively, to achieve a close-to optimal

Table 1: Sample Abstraction Categories and Associated Techniques

Base Abstraction Type	Abstraction Technique
Data Abstraction	Symbolic Value
	Mean, Variance
	Interval, Ratio
	Fuzzy Number
Structural Abstraction	Conceptual Modeling
	Declaration Modeling
	Functional Modeling
	Constraint Modeling
	Spatial Modeling
	Multimodeling
Behavioral Abstraction	Regression
	System Identification
	Neural Network
	Wavelet
	Genetic Algorithm

mapping between the actual values of the system and the approximate abstract system. Commonly used parameter models are ARX, ARMAX, OE (Output Error) and BJ (Box-Jenkins) (Tan et al. 1995; Johansson 1993). Brief explanations of these models are shown in Section 3.2.2.

Neural networks have been established as a general approximation tool for fitting models from input/output data (Cynbenko 1989; Tang et al. 1991; Tang and Fishwick 1993). From the system identification perspective, a neural network is just another model structure (Ljung and Soderstrom 1983; Barron 1989). The inputs are linearly combined at the nodes of the hidden layer(s) and then subjected to a threshold-like non-linearity, and then the procedure is repeated until the output nodes are reached. Backpropagation, recurrent and temporal neural networks have been shown to be applicable to modeling an identification (Fishwick and Lee 1996; Mills et al. 1995). On the other hand, recently introduced *wavelet decomposition* achieves the same quality of approximation with a network of reduced size by replacing the neurons by "wavelons", i.e. computing units obtained by cascading an affine transform and multidimensional wavelets (Ahang and Benveniste 1992).

Table 1 summarizes the based categories along with some sample abstraction techniques discussed so far. Having defined the model abstraction taxonomy, we now proceed to illustrate the different abstraction techniques using the following example.

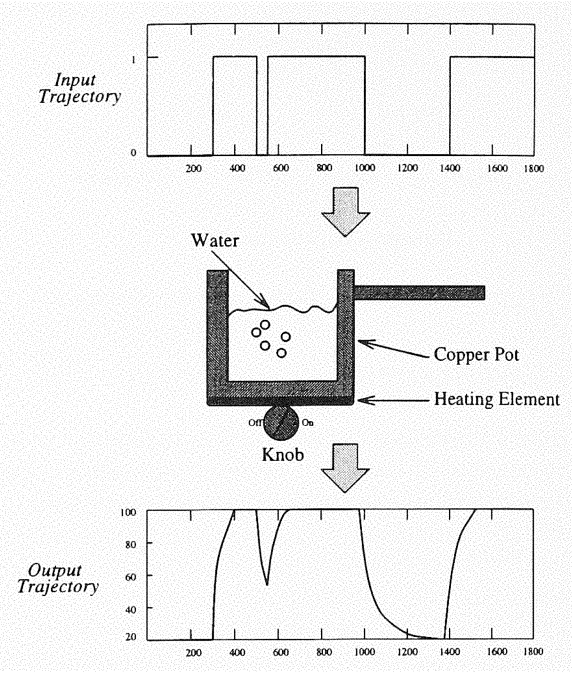

Figure 2: Boiling Water System

3 EXAMPLE I: BOILING WATER MODEL

Consider a pot of water in Figure 2. Here we show a picture of the boiling pot along with an input and output trajectory. The input reflects the state of the knob, which serves to specify external events for the system. The output defines the temperature of the water over time. Newton's law of cooling states that $Rq_h = \Delta T = T_1 - T_2$ where T_1 is the temperature of the source (heating element), and T_2 is the temperature of the water. q_h is heat flow. Since T_2 is our state variable we let $T = T_2$ for convenience. By combining Newton's law with the capacitance law, and using the law of capacitors in series, we arrive at:

$$k = \frac{C_1 + C_2}{RC_1C_2}$$
$$\dot{T} = k(T_1 - T)$$

3.1 Structural Abstraction

The structural approach to system abstraction for the boiling water is defined in a recent text (Fishwick 1995) where the boiling water is included as a sub-system within a system of two flasks and a human operator who mixes the flasks once the liquids are boiling. In the structural abstraction approach to systems, we first need to define our levels of abstraction and then choose which models types to use at each level.

We show part of the multimodel in Figures 3 and 4. The first model is a compressed version of all the

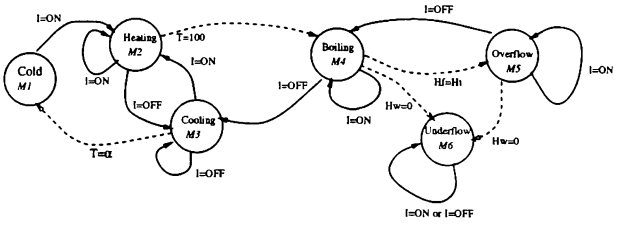

Figure 3: Six State Automaton Controller for the Boiling Water Multimodel

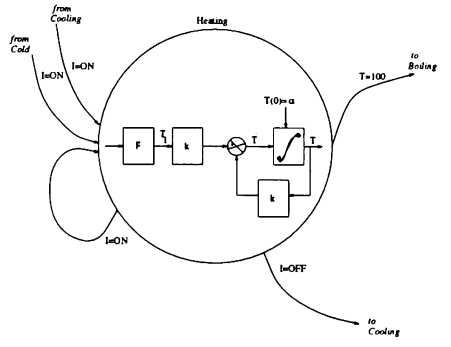

Figure 4: Decomposition of *Heating* State

hierarchy. Figure 4 shows Newton's law of cooling in a functional block form.

3.2 Behavioral Abstraction

3.2.1 Static Approach

In the static approach, we're interested only in the *final* (i.e., steady state) temperature of the water. Our two inputs are: (1) total amount of elapsed time for the input trajectory and (2) integral value of the input trajectory integrated over time. The output is the temperature of the water at the elapsed time. A graph of elapsed time versus temperature is shown in Figure 5. This information is obtained directly from the underlying simulation of the boiling water system. We chose a subset of all possible input time trajectories in such a way that some nonlinearity was introduced into the graph in Figure 5. This was done to challenge the behavioral parameter estimation methods in creating a good fit. This explains why Figure 5 contains a small area of discontinuity in the region between steady state temperature values of 20 and 40.

We will use linear regression to exemplify the static approach. In general, a polynomial fit to data in vectors x and y is a function p of the form:

$$p(x) = c_1 x^n + c_2 x^{n-1} + ... + c_d$$

The degree is n and the number of coefficients is $d = n + 1$. The regression coefficients $c_1, c_2, ..., c_n$ are determined by solving a system of simultaneous

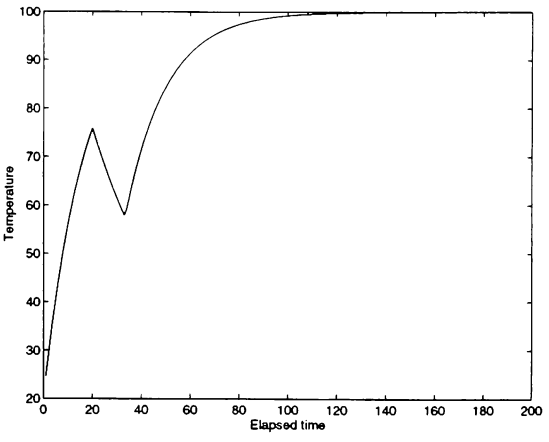

Figure 5: Elapsed Time vs. Temperature

linear equations: $Ac = y$ (Law and Kelton 1991). Figure 6 shows the result. The approximation is poor in the graph's central region because linear regression is done by polynomial fit, and so it generates a *monotonically increasing* function.

3.2.2 Dynamic Approach

In the dynamic approach, we're interested in time dependent behavior. In this case, we are concerned not only in the steady state temperature but also the way in which the temperature changes over time. For this approach, we chose a system with just one input and one output, both time-varying trajectories. The input is the input "knob off/knob on" trajectory and the output is the temperature trajectory.

The Box-Jenkins method is a frequently used linear system identification method in time series analysis (Tang, de Almeida, and Fishwick 1991; Tang and Fishwick 1993; The MathWorks 1991). Its structure is given by

$$y(t) = \frac{B(q)}{F(q)} u(t - nk) + \frac{C(q)}{D(q)} e(t)$$

with

$$\begin{aligned} y(t) &: \text{output signal} \\ B(q) &= b_1 + b_2 q^{-1} + ... + b_{nb} q^{-nb} \\ F(q) &= 1 + f_1 q^{-1} + ... + f_{nf} q^{-nf} \\ C(q) &= 1 + c_1 q^{-1} + ... + c_{nc} q^{-nc} \\ D(q) &= 1 + d_1 q^{-1} + ... + d_{nd} q^{-nd} \end{aligned}$$

The numbers nb, nc, nd and nf are the orders of the respective polynomials and q is the shift operator. The number nk is the number of delays from input to output. Figures 7 and 8 show the approximation result and the accompanying error. Successful identification of $y(t)$ depends on how well we guess the

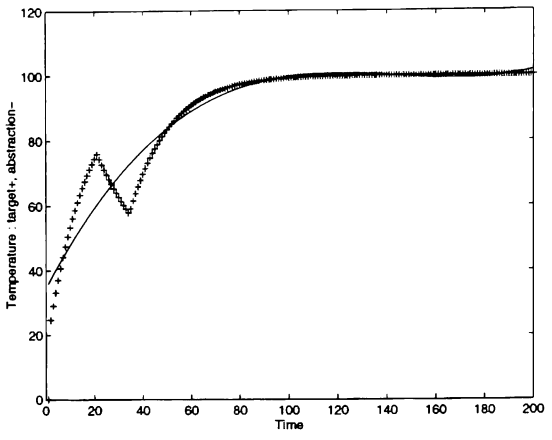

Figure 6: Linear Regression

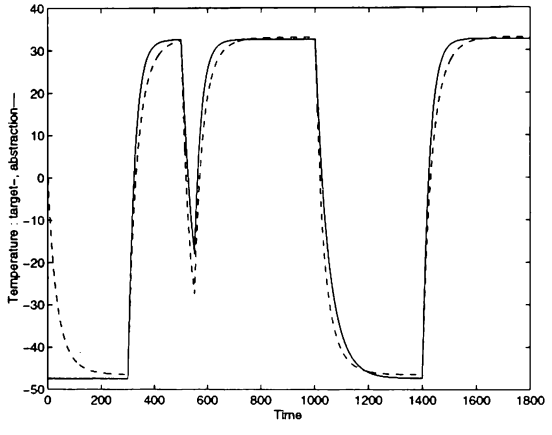

Figure 7: Box-Jenkins Method

values of nb, nc, nd, nf and nk. Heuristics and "expert rules," if available, aid us in choosing parameters. For example, too large a value for a parameter results in computational difficulties to generate $y(t)$, while too small a value results in a rough estimation. We often had to tune parameters by hand in order to get a good approximation.

4 EXAMPLE II: HEMATOPOIESIS MODEL

Though the abstraction methods discussed so far were good at linear system abstraction, non-linear system abstraction involves more complex behavior mapping. In this section, we show how these abstraction methods perform under non-linear conditions.

Our model deals with the regulation of hematopoiesis, the formation of blood cell elements in the body. Hematopoiesis is the process of blood creation in the body. White and red blood cells are produced in bone marrow. From the marrow they enter the blood circulatory system. As the oxygen level decreases in the body, there is a feedback back to the bone marrow–which produces more cells.

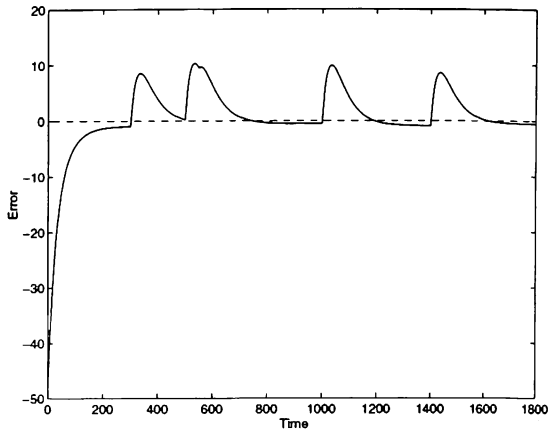

Figure 8: Abstraction Error in Box-Jenkins Method

Mackey and Glass (Mackey and Glass 1977) provide a delay model for hematopoieses of the following form:

$$\frac{dP(t)}{dt} = \frac{\lambda\theta^m P(t-T)}{\theta^m + P^m(t-T)} - gP(t)$$

where, λ : the flux of cells into the blood stream, P(t) : the concentration of cells(the population species) in the circulating blood ($cells/mm^3$), g : day^{-1}, cell loss rate per day and T : maturation delay.

We use λ as an input. Depending on the maturation delay T, we can generate different solutions. In a lower maturation delay, the system shows periodic behaviors, but, as the delay moves upward, nonperiod trajectories appear. Figure 9 shows a nonperiodic trajectory when the delay is 20.

Since we are interested in abstracting the time dependent behavior of cell concentration in the circulating blood, we will restrict out experiments to dynamic-behavioral abstractions. Also, to see how abstraction techniques perform under heavy nonperiodic and nonlinearity, we choose maturation delay 20. Now, the dynamic-behavioral abstraction of hematopoiesis model is to approximate equation (1) with a discrete model of the form where f is a nonlinear function to be estimated with order na:

$$P(t) = f(P(t-1), ..., P(t-na))$$

Small sampling period for the discretization makes the order of the discrete model very high due to the long dependence of $P(t)$ on $P(t-20)$, which results in numerical difficulties to compute the optimal function of f. Therefore, increasing sampling period is needed as long as the discretization is not too

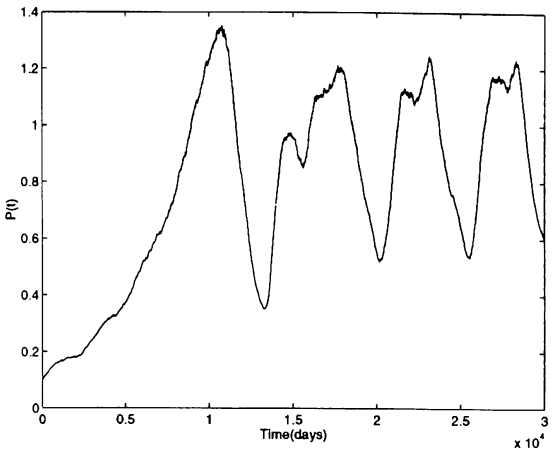

Figure 9: Cell Concentration vs. Time for Delay T = 20 Days

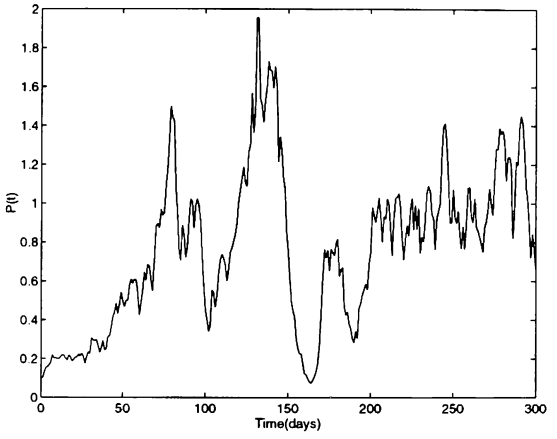

Figure 10: Hematopoiesis Model for Delay = 20 Days with Increased Sampling Period: Abstraction Target

rough. Figure 10 displays the time trajectory for the total concentration of blood cells when the sampling period is increased by 100, which introduces more nonlinearity and instability. We choose Figure 10 as the abstraction target and use λ for input.

ADALINE was developed by Widrow and Hoff (Widrow and Sterns 1985). Their neural network model differs from the perceptronq in that ADALINE neurons have a linear transfer function. The ADALINE network also enables the Widrow-Hoff learning rule, known as the Least Mean Square (LMS) rule, to adjust weights and biases according to the magnitude of errors.

The ADALINE neural network for the hematopoiesis model performs abstraction as shown in Figure 11. An ADALINE neural network takes initial weights and biases, an input signal and a target signal, and then filters the signal adaptively based on

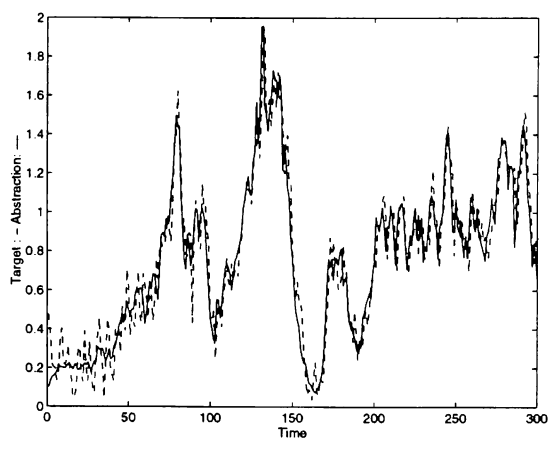

Figure 11: ADALINE Network for Hematopoiesis Model

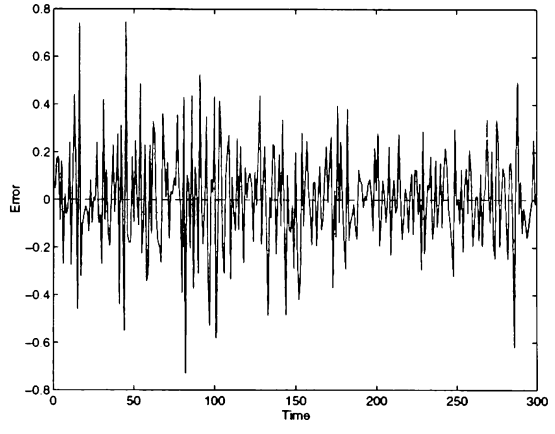

Figure 12: Abstraction Error in ADALINE Network

input delay and learning rate parameters. In most cases, input delay can be guessed by the modeled system itself. For hematopoiesis model, we know an output at time t is determined by 20 most recent inputs, which could be inferred by the the delay-differential equation (1). A proper learning rate is determined by repetitive trials until a good fit is achieved with fewer learning-stage perturbations.

5 CONCLUSIONS

We have presented a new taxonomy for model abstractions in dynamic systems. The taxonomy of abstraction types, with multimodeling at the top level, is constructed by *model engineering* perspective: when a system is first being developed, one should use structural abstraction to organize the whole system hierarchically with simple system types, and then graduate to more complex model types. Below the structural abstraction, each component is black-box

with no detailed internal structure. Behavioral abstraction is used to represent those black-boxes by approximating the behavior of the system components. By combining structural and behavioral abstraction together, each level of abstraction is independent from the lower abstraction levels, so a level can be executed apart from the rest of the hierarchy. These two concepts: structural and behavioral abstraction are blended together to form a comprehensive taxonomy. In addition to the taxonomy, we discussed several abstraction methods according to the categories they belong to and showed how they perform in linear and nonlinear system abstractions. We felt it important to provide both linear and nonlinear models since one technique may fare well for one type of system and then poorly on the other.

Given that we have developed this taxonomy, a good question is "What to do with it?" We are developing a system called MOOSE (Fishwick 1996b), standing for multimodeling object oriented simulation environment, in which the taxonomy is to be applied. MOOSE models are constructed using a graphical user interface which begins with the user specifying an object oriented class hierarchy. This procedure takes advantage of structural abstraction. For exploiting behavioral model abstraction, our current plans are to provide two or three basic techniques and allow the user to choose which they would like. Moreover, we are developing a semi-automated approach to developing behavioral abstractions of multimodel components which can benefit most from the computational gain afforded by not having to simulate at the lowest level.

ACKNOWLEDGMENTS

We would like to acknowledge the following funding sources which have contributed towards our study of modeling and implementation of a multimodeling simulation environment: (1) Rome Laboratory, Griffiss Air Force Base, New York under contract F30602-95-C-0267 and grant F30602-95-1-0031; (2) Department of the Interior under grant 14-45-0009-1544-154 and the (3) National Science Foundation Engineering Research Center (ERC) in Particle Science and Technology at the University of Florida (with Industrial Partners of the ERC) under grant EEC-94-02989.

REFERENCES

Ahang, Q. and A. Benveniste. 1992. Wavelet Networks. *IEEE Transactions on Neural Networks 3*(6).

Barron, A. R. 1989. Statistical Properties of Artificial Neural Networks. In *Proceedings of the 28th IEEE Conference on Decision and Control*, pp. 280–285.

Berzins, V., M. Gray, and D. Naumann. 1986. Abstraction-Based Software Development. *Communications of the ACM 29*(5), 402–415.

Booch, G. 1991. *Object Oriented Design with Applications.* The Benjamin/Cummings Publishing Company, Inc.

Cynbenko, G. 1989. Approximation by Superposition of a Sigmoidal Function. *Mathematics of Control, Signals and Systems 2*, 303–314.

Fishwick, P. A. 1987. A Taxonomy for Process Abstraction in Simulation Modeling. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, Volume 1, Alexandria, Virginia, pp. 144 – 151.

Fishwick, P. A. 1988. The Role of Process Abstraction in Simulation. *IEEE Transactions on Systems, Man and Cybernetics 18*(1), 18 – 39.

Fishwick, P. A. 1989. Abstraction Level Traversal in Hierarchical Modeling. In B. P. Zeigler, M. Elzas, and T. Oren (Eds.), *Modelling and Simulation Methodology: Knowledge Systems Paradigms*, pp. 393 – 429. Elsevier North Holland.

Fishwick, P. A. 1991. Heterogeneous Decomposition and Coupling for Combined Modeling. In *Proceedings of the 1991 Winter Simulation Conference*, Phoenix, AZ, pp. 1199 – 1208.

Fishwick, P. A. 1993. A Simulation Environment for Multimodeling. *Discrete Event Dynamic Systems: Theory and Applications 3*, 151–171.

Fishwick, P. A. 1995. *Simulation Model Design and Execution: Building Digital Worlds.* Prentice Hall.

Fishwick, P. A. 1996a. A Taxonomy for Simulation Modeling Based on a Computational Framework. *IIE Transactions on IE Research.* Revised and re-submitted May 1996.

Fishwick, P. A. 1996b. Extending Object Oriented Design for Physical Modeling. *ACM Transactions on Modeling and Computer Simulation.* (submitted for review).

Fishwick, P. A. 1996c. Toward a Convergence of Systems and Software Engineering. *IEEE Transactions on Systems, Man and Cybernetics.* Submitted May 1996.

Fishwick, P. A. and K. Lee. 1996. Two Methods for Exploiting Abstraction in Systems. In *Proceedings of the AI, Simulation and Planning Conference in High Autonomous Systems Conference*, pp. 257–264.

Fishwick, P. A., H. Narayanan, J. Sticklen, and A. Bonarini. 1994. Multimodel Approaches

for System Reasoning and Simulation. *IEEE Transactions on Systems, Man and Cybernetics 24*(10), 1433–1449.

Fishwick, P. A. and B. P. Zeigler. 1992. A Multimodel Methodology for Qualitative Model Engineering. *ACM Transactions on Modeling and Computer Simulation 2*(1), 52–81.

Johansson, R. 1993. *System Modeling and Identification.* Prentice Hall, Englewood Cliffs, New Jersey.

Law, A. M. and D. W. Kelton. 1991. *Simulation Modeling and Analysis.* McGraw-Hill.

Ljung, L. and T. Soderstrom. 1983. *Theory and Practice of Recursive Identification.* MIT Press, Cambridge, Mass.

Mackey, M. and L. Glass. 1977. Oscillations and Chaos in Physiological Control Systems. *Science 197*, 287–289.

Miller, V. T. and P. A. Fishwick. 1992. Heterogeneous hierarchical models. In *Proceedings of the Conference on Artificial Intelligence X: Knowledge Based Systems*, Orlando, FL. SPIE.

Mills, P., M. Tade, and A. Zomaya. 1995. Identification and Control Using a Hybrid Reinforcement Learning System. *International Journal in Computer Simulation 5*, 109–126.

Tan, K., Y. Li, D. Murray-Smith, and K. Sharman. 1995. System Identification and Linearization Using Genetic Algorithms with Simulated Annealing. *Proceedings of the First IEE/IEEE International Conference on GA in Engineering Systems: Innovations and Applications*, 164–169.

Tang, Z., C. de Almeida, and P. A. Fishwick. 1991. Time Series Forecasting using Neural Networks vs. Box-Jenkins Methodology. *Simulation 57*(5), 303–310.

Tang, Z. and P. A. Fishwick. 1993. Feed-Forward Neural Nets as Models for Time Series Forecasting. *ORSA Journal of Computing 5*(4), 374–386.

The MathWorks, I. 1991. *System Identification Toolbox.* The MathWorks, Inc.

Widrow, B. and S. Sterns. 1985. *Adaptive Signal Processing.* Prentice-Hall.

Zeigler, B. P. 1972. Towards a Formal Theory of Modeling and Simulation: Structure Preserving Morphisms. *Journal of the Association for Computing Machinery 19*(4), 742 – 764.

Zeigler, B. P. 1990. *Object Oriented Simulation with Hierarchical, Modular Models: Intelligent Agents and Endomorphic Systems.* Academic Press.

AUTHOR BIOGRAPHIES

PAUL A. FISHWICK is an Associate Professor in the Department of Computer and Information Science and Engineering at the University of Florida. He received the PhD in Computer and Information Science from the University of Pennsylvania in 1986. He also has six years of industrial/government production and research experience working at Newport News Shipbuilding and Dry Dock Co. (doing CAD/CAM parts definition research) and at NASA Langley Research Center (studying engineering data base models for structural engineering). His research interests are in computer simulation modeling and analysis methods for complex systems. He is a senior member of the IEEE and the Society for Computer Simulation. He is also a member of the IEEE Society for Systems, Man and Cybernetics, ACM and AAAI. Dr. Fishwick founded the `comp.simulation` Internet news group (Simulation Digest) in 1987. He has chaired workshops and conferences in the area of computer simulation, and will serve as General Chair of the 2000 Winter Simulation Conference. He was chairman of the IEEE Computer Society technical committee on simulation (TCSIM) for two years (1988-1990) and he is on the editorial boards of several journals including the *ACM Transactions on Modeling and Computer Simulation*, *IEEE Transactions on Systems, Man and Cybernetics*, *The Transactions of the Society for Computer Simulation*, *International Journal of Computer Simulation*, and the *Journal of Systems Engineering.* Dr. Fishwick's WWW home page is `http: //www.cise.ufl.edu /~fishw ick` and his E-mail address is `fishw ick@ cise.ufl.edu`.

KANGSUN LEE received the B.S and M.S degree in Computer Science from Ewha Womans University, Korea in 1992 and 1994, respectively. She is currently a Ph.D student in the Computer and Information Sciences and Engineering department at the University of Florida, Gainesville. Her research interests are in Modeling methodology, Abstraction techniques and simulation. Kangsun Lee's WWW home page is `http://www.cise.ufl.edu/~kslee` and her E-mail address is `kslee@cise.ufl.edu`