A semi-automated method for dynamic model abstraction

Kangsun Lee and Paul A. Fishwick

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

ABSTRACT

As complex models are used in practice, modelers require efficient ways of abstracting their models. Through the use of hierarchy, we are able to simplify and organize the complex system. The problem with the hierarchical modeling is that system components in each level are dependent on the next-lowest level so that we are unable to run each level independently. We present a way to augment hierarchical modeling where abstraction can take place on two fronts: structural and behavioral. Our approach is to use structural abstraction in order to organize the system hierarchically, and then apply behavioral abstraction to each level in order to approximate lower level's behavior so that it can be executed independently. The proposed abstraction method is done by semi-automatic way and gives advantages to view and analyze complex systems at different levels of abstraction.

Keywords: Abstraction, Multimodeling, Object-Oriented Modeling, System Identification

1. INTRODUCTION

Real world dynamic systems involve a large number of variables and interconnections, which sometimes make the modeling process untractable. Model abstraction is a method for reducing the complexity of a simulation model while maintaining the validity of the simulation results with respect to the question that the simulation is being used to address.[1] Computational efficiency and representational economy are main reasons of using abstract models in simulation[2–4] and well as in programming languages.[5–7] Much recent research has recognized the need for multiple levels of abstraction,[3,8,9] good representations and ability to change from one representation to another for efficient problem solving. Use of abstraction hierarchy has been standard answer to this problem.[9–12] While the hierarchical approach is sound for well-structured models defined in terms of state space functions and set-theoretic components, selection of system components in each level are dependent on the next-lowest level, since actual functional semantics are contained in the lowest level. This means we are unable to execute each level independently.

We propose an abstraction method which can better handle multiple levels of abstraction within hierarchical modeling framework. Our method is to construct abstraction hierarchy first and then make each level an independent structure by approximating its lower levels behaviors so that each level can be executed and analyzed apart from the rest of hierarchy. We define our system abstraction to be one of two types: *structural* or *behavioral*. Structural abstraction is a process of organizing the system hierarchically using refinement and homomorphism. Refinement is the process of refining a model to more detailed models of the same type (homogeneous refinement) or different types (heterogeneous refinement), while homomorphism is a mapping that preserves the behavior of the lower-level system under the set mappings.[12] In structural abstraction, one constructs an abstraction hierarchy with simple model types at first, refining them with more complex model types later. Structural abstraction corresponds to this iterative procedure[13–15] and resulting hierarchy can be used to provide multiple abstractions for the system. After creating the hierarchy, we may want to isolate abstraction levels, so a level can be executed alone. This is where the behavioral approaches are employed. Behavioral abstraction focuses only on behavioral equivalence without structural preservation. Below the structural abstraction, each component is black-box with no detailed internal structure. Behavioral abstraction is used to represent the black-box by approximating the behavior of lower level system components. By combining structural and behavioral abstraction together, each level of abstraction is

Other author information:
P.A.F.:Email: fishwick@cise.ufl.edu; Telephone and Fax: 352-392-1414; WWW home page: http://www.cise.ufl.edu/~ fishwick
K.L.: Email: kslee@cise.ufl.edu; Telephone: 352-392-1435; WWW home page: http://www.cise.ufl.edu/~ kslee

independent from the lower abstraction levels, so a level can be executed apart from the rest of the hierarchy. In depth discussions of each abstraction technique follow in the subsequent sections.

This paper is organized as follows : In section 2, we present a system abstraction method after examining existing abstraction techniques. Section 3 shows how our abstraction method works in an actual example and then we conclude this paper with future works to be achieved.

2. DYNAMIC MODEL ABSTRACTION

Models must be multi-layered so that different abstraction levels of the model respond to different needs of the analyst.[11,3,9] The important issue for the multi-layer modeling is that *how to organize a series of models in a way such that it facilitates multiple resolutions of the system?*. We follow the *multimodeling* methodology[16–20] to organize the system for this purpose. Multimodeling is a modeling process in which we model a system at multiple levels of abstraction. It provides a way of structuring a different model types together under one framework so that each type performs its part, and the behavior is preserved as levels are mapped.[4,21,22] Since it has different model types together under one structure, unlike the other homogeneous-structural abstraction, where dynamical systems are abstracted with only one model type and refined with the same model type, multimodeling can employ a number of abstraction perspectives and accommodate a larger variety of questions. Detailed discussion on multimodeling and examples of modeling techniques can be found in Refs. 16–20.

While the multimodeling approach is sound for well-structured models defined in terms of state space functions and set-theoretic components, selecting system components in each level is dependent on the next-lowest level due to hierarchical structure. This implies that we are unable to *simulate* each level *independently*. It is possible to obtain *output* for any abstraction level but, nevertheless, the system model must be executed at the lowest level of the hierarchy, since there is where we find the actual functional semantics associated with the model. A new definition and methodology is needed to better handle abstraction of systems and components. This is where the behavioral abstraction approaches are employed. By incorporating behavioral abstraction approaches into multimodeling methodology, abstraction in multimodeling allows each level to be understood independently of the others, so that discarding all the abstractions below, any given level will still result in a complete behavioral description of a system.[13] The components are "black boxes" with no detailed internal structure. Behavior is described as a set of input-output pairs defining a black box. We have two approaches for specifying system behavior:

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

Though the static and dynamic approaches describe different allowable behaviors of the same phenomenon, abstraction techniques for the dynamic approach can be applied to static approach too. Therefore, we'll focus on dynamical behavioral abstraction for illustrating abstraction techniques we'll use.

We denote the output of the dynamical system at time t by $y(t)$ and the input by $u(t)$. The data, defining system behavior, are assumed to be collected in discrete time. At time t we have the data set

$$Z^t = y(1), u(1), ..., y(t), u(t) \tag{1}$$

A model of a dynamical system can be seen as a mapping from past data Z^{t-1} to the next output $y(t)$:

$$\hat{y}(t) = \hat{g}(Z^{t-1}) \tag{2}$$

$\hat{y}$ represents the predicted value whereas $y(t)$ is the exact value. The problem of dynamical behavioral abstraction is to find a mapping $\hat{g}_t$ that gives good prediction in (2) using the information in a data record Z^t. Parametric model estimation in system identification[23,24] is the theory and art of building *mathematical model* of g_t. 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. A Parameter estimation procedure provides a search through parameter space, effectively, to achieve a close-to optimal 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). For the detailed structure of these models, see 23,25.

Neural networks have been established as a general approximation tool for fitting models from input/output data.[26–29] From the system identification perspective, a neural network may be considered as another candidate model structure.[24,30] 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. These produce the values that should be matched to the variable $y(t)$ in the observed pair $(y(t), u(t))$. Thus, behavioral abstraction by neural networks is to find $g(t, \theta, u(t))$, where θ represents the weights of the linear combinations involved in network structure. Backpropagation, recurrent and temporal neural networks have been shown to be applicable to system identification.[31,13,32] 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.[33]

By integrating multimodeling techniques and behavioral techniques, we propose a method for dynamic model abstraction. An execution will be shown in detail in the following section.

```
Do structural abstraction using Multimodeling
While ( there's a need to abstract a system further) do
begin
        For the entire multimodeling structure
        begin
            Check if a model component is within user interest;
            If not // irrelevant to user interest
              Box it for behavioral abstraction;
        end
        Apply one of behavioral abstraction techniques to the box
end
```

After structural abstraction, the user specifies the parts where behavioral abstraction is needed for a question he wants to ask, and then a search process is started to find out actual model components relevant to user interest. During behavioral abstraction, user intervention is needed to find good parameter values upon which a behavioral abstraction technique is executed. For example, order of equation is needed for parametric model estimation technique and number of hidden layer for neural network.

3. FULTON : steamship modeling

Consider a steam-powered propulsion ship model, named FULTON*, as shown in Fig. 1. In our system, furnace heats water in boiler: when the fuel valve is OPEN,fuel flows and furnace heats; when fuel valve is CLOSED, no fuel flows and furnace is at ambient temperature. Heat from the fuel is added to the water to form high-pressure steam, which performs work by expanding against the turbine blades. To be compressed at low pressure, the steam is condensed in a condenser, which requires the exhaust of heat from the water. There is a heat exchange is accomplished by circulating sea water and the steam is condensed again into liquid, at which point it can be pumped back to the boiler.[8,34]

3.1. Structural Abstraction of FULTON

Fig. 2 and Fig. 3 shows structural abstraction for the FULTON system. Since FULTON has 4 components whose function is distinct from others, we start with 4 Functional Block Models, $L_1, ..., L_4$.

Low level continuous models of BLOCK $L_2, ..., L_4$ are defined by the laws of thermodynamics and energy conservation.[34]

1. (L_1) BOILER ASSEMBLY : defined as FSM(Finite State Machine) in Fig. 3

*We name it "FULTON", after Robert Fulton, who designed and oversaw construction of the first steam-powered warship, the "USS FULTON", for the US Navy.

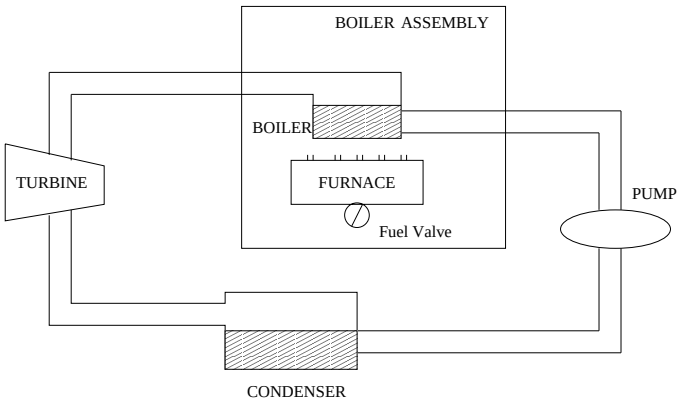


Figure 1. High-level view of a shipboard steam-powered propulsion plant

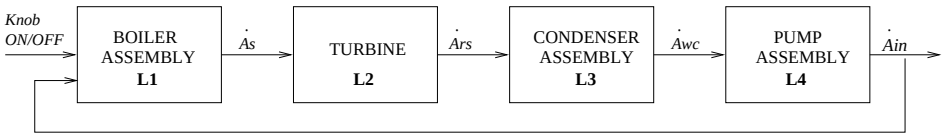


Figure 2. Top level of structurally abstracted model for FULTON

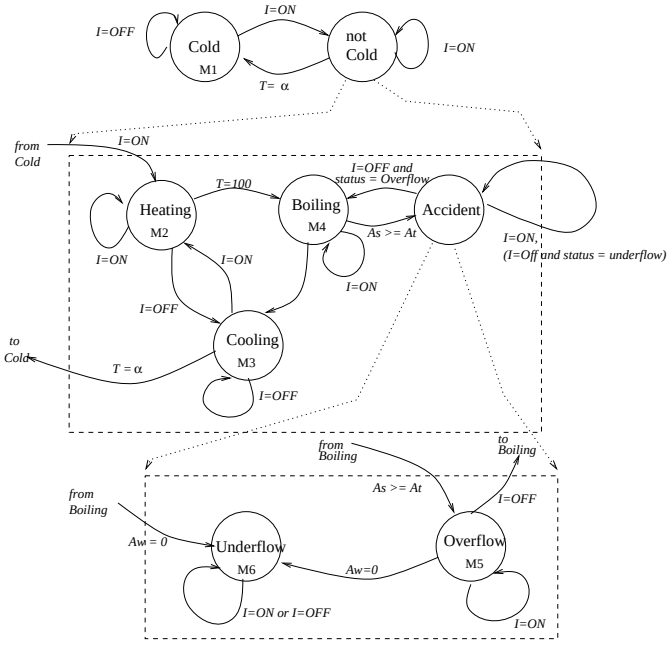


Figure 3. Structural abstraction of L_1

Table 1. Question-answering using the multimodel from structural abstraction

Question	Model	Answer
If the system is warm, how can it become cold?	FSM-1	System becomes cold when the water temperature equals the ambient temperature
Why did the water start to boil?	FSM-2	The water was being heated and the temperature reached 100∘C which caused the water to boil

2. (L_2) TURBINE : $\dot{A}_{rs} = \dot{A}_s * k_l$
3. (L_3) CONDENSER ASSEMBLY : $\dot{A}_{wc} = \dot{A}_{rs} - \dot{A}_{wp}$
4. (L_4) PUMP ASSEMBLY : $\dot{A}_{in} = \dot{A}_{wc} * k_p$

where, A_s : Amount of steam into the turbine, A_{rs} : Amount of steam remained after being exhausted in turbine, k_l : steam loss rate in turbine, A_{wc} : Amount of water in condenser assembly, A_{in} : Amount of water increased in boiler assembly by pumping water from pump assembly.

Since BOILER ASSEMBLY has several states with transitions among them, we model it by FSM as shown in Fig. 3. The top most FSM in Fig. 3 shows a two-state FSM with input. We label this FSM-1. The second label includes a detailed representation of state "Not Cold". By combining this new FSM with FSM-1, we create FSM-2 (a complete model of the boiling process). FSM-3 is constructed similarly. Each state is modeled as continuous models, $M_1, ..., M_6$: The continuous models contained within states *heating* and *cooling* are made by combining Newton's Law with the capacitance law .[12]

1. (M_1) COLD : $T = \alpha, \dot{A}_w = \dot{A}_{in}, \dot{A}_f = 0, \dot{A}_s = 0.$
2. (M_2) HEATING : $\dot{T} = k_1(100 - T), \dot{A}_w = \dot{A}_{in}, \dot{A}_f = 0, \dot{A}_s = 0.$
3. (M_3) COOLING :$\dot{T} = k_2(100 - T), \dot{A}_w = \dot{A}_{in}, \dot{A}_f = 0, \dot{A}_s = 0.$
4. (M_4) BOILING :T=100, $\dot{A}_w = -k_4 + \dot{A}_{in}, \dot{A}_f = k_5, \dot{A}_s = \dot{A}_f - \dot{A}_w.$
5. (M_5) OVERFLOW : same as BOILING with constraint $A_f = A_t$
6. (M_6) UNDERFLOW : T = $undefined$, $A_w = A_f = 0.$

where, T (Temperature), A_w (amount of water), and A_f (amount of foam on the top of water), A_t (maximum steam amount of boiler), A_s (amount of steam), I indicates fuel valve's position and α is the ambient temperature of the water and k_i (constants).

Now that we have specified 3 FSM levels, and one functional block level, we can use these models to answer questions about the system. A transition in FSM may sometimes refer to a more detailed state specification than is available at the current level of abstraction, according to the questions. We present sample questions and answers that can be practically derived from the presented multimodel. Given an arbitrary questions as shown in Table 1, a model is chosen on which to base the answer. An actual natural language processing system does not currently exist, therefore processing implied by Table 1 is idealized. By examining this table, we found that structural abstraction is useful not only for model building but also for facilitating question-answering using a variety of abstraction.

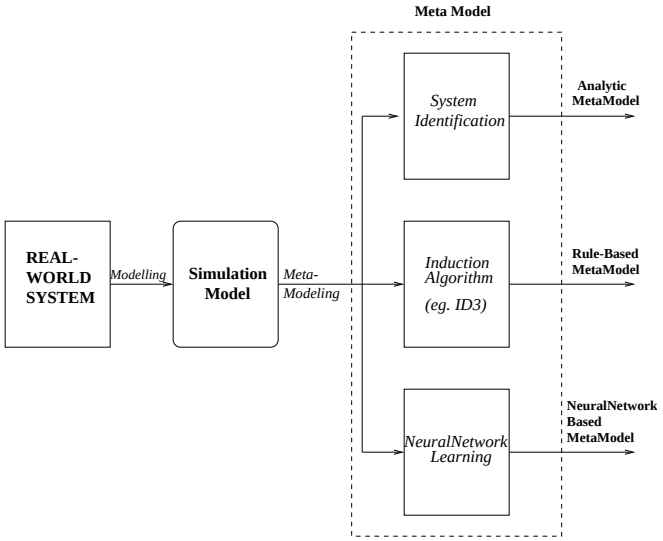


Figure 4. The process of behavioral abstraction

3.2. Behavioral Abstraction of FULTON

Behavioral abstraction is to build a simulation metamodel,[35–37] that is a model of simulation model. We start with set of input-out data pair from simulation results as shown in Fig. 4. With this *priori* knowledge, the method of behavioral abstraction is to find how the inputs relate to outputs by applying one of abstraction techniques. We showed several techniques to build a simulation metamodel in section 2. A mathematical metamodel is generated by applying a parametric estimation technique, while neural network or wavelet network models can be generated by training neurons or wavelons. In the following section, we'll see how one can earn benefits from behaviorally abstracted models for given questions.

3.2.1. Question 1 : *How are the four system components interacting?*

To focus on interaction between components at high level, we don't need detailed model for each. Since *Boiler* component is refined into FSM and low level continuous functional block models, we apply behavioral abstraction technique and replace it with neural network BLOCK model as shown in Fig. 5. The shaded box represents the behaviorally abstracted BLOCK model. Newly generated model should have the outputs which the original model generates based on the input, but has some accuracy loss. It runs with time-saving, since we don't have to go down the abstraction hierarchy in order to find the model's functional semantics. We experimented with ADALINE(ADAptive LINear Element) neural network using Matlab's neural network toolbox.[38] Fig. 6 shows how closely the abstracted model performs for this given question. The solid line represents the actual output from the original and dotted line represents the output from further abstracted model by applying the behavioral abstraction technique.

3.2.2. Question 2 : *What is the behavior trajectory of the Boiler over time?*

To refine a certain component, details of other components could be omitted as long as the resulting model provides the same outputs with a certain accuracy confidence. We search model components irrelevant to the question, and box it for behavioral abstraction. To answer question 2 efficiently, we abstract other components except boiler into one unit as shown in Fig. 7. The turbine, condenser and pump components are abstracted into one black-box model, but still provide the input to the boiler component with an allowable inaccuracy. We experimented this by ARX model using Matlab's system identification toolbox.[23] Result from this newly abstracted model is good enough to see almost overlapped line as shown in Fig. 8.

3.2.3. Question 3: *What is the work trajectory according to fuel provided?*

The *work* trajectory is the output of the FULTON model. Since the question only concerns the behavior of the entire FULTON model without structural details, all components can be fully abstracted into one BLOCK model. Fig. 9 shows this model. Abstraction results and loss of accuracy are shown in Fig. 10.

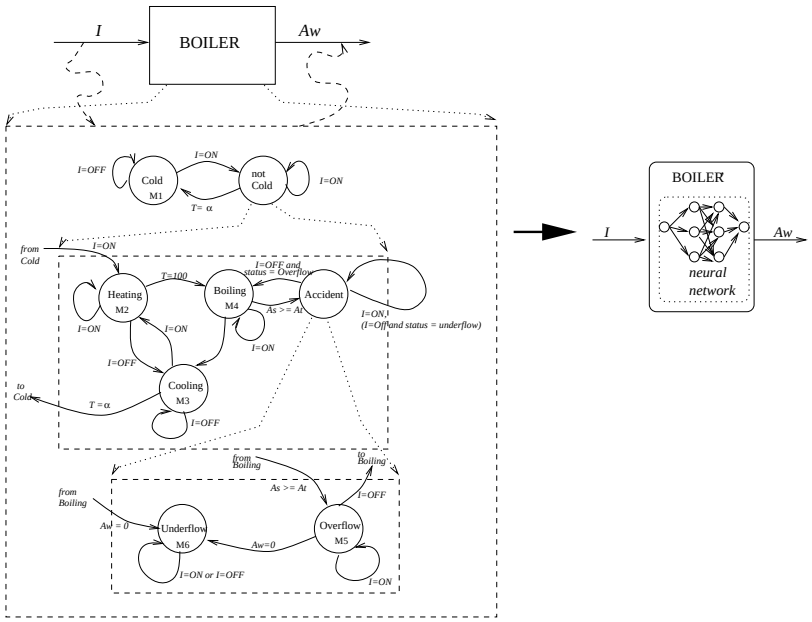
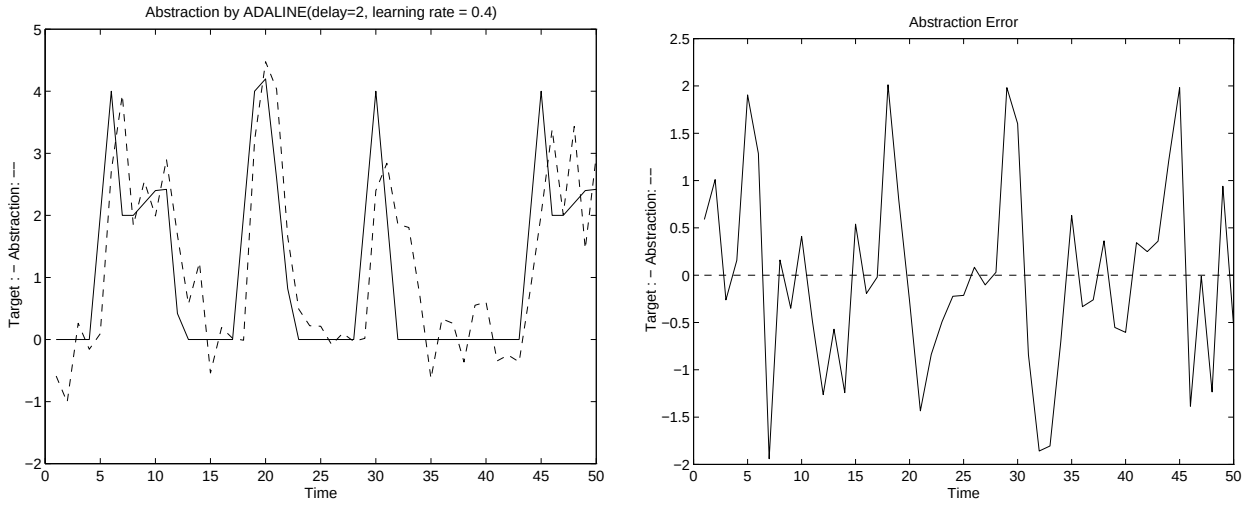


Figure 5. Behavioral Abstraction for question 1



(a) Result from Behavioral abstraction

(b) Abstraction Error

Figure 6. Behavioral Abstraction Result for question 1

4. CONCLUSIONS AND FUTURE WORK

We have presented a dynamic model abstraction method. Our approach is to use structural abstraction to organize the whole system hierarchically with simple system types first, and then graduate to more complex model types. The multimodeling method is used for this purpose. Below the structural abstraction, each component is black-box with no detailed internal structure. Behavioral abstraction is used to represent those black-boxes which approximate the behavior of the system components with various abstraction techniques discussed in section 2. 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 hierarchy. We showed how this method can provide benefits

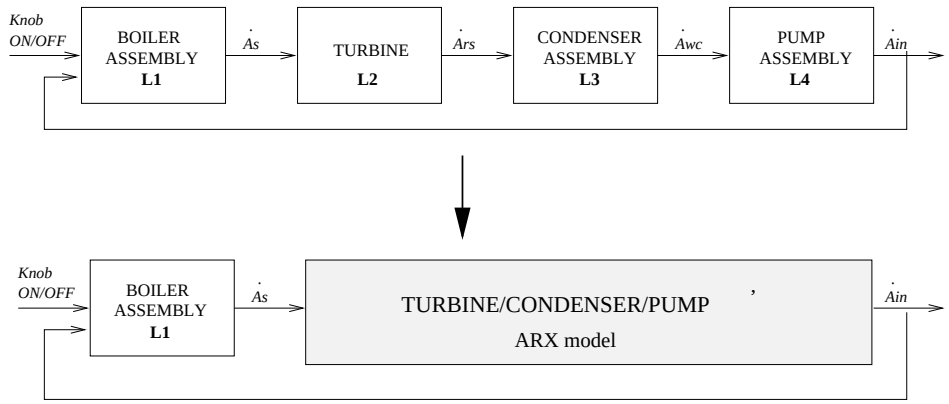
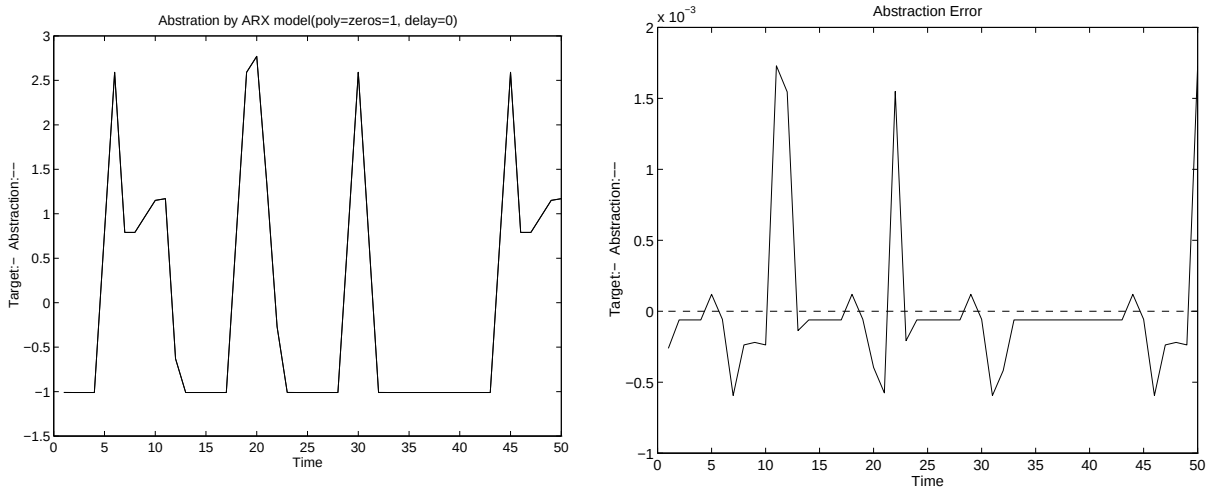


Figure 7. Behavioral Abstraction for question 2



(a) Result from Behavioral abstraction

(b) Abstraction Error

Figure 8. Behavioral Abstraction Result for question 2

by question-answering examples to the system. Unnecessary details could be efficiently omitted and answers were given with time-saving way by not having to simulate at the lowest level. We're developing MOOSE(Multimodeling Object-Oriented Simulation Environment),[39] where the proposed abstraction method is being implemented. MOOSE models are constructed using a graphical user interface which begins with the user specifying an object oriented class hierarchy by following OOPM[40](Object-Oriented Physical Modeling). This procedure takes advantage of structural abstraction. For exploiting behavioral model abstraction, we provide three basic techniques(parametric estimation, neural network, wavelet network) and allow the user to choose which they would like.

Having proposed framework to organize models according to abstraction level, our question is how we can select an optimal abstraction level under a certain time and accuracy constraints. For applications with significant time constraints, we may need to use simple models that are computationally less complex even at the expense of reduced accuracy. The optimal model for this application is the model that maximizes the tradeoff between model accuracy and model cost. We're studying a method that selects, from the candidate models, a model that is accurate and computable within a time constraint and planning to add this capability to MOOSE.

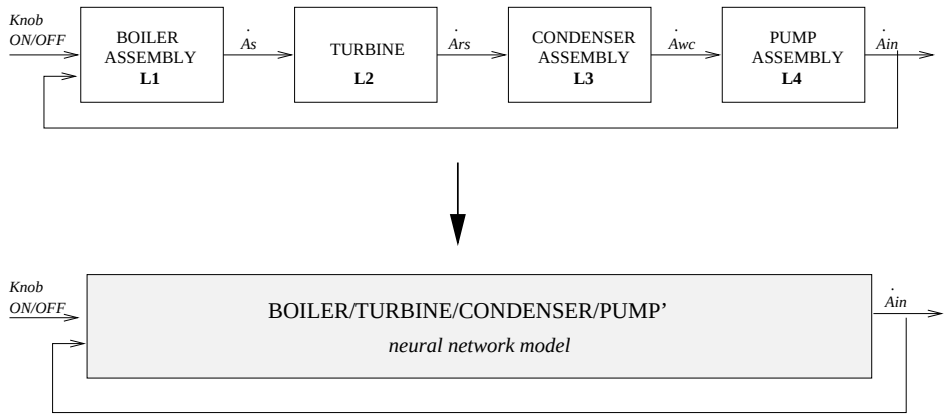
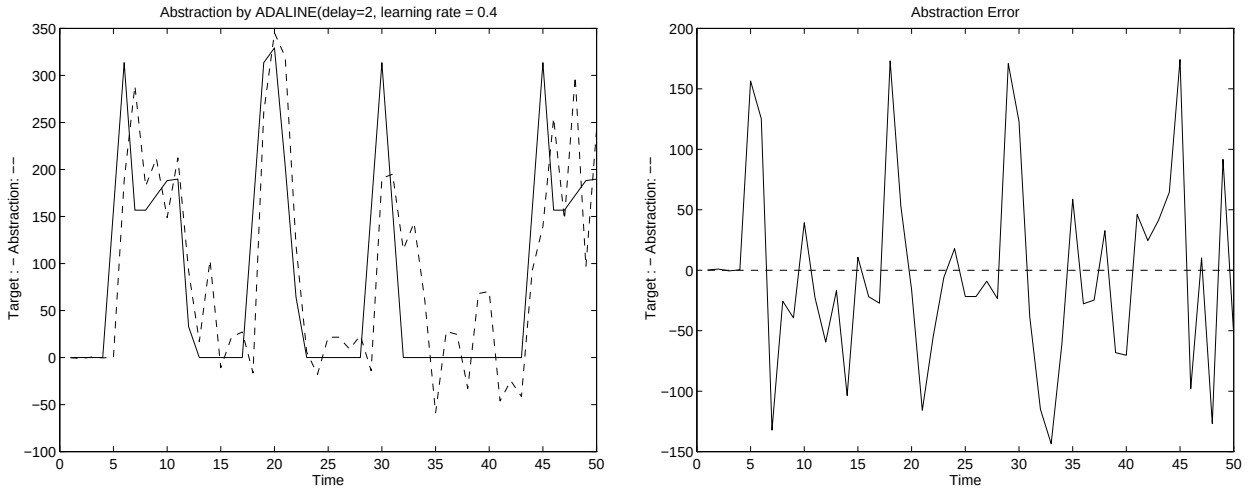


Figure 9. Behavioral abstraction for question 3



(a) Result from behavioral abstraction

(b) Abstraction error

Figure 10. Behavioral abstraction result for question 3

ACKNOWLEDGEMENTS

We would like to thank the following funding sources that have contributed towards our study of modeling and implementation of a multimodeling simulation environment for analysis and planning: (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

1. F. K. Frantz, "A Taxonomy of Model Abstraction Techniques," *Proceedings of the 1995 Winter Simulation Conference* , pp. 1413–1420, 1995.
2. P. A. Fishwick, "A Taxonomy for Process Abstraction in Simulation Modeling," in *IEEE International Conference on Systems, Man and Cybernetics*, vol. 1, pp. 144 – 151, (Alexandria, Virginia), October 1987.

3. P. A. Fishwick, "Abstraction Level Traversal in Hierarchical Modeling," in *Modelling and Simulation Methodology: Knowledge Systems Paradigms*, B. P. Zeigler, M. Elzas, and T. Oren, eds., pp. 393 – 429, Elsevier North Holland, 1989.
4. B. P. Zeigler, "Towards a Formal Theory of Modelling and Simulation: Structure Preserving Morphisms," *Journal of the Association for Computing Machinery* **19**(4), pp. 742 – 764, 1972.
5. M. G. Valdis Berzins and D. Naumann, "Abstraction-Based Software Development," *Communications of the ACM* **29**(5), pp. 402–415, 1986.
6. G. Booch, *Object Oriented Design with applications*, The Benjamin/Cummings Publishing Company, Inc., 1991.
7. R. W. Sebesta, *Concepts of Programming Languages*, The Benjamin/Cummings Publishing Company, Inc., 1992.
8. B. Falkenhainer and K. D. Forbus, "Compositional modeling: finding the right model for the job," *Artificial Intelligence* **51**, pp. 95–143, 1991.
9. P. K. Davis and R. Hillestad, "Aggregation, Disaggregation, and the Challenge of Crossing Levels of Resolution When Designing and Connecting Models," in *Proceedings of AI, Simulation and Planning in High Autonomous S ystems(AIS)*, pp. 180–188, 1993.
10. J. R. Hobbs, "Granularity," *Proceedings of the Seventh National Conference on Artificial Intelligence* , 1985.
11. C.-J. Luh and B. P. Zeigler, "Abstraction Morphisms for Task Planning and Execution," in *Proceedings of AI, Simulation and Planning in High Autonomous Systems(AIS)*, pp. 50–59, 1991.
12. P. A. Fishwick, *Simulation Model Design and Execution: Building Digital Worlds*, Prentice Hall, 1995.
13. P. A. Fishwick and K. Lee, "Two Methods for Exploiting Abstraction in Systems," *AI, Simulation and Planning in High Autonomous Systems* , pp. 257–264, 1996.
14. P. A. Fishwick, "A Taxonomy for Simulation Modeling Based on a Computational Framework," *IIE Transactions on IE Research, Accepted May 1996* , 1996.
15. P. A. Fishwick, "Toward a Convergence of Systems and Software Engineering," *IEEE Transactions on Systems, Man and Cybernetics* , 1996. Submitted May 1996.
16. P. A. Fishwick, "Heterogeneous Decomposition and Coupling for Combined Modeling," in *1991 Winter Simulation Conference*, pp. 1199 – 1208, (Phoenix, AZ), December 1991.
17. P. A. Fishwick and B. P. Zeigler, "A Multimodel Methodology for Qualitative Model Engineering," *ACM Transactions on Modeling and Computer Simulation* **2**(1), pp. 52–81, 1992.
18. P. A. Fishwick, "An Integrated Approach to System Modelling using a Synthesis of Artificial Intelligence, Software Engineering and Simulation Methodologies," *ACM Transactions on Modeling and Computer Simulation* , 1992.
19. P. A. Fishwick, "A Simulation Environment for Multimodeling," *Discrete Event Dynamic Systems: Theory and Applications* **3**, pp. 151–171, 1993.
20. P. A. Fishwick, H. Narayanan, J. Sticklen, and A. Bonarini, "A multimodel approache to reasoning and simulation," *IEEE Transactions on Systems, Man and Cybernetics* (10), pp. 1433–1449, 1994.
21. P. A. Fishwick, "The Role of Process Abstraction in Simulation," *IEEE Transactions on Systems, Man and Cybernetics* **18**, pp. 18 – 39, January/February 1988.
22. B. P. Zeigler, *Object Oriented Simulation with Hierarchical, Modular Models: Intelligent Agents and Endomorphic Systems*, Academic Press, 1990.
23. *System Identification Toolbox*, The MathWorks, Inc., 1991.
24. L. Ljung and T. Soderstrom, *Theory and Practice of Recursive Identification*, MIT Press, Cambridge, Mass, 1983.
25. K.C. Tan, Y. Li, D.J. Murray-Smith and K.C. Sharman, "System Identification and Linearisation Using Genetic Algorithms with Simulated Annealing," *Proc. First IEE/IEEE Int. Conf. on GA in Eng. Syst.: Innovations and Appl.* , pp. 164–169, 1995.
26. G. Cynbenko, "Approximation by Superposition of a Sigmoidal Function," *Mathematics of control, signals and systems* **2**, pp. 303–314, 1989.
27. S. M. Carrol and B. W. Disckinson, "Construction of Neural Nets using the Radon Transform," in *IJCNN*, 1989.
28. Z. Tang, C. de Almeida, and P. A. Fishwick, "Time Series Forecasting using Neural Networks vs. Box-Jenkins Methodology," *Simulation* **57**, pp. 303–310, November 1991.

29. Z. Tang and P. A. Fishwick, "Feed-Forward Neural Nets as Models for Time Series Forecasting," *ORSA Journal of Computing* **5**(4), pp. 374–386, 1993.
30. A. R. Barron, "Statistical Properties of Artificial Neural Networks," *Proceedings of the 28th IEEE Conference on Decision and Control* , pp. 280–285, 1989.
31. D. E. Rumelhart, G. E. Hinton, and R. W.Williams, *Learning Internal Representations by Error Propagation*, In parallel Distributed Processing: Explorations in the Microstructure of Cognition, Cambridge, MA:MIT Press, 1986.
32. P. M. Mills, M. O. Tade, and A. Y. Zomaya, "Identification and Control Using a Hybrid Reinforcement Learning System," *International Journal in Computer Simulation* **5**, pp. 109–126, 1995.
33. Q. Zhang and A. Benveniste, "Wavelet Networks," *IEEE transactions on Neural Networks* **3**(6), 1992.
34. F. J. K. W. Edward Gettys and M. J. Skove, *Physics*, McGraw-Hill, 1989.
35. D. Caughlin, "Model Abstraction Via Solution of A General Inverse Problem to Define a Metamodel," *SES transactions of Computer Simulation, Submitted Sept.* , 1996.
36. H. Pierreval, "A Metamodeling Approach Based on Neural Networks," *International Journal in Computer Simulation* **6**, pp. 365–378, 1996.
37. J. Signrandt and M. Smith, "Nonlinear System Identification using Wavelets," *CSC(Centre for Systems and Control) report* , 1996.
38. *Neural Network Toolbox*, The MathWorks, Inc., 1992.
39. T. G. Robert Cubert and P. A. Fishwick, "MOOSE: architecture of an object-oriented multimodeling," *SPIE Proceedings* , 1997.
40. P. A. Fishwick, "Extending Object-Oriented Design for Physical Modeling," *ACM transactions on Modeling and Computer Simulation, Submitted July* , 1996.