MODELING AND SIMULATION METHODS FOR DESIGN OF ENGINEERING SYSTEMS

Rajarishi Sinha, *Student Member, ASME*
Institute for Complex Engineered Systems
Carnegie Mellon University
Pittsburgh, PA 15213, USA
Email: rsinha@cs.cmu.edu

Vei-Chung Liang, *Student Member, ASME*
Institute for Complex Engineered Systems
Carnegie Mellon University
Pittsburgh, PA 15213, USA
Email: vliang@cs.cmu.edu

Christiaan J.J. Paredis, *Member, ASME*
Institute for Complex Engineered Systems and
Dept. of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213, USA
Email: paredis@cmu.edu

Pradeep K. Khosla, *Member, ASME*
Dept. of Electrical and Computer Engineering and
Institute for Complex Engineered Systems
Carnegie Mellon University
Pittsburgh, PA 15213, USA
Email: pkk@cs.cmu.edu

ABSTRACT

This article presents an overview of the state-of-the art in modeling and simulation, and studies to which extent current simulation technologies can effectively support the design process. For simulation-based design, modeling languages and simulation environments must take into account the special characteristics of the design process. For instance, languages should allow models to be easily updated and extended to accommodate the various analyses performed throughout the design process. Furthermore, the simulation software should be well integrated with the design tools so that designers and analysts with expertise in different domains can effectively collaborate on the design of complex artifacts. This review focuses in particular on modeling for design of multi-disciplinary engineering systems that combine continuous time and discrete time phenomena.

INTRODUCTION

Modeling and simulation enables designers to test whether design specifications are met by using virtual rather than physical experiments. The use of virtual prototypes significantly shortens the design cycle and reduces the cost of design. It further provides the designer with immediate feedback on design decisions which, in turn, promises a more comprehensive exploration of design alternatives and a better performing final design. Simulation is particularly important for the design of multi-disciplinary systems in which components in different disciplines (mechanical, electrical, embedded control, etc.) are tightly coupled to achieve optimal system performance.

This article surveys the current state of the art in modeling and simulation and examines to which extent current simulation technologies support the design of engineering systems.

We limit the scope of the survey by concentrating on system-level modeling. At a systems level, components and sub-systems are considered as black boxes that interact with each other through a discrete interface. In general, such systems can be modeled using differential algebraic equations (DAEs) [1] and/or discrete event systems specifications (DEVS) [2]. We will not consider the systems that require partial differential equations or finite element models to model system components or component interactions.

We further focus our attention on these aspects of modeling and simulation that are particularly important in the context of design. Specifically, we evaluate the current state-of-the-art with respect to model expressiveness, model reuse, integration with design environments, and collaborative modeling.

One of the most basic requirements for simulations in the context of design is that the modeling language be *sufficiently expressive* to model the non-linear, multi-disciplinary, hybrid continuous-discrete phenomena encountered in the design prototypes. Over the years, many modeling and simulation languages have been developed, but only a few of these languages are well suited for modeling of multi-disciplinary systems. The earliest simulation languages, based on CSSL (Continuous System Simulation Language), were procedural and provided a low-level description of a system in terms of ordinary differential equations. From these languages emerged two important developments: declarative (or equation-based) modeling, and object-oriented modeling. Current research

further builds on these developments by moving towards component-based modeling and by providing support for hybrid (mixed continuous-discrete event) systems.

Another requirement is that *simulation models be easy to create and reuse*. Creating high-fidelity simulation models is a complex activity that can be quite time-consuming. Object-oriented languages provide clear advantages with respect to model development, maintenance, and reuse. In addition, to take full advantage of simulation in the context of design, it is necessary to develop a modeling paradigm that is integrated with the design environment, and that provides a simple and intuitive interface that requires a minimum of analysis expertise.

Finally, we address the issue of *collaborative modeling*. Design of complex multi-disciplinary systems requires the expertise of a group of collaborating specialists. Designers with backgrounds in different disciplines collaborate with analysts, manufacturing engineers, marketing specialists, and business managers. To support this collaborative aspect of simulation and design, it is important to carefully document the models, capture their semantics, and make them available in well-organized repositories that fit within the context of the world-wide-web.

MODELING PARADIGMS AND LANGUAGES

Several general-purpose simulation modeling paradigms and languages have been developed. They can be classified according to the following criteria [3]: graph-based versus language-based paradigms, procedural versus declarative models, multi-domain versus single-domain models, continuous versus discrete models, and functional versus object-oriented paradigms.

We will illustrate the differences between the modeling paradigms with the example in Figure 1. It is a simplified servo system with one rotational degree of freedom used to control the read head on a disk drive. This mechatronic system has five major components: a power source, a DC motor, a load that represents the drive head, an encoder that measures the angular velocity of the load and integrates it to output an angular position, and a PWM controller (Pulse Width Modulation [4]) that controls the positioning of the head.

Graph-Based Modeling

Graphs have been used to represent interconnected systems in many different modeling domains [5, 6]**.** In systems modeling, research has focused on three graph-based paradigms: bond graphs, linear graphs, and block diagrams.

Bond graph modeling [7, 8] is based on energy-conserving junctions that connect energy storing or transforming elements through bonds. The bonds represent the power flow between the modeling elements as a product of a flow and effort variable. Elements are connected to each other through 0- and 1-junctions that represent Kirchhoff's current and voltage laws, respectively. Although bond graphs are domain independent, they are not very convenient for the modeling of 3D mechanics [6] or continuous-discrete hybrid systems [9, 10].

In Figure 3, the servo system introduced previously is modeled using bond graphs. Even for this simple system, beginning users may find it counterintuitive that the topology of the bond graph is quite different from the topology of the corresponding physical system. For example, the encoder component in the conceptual model does not map to a particular bond graph element; rather, it is represented as a link between a 1-junction and the controller. Furthermore, the causality of this model is fixed at the time of model creation. This may pose problems for models in which the causality changes dynamically (e.g. at zero velocity for Coulomb friction models). Although bond graph elements are normally linear, some languages such as CAMP-G and SIDOPS+ [11] support nonlinear multi-dimensional bond-graph models that can contain both continuous-time and discrete-time parts.

The second graph-based modeling paradigm builds on linear graph theory. The relationship between physical systems and linear graphs was first recognized by Trent [12] and by Branin [13]. Similar to bond graphs, these linear graphs represent the energy flow through the system, expressed by through and across variables (also called terminal variables). An edge in the linear graph indicates the existence of an energy flow in a system component, while the terminals of the component correspond to the nodes of the graph. For each edge, there is a terminal equation expressing the relation between its terminal variables. One or more edges and associated terminal equations completely define the dynamics of a component. Such *terminal graphs* of individual components can be composed into *system graphs* by merging the nodes between which physical connections exist. Unlike bond graph models, linear graph models do reflect the system topology directly [14, 15]**.** They are domain independent and can be easily extended to model 3D mechanics [15, 16] and hybrid systems [17]. Linear graphs form the underlying representation for the VHDL-AMS simulation language. In Figure 2, edge e_1 represents the power source and connects two electrical nodes *a* and *b*. The DC motor participates in two energy domains: electrical and mechanical. It is represented by two edges e_2 and e_3 in the electrical and mechanical domains, respectively. The load is a purely

mechanical component, and therefore is represented by the single edge e_4 connecting nodes *d* and *e* in the mechanical domain. The signal part of the linear graph is represented by block icons connected to each other and to the energetic part by dashed lines. The encoder measures the angular velocity of the load, and integrates it to obtain the angular position. This value is passed to the controller that outputs a voltage to the DC motor.

The third group of graph-based modeling paradigms is based on block diagrams, as in SimuLink or Easy5. Here, models are specified by connecting inputs and outputs of primitive models such as integrators, multipliers, or adders. Complex systems are modeled by hierarchically configuring lower-level encapsulated models of subsystems.

Typically, the individual models in block diagrams are defined procedurally (see next section). As a result, the solvers cannot break possible algebraic loops, and the user is required to reorganize the system equations manually. In the block diagram model for the servo system, shown in Figure 4, the motor and the load models are combined to avoid an algebraic loop. To make the system solvable, the load and motor inertias are summed and assigned to a single gain block.

Figure 5 illustrates the use of port-based icons to represent the signal flow through the system. Similar block representations have been used for declarative object oriented modeling [18] and modular discrete event modeling [19]. Each component in the conceptual model is represented as a component in the port-based model of Figure 5. The load is represented as a hierarchical composition of an inertia model and a viscous damping model. The nodes correspond to the nodes of the linear graph in Figure 2.

Declarative and Procedural Modeling

Many early simulation languages were based on the Continuous System Simulation Language (CSSL) [20]. They were procedural in nature, meaning that models were defined through *assignments* as is common in most programming languages. Assignments express a dependent variable as a function of independent variables (fixed causality), and have to be evaluated in the order defined by the user. This limits the reuse of procedural models and prohibits symbolic manipulation.

Declarative or equation-based languages, on the other hand, do not impose a fixed causality on the model. In these languages, the model is defined by a set of equations that establishes relations between the states, their derivatives, and time. The simulation engine is responsible for converting these equations into software procedures that can be evaluated by the computer. The advantage of declarative languages is that users do not have to define the mathematical causality of the equations, so that the same model can be used for any causality imposed by other system components.

Several declarative languages have been developed, such as VHDL-AMS, MOSES [21], Smile [22], ObjectMath, SIDOPS+, and Modelica [23]. Some languages like ASCEND [24] take the declarative paradigm even further by allowing models to be non-causal even in their parameters, implying that it is possible to solve for model parameters, given model inputs and outputs.

Many of the declarative languages such as Modelica and VHDL-AMS provide constructs for procedural models as well. This is useful for including discrete event models or embedded control programs that are more conveniently specified as procedures (e.g. the PWM controller in the example).

Discrete and Continuous Simulation

The behavior of multidisciplinary systems is a combination of continuous time physical phenomena and events occurring at discrete space and time coordinates. For high-fidelity simulation of such systems, *hybrid* modeling and simulation is required in which both continuous and discrete event phenomena can be represented.

Many physical phenomena, such as rigid body motion, flow of electric currents, fluid flow, or heat flow, evolve as continuous functions of time and are therefore best modeled by a set of differential algebraic equations (DAEs) [1, 25].

Physical events and digital components, on the other hand, generate outputs at discrete points in time and space; they are best modeled using discrete variables or impulse functions [2, 26, 27]. Examples include rigid body collisions, data buses, and digital controllers. In addition, discrete event simulation is applied to a variety of other disciplines, including logistics, transportation, material handling, and military simulation [28]. A good overview of the principles and industrial applications of discrete-event simulation can be found in [2, 28].

Because mechatronic systems combine both continuous time phenomena and discrete events, they require mixed continuous-discrete models [2, 29, 30]. Several simulation languages, including Modelica and VHDL-AMS, support mixed systems modeling. These models also require advanced solvers that efficiently synchronize between DAE solving and discrete event propagation. Most commercial simulation software packages include this capability now.

Object-Oriented Modeling

The software design methodology of object-oriented programming can be applied to systems modeling as well, with the benefits of simplified model creation and maintenance.

An important principle of object-oriented programming is that of *information hiding or encapsulation*: an object can only be accessed through its public interface, which is independent of the underlying implementation. The same principle can be applied to modeling by making a clear distinction between the physical interactions of an object with its environment (*interface*) and its internal behavior (*implementation*) [31, 32]. A model interface consists of ports that discretize the exchange of energy, mass, or information to a finite number of points on the component's interface. When connecting ports, Kirchhoff's network laws are imposed on the port variables. As for the equations describing the internal behavior of the components, the causality of the energy connections is assigned by the solver. The advantage of encapsulation is that a system can be modeled by composing and connecting the interfaces of its sub-systems, independently of the future implementations of these subsystems [32-34].

In Figure 5, the DC-motor has an interface consisting of two electrical connections and two mechanical connections (stator and rotor). The actual equations describing the conversion of electrical energy to mechanical energy are hidden from the rest of the system by encapsulating them in the implementation of the motor.

A second important principle of object-oriented programming is *inheritance:* objects that are derived from a parent class inherit its interface and data members. Similarly, in modeling, a model that derives from a parent model inherits the parent's interface and equations. The child model can be extended by including additional physical interactions (ports) in the interface, or additional equations in the implementation [18, 19, 24].

Object oriented model design results in a hierarchical organization of models and simplifies the tasks of reusing, maintaining, and extending families of simulation models. Several research groups have developed object-oriented languages for discrete event systems [2, 19] as well as continuous systems [18, 24, 30, 35]. Not all these languages support the object-oriented paradigm to the same extent; the most comprehensive support for object-oriented principles is contained in Modelica [18].

Comparison between Modelica and VHDL-AMS

Recently, there have been two important efforts to define advanced simulation languages capable of modeling complex multi-disciplinary systems: Modelica [18] and VHDL-AMS. Modelica has been developed primarily by the continuous systems community in Europe, while VHDL-AMS has evolved as an extension from VHDL to support **A**nalog and **M**ixed-**S**ignal components.

In general, the scope and expressiveness of both languages are very similar: Both support continuous time and discrete time modeling in multiple energy domains, and both support declarative modeling and hierarchical encapsulation.

In addition, Modelica has the advantage that is truly object-oriented with support for model inheritance and sub-typing (VHDL is limited to encapsulation). It further allows the definition of aggregate connections in which multiple across, through, and signal variables from different energy domains can be combined.

VHDL-AMS, on the other hand, as an extension of VHDL, has more modeling primitives for discrete-event simulation. In addition to continuous time simulation, it offers quiescent point, small-signal AC, and small-signal noise analyses.

A final important difference between the two languages relates to the solvers that exist to evaluate these models. Modelica is currently only supported by Dynasim. Through symbolic index reduction, their solver is capable of to handling index 3 problems that are common in mechanical models. (The index of a system of DAEs is the minimum number of differentiations required to obtain an equivalent explicit system of ordinary differential equations [1]). VHDL-AMS has only recently been standardized by IEEE, and has limited support currently, although several electronic CAD vendors are in the process of developing solver implementations.

SINGLE-DOMAIN SIMULATION

Simulation modeling defined for a single domain such as mechanical or electrical systems is a mature area, with several companies offering robust simulation packages. In this section, we examine those aspects of single-domain modeling that have an impact on multi-domain systems-level modeling.

Multi-body Dynamics Simulation. The mechanical interactions between multiple rigid bodies constitute an important aspect of the behavior of mechatronic systems. One method that is particularly relevant to component-based systems modeling is based on graph theory [15, 16]. Multi-body systems consist of two basic elements: bodies

(e. g., rigid bodies) and connections (e. g., joints). These elements map to vertices and edges, respectively, in a systems graph, from which standard graph algorithms can symbolically extract the DAEs describing the multi-body system [15, 36]. A second approach is based on modular, object-oriented models that can be hierarchically combined into complete systems [37-40]. Instead of a closed form symbolic expression, this approach generates a set of DAEs, often with an index of two or even three [38].

Based on these research results, several software systems, including ADAMS and DADS, provide efficient multi-body analysis capabilities [41, 42]. Some of these analysis systems are integrated with design tools allowing the mechanism data to be transferred to the simulator directly from CAD models. A more in-depth review of modeling and simulation of rigid body mechanics can be found in [43]; research in flexible multi-body systems modeling is surveyed in [44], while specific numerical methods are treated in [45-47].

Electrical and Electronics Systems. For the simulation of electrical systems, there exist a variety of simulation languages. For analog circuits the family of SPICE dialects is most popular, while for digital circuits VHDL and Verilog are common. More recently, languages, such as VHDL-AMS, have been developed for the simulation of mixed analog-digital circuits and multi-disciplinary systems. Most commercial vendors of simulation software for electrical systems integrate their simulation engines tightly with ECAD software. This results in a complete design and simulation environment in which the designer can define the circuit topology, create a physical layout, and verify the circuit's behavior.

Control Systems. Embedded controllers have a high degree of interaction with the electrical and mechanical components in the system. They also tend to be digital so that a hybrid continuous/discrete simulator is required to evaluate their interaction with the environment [48, 49].

An important aspect of the evaluation of embedded controllers is Hardware-in-the-Loop (HWIL) testing [50, 51]. This allows the evaluation of a physical controller prototype interacting with a virtual electro-mechanical system, as well as the evaluation of a virtual embedded controller interacting with a real physical system. Several development environments for control algorithms provide special extensions for HWIL testing and automatic code generation [50].

Hydraulic and Thermal Systems. Hydraulic and thermal systems are often modeled as interacting with each other and with mechanical components [52, 53]. The behavior of both thermal and hydraulic systems depends strongly on the geometry of the components and their physical configurations. As for mechanical systems, a tight integration with the 3D design environment is essential.

From the review above, it is clear that many single-domain simulation environments are closely integrated with design tools. This trend is currently expanding towards the simulation and design of multi-disciplinary systems in general.

INTERLEAVING DESIGN AND SIMULATION

One can think of design as a process that consists of *decomposition* and *composition*. High-level functions are hierarchically decomposed into functions for subsystems; these sub-functions are then mapped to physical components that are in turn recomposed into a complete system [40, 54, 55]. During the process of composition (i.e. synthesis), the designer defines which components are used and how they interact with each other. This process parallels the hierarchical modeling process: models of components are connected to each other via interaction models (describing the dynamics of the component interactions) to define the behavior of a system or subsystem. Both processes are based on hierarchical composition: composition of form in design and behavioral models in simulation [33, 56].

As pointed out in the previous section, many electronic CAD vendors offer close integration between the circuit design process and the behavioral modeling of these circuits using industry-standard languages. When the user defines the circuit topology, the corresponding models of components and component interactions are automatically composed into a system-level model. This can be easily accomplished because the interactions between the electrical components are very simple: the voltages of two connecting terminals are equal and the currents sum to zero.

In the mechanical domain, however, parts can interact with each other according to a variety of connections, such as lower pairs, gear contacts, or rolling contacts. These connections result in both kinematic and dynamic constraints on the positions of the interacting parts. The information needed to instantiate these interaction models (the model structure and corresponding parameters) either can be obtained directly from the mechanical CAD model, or has to be provided by the designer [57-60].

Some CAD tools, such as Pro/Engineer Behavioral Modeler™ [61], allow for the integration between CAD data and behavioral modeling. By including evaluation procedures as features in the feature tree, one can automatically

perform behavioral analysis whenever a part or assembly is modified. For example, an evaluation feature could compute the mass of the design from the CAD geometry and the material density.

The current research trend is to extend the single-domain integration between design tools and simulation tools to multiple domains. This can be achieved through component models [40, 62, 63] that include a description of both the form and the behavior. The act of connecting two components configures not only the form, but also the behavioral models of the components. This component-based paradigm can be extended even further towards object-oriented intelligent components in which knowledge-based systems containing design rules are integrated with the CAD and behavioral models [64, 65].

A domain in which component based modeling has already been demonstrated is MEMS (Micro Electro-Mechanical Systems) [66-69]. Although MEMS systems are multi-disciplinary in nature (often consisting of electrical and mechanical components), the mechanical behavior can frequently be approximated as being limited to two dimensions. The resulting mechanical equations are sufficiently simple to be modeled within the existing modeling paradigms of electrical CAD tools. In certain simulation experiments, it may be necessary to model the MEMS device not as a collection of rigid bodies, but as a more detailed finite-element model. This is an area of ongoing research [70, 71].

COLLABORATIVE MODELING

Design of complex multi-disciplinary systems requires the expertise of a group of collaborating specialists: Designers with backgrounds in different disciplines collaborate with analysts, manufacturing engineers, marketing specialists, and business managers. To coordinate design processes among geographically dispersed and multi-disciplinary teams, many global enterprises have taken advantage of computer aided engineering (CAE) technologies that provide sharing, visualization, documentation, and management of product models [72-77]. However, the aspect of collaborative simulation modeling is still in its infancy. To support collaborative modeling, design teams need common, shared model representations, repositories to manage model components, and model abstraction capabilities to provide different views of models to designers.

Common Representation

To share simulation models within a collaborative modeling environment, designers need a common model representation. The Very High-Speed Integrated Circuit Hardware Description Language (VHDL), for instance, has been used as a design automation tool in all phases of modern very large-scale integration (VLSI) design. Similarly, the U.S. Department of Defense and its contractors have used the High Level Architecture (HLA) for simulation of battlefield scenarios [78, 79]. HLA is a set of specifications that defines how multiple simulations, called federates, can interoperate within a federation to form a larger simulated environment. These standards facilitate collaborative modeling by eliminating the conversion between the representations of various modeling and simulation tools. Further research will have to address the issue of integration and interoperability of models developed in different application domains.

Model Management

In a collaborative modeling environment, product models are stored in repositories, usually implemented as databases. To manage such model repositories, researchers have developed several organizational schemes [79-81], domain ontologies [82-84], and unified data formats [85].

Park and Kim [79] introduced a relational algebraic framework for the management of VHDL models. Within this framework, a family of hierarchical system structures is organized as a VHDL model structure. First, a candidate model structure that meets design objectives is selected from the family. Then, the VHDL model for the structure is systematically constructed by combining primitive VHDL models in a VHDL library.

Breunese et al. [80] provide a framework for model reuse. There are three levels of abstractions for a model: engineering components, conceptual descriptions in terms of physical processes, and mathematical relations. Each component is associated with a number of alternative conceptual descriptions. Concepts, in turn, are further specified by selecting from a number of alternative mathematical relationships. By storing the models assembled from these generic building blocks also in the library, one can combine models with different levels of detail and validation status.

A very comprehensive effort to organize and store simulation models is part of the NIST design repository project [86, 87]. In contrast to traditional design databases, the goal of design repositories is to capture, share, and

reuse *design knowledge*. This knowledge goes well beyond just CAD data and includes behavioral models, functional representations, process plans, and possibly design rationale.

Model Abstraction

In the context of design, it is important to be able to simulate a system at different levels of abstraction. Different stages of the design process require different levels of analysis—high-level analysis in the early, conceptual stage when only few design details are known, and more detailed towards the end of the design process. Furthermore, the design of a particular sub-system may require only a high-level description of the behavior of other sub-system with which it does not interact directly. The goal of model abstraction is to provide users with a model of the design prototype that captures only the relevant dynamics [74, 88-91].

DISCUSSION

Although a vast body of work exists in the area of modeling and simulation of engineering systems, additional research is needed to address the following issues and challenges.

Modeling

Modeling at the component level. Simulation languages have evolved from procedural and functional languages towards equation-based and object-oriented languages, moving away from the differential equation level towards the use of more intuitive and user-friendly building blocks. The next step in this evolution is the development of simulation languages that operate at the level of components and sub-systems. These component models include multiple behavioral models describing the component from different perspectives and at different levels of detail. As the designer builds a system from components, the corresponding system-level model is instantiated from the component models. Depending on the type of the simulation experiment, the appropriate behavioral models for each of the components and component interactions can be instantiated.

Component interaction modeling. When a designer composes a system from components, he also defines how these components interact with each other (their relative positions, electrical contacts, etc.). Currently, the dynamics of these interactions need to be modeled explicitly by the user. However, often the information to extract these interaction models has already been provided by the designer. The information needed to determine the interaction models can be automatically obtained from the domain-specific design tools that are integrated with the modeling environment.

Selecting an adequate level of detail. At different stages of the design process, different analyses need to be performed. Correspondingly, the simulation model of a system needs to be adapted to be adequately accurate without being overly detailed to avoid wasting computational resources. The current state of the art does not allow simulation models to be easily converted to different levels of abstraction. Future research should focus on methods to automatically select the most appropriate model for each component and component interaction in a system.

Improvement of numerical solvers. Currently, most VHDL-AMS simulators are not suited for modeling of systems with mechanical components. They often fail to find algebraically consistent initial conditions and have difficulty solving index-3 problems that are common in mechanical systems. With the simulation of multi-disciplinary systems becoming more frequent, solvers need to address the special characteristics posed by all the different application domains. They should also provide the user with finer control over the numerical integration, beyond merely selecting the step size and integration algorithm.

Integration with design tools

Integration with CAD. As described in previous sections, close integration between design tools and the modeling and simulation environment simplifies model creation and design verification. However, such tight integration has occurred successfully only for electronic CAD (with some limited integration in other domains). Further integration of single-domain CAD tools with a common simulation environment is needed so that a team of designers can create virtual prototypes from within the design environments with which they are familiar and that are custom-tailored to their needs.

Finite-element modeling. Finite-element tools are currently not integrated with systems-level modeling environments—partly due to their computational requirements, partly due to the difficulty interfacing finite element models with lumped models. With the increases in computational resources, future research should focus these interfacing issues so that we can includes models of distributed physical phenomena like mechanical flexure, or complex electromagnetic and thermal behavior in system-level models.

Integration with optimization and synthesis tools. The support for integration with optimization and synthesis tools within current simulation environments is mostly limited to some scripting capabilities (for instance to run a particular simulation for a range of parameter values). Based on the results, the designer has to update the design parameters manually. Through a closer integration between simulation and design tools, this process should be automated in the future. First steps in this direction have already been taken in [61, 64].

Collaborative Modeling

Unified model representation. Recently, several standardization and unification efforts have resulted in modeling languages and frameworks for simulations in multiple domains—e.g. Modelica, VHDL-AMS, and HLA. However, most single domain simulation environments still use proprietary data formats and solvers that are tightly integrated with the modeling environment. Future modeling and simulation environments should allow for a tight integration between application domains, either by interfacing the solvers or by using common model representations.

Ontologies for modeling and simulation. To further improve the exchange of model information (between both agents and humans), researchers have started to develop domain ontologies [83, 84]. For instance, Ozawa [82] proposed a common ontology to support different levels of information sharing between humans and multiple modeling and simulation software agents. Upon these domain ontologies, unified taxonomies and keyword networks can be built to support model retrieval and repository management.

CONCLUSIONS

This paper reviews the state-of-the-art in modeling and simulation of engineering systems. Simulation is a very broad area that comprises many research issues that are not included in this survey. We have limited our attention to issues that are particularly important with respect to system-level modeling in support of design.

The modeling of mechatronic systems requires a language capable of describing physical phenomena in multiple energy domains, in continuous time as well as in discrete time. Recent advances in modeling have resulted in several modular, object-oriented languages that satisfy these requirements.

To further simplify the modeling process and avoid unnecessary duplication of data entry, it is critical that the simulation environment be integrated with the design environment. In single-domain simulation environments, this is already common practice. Current research is expanding this integration towards simulation of multi-disciplinary systems.

Finally, due to the multi-disciplinary nature of mechatronic systems, the design requires a team of experts with different backgrounds. Systems modeling, therefore, must support collaborative modeling, including support for standardized languages, model management tools, and model abstraction tools.

ACKNOWLEDGMENTS

We would like to thank Antonio Diaz-Calderon and the other members of the Composable Simulations team for sharing their insights in simulation-based design and virtual prototyping. We also acknowledge the valuable comments provided by the reviewers; they greatly improved the quality of the manuscript.

This research was funded in part by DARPA under contract ONR # N00014-96-1-0854, by the National Institute of Standards and Technology, by the National Science Foundation under grants # CISE/115/KDI 98 73005 and # EIA-97 29827, by the Pennsylvania Infrastructure Technology Alliance, and by the Institute for Complex Engineered Systems at Carnegie Mellon University.

REFERENCES

[1] Ascher, U. M. and Petzold, L. R., *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. Philadelphia, Pennsylvania: SIAM, 1998.

[2] Zeigler, B. P., Praehofer, H., and Kim, T. G., *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems*, 2nd ed. Academic Press, 2000.

[3] Fishwick, P. A., "A Taxonomy for Simulation Modeling Based on Programming Language Principles," *IIE Transactions*, vol. 30, pp. 811-20, 1998.

[4] Histand, M. B. and Alciatore, D. G., *Introduction to Mechatronics and Measurement Systems*. Boston: Mc Graw-Hill, 1998.

[5] Seshu, S. and Reed, M. B., *Linear Graphs and Electrical Networks*. Reading, Massachusetts: Addison-Wesley, 1961.

[6] Triengo, M. J. L. and Bos, A. M., "Modeling the Dynamics and Kinematics of Mechanical Systems with Multibond Graphs," *Journal of the Franklin Institute*, vol. 319, pp. 37-50, 1985.
[7] Paynter, H. M., *Analysis and Design of Engineering Systems*. Cambridge, MA: MIT Press, 1961.
[8] Rosenberg, R. C. and Karnopp, D. C., *Introduction to Physical System Dynamics*. New York: McGraw-Hill, 1983.
[9] Edström, K., "Simulation of Newton's Pendulum Using Switched Bond Graphs," *1999 Western MultiConference*, San Francisco, California, 1999.
[10] Ferris, J. B. and Stein, J. L., "Development of Proper Models of Hybrid Systems: A Bond Graph Formulation," *International Conference on Bond Graph Modeling and Simulation*, Las Vegas, Nevada, 1995.
[11] Breunese, A. P. J. and Broenink, J. F., "Modeling Mechatronic Systems Using the SIDOPS+ Language," *ICBGM '97*, Phoenix, AZ, pp. 301-306, 1997.
[12] Trent, H. M., "Isomorphisms between Oriented Linear Graphs and Lumped Physical Systems," *The Journal of the Acoustical Society of America*, vol. 27, pp. 500-527, 1955.
[13] Branin, F. H., "The Algebraic-Topological Basis for Network Analogies and the Vector Calculus," *Symposium on Generalized Networks*, Brooklyn, New York, pp. 453-491, 1966.
[14] Durfee, W. K., Wall, M. B., Rowell, D., and Abbott, F. K., "Interactive Software for Dynamic System Modeling Using Linear Graphs," *IEEE Control Systems*, vol. 11, pp. 60-66, 1991.
[15] McPhee, J. J., "On the Use of Linear Graph Theory in Multibody System Dynamics," *Nonlinear Dynamics*, vol. 9, pp. 73-90, 1996.
[16] Baciu, G. and Kesavan, H. K., "From Particle-Mass to Multibody Systems: Graph-Theoretic Modeling," *IEEE Trans. Systems, Man and Cybernetics*, vol. 27, pp. 244-250, 1997.
[17] Muegge, B. J., *Graph-Theoretic Modeling and Simulation of Planar Mechatronic Systems*, Thesis, University of Waterloo, Systems Design Engineering, Waterloo, 1996.
[18] Elmqvist, H., Mattsson, S. E., and Otter, M., "Modelica: The New Object-Oriented Modeling Language," *The 12th European Simulation Multiconference*, Manchester, UK, 1998.
[19] Zeigler, B. P., *Object-Oriented Simulation with Hierarchical, Modular Models*. Academic Press, 1990.
[20] Strauss, J. C. and others, "The SCI Continuous System Simulation Language (CSSL)," *Simulation*, vol. 9, pp. 281-303, 1967.
[21] Maffezzoni, C. and Girelli, R., "Moses: Modular Modeling of Physical Systems in an Object-Oriented Database," *Mathematics and Computer Modelling of Dynamical Systems*, vol. 4, pp. 121-147, 1998.
[22] Kloas, M., Friesen, V., and Simons, M., "Smile : A Simulation Environment for Energy Systems," *5th International IMACS-Symposium on Systems Analysis and Simulation*, Berlin, Germany, pp. 503-506, 1995.
[23] Mattsson, Elmqvist, and Otter, "Physical System Modeling with Modelica," *Control Engineering Practice*, vol. 6, pp. 501-510, 1998.
[24] Piela, P. C., Epperly, T. G., Westerberg, K. M., and Westerberg, A. W., "ASCEND: An Object Oriented Computer Environment for Modeling and Analysis. 1 - the Modeling Language," *Computers and Chemical Engineering*, vol. 15, pp. 53-72, 1991.
[25] Cellier, F. E., *Continuous System Modeling*. Springer-Verlag, 1991.
[26] Glynn, P. W., "A GSMP Formalism for Discrete Event Systems," in *Discrete Event Dynamic Systems: Analyzing Complexity and Performance in the Modern World*, Y.-C. Ho, Ed. IEEE Press, pp. 11-20, 1992.
[27] Koenig, H. E., Tokad, Y., Kesavan, H. K., and Hedges, H. G., *Analysis of Discrete Physical Systems*. New York: McGraw-Hill, 1967.
[28] Banks, J., *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice*. John Wiley & Sons, 1998.
[29] Barton, P. I. and Pantelides, C. C., "Modeling of Combined Discrete/Continuous Processes," *AIChE Journal*, vol. 40, pp. 966-979, 1994.
[30] Fishwick, P. A., "Integrating Continuous and Discrete Models with Object Oriented Physical Modeling," *1997 Western Simulation Multiconference*, Phoenix, Arizona, 1997.
[31] Cellier, F. E., "Object-Oriented Modeling: Means for Dealing with System Complexity," *15th Benelux Meeting on Systems and Control*, Mierlo, Netherlands, 1996.
[32] Diaz-Calderon, A., Paredis, C. J. J., and Khosla, P. K., "Reconfigurable Models: A Modeling Paradigm to Support Simulation-Based Design," *2000 Summer Computer Simulation Conference*, Vancouver, Canada, 2000.
[33] Zhang, G. and Zeigler, B. P., "The System Entity Structure: Knowledge Representation for Simulation Modeling and Design," in *Artificial Intelligence, Simulation and Modeling*, L. E. Widman, K. A. Loparo, and N. R. Nielsen, Eds. New york: Wiley, pp. 47-73, 1989.

[34] Zeigler, B. P. and Luh, C.-J., "Model Based Management for Multifacetted Systems," *ACM Transactions on Modeling and Computer Simulation*, vol. 1, pp. 195-218, 1991.
[35] Anderson, M., *Object-Oriented Modeling and Simulation of Hybrid Systems*, Thesis, Lund Institute of Technology, Department of Automatic Control, Lund, Sweden, 1994.
[36] Diaz-Calderon, A., Paredis, C. J. J., and Khosla, P. K., "Automatic Generation of System-Level Dynamic Equations for Mechatronic Systems," *Journal of Computer-Aided Design*, vol. 32, pp. 339-354, 2000.
[37] Kubler, R. and Schiehlen, W., "Modular Modeling and Simulation of Multibody Systems," *ASME DETC 1999 17th biennial Conference on Mechanical Vibration and Noise*, Las Vegas, Nevada, pp. DETC99/VIB-8227, 1999.
[38] Otter, M., Elmqvist, H., and Cellier, F., "Modeling of Multibody Systems with the Object-Oriented Modeling Language Dymola," *Nonlinear Dynamics*, vol. 9, pp. 91-112, 1996.
[39] Lückel, J., Junker, F., and Toepper, S., "Block-Oriented Modeling of Rigid Multibody Systems with Regard to Subsystem Techniques," in *Advanced Multibody System Dynamics Simulation and Software Tools*, W. Schiehlen, Ed. Dordrecht, Netherlands: Kluwer Academic Publishers, pp. 49-66, 1993.
[40] Sinha, R., Paredis, C. J. J., and Khosla, P. K., "Integration of Mechanical CAD and Behavioral Modeling," *IEEE/ACM Workshop on Behavioral Modeling and Simulation*, Orlando, FL, USA, 2000.
[41] Crolla, D. A., Horton, D., and Firth, G. R., "VDAS-a Toolkit Approach to Vehicle System Simulation," in *Advanced Multibody System Dynamics Simulation and Software Tools*, W. Schiehlen, Ed. Kluwer Academic Publishers, pp. 367-372, 1993.
[42] Wittenburg, J. and Wolz, U., "MESA VERDE: A Symbolic Program for Nonlinear Articulated-Rigid-Body Dynamics," *ASME Design Engineering Technical Conference*, Cincinnati, OH, 1985.
[43] Schiehlen, W., "Multibody System Dynamics: Roots and Perspectives," *Multibody System Dynamics*, vol. 1, pp. 149-188, 1997.
[44] Shabana, A., "Flexible Multibody Dynamics: Review of Past and Recent Developments," *Multibody System Dynamics*, vol. 1, pp. 189-222, 1997.
[45] Gillespie and Colgate, "A Survey of Multibody Dynamics for Virtual Environments," *ASME Dynamic Systems and Control Division*, Dallas, TX, 1997.
[46] Schwerin, V., *Multibody System Simulation: Numerical Methods, Algorithms, and Software*., 1999.
[47] Richard, M. and Gosselin, C., "A Survey of Simulation Programs for the Analysis of Mechanical Systems," *Mathematics and Computers in Simulation*, North-Holland, pp. 103-121, 1993.
[48] Broenink, J. F., Hilderink, G. H., and Bakkers, A. W. P., "Conceptual Design for Controller Software of Mechatronic Systems," *1998 Lancaster International Workshop on Engineering Design*, Lancaster UK, pp. 215-229, 1998.
[49] Rai, S. and Jackson, W., "Collaborative Design of Modular Electromechanical Systems with Distributed Controls," *1999 ASME Design Engineering Technical Conference*, Las Vegas, Nevada, pp. DETC99/DTM-8775, 1999.
[50] Le, T., Renner, F. M., and Glesner, M., "Hardware in-the-Loop Simulation - a Rapid Prototyping Approach for Designing Mechatronics Systems," *8th International Workshop on Rapid System Prototyping (RSP '97)*, Chapel Hill, NC, 1997.
[51] Boot, R., Richert, J., and Schütte, H., "Automated Test of ECUs in a Hardware-in-the-Loop Simulation Environment," *ASIM 98*, Zürich, Switzerland, 1998.
[52] Tummescheit, H. and Eborn, J., "Design of a Thermo-Hydraulic Model Library in Modelica," *12th European Simulation Multiconference*, Manchaster, UK, 1998.
[53] Ludecke, A., Trieu, H.-K., Hoffmann, G., Weyand, P., and Pelz, G., "Modeling in Hardware Description Languages for the Simulation of Coupled Fluidic, Thermal and Electrical Effects," *1999 Behavioral Modeling and Simulation*, Orlando, Florida, 1999.
[54] Pahl, G. and Beitz, W., *Engineering Design: A Systematic Approach*, 2nd ed. London, U.K.: Springer-Verlag, 1996.
[55] Shooter, S. B., Keirouz, W., Szykman, S., and Fenves, S. J., "A Model for the Flow of Design Information," *ASME DETC 2000, 12th International Conference on Design Theory and Methodology*, Baltimore, MD, pp. DETC2000/DTM-14550, 2000.
[56] Sydow, A., "Hierarchical Concepts in Modeling and Simulation," in *Progress in Modeling and Simulation*, F. E. Cellier, Ed. London: Academic Press, 1982.
[57] Shah, J. J. and Mantyla, M., *Parametric and Feature-Based CAD/CAM: Concepts, Techniques, Applications*. New York, NY: John Wiley & Sons, 1995.

[58] Sinha, R., Paredis, C. J. J., Gupta, S. K., and Khosla, P. K., "Capturing Articulation in Assemblies from Component Geometry," *ASME Design Engineering Technical Conference*, Atlanta, GA, 1998.
[59] Daberkow, A. and Kreuzer, E. J., "An Integrated Approach for Computer-Aided Design in Multibody System Dynamics," *Engineering with Computers*, vol. 15, pp. 155-170, 1999.
[60] Lin, S.-T. and Lin, J.-H., "Computer Aided Dynamic Analysis and Simulation of Multibody Mechanical Systems in AutoCAD," *Simulation*, vol. 71, pp. 328-335, 1998.
[61] Duckering, B. C., "Behavioral Modeling Technology: Leveraging Engineering Knowledge in CAD Models," *The Fourth IFIP Working Group 5.2 Workshop on Knowledge Intensive CAD (KIC-4)*, Parma, Italy, pp. 126-135, 2000.
[62] Qureshi, S., Shah, J., Sunderarajan, K. G., Urban, S., Harter, E., Parazzoli, C., and Bluhm, T., "A Framework for Providing an Integrated View of Electromechanical Product Design Information," *TeamCAD: GVU/NIST workshop on Collaborative Design*, Atlanta, Geogia, 1997.
[63] Stein, J. L. and Louca, L. S., "A Component-Based Modeling Approach for Systems Design: Theory and Implementation," *International Conference on Bond Graph Modeling and Simulation*, Las Vegas, NV, 1995.
[64] Susca, L., Mandorli, F., and Rizzi, C., "How to Represent "Intelligent" Components in a Product Model: A Practical Example," *The Fourth IFIP Working Group 5.2 Workshop on Knowledge Intensive CAD (KIC-4)*, Parma, Italy, pp. 197-208, 2000.
[65] Bettig, B., Summers, J. D., and Shah, J. J., "Geometric Examplars: A Bridge between CAD and AI," *The Fourth IFIP Working Group 5.2 Workshop on Knowledge Intensive CAD (KIC-4)*, Parma, Italy, pp. 57-71, 2000.
[66] Neul, R., Becker, U., Lorenz, G., Schwarz, P., Haase, J., and Wünsche, S., "A Modeling Approach to Include Mechanical Microsystem Components into the System Simulation," *1998 Design Automation and Test in Europe*, Paris, FRANCE, 1998.
[67] Dewey, A., Srinivasan, V., and Icoz, E., "Towards a Visual Modeling Approach to Designing Microelectromechanical System (MEMS) Transducers," *Journal of Micromechanics and Microengineering*, vol. 9, pp. 332-340, 2000.
[68] Romanowicz, B. F., *Methodology for the Modeling and Simulation of Microsystems*. Kluwer Academic Publishers, 1998.
[69] Mukherjee, T. and Fedder, G. K., "Hierarchical Mixed-Domain Circuit Simulation, Synthesis and Extraction Methodology for MEMS," *Journal of VLSI Signal Processing-Systems for Signal, Image and Video Technology*, pp. 233-249, 1999.
[70] Hofmann, K., Lang, M., Karam, J. M., Glesner, M., and Courtois, B., "Generation O: A Behavioral Model of an Acceleration Sensor from Its Finite-Element-Description," *Third France-Japan Congress and First Europe-Asia Congress on Mechatronics*, Besancon, France, 1996.
[71] Wilson, N. M., Dutton, R. W., and Pinsky, P. M., "Investigation of Tetrahedral Automatic Mesh Generation for Finite-Element Simulation of Micro-Electro-Mechanical Switches," *International Conference on Modeling and Simulation of Microsystems, Semiconductors, Sensors and Actuators*, San Juan, Argentina, pp. 305-308, 1999.
[72] Finger, S., Konda, S., Prinz, F., Siewiorek, D., Subrahmanian, E., Tenenbaum, M., Cutkosky, M., Leifer, L., Bajcsy, R., and Birmingham, W., "Creating an Advanced Collaborative Open Resource Network," *6th International ASME Conference on Design Theory and Methodology*, Minneapolis, MN, 1994.
[73] Sriram, D., Logcher, R. D., Groleau, N., and Cherneff, J., "Dice: An Object-Oriented Programming Environment for Cooperative Engineering Design," in *Artificial Intelligence in Engineering Design*, vol. III, T. Tong and D. Sriram, Eds. Academic Press, pp. 303-366, 1992.
[74] Choi, K. K., Chang, K. H., Wang, J. Y., Tsai, C. S., and Steele, J. S., "A Simulation-Based Design Approach for Concurrent Engineering," University of Iowa, Iowa City, Technical Report R96-08, November 1996.
[75] Cutkosky, M. R., Tenenbaum, J. M., and Glicksman, J., "Madefast: Collaborative Engineering over the Internet," *Communications of the ACM*, vol. 39, pp. 78-87, 1996.
[76] Bajaj, C. and Cutchin, S., "Web Based Collaborative Visualization of Distributed and Parallel Simulation," *1999 IEEE Parallel Visualization and Graphics Symposium*, San Francisco, California, pp. 47-54, 1999.
[77] Iwasaki, Y., Farquhar, A., Fikes, R., and Rice, J., "A Web-Based Compositional Modeling System for Sharing of Physical Knowledge," *International Joint Conference on Artificial Intelligence*, pp. 494-500, 1997.
[78] Lutz, R., Scrudder, R., and Graffagnini, J., "High Level Architecture Object Model Development and Supporting Tools," *Simulation*, vol. 71, pp. 401-409, 1998.
[79] Park, H. C. and Kim, T. G., "A Relational Algebraic Framework for VHDL Models Management," *Transactions of the Society for Computer Simulation International*, vol. 15, pp. 43-55, 1998.

[80] Breunese, A. P. J., Top, J. L., Broenink, J. F., and Akkermans, J. M., "Library of Reusable Models: Theory and Application," *Simulation*, vol. 71, pp. 7-22, 1998.
[81] Murdock, J. W., Szykman, S., and Sriram, R. D., "An Informaion Modeling Framework to Support Design Databases and Repositories," *1997 ASME Design Engineering Technical Conferences*, Sacramento, CA, pp. DETC97/DFM-4373, 1997.
[82] Ozawa, M., Cutkosky, M. R., and Howley, B. J., "Model Sharing among Agents in a Concurrent Product Development Team," *Workshop on Knowledge Intensive CAD, KIC-3, IFIP 5.2 Working Group*, Tokyo, Japan, pp. 1-12, 1998.
[83] Schlenoff, C., Denno, P., Ivester, R., Szykman, S., and Libes, D., "An Analysis of Existing Ontological Systems for Applications in Manufacturing," *ASME DETC 19th Computers and Information in Engineering Conference*, Las Vegas, Nevada, pp. DETC99/EIM-9024, 1999.
[84] Devedzic, V., "A Survey of Modern Knowledge Modeling Techniques," *Expert systems with applications*, vol. 17, pp. 275-294, 1999.
[85] Szykman, S., Senfaute, J., and Sriram, R. D., "The Use of XML for Describing Functions and Taxonomies in Computer-Based Design," *19th DETC/Computers and Information in Engineering Conference*, Las Vegas, Nevada, pp. DETC99/EIM-9025, 1999.
[86] Szykman, S., "Design Respositories: Engineering Design's New Knowledge Base," *IEEE Intelligent Systems*, vol. 15, pp. 48-55, 2000.
[87] Szykman, S., Fenves, S. J., Shooter, S. B., and Keirouz, W., "A Foundation for Interoperability in Next-Generation Product Development Systems," *2000 ASME Design Engineering Technical Conferences (20th Computers and Information in Engineering Conference), paper No. DETC2000/CIE-14622*, Baltimore, MD, 2000.
[88] Nguyen, "Model Validation and Abstraction," *HW/SW Codesign MEDEA/ESPRIT conference*, Grenoble, France, 1998.
[89] Lee, K. and Fishwick, P. A., "A Semi-Automated Method for Dynamic Model Abstraction," *SPIE*, 1997.
[90] Hsieh, Y.-W. and Levitan, S. P., "Model Abstraction for Formal Verification," *1998 Design Automation and Test in Europe*, Paris, 1998.
[91] Ozawa, M., Biswas, G., and Zhu, "Task Distribution and Lumped Parameter Modeling in Multi-Disciplinary Product Development," *ASME Design Engineering Technical Conferences*, Las Vegas, Nevada, 1999.

FIGURE CAPTIONS

Figure 1. Conceptual model for a servo system of a disk drive head.

Figure 2. Linear graph representation of the drive head servo system. A dark node (*a*, *b*, *c*) represents an electrical domain connection and a light node (*d*, *e*) represents a mechanical domain connection. Solid edges (e_1, e_2, e_3, e_4) indicate energy flow between nodes. Dashed lines represent signal flow between the energetic portion and the signal portion of the graph.

Figure 3. Bond graph representation of the drive head servo system.

Figure 4. Block diagram representation of the drive head servo system generated using SimuLink and Matlab.

Figure 5. Port-based model representation of the drive head servo system. Nodes *a* through *e* correspond to the nodes in Figure 2.