ELSEVIER

Mathematics and Computers in Simulation 36 (1994) 91–101

MATHEMATICS AND COMPUTERS IN SIMULATION

# Self-explanatory simulators: making computers partners in the modeling process

Kenneth D. Forbus

*The Institute for the Learning Sciences, Northwestern University, Evanston, IL, 60201, USA*

**Abstract**

Digital computers have dramatically aided the application of mathematical models to problems of all kinds. For the most part computers have served as calculating engines, data retrieval and sorting devices, and lately as a powerful medium for visualization. Rarely have computers been used to assist in the more conceptual stages of investigations, for example in formulating the models to be analyzed, or in helping to interpret their results. Qualitative physics provides exactly the kinds of representations and reasoning techniques that can allow computers to shoulder more of the burden of creating and interpreting simulation models. This paper outlines the idea of *self-explanatory simulations*, an integration of qualitative and quantitative techniques, which provides one example of how qualitative physics can do this. A self-explanatory simulator combines the precision and efficiency of numerical simulation with the flexibility and interpretive power of qualitative representations. This synthesis makes them easier to use, and makes their results easier to understand. Moreover, such simulators can be written automatically from a physical description of the situation to be simulated, by using the qualitative analysis of the situation to guide the application of the appropriate quantitative and numerical models. Consequently, computers can become more like partners in the modeling process. This paper describes the basic ideas of self explanatory simulators, the results we have achieved so far, and our plans to push this technology towards applications.

## 1. Introduction

A central goal of qualitative physics is to make precise the representations and reasoning techniques that enable scientists, engineers, and just plain folks to understand the physical world. This ranges from understanding the common sense notions that allow children to do things in the physical world long before they have had mathematical training, to understanding the kinds of knowledge that characterizes the physical intuition or insight of world-class scientists and engineers. An important claim is that qualitative knowledge serves to organize other kinds of knowledge about the physical world, and that reasoning about the physical world frequently entails using qualitative knowledge to orchestrate the application of more detailed, quantitative knowledge. Furthermore, many suspect that in any domain where mathematical modeling is appropriate, not just physics and engineering, some form of qualitative representa-


*SSDI* 0378-4754(93)E0052-7

tions and reasoning will be required to build computers which can match the power and flexibility people bring to the formulation and interpretation of models.

This paper describes one such effort, carried out in collaboration with Brian Falkenhainer of Xerox PARC, to use qualitative physics to allow computers to help formulate and interpret models. In this research, we focus on extending the idea of numerical simulation. Numerical simulation is already important in science and engineering, and growing more so daily. As computing costs fall, simulations are also becoming common tools in education and training. While traditional simulations are precise, they can be difficult to build and debug and their results can be difficult to interpret. Our extension is the idea of *self-explanatory simulations*. A self-explanatory simulation combines the precision of numerical simulation with the explanatory power of qualitative representations. Self-explanatory simulators have three advantages:

(1) *Better explanations*: By tightly integrating numerical and qualitative models, the simulator can explain behavior as well as just predict it. In instruction these explanations can help students understand the artifact. In design these explanations can help pinpoint which parts of the artifact contribute to different aspects of its behavior.

(2) *Improved self-monitoring*: Most of the modeling assumptions underlying today's numerical simulators remain in the heads of their authors, and are only partially explicated in the program's documentation or listing. By incorporating the intuitive knowledge of the domain in an explicit qualitative model, the simulator itself has more opportunities to ensure that its results are consistent.

(3) *Increased automation*: While many numerical simulators are still built by hand, an increasing number of domain-specific toolkits for easing simulation construction are being used and developed (e.g. SPICE, ADAMS, DADS [7]). Such systems do not go far enough. Even in the best of them, most of the modeling choices are made by a human user, and explanation facilities are limited to graphics. Self-explanatory simulations offer a higher degree of automation, since the explicit representation of domain theories and modeling assumptions (e.g., [3]) allows the compiler to shoulder more of the modeling burden.

This paper describes the ideas of self-explanatory simulators, focusing on the improvements they provide in model formulation and interpretation. Section 2 describes the basic concepts, including some results obtained to date. Section 3 outlines some obstacles which must be overcome in order to make self-explanatory simulation a robust technology for applications. Section 4 describes some of our current efforts in this direction.

## 2. Self-explanatory simulators

The basic idea of self-explanatory simulations is to use the qualitative analysis of a system as a framework to create and organize a numerical simulator of that system. Consider the traditional *state space* formulation of dynamical systems. In this formulation a system has a set of *state variables* which suffice to completely determine its properties. The state variables for a spring-block oscillator, for instance, might be the position of the block and its velocity. These state variables may in turn determine other properties of the system (e.g., kinetic energy), but all other parameters are functions of these state variables. In Qualitative Process theory [4], such state variables correspond to *directly influenced* quantities. In QP theory the directly

influenced quantities cause in turn changes in other quantities, just as other parameters are determined by state variables in the state-space formulation. In a state-space the state of the system is described by a vector of values for the state parameters, while in qualitative terms it is described by a set of ordinal relationships among the system's variables. Effectively, the qualitative description quantizes state space into regions of "equivalent" (i.e., qualitatively identical) behavior. These qualitative distinctions are sufficient for a surprising number of inferences. Furthermore, qualitative explanations abound in instructional materials concerned with physics and engineering. Thus QP theory provides a formal language that can be used in attempts to express mental models.

A simple system like a spring-block oscillator can be modeled by a single set of equations. But many systems are not so simple. Traditional mathematical techniques focus on analyzing a given set of equations, and tend to ignore the modeling process per se. Qualitative physics provides tools for formalizing the modeling process itself, enabling the implicit assumptions of the modeler to be expressed and reasoned about. For example, a spring might behave linearly for small excursions but non-linearly for large displacements. Such changes in the relevant equations are then reflected in the quantization of state-space introduced by the qualitative description. This is an example of how qualitative physics can be used to capture the *tacit knowledge* of scientists and engineers, which is an important part of their expertise.

Already we see one role that qualitative analysis plays in simulation generation: constructing the relevant sets of equations. But there is more to it than that. First, suppose our simulation includes a notion of qualitative state as well as numerical state. This provides the potential for improved *self-monitoring*, i.e., detecting clashes between expectations expressed qualitative and numerical predictions. Suppose, for instance, that one has modeled a spring-block oscillator in the usual way with the spring providing a restoring force. A naive user might try a spring constant with the wrong sign (corresponding to a "exacerbating force"), and the simulation will detect the inappropriate behavior which results and flag the relevant parameters for correction. Integrating the qualitative state with the numerical state also provides the basis for powerful explanation facilities. The qualitative state can be used to access the set of equations describing the current behavior, the artifact's causal structure, and the qualitatively distinct behaviors which may occur next. These capabilities are illustrated in [5].

Besides generating numerical predictions, a self-explanatory simulator can use the conceptual and causal structure of the qualitative model to provide explanations of that behavior on demand. Fig. 1 provides an illustration. The causal structure imposed by the qualitative analysis is more than just a device for explanation. It plays a central role in building the numerical aspect of the simulator itself. For example, the order in which parameters are updated can be read directly off the graph of influences in the QP analysis of the system. Also, tests are incorporated to determine when the qualitative state changes, so that different code is used to reflect the different set of governing equations.

To summarize, a self-explanatory simulator combines qualitative and numerical representations to generate predictions and explanations of an artifact's behavior. The numerical component provides precise predictions of behavior, while the qualitative component explains this behavior in terms of the underlying qualitative model and set(s) of equations. Since the qualitative theory of the domain must be compatible with the equations, the explanations the system provides give the student a consistent picture of the system.

This self-explanatory simulation concerns a large can containing a small amount of water sitting on a stove. In the dialogue below, the user has run the simulation and is viewing plots of the amounts of water and steam and their temperatures. Questions are posed in a formal query language, and paraphrased here in English. The answers were generated automatically by the simulator's explanation facility. Commentary is in italics.

```
Q: What is happening at t = 100.0 seconds?
A: Heat is flowing from STOVE to the water in CAN.
```

*Qualitative explanations are available for every time tick in the simulation.*

```
Q: What is the temperature of the steam?
A: Sorry, the temperature of the steam in CAN does not exist at 100.0
seconds.
```

*The simulator knows that the numerical value it has for this tick is meaningless and does not report it.*

```
Q: What happens next?
A: At 238.6 seconds the boiling temperature of WATER and the temperature
of the water in CAN become equal.  Then, heat is flowing from STOVE to
the water in CAN and the water in CAN is boiling.
```

*Here we see the qualitative and quantitative components interacting: The qualitative aspect identifies this point as interesting, while the numerical aspect ascertains its exact time.*

```
Q: What else might have happened?
A: The only other possibility was: The temperature of the water in CAN
and the temperature of STOVE become equal, which leads to S2. In S2,
nothing is happening.
```

*The qualitative aspect of the simulation provides alternate possible behaviors.*

Fig. 1. Sample output from a self-explanatory simulator.

In engineering analysis tasks, the major role we see for self-explanatory simulators is in system-level simulation. That is, the technology we have developed works best for systems that can be described by sets of ordinary differential equations. One potential application is running trade-off studies in design: Given a particular feed system design in a power plant, for instance, examine the relative merits of one brand of pump versus another. By including a design's behavioral constraints in the model, the simulator itself can help verify them.

### *2.1. Compiling self-explanatory simulators*

Self-explanatory simulators can be built automatically, assuming the following three inputs:

(1) A *domain theory*, which provides the fundamental knowledge of the physical laws, processes, and conceptual structure of the domain.

(2) A collection of *modeling assumptions*, which expresses the requirements and constraints of the task at hand.

(3) A *structural description* of the artifact or system to be modeled.

We describe each in turn.

The domain theory must contain descriptions of the physical phenomena in both qualitative and quantitative terms. The qualitative aspect allows the automatic identification of when particular phenomena may be relevant, and the quantitative aspect provides detailed models which are woven together via qualitative analysis to build the simulator. A domain theory for engineering thermodynamics, for example, will have a model of the process of heat flow which

```
(defProcess (heat-flow ?src ?dst ?path)  ;; The physical process being defined
  Individuals ((?src :conditions (Quantity (heat ?src)))  ;; Kinds of entities
               (?dst :conditions (Quantity (heat ?dst)))  ;;  it occurs among
               (?path :type Heat-Path
                      :conditions (Heat-Connection ?path ?src ?dst)))
  Preconditions ((Heat-Aligned ?path))  ;; Non-dynamical antecedents
  QuanityConditions ((greater-than (A (temperature ?src))   ;; Dynamical
                                   (A (temperature ?dst))))) ;;  antecedents
  Relations ((Quantity flow-rate)  ;; Consequences
             (Q= flow-rate (- (temperature ?src) (temperature ?dst))))
  Influences ((I+ (heat ?dst) (A flow-rate)) ;; Direct causal effects
              (I- (heat ?src) (A flow-rate))))

(defSimParameter (temperature ?self) ;; parameter being defined
   (Physob ?self) ;; When part of this kind of entity
   :min 0.0       ;; Minimum possible value
   :default 25.0) ;; Default offered in simulator

(defSimParameter (thermal-conductance ?path)
   (Heat-Path ?path) :min 0.0 :default 0.1)

(defSimParameter (heat-flow-rate ?self)
   (quantity (heat-flow-rate ?self)) :min 0.0 :default 0.5)

(defSimEquation heat-flow-rate-definition ;; name of model
   (flow-rate ?self) ;; parameter being defined
   ((?self src ?src) ;; Additional antecedent patterns
    (?self dst ?dst) (?self path ?path))
   (heat-flow ?src ?dst ?path) ;; Entity it belongs to
   ((temperature ?src)  ;; Antecedent parameters
    (temperature ?dst) (thermal-conductance ?path))
   ((qprop (flow-rate ?self) (temperature ?src))  ;; Prerequisite
    (qprop (flow-rate ?self) (temperature ?dst))) ;;  causal structure
    (* (A (thermal-conductance ?path)) ;; Expression for the parameter
       (- (A (temperature ?src)) (A (temperature ?dst)))))
```

Fig. 2. A small sample of a simple domain theory.

describes in qualitative terms when heat flow arises and its rough consequences (e.g., the transfer of heat from the body with higher temperature to the body with lower temperature). In addition, a rich domain theory will have a variety of quantitative models and numerical approximations that provide numerical models for important classes of heat flow.

Fig. 2 provides an example of a piece of a domain theory. Notice that the qualitative description of heat flow (e.g., the *defProcess* statement) is very simple, only providing basic qualitative, causal information about the relationships between the parameters. The rest of the information in the process definition concerns what kinds of situations this process can occur in (i.e., the *Individuals* field) and the exact circumstances under which it acts (i.e., the *Preconditions* and *QuantityConditions* fields). The *defSimParameter* statement provides basic information about the numerical properties of a quantity, while the *defSimEquation* statement provides quantitative information about the relationships expressed in the qualitative aspect of the model. Notice that, as in the qualitative model, much of the information in the quantitative specification concerns when it is relevant (i.e., when a particular variety of conceptual entity, the physical process *heat-flow*, is acting and when the rate parameter is causally determined in a particular way). This relevance information is expressed using the conceptual structure set up

```
(defprocess (fluid-flow ?src-cs ?dst ?path)
   Individuals ((?path :type Fluid-Path
                       :conditions (Possible-Path-State ?path ?st)
                                   (Connects-To ?path ?src ?dst))
                (?src-cs :type Contained-Stuff
                         :form (c-s ?sub ?st ?src)
                         :conditions (Filled ?path ?src-cs))
                (?dst :type container))
   Preconditions ((Aligned ?path))
   QuantityConditions ((greater-than (A (pressure ?src-cs))
                                     (A (pressure ?dst))))
   Relations ((Quantity flow-rate))
   Influences ((I+ (amount-of-in ?sub ?st ?dst) (A flow-rate))
               (I- (amount-of-in ?sub ?st ?src) (A flow-rate))))

(defperspective (simple-fluid-rate ?pi)
   Individuals ((?pi :type (process-instance fluid-flow)
                     :conditions (Active ?pi)
                                 (?pi src-cs ?src-cs)
                                 (?pi dst ?dst))
                (?path :conditions (?pi path ?path)
                                   (not (Consider (fluid-conductance ?path)))))
   Relations ((Q= (flow-rate ?pi) (- (pressure ?src-cs)
                                     (pressure ?dst)))))

(defperspective (variable-fluid-rate ?pi)
   Individuals ((?pi :type (process-instance fluid-flow)
                     :conditions (Active ?pi)
                                 (?pi src-cs ?src-cs)(?pi dst ?dst))
                (?path :conditions (?pi path ?path)
                                   (Consider (fluid-conductance ?path))))
   Relations ((Quantity (pressure-delta ?src-cs ?dst))
              (Q= (pressure-delta ?src-cs ?dst)
                  (- (pressure ?src-cs) (pressure ?dst)))
              (Q= (flow-rate ?pi) (* (pressure-delta ?src-cs ?dst)
                                     (fluid-conductance ?path)))))
```

Fig. 3. A simplified example of how modeling assumptions can be represented.

by the qualitative aspect of the vocabulary. Often domain theories will contain multiple, conflicting perspectives on a phenomenon (e.g., laminar versus turbulent flow, as described in [3]). The causal requirements specified in the *defSimEquation* form allow the appropriate quantitative model to be selected.

Modeling assumptions provide information about the goals of the analysis. Often there are multiple models of a phenomena, e.g., a "black box" description of a system versus its decomposition into components, or different numerical approximations for the same phenomena, as mentioned above. The choice of which model is appropriate depends in part on the goal of the analysis. For example, a simulator built to confirm that the overall behavior of a system will be appropriate does not need to model failure modes of components, while conversely, a simulator built for Failure Effects and Modes Analysis (FEMA) must incorporate fault models. Fig. 3 illustrates how such modeling assumptions can be represented. Notice that fluid-flow can, in the simplest case, ignore path conductance. (At least for purely qualitative analyses, since some multiplier is needed to scale the pressure differential for even the simplest quantitative model.) Path conductance can be factored in as a single parameter, or this parameter in turn could be viewed as dependent on properties of the geometry of the fluid

```
(Substance water)
(State liquid)
(State gas)
(Container Can)
(Temperature-Source Stove)
(Heat-Path Burner)
(Heat-Connection Burner Stove (c-s water gas can))
(Heat-Connection Burner Stove (c-s water liquid can))
```

Fig. 4. The structural description used to create the simulator shown in Fig. 1.

path. Notice that in each of these different perspectives, the underlying commonalties of fluid flow do not change. The representation language supports this by allowing the *composition* of model fragments to create a task-specific model of a situation.

The structural description of a system formally describes the parts which comprise that system and the relationships between them which are relevant to the analysis. In most qualitative physics work to date, the level of detail used in structural descriptions is roughly that found in schematics of electronic circuits; that is, the connections and presumed interactions between parts are noted, while geometry tends to be ignored. Fig. 4 illustrates the structural description used to compile the simulator whose output is depicted in Fig. 1. Notice that we only needed to specify what kinds of objects exist (e.g., the existence of a substance, container, etc.) and the structural relationships that can hold between them (e.g., that the burner provides a heat path between the stove and any contained stuffs in the container). We did not need to specify what physical processes might occur: The compiler figured that out by itself, as part of its model formulation procedure.

Here we only outline the process of simulator compilation; see [6] for details. Given the structural description, the first step is to apply the model fragments of the domain theory, appropriately modulated by the modeling assumptions, to create a qualitative description of the kinds of physical phenomena which can occur in the artifact to be simulated. Second, appropriate quantitative models for the physical phenomena are selected. Third, the qualitative and quantitative models are used to write the code for the simulator. The code includes an explanation facility which consists of two components. The first is a *concise history*, which summarizes in qualitative terms the behavior generated by the numerical component of the simulator. The second is a *structured explanation system*, a repository of information about the original qualitative and quantitative models which can be used to explain and justify behaviors in terms of the original models.

### 2.2. State of the art

In our original work on self-explanatory simulators ([5]), the compilation technique we used was based on the notion of *envisioning*. The envisionment of an artifact is a graph describing every qualitatively distinct state of the artifact's behavior and the possible transitions between them. Using an envisionment as the basis for building selfexplanatory simulators had two advantages. First, self-monitoring was straightforward: Any time-point whose qualitative description did not match some state of the envisionment must represent a divergence between the quantitative and qualitative models. Second, counterfactual explanations (e.g., "What else might have happened if $X$ didn't?") could be cheaply generated, without any qualitative

reasoning at run-time. However, using envisionments has two problems. First, when using envisionments the most natural way to organize the simulator is around envisionment states. Since the same underlying physical mechanisms can generate a variety of distinct qualitative states, this led to greatly redundant code. Second, and more serious, is the fact that envisionments are exponential in the size of the artifact being modeled. Any straightforward envisionment-based system will not scale up.

We overcame these limitations by developing a new technique for generating selfexplanatory simulators that does not rely on envisioning. In fact, it does not require any qualitative simulation at all, nor even the generation of a single globally consistent qualitative state! Instead, we have developed a more local qualitative method of analysis that, while fundamentally simpler than qualitative simulation, still yields enough information to generate self-explanatory simulators. Roughly, we mimic the local case analyses carried out by human simulation writers when dealing with large physical systems; the details may be found in [6]. This approach has been reasonably successful: We have generated self-explanatory simulators for artifacts which are far too large to tackle with any qualitative simulation-based approach. The largest example is a model of a twenty-stage distillation column, containing hundreds of parameters, carried out by Nikitas Sgouros. For details, see [10].

One of our motivations is to develop simulation-based training and tutoring systems (e.g., [8,9,11]). For example, we would like engineering professors to be able to quickly and easily develop self-explanatory simulators which they can pass out to students to use in their classes. To this end we have demonstrated the ability to produce stand-alone runtime systems which can be executed on small microcomputers that would be incapable of building the simulator in the first place. Our original demonstration runtime systems worked on IBM PS/2's and 386 notebooks, while the simulation compiler itself resides on an IBM RS/6000. Both used Lucid Common Lisp and CLIM, but we are moving to C + + for the runtime system, as outlined below.

The idea of self-explanatory simulators is catching on; both Weld's group at University of Washington and the Knowledge Systems Lab at Stanford have developed selfexplanatory simulation systems. Their systems are focusing primarily on engineering design, and thus tightly interleave the creation of simulation code with its execution. Considering that engineers typically have access to high-powered workstations, this is an excellent choice in the tradeoff space for such systems. Falkenhainer and I are pursuing similar techniques for assisting Xerox engineers, while continuing to explore off-line compilation algorithms for creating educational software, which typically must run on very small computers.

## 3. Towards applications

We believe our progress to date on self-explanatory simulators raises some very exciting prospects for applications. This section explores some of the problems, both theoretical and practical, which we think must be addressed in order to make self-explanatory simulators useful in practical settings. We begin with general issues which cut across all classes of applications, and then look at problems specific to two particular applications: (1) Engineering design and analysis, and (2) Simulation-based training and tutoring systems.

The first problem is making the compilation process more efficient. Our current compiler avoids being exponential in the size of the artifact by limiting the complexity of interactions it considers [1]. There is a definite trade-off in the resources consumed by the compiler versus the quality of code produced, which must be explored to ascertain what choices are reasonable for different purposes.

The second problem is basically software engineering: we need to make the code produced by the compiler more efficient. Currently the compiler produces Common Lisp programs without declarations, which increases the computational overhead of the runtime system. From an applications standpoint, using Common Lisp makes such simulators harder to accept in organizations which due to tradition (e.g., engineering groups) or economics (e.g., universities) cannot provide high-quality Lisp environments for every user.

Perhaps the most difficult general problem is the current lack of rich, integrated domain theories which cover a broad range of physical phenomena. Little effort has been spent on developing such theories to date, [2] because there was little which could be done with them: The reasoning techniques developed in qualitative physics did not show signs of scaling up. Our work on self-explanatory simulators, as well as recent progress on incremental qualitative simulation systems [2], suggest that qualitative reasoning techniques which scale up to large artifacts can be built. Consequently, it is now worth developing domain theories that could meet the needs of engineering applications.

For engineering analysis tasks, such as supporting the design process, two other issues seem very important for near-term acceptance and deployment of self-explanatory simulators. First, the compilation process should exploit existing software tools, such as equation solvers and algebraic manipulation packages, rather than reinventing the wheel. The ideas of self-explanatory simulation can still be used in such circumstances, and in fact the modeling abilities provided by qualitative physics could yield automatic methods for setting up numerical models and switching between them as appropriate. Second, interfaces which make domain theories easily comprehensible and extensible must be developed, since exploring approximations is an important part of an engineer's task.

For simulation-based training and tutoring, there are two problems which must be addressed for most practical applications. First, tools which enable educators to easily specify structural descriptions and to explore and extend domain theories are needed. In the case of engineered artifacts, some way to capture and exploit the models developed during the design of an artifact should be developed. Obviously, if self-explanatory simulators were used for analyses during the design process then model capture is greatly simplified! (The computational requirements for training simulators, however, are sufficiently different, e.g., driving real-time displays, that new simulators may often be required.) Second, methods for integrating self-explanatory simulators to other kinds of instructional software, e.g. intelligent multimedia, need to be developed.

---

[1] Specifically, it places an upper bound on the size of ATMS environments which can be created. Such bounds can be set by analyzing the domain theory in advance.

[2] Some exceptions include [1,3].

## 4. Summary

Qualitative physics can provide the intellectual tools that allow computers to become partners in the modeling process, beyond being calculation engines, database maintainers, and visualization media. Qualitative representations provide a formal language for encoding physical intuitions, laws, and insights, and which provides the context for the expression of quantitative knowledge. Qualitative reasoning can provide help in formulating and interpreting models, and in orchestrating the use of both qualitative and other kinds of knowledge to solve interesting problems. Our work on self-explanatory simulators provides one illustration; the qualitative physics literature contains many others.

Our current work on self-explanatory simulators is designed to bridge the gap between research and application. This includes:

- *More efficient compilation strategies*: We are exploring the compilation resources/quality tradeoff to identify useful regions for particular applications.
- *More efficient runtime systems*: We are developing a C + + output mode for the simulation compiler, in order to (a) produce stand-alone executables for popular microcomputers used by engineering students and (b) produce efficient code that integrates smoothly with other software tools practicing engineers use.
- *Developing rich integrated domain theories*: We are working on a domain theory for engineering thermodynamics. The goal of this domain theory is to be powerful enough to support the creation of self-explanatory simulators for (a) the external Thermal Control System for the Space Station Freedom, (b) turbofan engines and other aircraft subsystems, and (c) the thermodynamic aspects of Navy propulsion plants.
- *Better modeling support tools*: While we believe that AI researchers can amply contribute to building broad-coverage integrated domain theories, we also believe that ultimately engineers and academics who are domain specialists must be heavily involved, and in fact will probably need to do the majority of the work. Consequently, we are constructing tools to simplify the modeling process, both at the level of constructing domain theories and constructing models of specific artifacts, and aimed both at developers of models and users of models.

## Acknowledgements

Financial support for this work has been provided by NASA Johnson Space Center, NASA Langley Research Center, Xerox, IBM, and an NSF Presidential Young Investigator award.

## References

[1] J. Collins and K. Forbus, Building qualitative models of thermodynamic processes, unpublished manuscript.
[2] D. DeCoste and J. Collins, IQE: An incremental qualitative envisioner, Working papers of the Fifth International Workshop on Qualitative Reasoning, Austin, Texas, May, 1991.
[3] B. Falkenhainer and K. Forbus, Compositional modeling: Finding the right model for the job, Artificial Intelligence 51 (1991) 95–143.

[4] K. Forbus, Qualitative process theory, Artificial Intelligence 24 (1984).
[5] K. Forbus and B. Falkenhainer, Self-explanatory simulations: an integration of qualitative and quantitative knowledge, Proceedings of AAAI-90, Boston, MA, 1990.
[6] K. Forbus and B. Falkenhainer, Self-explanatory simulators: scaling up to larger models, Proceedings of AAAI-92, San Jose, July, 1992.
[7] E.J. Haug, Computer-Aided Kinematics and Dynamics of Mechanical Systems, Vol. 1: Basic Methods (Allyn and Bacon, Newton, MA, 1989).
[8] J. Hollan, E. Hutchins and L. Weitzman, STEAMER: An interactive inspectable simulation-based training system, AI Magazine (1984).
[9] A. Stevens, B. Roberts, L. Stead, K. Forbus, C. Steinberg and B. Smith, STEAMER: Advanced computer-aided instruction in propulsion engineering, BBN Tech report, 1981.
[10] N. Sgouros, Integrating qualitative and numerical models in binary distillation column design, Proceedings of the 1992 AAAI Fall Symposium on Design of Physical Systems, October, 1992.
[11] B. Woolf, D. Blegen, J. Jansen and A. Verloop, Teaching a complex industrial process, Proceedings of AAAI-86.