Machine-generated Explanations of Engineering Models: A Compositional Modeling Approach

Thomas R. Gruber and Patrice O. Gautier

Knowledge Systems Laboratory
Stanford University
701 Welch Road, Building C • Palo Alto, CA 94304
gruber@ksl.stanford.edu

Abstract

We describe a method for generating causal explanations, in natural language, of the simulated behavior of physical devices. The method is implemented in DME, a system that helps formulate mathematical simulation models from a library of model fragments using a Compositional Modeling approach. Because explanations are generated from models that are dynamically constructed from modular pieces, several of the limitations of conventional explanation techniques are overcome. Since the explanation system has access to the derivation of mathematical equations from the original model specification, the system can explain low-level quantitative behavior predicted by conventional simulation techniques in terms of salient behavioral abstractions such as physical processes, idealized components, and operating modes. Instead of relying on ad hoc causal models, crafted specifically for the explanation task, the program infers causal relationships among parameters in a constraint-based equation model. Rather than using canned, top-down templates, the text generator composes textual annotations associated with individual model fragments into coherent sentences. We show how these techniques can be combined to produce a variety of explanations about simulated systems.

1. Introduction

This paper describes a method for generating explanations of how devices work. The method is implemented in the Device Modeling Environment (DME), a system to help engineers to explore the behavior of modeled devices under various scenarios by assisting with the formulation of simulation models and the interpretation of results [12,17]. On the basis of an initial model and the behavioral predictions obtained through simulation, DME can answer a range of user queries about the structure and behavior of the modeled system.

The approach combines several techniques:

- automatically assembling mathematical models from high-level model specifications, and explaining the low-level simulation data in terms of the original specifications
- dynamically generating explanations in response to queries formulated by the user through direct manipulation of text and graphics displayed during simulation
- composing text bottom-up from annotations on modular model fragments, and processing the resulting text to produce smooth, concise sentences
- inferring causal influences from mathematical constraint models and filtering the resulting network to produce succinct causal explanations of behavior

In this paper we attempt to show how these techniques can be combined to produce a variety of explanations about device behavior without imposing unrealistic limitations on the descriptive power or scale of the models. We begin by briefly describing how compositional modeling and simulation are supported in DME. We demonstrate some of the explanations generated using a model of the Space Shuttle's Reaction Control System. We show how users ask queries, how text is generated by composition, and how causal explanations are inferred from the equation model. We conclude by comparing related work and listing important open problems.

2. DME: Integration of model formulation, simulation, and explanation

Model formulation is the task of constructing a model from available primitives to answer some questions about device behavior. Every physical device can be modeled in a myriad of ways. Each model is based on a set of approximations, abstractions, and other assumptions about what is relevant to produce the desired data at a reasonable cost. Simulation is the prediction of behavior, typically described by the time-varying values of model parameters, from a device model and a specification of initial conditions. Explanation is the communication of the predicted data and the relevant information about the underlying model to enable the human engineer to interpret the results. Each model and simulation method will lead to a different explanation of how the device works.

In conventional mathematical simulation systems, the input to the simulator is a set of equations or low-level transfer functions, and the output is a set of numbers corresponding to the time-varying values of parameters that represent physical quantities. To the simulation engine, the equations are mathematical constraints over the values of parameters. To the engineer, the equations embody knowledge about high-level

Funding was provided by NASA Grant NCC2-537 and NASA Grant NAG 2-581 (under ARPA Order 6822).

concepts such as idealized components, physical processes, and physical laws.

A method for generating explanations, then, must take into account the process by which the model was developed and simulated; it must explain the model, not just the predictions.

DME is designed to explore this idea. Model formulation, simulation, and explanation services are integrated in an interactive system to help engineers build and understand models of physical devices. The explanation system in DME uses information about the derivation of the mathematical model from the original primitives, knowledge about the simulation method, and the initial conditions to answer user questions. To describe how this works, we need to look at how the system does model formulation and simulation.

2.1. Compositional Modeling in DME

DME's model formulation is based on the *compositional modeling* approach [4]. In compositional modeling, the model formulation system assembles a complete device model by selecting and composing modular primitives. The system starts with a library of *model fragments*, representing idealized components (e.g., resistors, transistors, logical gates, valves, pipes), physical processes (e.g., flows, heat transfers), operating modes of components (valves being open, transistors being saturated), and other modular descriptions of physical mechanisms. In DME, model fragments may inherit properties from more general model fragments in a subsumption hierarchy. For example, a generic model fragment for hydraulic components, which posits terminals for fluid flow, might be inherited by model fragments for pipes and valves. Each model fragment has a set of *applicability conditions* that specify when the model makes sense (e.g., a valve operating mode makes sense only for components modeled as valves). Model fragments also have *activation conditions* that indicate when an applicable model holds in a given simulation (e.g., the model of a boiling process can only hold when the temperature of the water is above some threshold). Each model fragment contributes a partial description of the behavior that occurs when the conditions hold. The behavior is specified in terms of algebraic and logical constraints on the values of simulation parameters.

The user specifies a set of abstract components and initial values for parameters in a *scenario description*. If the initial conditions are incomplete (i.e., the resulting model is under constrained), the system prompts the user for more constraints. From the scenario description, the system assembles a set of applicable model fragments and composes their partial behavior descriptions into a mathematical behavior model (a system of constraint equations and rules for discrete events).

The mathematical model is then used to drive a simulation, which predicts time-varying values of parameters. In DME, the simulation is performed by a distinct module, using off-the-shelf techniques for numerical integration [2] and qualitative simulation [15]. During any point in the simulation, the behavior of the modeled system is defined by a set of governing equations, called the *equation model*, and the values of the parameters constrained by those equations. The equation model is the union and composition of the equations contributed by model fragments whose activation conditions are satisfied.

A qualitative *state* is a period during which a set of equations remains unchanged. Within a qualitative state, the values of continuous parameters change. When parameter values cross limit points, the activation conditions of some model fragments become true and others become false, leading to a state transition. DME continually monitors the numerical simulation for such changes. When limit points are reached, or when discrete events trigger a state change, DME updates the equation model for the new state. Thus, in DME, model formulation and simulation are performed dynamically and synchronously.

We will show examples of DME's operation from the domain of the Space Shuttle's Reaction Control System (RCS). The RCS is a system of thrusters that are used to steer the vehicle. To investigate the design and testing of operator procedures, a model library for the pipes, tanks, and valves of the RCS was built in DME. Figure 1 shows a schematic diagram of the model. The RCS model contains 160 components, 150 equations, and 180 parameters. The scenario description is a description of these components, their connection topology, and the initial states of valves and tanks.

3. Explanation Generation

DME uses a variety of techniques to produce reasonable quality natural language text in an interactive setting without requiring sophisticated models of users, discourse structure, or natural language. We describe the essential elements of the design in this section.

3.1. Query-driven explanation generation

At any time during a simulation, the user can ask DME for an explanation. Each question is a query parameterized for some class of objects (a component, operating mode, qualitative event, parameter, or state). For each class of questions,

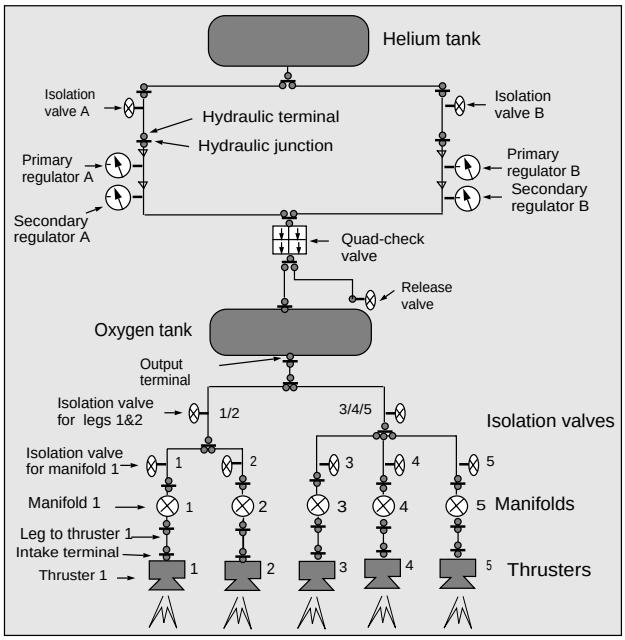

Figure 1: Schematic diagram of the RCS model

there is a task-specific, domain-independent procedure for generating answers called an explanation schema [18]. The explanation types currently implemented include:

- *What is this component?* Describes the parts, parameters, and possible behaviors (e.g., Figure 2)
- *What is the definition of this parameter, relation, component, or operating mode?*
- *What are the initial conditions of this simulation scenario?*
- *What is happening in this state?* Describes a snapshot of the simulation for a qualitative state
- *What just happened?* Summarizes salient changes in a simulation state (Figure 3)
- *What caused this event?* Explains the logical preconditions of a model fragment that asserts a discrete change
- *What caused this change in behavior?* Explains activation conditions of an operating mode (Figure 4)
- *What influences this parameter?* Gives causal account of the how this parameter obtains its value in a given current state (Figure 5)

To initiate an explanation, the user clicks on some graphics or text that has been presented. The system schematic is a useful starting point. A schematic like that shown in Figure 1, created using DME modeling tools, is shown to the user in a window. Each component and link is mouse sensitive.When the user clicks on a presented object, the DME offers a set of relevant questions. For example, in Figure 2 the user clicks on one of the thruster icons in the picture and asks "What is this component?" The resulting explanation is generated from the static structure of the thruster model and the set of model fragments representing its possible operating modes.

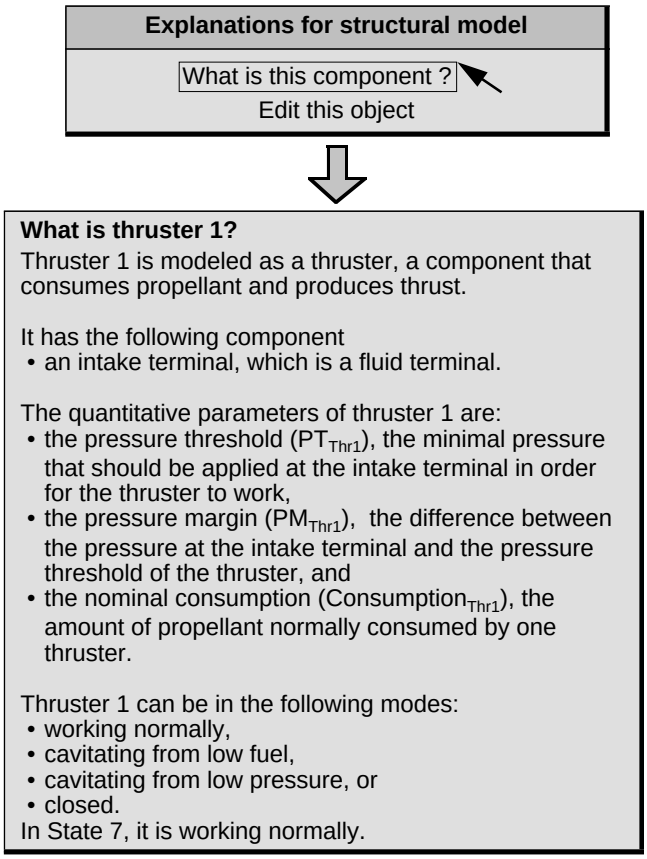

Figure 2: Describing component structure and operating modes

3.2. Identifying salient information

By clicking on the phrase "in state 7," the user then asks for an explanation of "what just happened." The answer is shown in Figure 3. In this type of explanation, only those events that can cause qualitative state transitions are reported. In Figure 3, the system reports that the value of a quantitative parameter (pressure margin of thruster 1) crossed a limit point (0 Pascals). Limit points are derived from an analysis of the activation conditions of model fragments; they are values that could cause models fragments to become active or inactive. In this example, the model fragment for thruster 1 working normally became inactive, and the model fragment for thruster 1 cavitating became active.

Limit point analysis, which is common in qualitative simulation, is used as a salience heuristic for explanation. This achieves a dramatic reduction in the information that *might* be displayed: of the 180 parameters in the RCS model, the "what just happened" explanation only mentioned PM_{Thr1}. In a conventional simulator, the trajectories of some subset of the modeled parameters can be displayed. However, they would typically be chosen in advance and fixed throughout the simulation. Because DME is responsible for updating the equation model, it can dynamically determine salient events and filter the information presented.

3.3. Follow-up questions grounded in presentations

All explanations are presented using mouse-sensitive text that allows the user to ask follow-up questions. Every noun phrase, verb phrase, or sentence in an explanation represents some object, event, or fact for which there are anticipated questions and corresponding explanation schemata. For example, in Figure 3 the strings "pressure margin" "thruster 1" "PM_{Thr1}" and the entire sentence are all mouse-sensitive regions with separate menus of relevant follow-up questions. This technique of associating follow-up questions with mouse-sensitive text has been used by Moore and Swartout [19] to help manage an interactive dialog based on a text plan. Although DME does no text planning, the technique does allow the user some control over the level of detail and focus of the explanation. This lessens the need for a user or dialog model, especially when tutoring is not the explanation task.

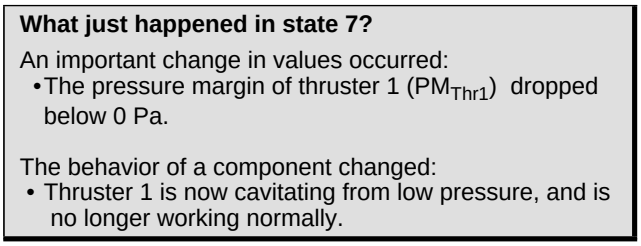

Figure 3: Summarizing salient changes

3.4. Compositional text generation

The text presented in DME explanations is composed from textual annotations on various model fragments. In Figure 2, "Thruster 1" is an annotation on the relation between the RCS system model fragment and the thruster model. The name and definitions of "intake terminal" comes from the generic fluid terminal model, which is a component of the thruster model. The text used to describe the quantitative parameters comes from text fragments associated with the models for those parameters. For instance, "pressure threshold" is the name for the one of the parameters of a thruster; "PT" is its abbreviation, and "the minimal pressure that should be applied..." is its definition. The texts identifying the four operating modes for a thruster come from the four model fragments representing these behaviors. The generation of even this simple summary is completely driven by the model library; model fragments and textual annotations can be added, deleted, and modified independently. This modularity is possible because the behavior and structure of the thruster are determined by composing model fragments from a library.

Noun phrases that identify subcomponents also reflect the part composition of the device; for example, in Figure 5 the phrase "the pressure at the intake terminal of thruster 1" is composed from the annotations on the pressure parameter ("pressure" "at"), the relation between the thruster and the terminal ("intake terminal"), and the relation between the RCS and thruster 1. A similar set of annotations was used to generate the associated glyph "$P_{In(Thr1)}$."

The model fragment representation is composable because the partial descriptions of behavior represented by each fragment can be composed into a meaningful global behavior. Compositionality allows libraries of reusable model fragments. Similarly, the text generation technique is composable because the text fragment annotations can be composed into English sentences. Writing these annotations requires no knowledge of linguistics.

The problem with generating sentences by composing text fragments is that the compositions can be repetitive and verbose. To address this problem, DME applies several transformations to composed text. Abbreviations for components and model parameters are introduced, and subsequent references are treated as if the abbreviations were pronouns ($P_{In(Thr1)}$). Since descriptions of device structure and behavior often introduce parallel structure, factoring transformations are used to eliminate repeated references to the same component, parameter, or parameter value. Other transformations include capitalization, punctuation, indentation, and enumeration. This class of language transformation is described in [11].

4. Causal explanation

The explanations shown so far reflect the structure of the modeled device and state of the simulation program. They are relatively easy to generate because the necessary information about the models and simulation history is represented explicitly in a knowledge base. More difficult are explanations of the *causes* of events or states. DME can answer two classes of questions about causality: what causes a change in qualitative state, and what causes a quantitative parameter to have its current value.

4.1. Explaining logical preconditions of events

The first type of causal explanation is a logical account of a discrete change in value or state. Discrete changes in the operating regions of components or discontinuous changes in quantitative parameters are reflected in the changing activation of model fragments. The "cause" of a discrete change, then, can be explained in terms of the predicates and parameter values that are mentioned in the activation conditions of a model fragment. DME explains how the activation conditions of model fragments become satisfied (or cease to hold) in a given state or at a state transition. It can then recursively explain the causal ancestry of each of the parameter values mentioned in the conditions (on request). For instance, in Figure 4, the user asks why the thruster is now cavitating. The resulting explanation describes the conditions under which the "cavitating from low pressure" operating mode is active, and why those conditions held in state 7. The first condition is that the discrete parameter "flow status" must have the value "open." In this case, the parameter got its current value because of an operator action in an earlier state. The second condition in Figure 4 is an inequality over the pressure margin parameter, which is a continuous quantity. Its current value is displayed to show that the inequality is satisfied.

Explaining how model fragment preconditions are satisfied is similar to explaining a sequence of rule firings in rule-based expert systems. In these systems, such explanations make sense only when the rules correspond to meaningful inferences or actions in the system being modeled [1]. Similarly, in compositional modeling, explanations of changes in qualitative state make sense if the transitions among model fragments reflect meaningful physical changes in the

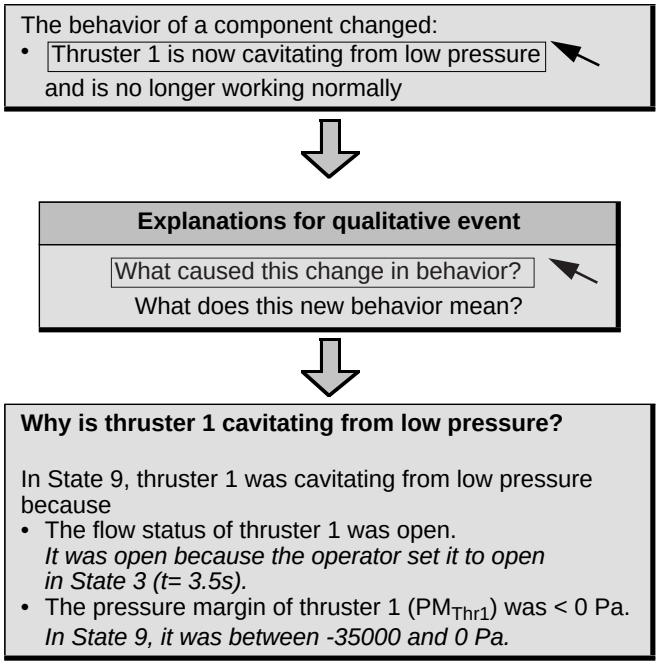

Figure 4: Explaining why a component operating mode became active.

modeled system. In engineering domains, components are designed to operate in predicable modes (e.g., open, closed, broken), and the operating assumptions of models of physical mechanisms can be made explicit (e.g., boiling starts at a boiling point).

4.2. Explaining causal influences on a continuous parameter

While the "cause" of a discrete event or parameter can be traced to the satisfaction of logical preconditions, the "cause" of the value of a continuous quantitative parameter is more complex. In Figure 5, the user asks why the pressure margin has a negative value. The answer is an account of the mathematical influences on the parameter in the current equation model.

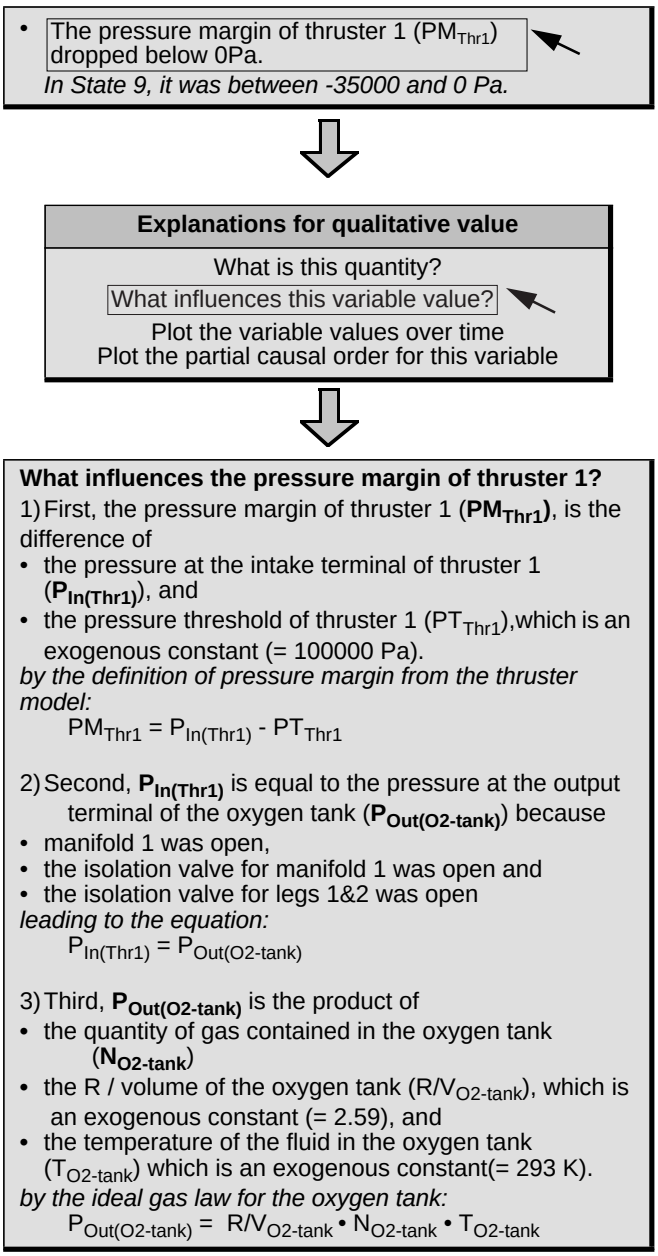

Figure 5: Explaining causal influences on a parameter

As explained in Figure 5, the pressure margin of thruster 1 (PM_{Thr1}), is determined by the intake pressure $P_{In(Thr1)}$, due to the equation that defines the pressure margin parameter (which comes from the thruster model) and the fact that the other term in that equation is an exogenous constant. The pressure at the intake terminal, in turn, is equal to the pressure at the output terminal of manifold 1, which is equal to the pressure at the intake terminal of the manifold. This chain of equalities (a=b, b=c,...) can be followed through seven subsequent terminals of isolation valves and pipes to the pressure at the output terminal of the oxygen tank ($P_{Out(O2-tank)}$). This would be tedious to explain to the user, so by default the system collapses this chain into the single equation shown in the second paragraph of Figure5. In this explanation, instead of showing the chain of equations, the system displays all the active operating modes that contributed an equation in the chain. For example, the phrase "manifold1 was open" is a description of the "open" operating mode of the "manifold1" component. If the manifold were closed, a different equation would be in force. Finally, the oxygen tank pressure is shown to be determined by the quantity of gas in the tank ($N_{O2-tank}$), as given by the ideal gas law associated with the tank.

The explanation goes beyond a literal presentation of the equations in several respects. First, it is causally directed; the explanation traces the path of influence back from the thruster through pipes to the oxygen tank. Second, it is an abstraction; most of the equations are hidden or summarized. It also relates the mathematical relationships among parameters to the high-level model fragments that contributed the relevant equations (the third equation is an application of the *ideal gas law*, which came from modeling the oxygen tank as an ideal container).

4.2.1 Causal ordering

This explanation happens to follow the connection topology shown in Figure 1, but to the simulator there are only constraint equations. In a different scenario, the direction of causality over the same components could be reversed (e.g., back pressure). Instead, DME infers the path of causal influence from oxygen tank to thruster using the *causal ordering* algorithm [13]. Given a set of equations constraining a set of model parameters, and the knowledge of which parameters are exogenous (i.e., determined outside the modeled system), the algorithm produces a dependency graph over the remaining parameters. The graph determines the order in which parameter values are computed within a numeric simulation loop, and usually corresponds to the intuitive assignment of causal influence in the model.

4.2.2 Filtering irrelevant information

In models of even moderate size, there may be many parameters that are involved in the behavior of interest. However, to be comprehensible, an explanation must be limited to just a few of the most salient parameters. In explaining the influences on a continuous parameter, DME makes use of several heuristics to determine which nodes of the causal order graph are worth mentioning. These heuristics include:

- recognizing and collapsing equality chains
- looking at the physical dimensions of the parameters involved (changes in dimensions are interesting)

- looking at the position of variables in the causal ordering (root nodes are interesting)

In the example of Figure 5, of the 15 non-exogenous parameters involved in the path from PM_{Thr1} to $N_{O2\text{-}tank}$, only $P_{In(Thr1)}$ and $P_{Out(O2\text{-}tank)}$ were explained. A long chain of equalities was collapsed to the single equation $P_{In(Thr1)} = P_{Out(O2\text{-}tank)}$. The procedure for filtering parameters and collapsing equality chains is not dependent on knowledge about flows or device topologies. Details of the use of causal ordering to produce explanations in DME are given in a separate paper [9].

The important consequence of inferring causal order is that it avoids the need for ad hoc "causal models" that are built specifically for explanation generation. In explaining how things work, people often use causal terminology. However, when analyzing the behavior of electromechanical devices, engineers use logical and mathematical constraints, which do not specify causality. Inferring causal dependencies allows the use of realistic engineering models without sacrificing the ability to generate causal explanations.

5. Related Work

Existing systems for generating explanations of device behavior typically depend on models built explicitly for the explanation or tutoring task [8,21]. When explanations are generated from more general behavior models, the explanations typically follow from hard-coded labeling of causal influence [24] or component function [14]. DME model libraries are designed primarily for describing behavior for engineering analysis. Textual annotations used for explanation are optional and have no behavioral semantics.

Integrating model formulation and explanation in DME is analogous to the approach used in the Explainable Expert System project [22]. In EES, the developer gives high-level specifications of a knowledge system, and the low-level code is synthesized. Then, the answers produced by the compiled knowledge system can be explained in terms of the original specifications. In DME, the model builder gives high-level descriptions of a device, behaviors of interest, and assumed operating environment, and the system generates a low-level equation model for simulation. It can then explain the predicted behavior in terms of the high-level descriptions.

DME's explanation approach is most similar to the approach taken in the SIMGEN systems [3,7]. Like DME, these systems use a compositional modeling method of model formulation, perform numeric simulation, and generate causal explanations. However, they use a qualitative model for explanation and a parallel quantitative model for simulation. DME generates qualitative explanations from the quantitative model. In SIMGEN, the qualitative model is used to produce an *envisionment* [5,6], which is a description of all possible qualitatively distinct states of the device. A numerical simulation is run in lock-step with the qualitative model. A fundamental limitation of this approach is that envisionment requires a complete enumeration of the state space of the modeled system; in complex devices, this space is very large and impractical to enumerate.

In SIMGEN, causal explanations are produced directly from the causal influences represented explicitly the qualitative model. Thus, SIMGEN requires that the model library ("domain theory") include an integrated network of qualitative and quantitative model fragments that anticipates the relevant causal dependencies. DME only requires model fragments containing the quantitative constraint equations and discrete event rules. The intent is to facilitate the development of model libraries by domain specialists not trained in qualitative modeling. In any case, useful model libraries are difficult to build and new development tools will be required to achieve this goal.

The successor to SIMGEN, SIMGEN.MK2 [3], addresses the efficiency and scale problems with a compilation approach. SIMGEN.MK2 precomputes the conditions for state transitions from an analysis of the qualitative model, without using a global envisionment. Nevertheless, the time required by SIMGEN.MK2 to produce a self-explanatory simulator for a non-trivial example is still significant (on the order of hours). In contrast, DME uses an interpreted approach, with a different performance profile. DME dynamically reconfigures the model at each qualitative state transition. Although it is fast at doing numerical simulation within a qualitative state, DME is slower at computing the set of active model fragments at the transition to each new state. However, it can generate a complex explanation of existing states almost instantaneously. This facilitates the rapid configuration and debugging of models, which is important for design tasks.

6. Summary and future work

We have described a collection of techniques that, when taken together, constitute a practical method for generating interactive explanations of device behavior in natural language. The main result is the ability to explain engineering models of the sort engineers use for analysis (constraint equations with ordinary differential equations). Integrating explanation with model formulation capabilities allows DME to explain numerical values predicted from equation models in terms of salient behavioral abstractions such as physical processes, idealized components, and operating modes. Using compositional text generation facilitates modularity and scalability in the development of model libraries. Just as the behavior of a component can be described without making assumptions about the connection topology in which it operates, the textual annotations on model fragments can be written without anticipating the sentences into which they will be composed. Because causal relationships are inferred, there is no need to build assumptions of causality into the models.

Our long-term objective is to make machine-generated explanation part of the engineer's toolbox. High-quality explanations could provide support for conceptual design, testing operator procedures, and documenting design intent. We believe that achieving these goals requires the following new capabilities:

- The ability to generate and explain alternative, hypothetical (what-if) scenarios
- Explaining the function of a device in terms of its structure and behavior (e.g., how a functional requirement is satisfied). This will build on work in representing function [23]

- *Active documentation* of device behavior and intended behavior, in which stand-alone, explainable simulations with interactive, query-driven explanations are used as a medium for documentation of designs [10]
- Explaining modeling assumptions, such as the reasons for making approximations. This will require model formulation systems to reason explicitly about the choice of models, as explored in recent work [4,16,20]

Work on these problems is in progress.

Acknowledgments

The DME system is the product of members of the How Things Work project, including Richard Fikes, Yumi Iwasaki, Alon Levy, Chee Meng Low, Fritz Mueller, and James Rice. Brian Falkenhainer and Ken Forbus have been very influential and provided valuable comments on the draft.

References

[1] W. J. Clancey. The epistemology of a rule-based expert system: A framework for explanation. *Artificial Intelligence*, **20**:215-251, 1983.

[2] G. Dahliquist & Å. Björck. *Numerical Methods*. Prentice-Hall, Englewood Cliffs, NJ, 1974.

[3] B. Falkenhainer & K. Forbus. Self-explanatory simulations: Scaling up to large models. *Proceedings of the Tenth National Conference on Artificial Intelligence*, San Jose, CA, pages 685-690. AAAI press / The MIT press, 1992.

[4] B. Falkenhainer & K. D. Forbus. Compositional modeling: Finding the right model for the job. *Artificial Intelligence*, **51**:95-143, 1991.

[5] K. D. Forbus. Qualitative Process Theory. *Artificial Intelligence*, **24**:85-168, 1984.

[6] K. D. Forbus. The qualitative process engine. In Daniel S. Weld & Johan de Kleer, Ed., *Readings in Qualitative Reasoning about Physical Systems*, pages 220-235. Morgan Kaufmann, San Mateo, CA, 1990.

[7] K. D. Forbus & B. Falkenhainer. Self-explanatory simulations: An integration of qualitative and quantitative knowledge. *Proceedings of the Eighth National Conference on Artificial Intelligence*, Boston, pages 380-387. AAAI Press/ The MIT Press, 1990.

[8] K. D. Forbus & A. Stevens. Using qualitative simulation to generate explanations. *Proceedings of the Third Annual Conference of the Cognitive Science Society*, 1981.

[9] P. O. Gautier & T. R. Gruber. Generating Explanations of Device Behavior Using Compositional Modeling and Causal Ordering. *Proceedings of the Eleventh National Conference on Artificial Intelligence*, Washington, D.C., AAAI Press/The MIT Press, 1993.

[10] T. R. Gruber & D. M. Russell. Generative design rationale: Beyond the record and replay paradigm. In Thomas Moran & John H. Carroll, Ed., *Design Rationale*, Lawrence Erlbaum, in press. Available as Technical Report KSL 92-59, Knowledge Systems Laboratory, Stanford University.

[11] E. H. Hovy & Y. Arens. Automatic generation of formatted text. *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 92-95. AAAI Press/The MIT Press, 1991.

[12] Y. Iwasaki & C. M. Low. Model Generation and Simulation of Device Behavior with Continuous and Discrete Changes. *Intelligent Systems Engineering*, **1**(2), 1993.

[13] Y. Iwasaki & H. Simon. Causality in device behavior. *Artificial Intelligence*, **29**:3-32, 1986.

[14] A. M. Keuneke & M. C. Tanner. Explanations in knowledge systems: The roles of the task structure and domain functional models. *IEEE Expert*, **6**(3):50-56, June 1991.

[15] B. Kuipers. Qualitative simulation. *Artificial Intelligence*, **29**:289-388, 1986.

[16] A. Y. Levy, Y. Iwasaki, & H. Motoda. Using Relevance Reasoning to Guide Compositional Modeling. *Proceedings of the Second Pacific Rim International Conference on Artificial Intelligence*, Seoul, South Korea, 1992.

[17] C. M. Low & Y. Iwasaki. Device Modeling Environment: An integrated model formulation and simulation environment for continuous and discrete phenomenon. *Proceedings of the First International Conference on Intelligent Systems Engineering*, The Institute of Electrical Engineers, London,1992.

[18] K. R. McKeown. Discourse Strategies for Generating Natural Language Text. *Artificial Intelligence*, **27**(1):1-42, September 1985.

[19] J. D. Moore & W. R. Swartout. Pointing: A way toward explanation dialog. *Proceedings of the Eighth National Conference on Artificial Intelligence*, Boston, pages 457-464. AAAI Press/The MIT Press, 1990.

[20] P. P. Nayak. *Automated modeling of physical systems.* Ph.D. Thesis, Computer Science Department, Stanford University, 1992. Technical report STAN-CS-92-1443.

[21] D. Suthers, B. Woolf, & M. Cornell. Steps from explanation planning to model construction dialogues. *Proceedings of the Tenth National Conference on Artificial Intelligence*, San Jose CA, pages 24 - 30. AAAI press / MIT press, 1992.

[22] W. Swartout, C. Paris, & J. Moore. Design for explainable expert systems. *IEEE Expert*, **6**(3):58 - 64, June 1991.

[23] M. Vescovi, Y. Iwasaki, R. Fikes, & B. Chandrasekaran. CFRL: A language for specifying the causal functionality of engineered devices. *Proceedings of the Eleventh National Conference on Artificial Intelligence*, Washington, D.C., AAAI Press/The MIT Press, 1993.

[24] B. White & J. Frederiksen. Causal model progressions as a foundation for Intelligent learning. *Artificial Intelligence*, **42**(1):99-155, 1990.