01 Jan 1993

Functional Representation and Reasoning About the F/A-18 Aircraft Fuel System

M. Pegah

J. Sticklen

William E. Bond
Missouri University of Science and Technology, bondw@mst.edu

Functional Representation and Reasoning About the F/A-18 Aircraft Fuel System

Mahmoud Pegah, University of Northern Iowa
Jon Sticklen, Michigan State University
William Bond, McDonnell Douglas Aerospace

THIS WORK TESTS HOW WELL FUNCTIONAL MODELING CAN SCALE UP TO MEET A FORMIDABLE REAL-WORLD PROBLEM, AND EXTENDS THE FUNCTIONAL MODELING APPROACH TO INCLUDE A LIBRARY OF STANDARD PARTS.

A PIVOTAL CHALLENGE OF THE 1990s is to develop engineering methodologies that robustly address design for manufacturability, design to requirements, and conceptual design. For several years we have explored how functional reasoning, a subfield of model-based reasoning, can address such issues. In general, model-based reasoning circumvents the brittleness of reasoning systems built solely on associational knowledge. Model-based reasoning is attractive because it captures an intuition that is especially cogent in engineering: to troubleshoot a device, redesign one to new specifications, and so on, it is useful to represent and reason with a model of the device — to know how the device "works."

There are two variations on model-based reasoning: One focuses on how models of behavior are derived,[1-4] while the other focuses on how models of behavior are used.[5-9] The latter includes functional reasoning, which assumes that when we know what a device will be used for (its purpose), we can better organize our causal knowledge of the device. This approach uses abstractions of a device's purpose to index behaviors that achieve that purpose. Our variation on this method — functional modeling — also uses simulation as a core reasoning strategy: We first decompose the complex causal knowledge of a device along functional lines, then we compose a causal story of how the device will operate in a particular situation given stated boundary conditions. That is, we use both representational decomposition for managing complexity and situation-specific composition for simulation. (A sidebar describes the basics of functional modeling in more detail.)

Our long-term goal is to demonstrate that the functional approach can capture causal understanding about complex devices across diverse domains; we focus here on a realistically sized aerospace application: the fuel system of the F/A-18 aircraft, manufactured by McDonnell Douglas for the US Navy. In this research, we did not want to implement and field an "industrial strength" computer system for the engineering mainstream. Rather, we hoped to demonstrate that the functional approach could scale to some real-world aerospace problems, and to extend the representational power of functional modeling to include libraries of standard parts.

Many model-based reasoning approaches have been difficult to scale up from demonstration-sized problems. This is especially evident in research aimed at digital electronic circuit diagnosis. We use the term "scalability" here in two senses. It refers both to a knowledge-based system's ability to adapt to larger domains and to a system's ability to apply to more domains. Our initial work using functional modeling was in a small, well circumscribed domain with about 20 components: the human complement system (part of the immune system). We have argued that the inherent modularity of our

0885/9000/93/0400-0065 $3.00 © 1993 IEEE

Basics of functional modeling

Functional modeling largely adopts an existing formalism for functional representation that centers on the organization of causal device knowledge.[1] To represent a device functionally, we first describe "what it is" by recursively decomposing it into its constituent subdevices, which are related by a ComponentOf relation. In engineered devices, this decomposition typically parallels major structural systems. We then describe "how it works" by enumerating the functions of each subdevice and describing the behaviors that accomplish those functions. A function has three elements: a Provided clause stating the conditions under which the function is applicable (a precondition); a ToMake clause stating the function's result (a postcondition); and a By clause pointing to the causal description of how the function is implemented. (The "functional role" we mentioned earlier is actually a fourth element of a function description; we describe it in the article.)

In functional modeling, *behaviors* implement abstractly stated functions. Behaviors are directed graph structures in which the start nodes test the device's state variables, and the other nodes describe changes in state variables. Behaviors resemble fragments of causal nets: Each fragment carries (in its start nodes) predicates that indicate when the fragment is applicable, but the edges of the directed graph are annotated and point to an elaboration of why each node transition takes place. These annotations are either pointers to "world knowledge" or to other parts of the functional description itself (lower level functions or behaviors).

Once we have constructed a functional representation, we can understand a device's functionality by following a chain through lower and lower levels of subdevices:

device => function => behavior
=> subdevice => function => behavior ...

However, we might not have to follow this path to the very lowest levels if we find a level where a particular functionality of some underlying subdevice can be "assumed true." This ability to probe only as far as needed follows directly from the modularity of representation. (Functional descriptions are naturally modular: A subdevice can be replaced by a totally different subdevice that accomplishes the same functions.) Put another way, there is a implicit natural "layering of understanding" from the most abstract levels of device description to the most detailed.

Overall, functional modeling comprehends complex devices using a twofold divide-and-conquer strategy: It decomposes devices into subdevices, and causal knowledge into behaviors that are indexed by abstractly stated functions. The representational aspects of functional modeling parallel those of the work of Sembugamoorthy and Chandrasekaran,[1] but the computational goals differ. We use our representations to find the consequences of a particular set of boundary conditions. This amounts to building a full state-change diagram (a specialized causal net) from the fragments that exist in the behaviors of the functional representation. There is symmetry between the foundation of our representational viewpoint (decomposition to handle complexity) and the core of our computational process (composition tailored to a particular context).

The algorithm for finding consequences works as follows:

(1) *Specify the initial conditions.* The device variables have "default values," so only variable bindings that are not the device's normal state need to be set. Likewise, only missing functions or altered functions need to be explicitly input.

(2) *Determine the starting functions/behaviors.* Once the initial conditions are specified, they can be used to index behaviors and functions that would be applicable under those conditions. Redundant functions/behaviors are first "filtered" out; for example, if functions F_a and F_b have the same Provided clause (the same precondition), and F_a contains a knowledge pointer to F_b, then we should filter out F_b. The filtering leaves us with the "invocable functions/behaviors." If there are no invocable functions/behaviors, the functional reasoner halts.

(3) *Starting with the invocable functions/behaviors, construct a new state-change graph structure for the current situation.* Each node in this structure — which is called a *particularized state diagram* (PSD) — is a partial state description that points to a variable of the device and a statement about how that variable is altered. The PSD is constructed by traversing each applicable behavior:

- When at a partial state, put a corresponding node into the PSD to mark a partial state change, and update the associated state variable database accordingly.
- When at an annotation that cannot be decomposed, remember that succeeding partial states assume whatever the annotation points to but make no changes in the PSD that is being built.
- When at a decomposable annotation (that is, another function or behavior), remember that succeeding partial states assume the function/behavior pointed to, and expand the function/behavior pointed to whenever possible. To determine whether a given function/behavior can be expanded, check its starting predicates.

(4) *Continue expanding the annotation links until there are no more decomposable links.*

In other words, a PSD is built by following all decomposable annotations that were in the starting behaviors and expanding them recursively until what is left is a PSD with only partial state transitions. Each node in the PSD contains knowledge of the state variable it alters and the nature of the alteration. In addition, each node contains a listing of the assumptions under which this state change takes place. Once the PSD has been constructed, it is easy to determine what the effect on the device will be by traversing the PSD and noting cumulative changes in the device's state description variables.

Reference

1. V. Sembugamoorthy and B. Chandrasekaran, "Functional Representation of Devices and Compilation of Diagnostic Problem-Solving Systems," in *Experience, Memory, and Learning*, J. Kolodner and C. Reisbeck, eds., Lawrence Erlbaum Associates, Hillsdale, N.J., 1986.

approach would let it scale to larger domains with few changes to the device representation or reasoning techniques, but we needed to test that claim empirically. We have also argued that the basic issues of device representation and reasoning cut across various types of domains, but we again needed a practical test to substantiate the claim. Our representation of the F/A-18 fuel system currently includes 89 component devices, 92 functions, 118 behaviors, and 181 state variables. Although still not overwhelmingly large, the system is an order of magnitude more complex than any system yet represented functionally.

Although a library of standard parts can make it easier to build device models, we expected the addition of a library facility to functional modeling to excite little theoretical interest. However, our work in this area pointed the way to another level of device organization that we had not anticipated — called the functional role — which we are now developing as an added feature

of the functional repertoire. Also, our experience acquiring knowledge for this model (by reverse engineering from a technical manual) indicates that a functional approach provides a strong backbone for reverse engineering.

Knowledge acquisition

We obtained most of the project's detailed knowledge from a technical manual of the F/A-18 fuel system, containing schematics of the fuel system and information about the operation of components. However, the manual included no direct information about the intended engineering use of the various components or subsystems. We gathered this information in three phases. We first obtained a top-level understanding of the fuel system in several interviews with McDonnell Douglas engineers. This phase was relatively short (about three weeks). We then used this understanding of the system's intended purpose as a starting place for reverse engineering a full functional model from the technical manual. Our top-level understanding of the entire system helped us organize our causal understanding of the components at the next lower level, which in turn guided the development of deeper and deeper levels of understanding. This recursive process continued until we reached the system's most detailed level. This part of the project took about two years, far more than any other (the researcher responsible for this task was a graduate student at the time; less time would likely be needed in an industrial setting).

Finally, we informally tested the completed model to see if the model properly captured system redundancies that are inherent in the actual fuel system and to see if McDonnell Douglas engineers found the functional approach promising for engineering modeling. We obtained positive results on both counts, and we based our final changes in the model on feedback from the engineers.

The difficulty of the reverse engineering phase illustrates why a description of a device's purpose or goal is an important part of its representation. Design engineers usually design physical artifacts to accomplish a specified set of requirements. Reverse engineering, on the other hand, involves developing a sufficient understanding of how an existing artifact "works" to, for example, redesign it to altered specifications. The central goal is to determine how the original engineer intended the device to function.

But most artifact descriptions are similar to blueprints, which represent the artifact's physical structure but not the function or purpose of its subsystems. In principle, such blueprint representations contain all the knowledge necessary to understand how the artifact works, but assimilating that knowledge typically involves assigning purpose to the subsystems based on their structure. This very formidable task requires determining large-scale behavior from structure and small-scale component behavior (qualitative physics), and selecting from those possible behaviors the small subset of behaviors intended by the original design engineer. We can avoid this task, though, if we include a description of purposes or goals from the start. Thus, it should not be surprising that functional techniques form a natural template for reverse engineering.

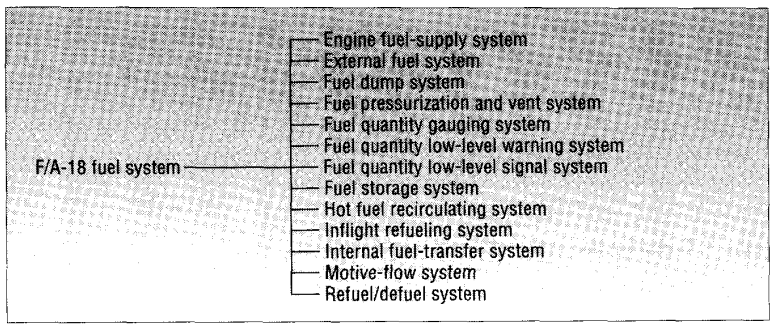

Figure 1. Top-level components of the F/A-18 fuel system.

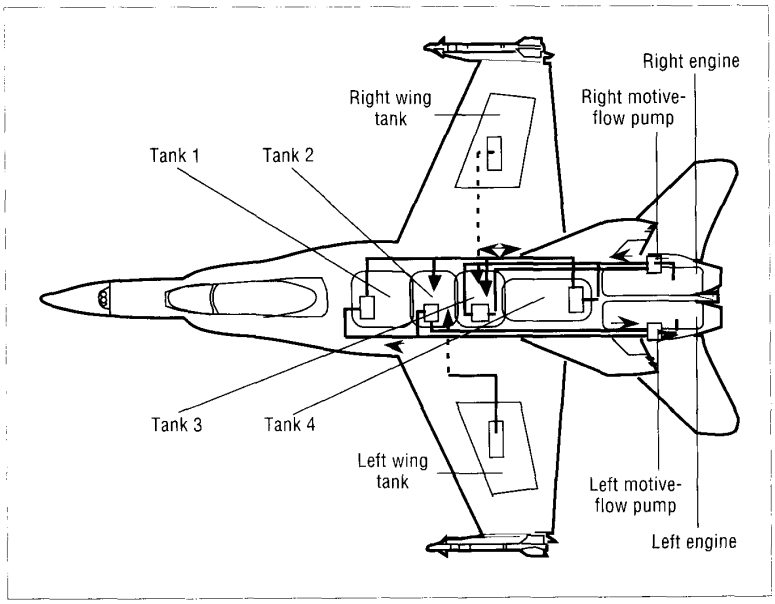

Figure 2. The F/A-18 fuel system.

The F/A-18 fuel system

We decomposed the F/A-18 fuel system into the 13 major subsystems shown in Figure 1, concentrating on two that contain about 70 percent of the system's components: the internal fuel-transfer system and the motive-flow system.

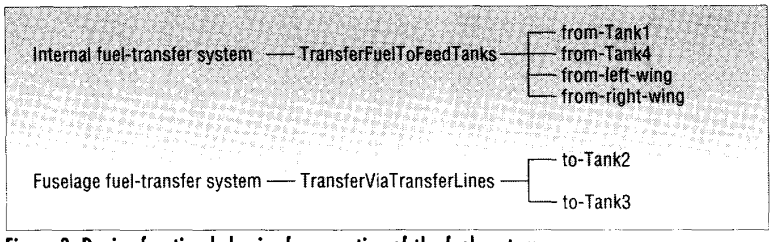

Figure 3. Device-function-behavior for a portion of the fuel system.

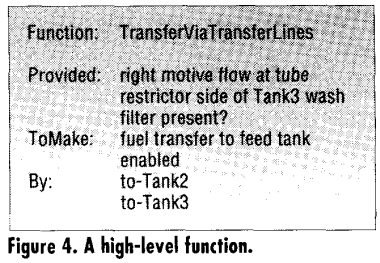

Figure 4. A high-level function.

The internal fuel-transfer system includes a fuselage fuel-transfer system and a wing fuel-transfer system. Its top-level goal is to deliver fuel from four transfer tanks (the two wing tanks and Tanks 1 and 4 in Figure 2) to two engine feed tanks (Tanks 2 and 3). The fuselage fuel-transfer system pumps enough fuel from either transfer tank to either feed tank to ensure that the feed tanks are always full. The wing fuel-transfer system pumps fuel from the right wing tank to Tank 3, and from the left wing to Tank 2.

An engine fuel-supply system then moves the fuel from the feed tanks to the engines. This system is powered by two motive-flow/boost pumps, which basically use the fuel as a hydraulic fluid. Tank 2 and the left motive-flow pump supply fuel to the left engine; Tank 3 and the right pump supply fuel to the right engine. The motive-flow pumps also power the engine fuel turbine pumps, the fuel dump system, and internal fuel transfer.

Let's look at part of the representation of the internal fuel-transfer system. Figure 3 indicates its function (expanded in Figure 4) and the behaviors that implement it, plus the function and behaviors for the fuselage fuel-transfer system. Figure 5 shows the to-Tank3 behavior that enables fuel transfer to Tank 3; the annotation on the first link points to another behavior (shown in Figure 6) that we can examine if we want more information about this link.

Figure 6 shows how a functional representation can "bottom out" when a causal transition need not be explained further for the reasoning task. The annotations all point to world knowledge about incompressible fluids rather than to deeper parts of the representation. The behavior basically states that four points in the fuel system are connected. We could have used more detail to explain the causality (fuel can be treated as an incompressible fluid, so when an upstream point has pressure, connected downstream points also have pressure), but for our purposes it was sufficient to refer to our store of world knowledge.

Reasoning about the internal fuel-transfer system. Now let's step through the algorithm we outlined in the sidebar.

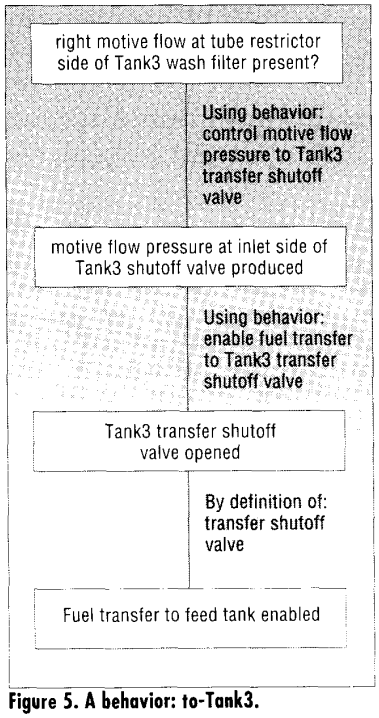

Figure 5. A behavior: to-Tank3.

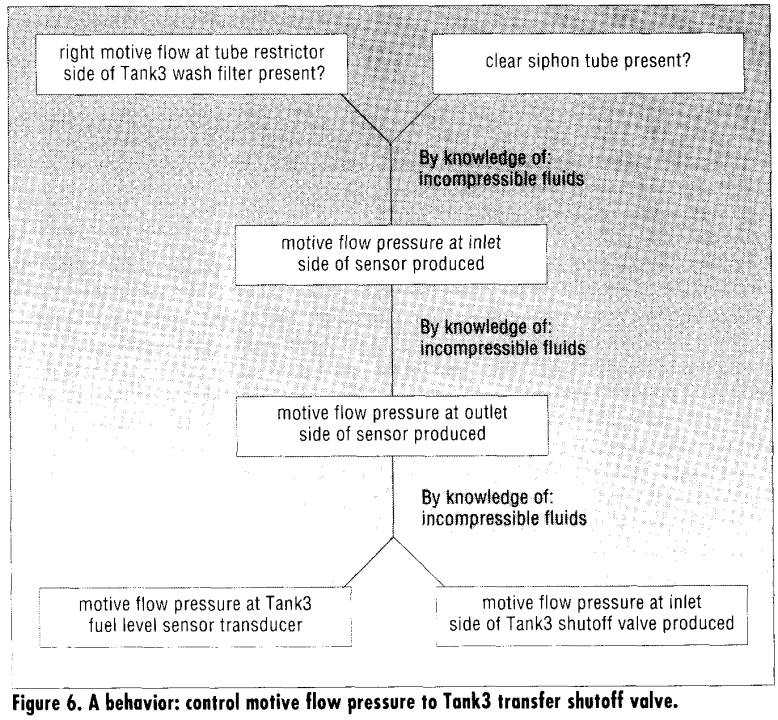

Figure 6. A behavior: control motive flow pressure to Tank3 transfer shutoff valve.

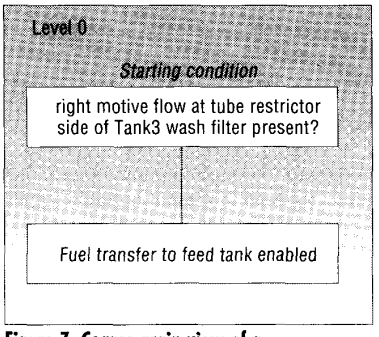

Figure 7. Coarse grain view of a particularized state diagram.

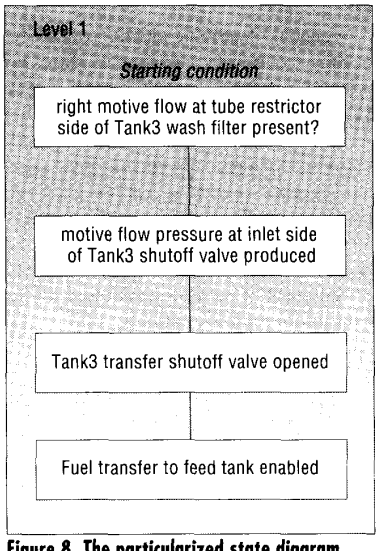

Figure 8. The particularized state diagram after one round of expansion.

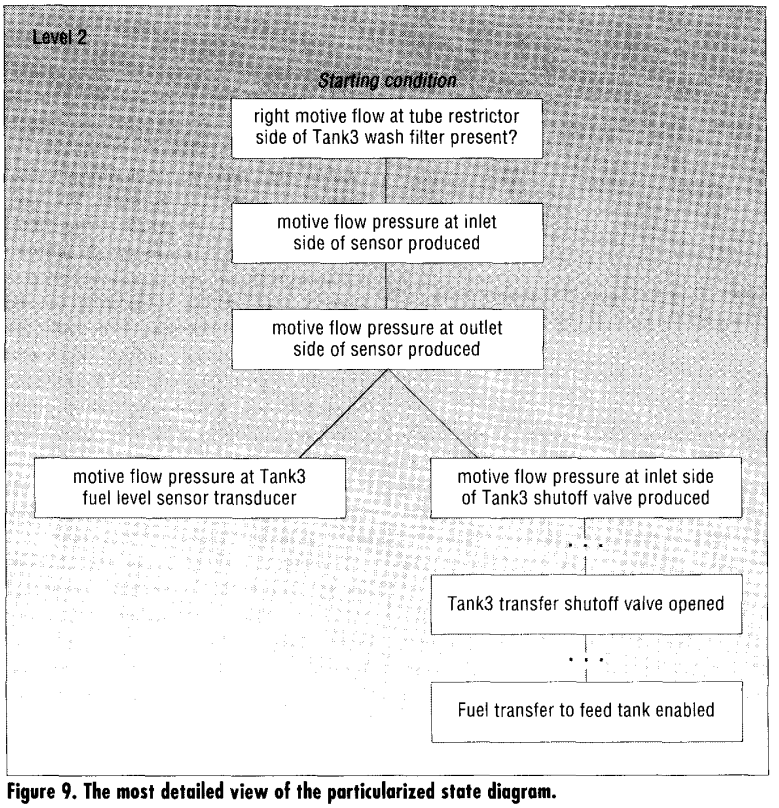

Figure 9. The most detailed view of the particularized state diagram.

First, we'll specify the initial condition as the presence of right motive-flow fuel at the tube restrictor side of the Tank 3 wash filter. Then we determine the applicable functions or behaviors based on this condition; in this case the function in Figure 4 applies.

Next, we build a particularized state diagram: Starting from the high-level functions (or behaviors) just identified, we use the link annotations to index lower level functions and behaviors whose preconditions are met. Those functions and behaviors whose predicates are true are expanded and "spliced" into the place originally held by their links in a process similar to macro expansion techniques in software languages. The process is recursively applied until no more decomposable links remain (until there are no more links pointing to behaviors or functions).

At the highest level, we would see a simple causal net-like structure, as in Figure 7. After the first iteration, the behavior in Figure 5 would be "spliced" into the diagram (see Figure 8). After another round, the behavior in Figure 6 would be added, yielding the final, detailed diagram in Figure 9. (The behavior in Figure 5 had two other annotations that we did not discuss; the fact that they would also add to this diagram is acknowledged by the dots near the bottom of Figure 9 acknowledge that.)

Once we have produced a complete particularized state diagram, it is straightforward to determine the cumulative effects (the consequences) of the initial conditions by traversing the graph structure and keeping a running tally of all changes made to state variables. The cumulative effects are then read from this "tally sheet."

The standard library

One of the most tedious and error-prone aspects of design is the need to copy the same type of component many times. We have developed a device library for the F/A-18 fuel system, similar to the standard parts libraries in most CAD systems. The library uses a type hierarchy that supports inheritance for modeling lower level objects (see Figure 10). For example, the fuel system has more than 90 different valves, each of which inherits functions for enabling and disabling flow from the device Valve. But for each type of valve we can specify behaviors that determine how those functions are carried out.

The library is straightforward to implement, but properly "connecting" a standard part with the rest of the model is not so straightforward. There are two types of connections in a functional model: The standard part's state variables must be mapped into the overall model, and any

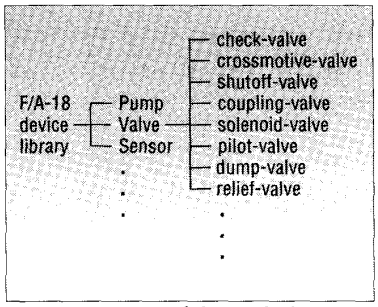

Figure 10. A portion of the standard parts library for the F/A-18 fuel system.

physical connections (pipes, in our case) must be properly attached from the standard part to the overall model. Our initial solution to both problems relies on the system's user to make the necessary connections. We rely on the computer system only for bookkeeping functions.

We initially developed the library facility to help developers build functional models more quickly. However, we have since noted a strong potential synergism between our library utility and research reported by Nayak, Addanki, and Joscowicz.[10] They suggest representing the primitive behaviors of high-level models in a context-dependent manner as a way to automatically select appropriate models. Our functional-modeling framework, especially the standard library, extends this notion by allowing selection of functional components based on context-dependent information.

A modeler could use the standard library as a static repository of parts. For example, when describing a portion of the fuel system model, the user could directly instantiate a "relief value." This approach does not require a hierarchical organization except to help the user to find the appropriate subdevice more quickly. A user could also use the library to automatically choose the appropriate device based on constraints imposed by the functional requirements. This situation would use a hierarchical organization but would not require any new epistemic analysis; the objects under the subdevice Valve are simply types of Valve.

Finally, a user could select Valve without the current functional constraints being strong enough to force the selection of a single type of valve. During simulation, however, the need for a given group of functions might become strong enough to force a choice. This situation is more interesting than the previous two. We have so far thought of devices as being made up of components, functions, and behaviors. Individual functions are context dependent because of their preconditions. We have recently developed the notion of the "functional role," which lets us group multiple functions within a device and determine a group's applicability depending on the context. So, for example, if a hydraulic system has a thermal ballast, we could represent it as having one set of functions that stabilize the system temperature, while a second set stores hydraulic fluid. The functional role is a natural extension of functional modeling that provides an additional and higher level indexing capability to causal understanding of a device.

We plan to fully elaborate the standard library facility to implement functional roles. We have experimented with functional roles as a CAD aid for building functional models and as a constraint on the choice of particular types of devices. However, we have yet to develop functional roles as a way of organizing alternative groups of behaviors in a device and selecting among these groups at runtime. We intend to accomplish both as a final extension to our work on the F/A-18 fuel system. We have also started to examine how a functional representation could directly support troubleshooting, specifically in the external thermal control system of Space Station Freedom.

IN THEIR SURVEY OF MODEL-based reasoning, Davis and Hamscher identified domain independence, scalability, and model selection as crucial research issues.[6] Although the survey is explicitly for the area of troubleshooting, we can raise the same three issues for all areas of model-based reasoning. In our research, we have successfully applied the same basic reasoning strategy and representation primitives to two very different domains (engineering and medical diagnosis), although we augmented the representational primitives for the engineering task. Also, the inherently modular nature of a device's functional representation makes scalability straightforward (by adding new subdevices).

Model selection is the most interesting of the three issues and the hardest to pin down, partly because it is a multidimensional task. Along one dimension, we must select the level at which we want to represent our model. As Davis points out, no model is complete. But functional modeling lets us point to "world knowledge" as the reason for a given state-variable transition (in a behavior), thereby letting us construct models that "bottom out" at whatever level is appropriate. We determine the appropriate level by evaluating whether the world knowledge can be treated as a monolithic entity for purposes of the current model.

We should determine which type of model to construct based on whether its representational primitives can express our device knowledge and whether the output of its reasoning matches our needs. This might seem self-evident, but for the most part, model-based reasoning has not dealt explicitly with these issues in these terms. One of the strongest arguments for the functional approach is the relative clarity of statement of its representational primitives and reasoning methods.

Acknowledgments

We are indebted to Ahmed Kamel of Michigan State University's AI/KBS Lab, and Barry Flachsbart of McDonnell Douglas, whose comments on earlier drafts of this article were most helpful. Gene Wallingford played an important role in the functional modeling-based initial implementation of the F/A-18 fuel system at Michigan State University, from which the current model evolved. The model is built in a task-specific language shell that was implemented by Jumein Sun and Kurt Patzer in the AI/KBS Lab.

This research is supported in part by Independent Research and Development funds from McDonnell Douglas Research Laboratories. The broad research to extend and apply functional-reasoning techniques in the AI/KBS Lab is also supported by DARPA (ARPA 8673) and the National Science Foundation Center for High-Speed, Low-Cost Polymer Composite Processing at Michigan State University. Equipment used in the AI/KBS lab has been generously supplied by Apple Computer.

References

1. J. deKleer and J.S. Brown, "A Qualitative Physics Based on Confluences," *Artificial Intelligence*, Vol. 24, 1984, pp. 7-83.
2. K.D. Forbus, "Qualitative Process Theory," *Artificial Intelligence*, Vol. 24, 1984, pp. 85-168.
3. B. Kuipers, "Commonsense Reasoning about Causality: Deriving Behavior from Structure," *Artificial Intelligence*, Vol. 24, 1984, pp. 169-203.
4. T. Bylander, "Consolidation: A Method for Reasoning about the Behavior of Devices," Ohio State University, Columbus, Ohio, 1986.
5. J. deKleer and B. Williams, "Diagnosing Multiple Faults," *Artificial Intelligence*, Vol. 32, 1987, pp. 97-130.
6. R. Davis and W. Hamscher, "Model-Based Troubleshooting," in *Exploring Artificial Intelligence*, H. Shrobe, ed., Morgan Kaufmann, San Mateo, Calif., 1988.
7. V. Sembugamoorthy and B. Chandrasekaran, "Functional Representation of Devices and Compilation of Diagnostic Problem-Solving Systems," in *Experience, Memory, and Learning*, J. Kolodner and C. Reisbeck, eds., Lawrence Erlbaum Associates, Hillsdale, N.J., 1986.
8. D. Franke, "Representing and Acquiring Teleological Descriptions," *Proc. 1989 Workshop on Model-Based Reasoning*, pp. 62-68.
9. J. Sticklen, B. Chandrasekaran, and W. Bond, "Distributed Causal Reasoning," *Knowledge Acquisition*, Vol. 1, 1989, 139-162.
10. P.P. Nayak, S. Addanki, and L. Joscowicz, "Modeling with Context Dependent Behaviors," *Working Papers for the Second AAAI Workshop on Model-Based Reasoning*, 1990, pp. 1-8.

Jon Sticklen is an assistant professor of computer science at Michigan State University and a principal in the AI/Knowledge-Based Systems Laboratory. His research interests center on model-based reasoning and extensions to the generic-task viewpoint for task-specific architectures. The domains in which he is working include designing and fabricating composite materials, capturing design knowledge in engineered artifacts, troubleshooting engineered devices, agromanagement decision support systems, and capturing model-level ecological knowledge. He received his PhD in computer science from Ohio State University, his MS in astronomy from Columbia University, and his BS in physics from Ohio State University. He is a member of the IEEE.

William E. Bond is a senior technical specialist in the New Aircraft and Missile Products division of McDonnell Douglas Aerospace, where his work focuses on developing and applying machine learning, neural networks, and model-based learning. He received a PhD from Rensselaer Polytechnic Institute.

Readers can reach the authors in care of Jon Sticklen, Michigan State University, Computer Science Department, A714 Wells Hall, East Lansing, MI 48824-1027; Internet, sticklen@cps.msu.edu

Mahmoud Pegah is an assistant professor of computer science at the University of Northern Iowa, a doctoral candidate in computer science at Michigan State University and a member of its AI/KBS Laboratory. (His photo was not available at press time.) His research interests include model-based reasoning, abductive problem solving, and distributed/visual processing systems. Pegah received his MS in computer science from Michigan State University. He is a member of AAAI, ACM and IEEE.