Device Representation and Reasoning with Affective Relations

James M. Crawford*

CIRL
1269 University of Oregon
Eugene, OR 97403
U.S.A.

Daniel L. Dvorak, Diane J. Litman
Anil K. Mishra, Peter F. Patel-Schneider

AT&T Bell Laboratories
600 Mountain Avenue
Murray Hill, NJ 07974
U.S.A.

Abstract

Device representation and reasoning with affective relations occupies a middle ground between classical model-based diagnosis and heuristic expert systems. A device is modeled by specifying a set of diagnostically motivated affective relations among its components. Reasoning is then performed by a set of inference rules that reason with the model to propagate symptoms through the components. Representation and reasoning with affective relations extends several benefits of classical model-based diagnosis—the model as a unifying framework for knowledge, methodical coverage of the domain, and diagnostic reasoning based on equipment design and causality—to a class of problems where classical model-based diagnosis cannot be applied because the required models cannot be reasonably obtained or represented. Our work evolved from our redesign of a heuristic expert system for monitoring long-distance telephone switching systems, and is applicable to highly complex self-checking (fault-tolerant) systems.

1 Introduction

Following their advent in the 1970's, expert systems were successfully applied in a number of commercially important areas. However, as these systems grew in size and complexity, it became clear that they had important limitations. For example, large collections of interacting rules turn out to be difficult to understand or modify. These limitations can be traced to a set of now well-known problems with "surface" or "first-generation" expert systems [?; ?], for example, unintended rule interactions, conflicting rules, and knowledge implicit in the rules.

To address these problems, a wide variety of "second-generation" expert systems have been proposed in the literature, encompassing both "deep" approaches (e.g., knowledge level analyses [?], deep compiled knowledge [?], model-based reasoning [?], functional reasoning [?]), as well as hybrid approaches that integrate multiple representation and reasoning techniques. For example, in the domain of device diagnosis, model-based approaches generally represent the structure and behavior of the device declaratively and then reason about the device from first principles (rather than via a collection of ad-hoc heuristics).

Generally speaking, the commercial impact of second-generation expert systems has not matched the impact of their simpler first-generation counterparts. A major factor in this relative lack of commercial impact of second generation approaches, especially the more sophisticated approaches based on model-based reasoning, is the high up-front cost of developing a model of complex real-world devices. To take an extreme but very real example, consider the problem of developing a model of a modern telephone switching system (or some other equally complex device such as a nuclear power plant or large chemical refinery). A telephone switch can easily have thousands of circuit packs, each of which is of the complexity of a personal computer. Clearly any behavioral or functional model of such a device will have to be at quite a high level. Unfortunately, there do not seem to be any good high level languages for even giving a behavioral or functional description of these complex devices. One might expect to be able to use some mathematical model, queuing theory for example, but we are not aware of any evidence that such an approach would be useful. It is clear that such an approach would be almost completely disjoint from the approaches that domain experts take to monitoring and diagnosing these types of devices.

To understand the approach that domain experts take, it is essential to understand that modern complex devices are *largely self-checking*.[1] That is, they generally

*Add CIRL grants. Previously at AT&T Bell Laboratories.

[1]We use the term "self-checking" to refer to built-in ca-

monitor their own operation and produce *alarms* that indicate the nature of most failures. This type of self-checking is, of course, essential precisely because humans cannot keep an adequate model of such complex devices in their head. To produce commercially important monitoring and diagnostic systems in these domains it suffices to automate the largely mechanical process that domain experts use to understand the alarms produced by these complex self-checking devices.

Unfortunately most existing work on first- and second-generation expert systems provides relatively little help in this endeavor. First generation approaches tend to break down due to their well-understood limitations. Second-generation model-based approaches are not directly applicable because of the lack any real *model* of these complex devices.

This paper presents an approach based on *affective relations*. Affective relations define a highly abstract non-behavioral representation for modeling devices. An affective relation is so named because one component *affects* another in some diagnostically important way. The affective relation model is not used to simulate device behavior (in fact it is much too weak for that). Instead the model is a way to organize knowledge about the domain in a coherent way; ad-hoc heuristics are replaced by a small set of general principles for monitoring and diagnosis from affective relation models. Unlike traditional device models, affective relations are easy to acquire from front-line experts such as operators and maintenance technicians because they represent precisely the kind of knowledge that is needed to diagnose routine problems, particularly in complex self-checking devices.

We illustrate our approach by presenting a rational reconstruction of an existing rule-based monitoring and diagnosis system for the 4ESS®telephone switching system (the 4ESS handles most AT&T domestic long-distance calls). By comparing the older rule-based design with the design based on affective relations, we demonstrate how the use of affective relations (like traditional model-based approaches) provides generality, clarity and conciseness that allows for increased inferential power and reduced maintenance costs.

The larger import of our work is the identification of a middle ground between surface approaches and traditional model-based approaches. Our experience shows that a model of the affective relations between the components of a device is a particularly useful level of representation because it brings many of the benefits of model-based reasoning to domains where full device models cannot be built.

pabilities ranging from verification tests that signal when a device is malfunctioning to limited diagnosis that identifies suspects from evidence localized in time and space.

2 Affective Relations

Affective relations models are applicable to domains satisfying two basic criterion:

1. Devices can be naturally broken down into components such that the components generally interact via known pathways.
2. These pathways can be characterized by a reasonably small set of affective relationships between components.

Abstractly, reasoning with affective relations proceeds by examining the output of the device to determine the state of key components, and then propagating this information through the affective relations to deduce the state of other components. Actions are then performed based on the nature of the alarm messages from the device and the states of the components. To make this more concrete the rest of this section presents an example of an affective relation model.

2.1 Device Representation using Affective Relations

The key elements of our device model are 1) a set of basic *classes* (e.g., Device), 2) a set of diagnostically motivated *affective relations* relating the classes, and 3) a set of properties and general relationships specifying the semantics of the affective relations, i.e., the knowledge that is implicit in any explicitly specified knowledge. See Figure 1 for a knowledge level [?] description of a portion of the device model[2].

Affective relations link components to other components in diagnostically useful ways. For example, `immediate-part-of` means that a direct subcomponent/component relationship exists between two devices. `Part-of` reflects direct as well as transitively closed subcomponent/component relations, while its inverse `subpart` reflects direct and transitive component/subcomponent relations. `Depends-on` means that the correct functioning of a device depends on the correct functioning of another device; `dependent` represents the inverse relation between the two devices. `Standby` means that if one device fails, its standby partner will take over automatically; it also means that reliable operation is jeopardized if one device fails. `Self-alarm` means that a device is signaling an alarm. However, a problem with one device may not necessarily cause that device to itself signal an alarm; instead, the problem may affect a second device, causing *it* to signal an alarm. `Part-alarm` and `dependent-alarm` link devices to alarms signaled by other devices. `Part-alarm` relates a device to an alarm on another device when the alarming device is a `subpart` of the device; `dependent-alarm` relates devices to alarms based on `depends-on`. Devices are either `functional` or

[2]We use typewriter font for `relations` and sans serif for classes (and their members).

Classes: Device Alarm Boolean

Affective relations, and their domain and range:

`immediate-part-of`	Device, Device	`dependent`	Device, Device	`dependent-alarm`	Device, Alarm
`part-of`	Device, Device	`standby`	Device, Device	`functional`	Device, Boolean
`subpart`	Device, Device	`self-alarm`	Device, Alarm	`simplex`	Device, Boolean
`depends-on`	Device, Device	`part-alarm`	Device, Alarm		

Properties, and the affective relations they characterize:

Symmetric: `standby`
Transitive: `part-of`, `subpart`
Inverses: (`part-of`, `subpart`), (`depends-on`, `dependent`)

General relationships between affective relations:

$\forall$ x $\forall$ y [`immediate-part-of`(x,y) $\Rightarrow$ `part-of`(x,y)]
$\forall$ x $\forall$ y [`part-alarm`(x,y) $\equiv$ $\exists$ z (`subpart`(x,z) $\wedge$ `self-alarm`(z,y))]
$\forall$ x $\forall$ y [`dependent-alarm`(x,y) $\equiv$ $\exists$ z (`depends-on`(z,x) $\wedge$ `self-alarm`(z,y))]
$\forall$ x $\forall$ y [(`depends-on`(x,y) $\wedge$ `functional`(y,false)) $\Rightarrow$ `functional`(x,false)]
$\forall$ x [`simplex`(x,true) $\equiv$ $\forall$ y (`standby`(x,y) $\Rightarrow$ `functional`(y,false))]

Figure 1: Device modeling with affective relations: A knowledge level description.

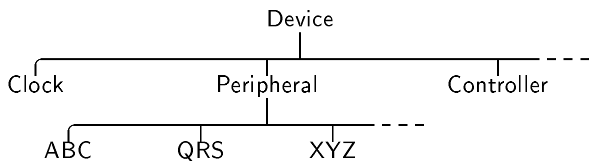

Figure 2: Partial taxonomy of 4ESS device types.

not. If a device depends on another device that is not functional, the dependent device is also not functional. If a device is in `simplex` mode, it means that none of the device's standbys are functional.

To illustrate the use of these relations, we examine in some detail a digital toll switching system known as a 4ESS switch. A portion of the basic class hierarchy is shown in Figure 2.[3] The class Device is broken down into a series of 4ESS-specific subclasses. Each major type of component of the 4ESS (down to the level of field-replaceable units) adds a class to the general device ontology shown in Figure 1.

Figure 3 shows how some of the key portions of the model of the 4ESS switch are instantiated using the general and domain-specific portions of the ontology shown in Figures 1 and 2, respectively. The actual 4ESS switch and its components are each defined as members of one of the device classes, e.g., Device(4ESS) defines the switch 4ESS to be an instance of type Device. Note that the model is *isomorphic* to the device: for each type of component in the device there is a class in the model and for each component in the device there is one instance in the model. Affective relations are asserted between device instantiations, e.g., `immediate-part-of` (Clock 0, 4ESS). Recall that the properties of and the general relationships between the affective relations (Figure 1) specify the knowledge that is implicit in an explicitly defined model such as Figure 3. Thus, because of the first general relationship in Figure 1, `immediate-part-of` (Clock 0, 4ESS) entails `part-of` (Clock 0, 4ESS), which in turn entails the inverse relation `subpart` (4ESS, Clock 0). Similar inferences can be drawn from the other `immediate-part-of` assertions. Then, transitivity entails further `subpart` and `part-of` relations. The `depends-on` assertions entail a set of inverse `dependent` relations. Because `standby` is defined to be symmetric, `standby` (A-CNTL 0, A-CNTL1) entails `standby` (A-CNTL 1, A-CNTL0), and similarly for the other `standby` assertions.

[3]To eliminate jargon, hardware component names have been replaced with ABC, etc.

Figure 4 pictorially represents the model. Each box represents a device. The immediate nesting of boxes depict `immediate-part-of` relations. The labeled arrows depict some of the other explicit (e.g., `standby` (A-CNTL 0, A-CNTL1)) and implicit (e.g., `standby` (A-CNTL 1, A-CNTL0)) affective relations linking devices.

Note that affective relations among components are *not* traditional relations of structure or behavior or function except in some very abstract sense. For example, unlike typical behavioral models, the relations do not define or constrain the input/output behavior of components. Rather, affective relations express aspects of the design at a level of abstraction that expert human troubleshooters use to link symptoms to suspects. This will be illustrated below. Furthermore, the use of the model for diagnostic reasoning not only motivates but also limits what must be explicitly represented in the model. We do *not* need to represent every type of component or every relationship between components, but only those which participate in an expert's causal analysis of alarms.

Device(4ESS)	`depends-on`(A-CNTL 0, Clock 0)	`immediate-part-of`(Clock 0, 4ESS)
Clock(Clock 0)	`depends-on`(A-CNTL 1, Clock 1)	`immediate-part-of`(Clock 1, 4ESS)
Clock(Clock 1)	`depends-on`(Q-CNTL 0, Clock 0)	`immediate-part-of`(ABC 6, 4ESS)
ABC(ABC 6)	`depends-on`(Q-CNTL 1, Clock 1)	`immediate-part-of`(A-CNTL 0, ABC 6)
Controller(A-CNTL 0)		`immediate-part-of`(A-CNTL 1, ABC 6)
Controller(A-CNTL 1)	`standby`(A-CNTL 0, A-CNTL1)	`immediate-part-of`(SPC 0, ABC 6)
SPC(SPC 0)	`standby`(SPC 0, SPC 1)	`immediate-part-of`(SPC 1, ABC 6)
SPC(SPC 1)	`standby`(Q-CNTL 0, Q-CNTL1)	`immediate-part-of`(QRS 0, 4ESS)
QRS(QRS 0)		`immediate-part-of`(Q-CNTL 0, QRS 0)
Controller(Q-CNTL 0)		`immediate-part-of`(Q-CNTL 1, QRS 0)
Controller(Q-CNTL 1)		

Figure 3: A portion of the device model of the 4ESS switch.

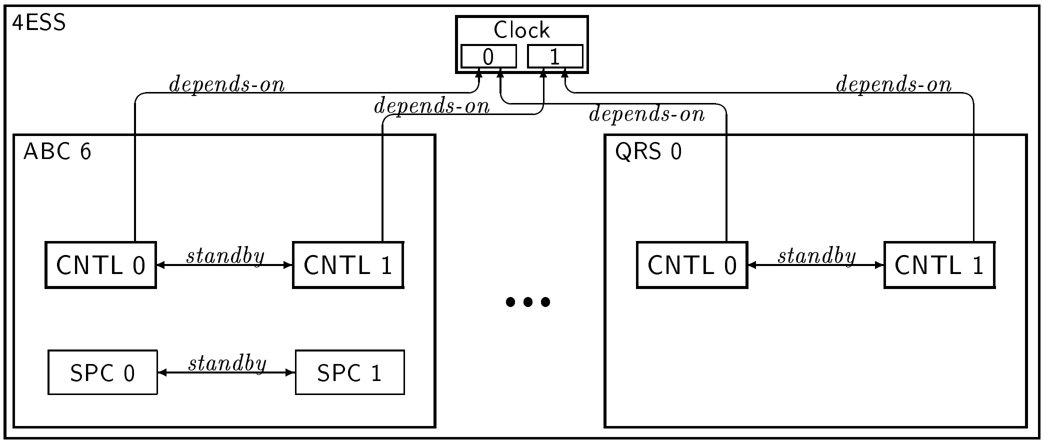

Figure 4: Pictorial representation of part of the 4ESS switch model.

2.2 Diagnostic Reasoning from the Device Model

Our use of an explicit model of the components of a monitored device and affective relations between components brings many of the benefits of traditional model-based reasoning to domains, such as 4ESS alarm monitoring, where conventional models are too costly to acquire. For example, one benefit is that the approach allows us to replace heuristic rules of first-generation expert systems with diagnostic rules that reason from the device model using general principles expressed in terms of the affective relations. In this section we informally contrast our approach to diagnostic reasoning in the 4ESS domain with a previously developed heuristic approach.

In our domain, diagnostic reasoning involves real-time monitoring of alarms and other informational messages from the 4ESS switches, e.g., to correlate alarm messages coming from the various components, ignore transient alarms, recognize chronic problems, run diagnostics, and signal problems requiring human attention. The monitoring system takes input in the form of alarm messages and produces output in the form of warning and action messages to field technicians. An expert system was originally developed for monitoring of the 4ESS switches in 1990. The system was designed as a conventional rule-based system using an OPS5-compatible language. It contained several hundred heuristics gleaned from many interviews with domain experts. Three example heuristics are shown below.

1. If *F-level* alarms are occurring concurrently for more than one component in a peripheral device, add the number of alarms together and threshold on the sum.

2. If a type-ABC component is in *normal* mode, perform thresholding of alarms, but if it is in *simplex* mode, ignore thresholding and alert on any *F-level* alarms.

3. If *F-level* alarms are occurring on different type-ABC and QRS components, but all alarms are on the same *controller* number, then suspect the *clock*.

Device representation and reasoning with affective relations allows us to replace such heuristics with diagnostic rules that reason from our device model using first principles. For example, one such general principle is:

- Concurrent alarms on multiple components of a larger device indicate a problem at the device level.

We capture this principle by using part-alarm to link alarms on components to any larger devices the components are part of. We then threshold on the number of such links (looking for a device linked to a large number of alarms on different subparts). This subsumes the first heuristic rule above and is clearly both simpler and more general. For example, the new rule covers all types of alarms and all levels of nesting in the part-of hierarchy. Of course this reformulation is only possible because of the explicit representation of the immediate-part-of, part-of, and subpart relations between components.[4]

We can also take advantage of the explicit encoding of standby. In the normal situation where a component is active and its standby is ready, alarms undergo thresholding before an alert is issued. However, if the standby is out of service then *any* alarm should generate an immediate alert. By making standby explicit in our model, and defining simplex in terms of the affective relations, the second heuristic rule above can be replaced with the following more general principle:

- If a device's standby is not functional, then alert on any alarm.

Again, note that the new rule covers *all* devices that have a standby, not just ABCs, and similarly covers all types of alarms.

Finally, a dependency-based rule of monitoring allows us to recognize the failure of a device from alarms on the device's dependents. For example, if Clock 0 in Figure 4 is failing, this may be indicated by alarms on the dependent components CNTL 0 in ABC 6 and CNTL 0 in QRS 0. By making depends-on explicit in our model, the third heuristic rule above (whose rationale for implicating the clock is particularly obscure) can be replaced with the following more general principle:

- Concurrent alarms on multiple dependents of a device indicate a problem with that device.

This rule clarifies that diagnostic reasoning traces up a dependency chain to identify a common point of failure. Depends-on can also be exploited to propagate state information, as was shown in the fourth relation of Figure 1.

2.3 A Detailed Example

The basic activity in the 4ESS expert system is the analysis of alarms. Alarm messages from the switch, in conjunction with the semantics of the model, trigger the dynamic creation of the relevant portions of the device model. From this model general purpose diagnostic rules determine the root cause of the alarm.

At startup, the model of the 4ESS only contains a small set of core device types, associated instantiations, and immediate-part-of relations between the instantiations. This is because in a complex device such as the 4ESS, we have found it to be prohibitively difficult to build up a complete device configuration in advance. In general, device types as well as the number and arrangement of actual components vary among switching installations. Most devices and affective relations are thus dynamically created, in response to both the computation of the entailments of the model, as well as to the rule-based processing of the alarms. For example, returning to the model of Figures 3 and 4, only 4ESS, CLOCK 0, CLOCK 1, and the two immediate-part-of relations relating the 4ESS and the clocks are initially asserted in the model.

New Alarm:
id:22257; type:F-LEV;
unit:ABC 6; subunit:A-CNTL 0

Assertions added to the device model:
Alarm(alarm1)
ABC(ABC 6)
immediate-part-of(ABC 6, 4ESS)
Controller(A-CNTL 0)
immediate-part-of(A-CNTL 0, ABC 6)
self-alarm(A-CNTL 0, alarm1)

Further implicit assertions:
part-of(ABC 6, 4ESS)
part-of(A-CNTL 0, ABC 6)
part-of(A-CNTL 0, 4ESS)
part-alarm(ABC 6, alarm1)
subpart(4ESS, ABC 6)
subpart(ABC 6, A-CNTL 0)
subpart(4ESS, A-CNTL 0)
part-alarm(4ESS, alarm1)

Figure 5: A simplified alarm message, and some device model additions.

An example of an alarm message from the switch is shown in the top portion of Figure 5.[5] The message assigns a unique identifier to the alarm, specifies the alarm type, and states that the alarm is occurring on the specified subunit of the specified unit of the 4ESS. That is, units link devices to the 4ESS as a component, while subunits link devices to units.

Processing of the alarm causes the device model to be updated as shown in the middle of Figure 5. Intuitively, a representation of the alarm is created and added to the model, the devices mentioned in the message are either found in the model or created and added to the model, and the alarm just created is related to the lowest level device mentioned in the alarm. In particular, the first

[4] Thresholds go up as one travels up the part-of hierarchy, e.g., many alarms on many components are necessary before concluding that there is a problem with the 4ESS.

[5] We have simplified the alarm, to enhance the clarity of this section.

assertion in the figure creates an instantiation alarm1 and declares it to be a member of the class Alarm. (Alarms also have an internal structure which is not shown, e.g., alarm1 has a slot "id" filled by "22257", etc.) The unit and subunit devices referred to in the alarm message will also be added to the model, along with the corresponding configurational relations between the 4ESS and the unit, and between the unit and the subunit. Thus, device ABC 6 is instantiated, declared to be a member of the class ABC, and linked to the device 4ESS (already instantiated at startup) via `immediate-part-of`. Similar processing occurs for A-CNTL 0. Finally, a `self-alarm` assertion is added which relates A-CNTL 0, the device component that actually signaled the alarm, to alarm1.

As at startup, the semantics of the model in conjunction with the `immediate-part-of` assertions will cause implicit `part-of` and `subpart` relations to be explicitly added to the model, as shown in the bottom of Figure 5. These assertions, in conjunction with the second general relationship in Figure 1, will in turn cause the assertions `part-alarm` (ABC 6, alarm1) and `part-alarm` (4ESS, alarm1) to be added to the model. Thus, the alarm is propagated to other devices to which it is relevant, by adding a `part-alarm` relation from any devices the alarming device is a component of, and the alarm. In this case it adds this relation between the two components which have A-CNTL 0 as a subpart (4ESS and ABC 6), and A-CNTL 0's alarm (alarm1). This captures the fact that a failure in 4ESS or in ABC 6 could have caused the alarm. At this point the computation of the implicit portions of the model is complete; initial processing of the alarm has resulted in dynamic construction of a portion of the model shown in Figures 3 and 4, and incorporation of the alarm into the model.[6]

Once our model is created, diagnostic rules of inference determine the root cause of the alarms by reasoning from the model. In particular, creation of the affective relations `self-alarm`, `part-alarm`, and `dependent-alarm` between devices and alarms trigger the diagnostic rules. For example, the following rule formalizes the first example from Section 2.2[7]:

$$\forall x \forall y \; [\; \texttt{part-alarm}(x, y) \wedge \texttt{part-threshold}(x, \mathsf{true}) \Rightarrow \texttt{functional}(x, \mathsf{false}) \;]$$

The rule is an example of a thresholding rule that determines when a non-alarming component is not functional, based on thresholding of the `part-alarm` relations in which it is involved. The domain specific relation `part-threshold` holds if there are a sufficient number of alarming devices that are components of the device x. When a threshold is met and a device asserted to be non-functional, a warning message can be sent to a technician.

Assume `part-threshold` is defined to be true when a device has at least two alarming components. The rule would thus not be satisfied for any device after the first alarm. However, if the switch then generated an alarm on another component of ABC 6, say `self-alarm` (SPC 0, alarm2), `part-alarm` (ABC 6, alarm2) would be among the newly entailed relations, and `part-threshold` (ABC 6, true) would now be satisfied. This would entail `functional` (ABC 6, false), and an operator could be sent a warning message. Again, this example illustrates how our approach supports reasoning from the device model using general principles involving affective relations.

2.4 Status and Implementation

Our implemented reconstruction of the original heuristic 4ESS alarm monitor includes a fairly simple ontology such as that shown in this paper, along with a small number of diagnostic rules about the 4ESS switch. This has already allowed us to achieve approximately 10% of the functionality of the original expert system. Furthermore, our implementation uses a much more principled and organized knowledge base with fewer rules. We are currently extending the functionality of our prototype, and it appears that our methodology will extend to the entire 4ESS monitoring problem.

The device model as well as the diagnostic rules of inference are implemented using R++ [**?**; **?**], an extension to C++ that supports forward chaining directly on C++ objects. The explicit encoding of the 4ESS device model is represented in a declarative, object-oriented manner using C++ objects corresponding to the physical devices and to the other basic classes in the ontology, and inter-object pointers corresponding to the affective relations between the classes. The entailments of the model, as well as the diagnostic rules of inference, are both represented using the *rule* construct of R++. R++ rules are if-then rules that are associated with C++ classes and that are inherited according to the C++ class hierarchy. Because the R++ rules are tightly integrated with the C++ object-oriented class mechanisms – which are used to define the explicit portion of the device model – it is natural to write rules that operate on the model. Furthermore, because the condition of an R++ rule must traverse inter-object pointers, rules in fact must be written using relations (such as affective relations) that are explicitly defined between classes. Finally, the association of rules with classes allows subclasses to inherit rules as well as attributes from superclasses. This allows us to write rules for one type of device and have them

[6] Although not illustrated here, other relations in the model (e.g., `depends-on`, `standby`) can also be dynamically created. This is done by supplementing the general relationships shown in Figure 1 with domain specific relationships, e.g., for all devices of type controller, the device depends-on the clock device having the same number. Thus, when an alarm causes a controller to be created, the appropriate depends-on assertion from the controller to a clock is entailed.

[7] In reality, a warning is sent and further evidence obtained before inferring non-functional.

inherit to all instances of that type.

3 Related Work

Work by Abu-Hanna et al. [?] argues that functional models are generally easier to obtain than behavioral models; furthermore, by representing design *intent*, functional models help identify relevant suspected causes of a symptom and are thus attractive for the suspect generation task. The need to represent diagnostically useful models at a level that is easy to obtain is also a conclusion drawn in our work. However, our affective relations do not encode purpose or intent (except perhaps at a very high level), and are thus at even a more abstract level of representation than the functional models of Abu-Hanna et al. Abbott's work on operative diagnosis [?] is also similar to ours in that her model includes a directed graph of "paths of propagation" to relate symptoms to suspects. Abbott's paths indicate only that one component may affect another, so they are the most abstract kind of affective relation. Our 4ESS switch monitoring example uses more detailed relations because they were readily available from 4ESS technicians, enabling more focused diagnosis. Because our work inputs information about a device, organizes and processes this information under an existing taxonomy of knowledge about related devices, and uses an existing taxonomy of knowledge to control the generation of a diagnosis, our work also fits within Clancey's general framework of heuristic classification [?].

Paragraph on belief nets. [?]

Although it has not been the focus of this paper, our work also seeks to bring some of the benefits of object-oriented modeling to bear on the task of building expert systems. The object-oriented basis of R++ has proven critical to our success in implementing our approach as it provides for both a taxonomy of classes to directly represent abstractions in the domain and a tight integration of rules with these abstractions, thus allowing us to write rules to do diagnosis within the model. There are several other attempts to integrate object-oriented and rule-based reasoning [?; ?; ?] that could be used to provide some of the same benefits as R++ does.

4 Conclusions

Device representation and reasoning with affective relations provides a middle ground between model-based reasoning and heuristic expert systems, for monitoring and diagnosing complex devices that are largely self-checking. While model-based diagnosis would seem to be well suited for such tasks, in practice the models of traditional approaches are often too difficult to obtain (e.g., due to system complexity, inadequate documentation, lack of an appropriate representation language). This has often led developers to rely on heuristic expert systems, which themselves have many well-known limitations. By providing an abstract level of device modeling based on a set of diagnostically motivated and easy to acquire affective relations, our work brings the advances of model-based reasoning to domains in which the development of traditional models is difficult. These advances include: (1) a unifying framework for knowledge and diagnosis based on general principles, (2) a useful tool for focusing knowledge capture (our models seem to correspond well with an expert's knowledge and can be visually displayed (as in Figure 4)), and (3) a more methodical coverage of the domain. Our work at the symbol level also brings to expert systems the benefits of relatively recent advances in object-oriented knowledge-representation — principally the use of objects to organize knowledge, and a tight integration of objects within the rule-based paradigm.

For the longer term, a further objective of our work is the development of a shared ontology similar to that of Figure 1, but which can be used to monitor and diagnose many different kinds of devices. A key question is the extent to which general principles of causality and diagnosis apply across different kinds of devices, e.g., how far should symptoms be propagated up the `part-of` hierarchy?

A driving force behind our work has been to develop a pragmatic solution, one that balances the fact that developers of commercial systems have only limited resources to invest, but that they also have a desire to obtain the benefits found in "ideal" approaches. In the end the essential contribution of device representation and reasoning with affective relations is the identification of a particularly useful level of abstraction for modeling complex devices – represent the affective relationships between components that are used by experts when reasoning about the device. This level of abstraction is high enough that large and complex devices can be represented but still detailed enough to allow effective monitoring. The use of such models brings the advantages of model-based reasoning to domains previously handled only with heuristic methods.

Acknowledgments

We would like to thank the management of the Operations Technology Center, especially Raj Dube and Pramod Warty, and the management of the Software and Systems Research Center, especially Mike Wish and Ron Brachman, for their support of this project. We would also like to thank Steve Correl for the implementation of many of these ideas using R++, Ben Kuipers for helpful comments and discussions, and David Etherington for comments on an earlier draft of this paper.

References

[Albert, 1994] Patrick Albert. ILog Rules, embedding rules in C++: Results and limits. In *Position Papers* *for the OOPSLA'94 Embedded Object-Oriented Production Systems Workshop (EOOPS)*, October 1994.

[Hal, 1993] The Haley Enterprise. *Rete++: Seamless Integration of Rules and Objects Using the Rete Algorithm and C++*, 1993.

[Miranker *et al.*, 1993] Daniel P. Miranker, Frederic H. Burke, Jeri J. Steele, David R. Haug, and John Kolts. The C++ embeddable rule system. *International Journal on Artificial Intelligence Tools*, 2(1):33–46, 1993.