RAPPER: The Copier Modeling Project

David Bell Daniel Bobrow Brian Falkenhainer Markus Fromherz
Vijay Saraswat Mark Shirley*

October 18, 1994

Abstract

The programme of research of model-based diagnosis is based on several assumptions, foremost of which is the availability of component-based models of the system being diagnosed. This assumption has been fairly innocuous for diagnosis of networks of combinatorial devices, for which the construction of models with appropriate levels of discriminatory power has been easy. A priori, however, there is no reason to believe that it will be possible to easily construct models of appropriate generality, power and usability for complex, real-time, computational electro-mechanical systems.

This paper reports on a project we started three years ago to construct a component-based model for the input document-handler of a photocopier. Our goal was to develop a model that was rich enough to allow the generation of diagnostic trees with the same discriminatory power as existing documentation (RAPs, or Repair Action Procedures), which are by and large constructed manually. This goal has been met in principle, but we have learnt several useful lessons along the way.

1 Introduction

Model-based diagnosis has been a central area of research for members of our group for several years.[1]

For some time it has been recognized that the central challenge in model-based diagnosis is developing appropriate models of structure and behavior for the systems under consideration. To be useful for diagnosis, such models should be *accurate* — they should be discriminating enough to correctly identify exactly those states of the system that are in fact consistent with the given observations. To be usable in practice, they should be *abstract* — so that relevant reasoning can be performed with appropriate time- and space-efficiency. To be economical to produce, they should be *veridical* — they should be designed to capture the physics of the device being modeled so that

*Xerox PARC, 3333 Coyote Hill Road, Palo Alto, CA 94304, U.S.A.; Name@parc.xerox.com

[1]Our group, the Scientific and Engineering Reasoning Area at Xerox PARC, includes David Bell, Danny Bobrow, Johan de Kleer, Brian Falkenhainer, Markus Fromherz, Olivier Raiman, Vijay Saraswat, Mark Shirley and Brian Williams.

they could be reused in all those settings in which the device would be used. Global behavior should be derived from local interaction between components. This is usually called the "No function in structure principle" [1].

Models with these properties have been easy to construct for networks of electronic devices. But will it be possible to construct such models with reasonable effort for other real-world designed systems? In particular, it is natural for us to ask this question for the very complex engineered devices about which detailed information is available to us, namely, reprographics machines. Typically, these machines consist of hundreds of electrical, mechanical, electro-mechanical and computational components — sensors and effectors — involved in moving sheets of paper through mechanical pathways, using complex xerographic processes to transfer optical images (or bit-patterns) onto marks on paper, and applying various physical transformations to streams of paper (stapling, stitching, shrink-wrapping, . . .) — while keeping the entire process under computational control.

Fortunately, there already exists extensive diagnostic documentation for such machines, principally in the form of detailed diagnostic trees called RAPs (Repair Analysis Procedures). This gives us a pre-existing standard for coverage and accuracy that we could try to match. The question we asked ourselves, therefore was:

> *For a complex, real-world, computational, electro-mechanical system, is it possible to systematically create an abstract and veridical model SD of structure and behavior that is such that the diagnostic tree generated by SD is query-equivalent to existing diagnostic documentation?*

In more detail: How much detail would be necessary, particularly when describing signal and error propagation? At what level of abstraction would we have to model continuous processes (e.g. air-knife)? How detailed would information about geometry and spatial organization and interaction have to be? How detailed would information about temporal interactions between sub-systems have to be? Which ontologies for space and time would be appropriate? Could the models plausibly be generated as a byproduct of design? Could models be constructed explicitly at a level of effort that would be justified when compared to the existing approach to creating service documentation?

This paper is a preliminary report on our work in answering these questions. We chose the Recirculating Document Handler (RDH) subsystem of a copier in wide use — a major reason was that this subsystem is the only subsystem that handles customer input documents, and is relatively isolated from other components that are concerned with imaging and moving marked sheets. In brief, we have built an abstract, qualitative model that is detailed enough to produce diagnostic trees that match existing documentation. But we have learnt several important lessons along the way, chief among them being the importance of understanding both current work-practice and existing processes for informal model construction, and the wider organizational context in which re-engineering of current work processes (such as that implicit in the model-based diagnosis approach) is to be done.

The rest of this paper is as follows. We first briefly describe the particular subsystem we modeled. We then consider the particular modeling viewpoint we adopted, and discuss some representational issues and the diagnostic algorithm we employed. Finally, we step back and review some lessons we learnt (are learning) from this (ongoing) work.

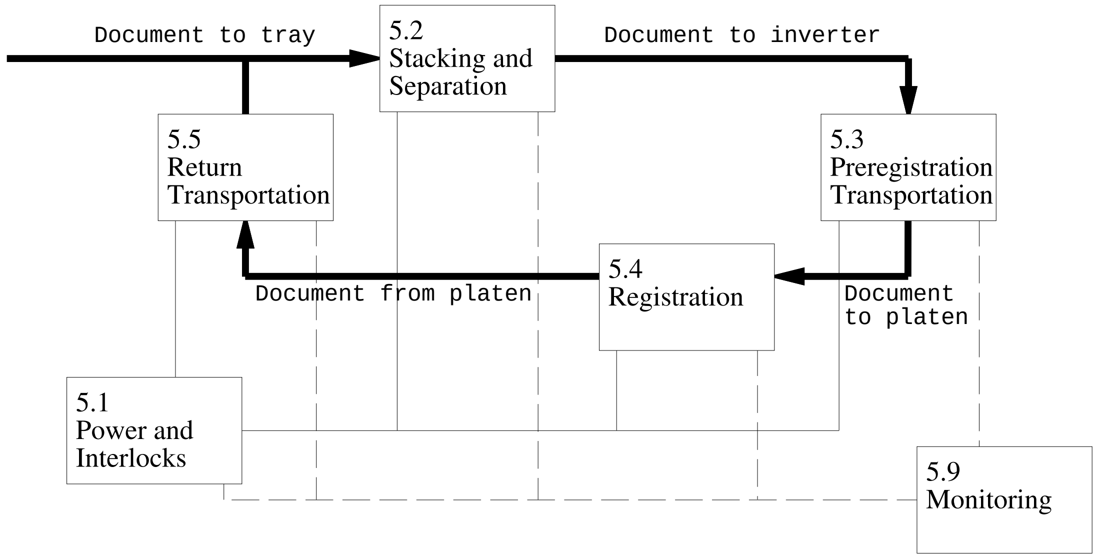

Figure 1: Schematic view of the Recirculating Document Handler

2 The task

The RDH for the particular machine we examined is responsible for feeding documents one at a time through a series of rolls from the tray to the platen glass, registering them on the platen for exposure, and moving them back to the document tray (Figure 1).

The system is broken up by into several subsystems.[2] The first subsystem handles power and interlocks. It provides motor drive to subsequent chains. It contains primarily the interlock mechanism and the AC motor that supplies torque to the rest of the document handler. The interlock mechanism ensures that the left-cover, right-cover and the input station are all latched shut; only then is power supplied to the motor. It contains the mechanical components for interlocking (a counter-balance and a magnetic catch) — these control interlock switches which electronically signal whether or not the latch is in place.

The second subsystem handles document stacking and separation in re-circulating mode. It's principle operations are to sense the presence of a document stack in the RDH (and report the height of the stack), achieve sheet separation using the air knife[3] and provide vacuum to transport the bottom-most sheet into the next chain. It contains solenoids, (position and light) sensors, valve assemblies, vacuum blower assembly, vacuum transport assembly, and stack height sensor.

The next subsystem handles the preregistration transportation of paper from the input tray (starting at the takeaway roll) to the platen. In simplex mode, it simply transports the paper around the paper-path loop so as to provide the paper to the platen face-down. In duplex mode, this chain is responsible for inverting the paper and providing the paper "face-up" to the platen. It consists primarily of roll stations, the inverter assembly (for duplex sheets) consisting of an inverter roll,

[2]We concentrate only on those subsystems that come into play during the operation of the *recirculating* document handler. The document handler also provides the functions of transporting computer forms and sheets fed from a side-feeder.

[3]A side-ways flow of air that flutters and thus separates top sheets from the bottom one.

an inverter gate and two different output paper paths, a bi-directional roll station (to handle both simplex and duplex), and forward and reverse drive assembly.

The fourth subsystem is responsible for driving the document onto the platen glass, stopping the document on the glass in the proper position for exposure, and after exposure driving the document off the glass into the return transport. It uses several main component assemblies. The document drive assembly drives the paper up to the registration fingers which stop the document. The clamp plate assembly then clamps the document in this registered position. Once registered and clamped, the fingers retract, clearing the paper path for future transport. After exposure, the clamp plate moves up allowing the document drive assembly to drive the document off the glass.

The fifth subsystem handles the return transportation of paper from the platen to the input tray, without curling the document. It consists of a document exit roll station, an exit roll drive clutch, decision gates, baffles, and a static eliminator.

Finally, an electronic subsystem monitors the state of the signals received from the switches and sensors within the document handler. If it detects a fault, the document handler operation is stopped and a message is displayed on the control panel.

Overall, the system we model consists of about 160 components, with approximately 40 different component types. The components are pneumatic, mechanical, electro-mechanical, electrical and computational and have time-varying behavior.

The diagnostic task Innumerable things can go wrong in this complex process. Sheets of paper may not separate properly, may jam during transport, may not register correctly (producing skewed copies), or may stick together because of electro-static charge.

When things go wrong a service technician is called. Part of his job is to *diagnose* the machine by making a series of tests on the system to refine his hypotheses until the faulty component(s) have been identified. In this task, the service technician is guided by a thick binder of *Repair Analysis Procedures* (RAPs). As the copier's builtin monitoring software reports a fault code in case of a malfunction, there is one RAP for each fault code. Starting from this initial symptom, a RAP includes a sequence of tests and observations for the service technician to follow, culminating in some suggested repair actions.

Currently, RAPs are generated manually by systematic experimentation. Our goal is to construct a qualitative model of the copier which would be adequate to either interactively guide a service technician through the repair process or automatically generate the Repair Analysis Procedures.

3 Model and Methodology

Diagnosis of time-varying systems is a complex problem. Our task is aided by the *sequential* and *cyclic* nature of activity in the RDH.

"Critical event" modeling. The correct overall functioning of the RDH is generally determined by the behavior of each component at certain "critical" periods, typically determined by the time that a sheet of paper usually passes a particular position in the path. Further, interactions between successive cycles of paper through the RDH are very rare and can be ignored to the first degree of approximation. This allows us to adopt the following modeling style. We model *one* "typical pass" of a "typical" sheet through the system, and for the typical pass, state the deviations from normality of various component behaviors at the critical time-events (cf.,[7]).

For instance, a segment of a paper-path takes on as value a "sheet". A sheet may be the value `none` (if in fact no sheet is transmitted during the prototypical pass) or `doc(P)`, where `P` is a complex description with attributes `size`, `dep_time` and `arr_time`. The normal value for `size` is `one` indicating a cleanly separated sheet of paper passing through that paper-path segment during the prototypical pass. It may be `shingled` if in fact two sheets stuck together were passed in. The values that `dep_time` and `arr_time` take on include `normal`, `early_1`, `early_2`, `late_1` and `late_2`, indicating the time (relative to normal) at which the document departed from the station at the beginning of the path, and arrived at the station at the end of the path. In addition, `arr_time` can take on the value `jam` to indicate that paper did not leave the station at the end of the path.

Other types of entities in the system (e.g., representing torque, fluid-path, sensor signals, electrical wires, etc.) could also be modeled as taking on values in a discrete set. (For example, a sensor signal is one of `step`, `clear`, `blocked`, or `pulse(T)`, where `T` specifies relatively a possible delay (`none`, `early`, `late`).)

Such a representation decision eliminates the need for explicit representation of time and complex temporal constraints. In some cases, a sheet passes the same device more than once (e.g., inverting roller for a duplex sheet). This is handled by having that component deal with a tuple of values, one for each moment of interaction between the device and the paper during the prototypical pass. However, genuine repetitive actions (e.g., the comparison between multiple passes) cannot be modeled. [4]

Classifying behavior For the task of diagnosis, we are interested in identifying and therefore modeling both correct and incorrect behavior. As an example, a roller might transport a sheet of paper with normal or low speed. With each behavior, we associate the (qualitative) states or *modes* of the process consistent with that behavior. For example, the roller behavior just described might be due to the roller being in normal or worn mode.

Correct and incorrect behavior is modeled for those cases only where the input is "in band", i.e. for each component, we make a deliberate choice of which values are accepted as valid input values. For all other inputs, we make the simplifying assumption that no information about the behavior of the device can be gained in these situations, and therefore usually provide some default behavior only. This is in line with another decision, namely the *single-fault assumption* that we will see at most one component in a fault mode. For the task at hand, this significant assumption is appropriate. Whenever an error is encountered as a sheet moves around the paper path, the physical system is shut down, thus masking any "down-stream" faults. This strongly sequential nature of the artifact allows an analysis of most problems as single independent faults.

Encapsulating complex processes. Since our task-objective was to be RAP-equivalent, we did not have to create detailed models of the internal working of most components; rather coarse

[4]This still allows us to handle some kinds of faults that arise only on multiple passes of a document through the system. For example, suppose that a large number of copies are being made of a given document. The state of the platen glass is such that a small amount of extra static charge builds up on the sheet on each pass. If the charge crosses a threshold then it can cause the sheet to stick and be detected as late at the tray exit sensor. This kind of multi-cycle interaction can still be tackled through the "prototypical-pass" approximation.

qualitative models sufficed. For example, the actual functioning of the air-knife involves a complex interaction between a stack of paper, air blowing on the stack to separate the sheets of paper, and a vacuum system that sucks the bottom sheet down onto a moving belt. To describe the *correct* functioning of such a system (e.g., in terms of air pressure, suction effects, the physics of paper separation, etc.) would be a significant undertaking in physical modeling. But for the task at hand, we are only interested in the (correct and incorrect) interactions of a process with its environment, *regardless* of its internal structure.

Similarly, the start-up circuit for an AC motor exhibits complex dynamics, which did not have to be modeled in detail. For our purposes it was sufficient to encapsulate it as a single "component" whose input/output behavior was equivalent to the RAP's view of the subsystem.

Hierarchical modes. Because of the sheer size of the system being modeled, we found it very convenient to associate hierarchical modes with subsystems. In general the mode of a subsystem was some function of the mode of its components; typically, it was just a tupling. This meant that the overall system had a mode that took on as value a tree of mode-values. This required us to handle the implementation of single-fault assumptions with care — just because the mode for one device was known to be ok, it was not possible to instantiate the modes for all other devices, since the shape of the mode-tree (i.e., the shape of the tree underlying other sub-systems) may not yet have been determined.

Implementation and system-level issues. The model is implemented in an early version of the language `cc(AL)`, a concurrent constraint language [8] with the "attribute-value" constraint system and naive arithmetic. Terms of the form `P[dep_time >> normal]` designate an av-list identical to P except that the value of the attribute `dep_time` is `normal`. Accessor notation (`P.dep_time`) can be used for obtaining values at attributes. Limited but useful constraint-solving over av-lists is allowed. AV-lists proved invaluable in passing, accessing and updating complex aggregate data-structures between processes.

Each component type is represented by a predicate whose clauses describe the possible behaviors for all modes of the component and for "out of band" inputs. Interconnection axioms establish local component interactions via shared *node* variables. Figure 2 shows a simple example, the model of a roll station.

A simple compiler translates `cc(AL)` programs into (Sicstus) Prolog, using delay primitives to control evaluation of non-ground arithmetic expressions. The diagnostic algorithm is implemented as an invocation of the model, under a pro-active single-fault assumption (which is not in pure Prolog). This version of the compiler does not generate code that does constraint propagation at run-time. Hence, in essence the run-time behavior is that of Prolog-style depth-first generate-and-test search.

4 Diagnosis in RAPPER

Given an observed fault code which indicates faulty behavior, we can query the RDH model for all possible modes of components consistent with this observation. As explained above, we make the single-fault assumption.

Roll Station. If a roll station receives no document or low torque, then it does not pass any paper through. In other (in-band) cases, its behavior is determined by whether it is ok or worn. If more than one sheet of paper is simultaneously received (shingled), then a jam is asserted. Otherwise, the sheet is transferred normally (if the device is ok) and with an incremental delay otherwise.

```
Mode is {ok, worn_roller}
ROLL_STATION(Mode,        Torque, PaperIn, PaperOut) */

roll_station(_M,          _T,     none,    none).
roll_station(_M,          Torque, doc(P[arr_time >> jam]),  none):- low_torque(Torque).
roll_station(worn_roller, normal, doc(P[size >> shingled, arr_time >> jam]),  none).
roll_station(ok,          normal, doc([size >> shingled, arr_time >> jam]),  none).
roll_station(worn_roller, normal, doc(P[size >> one]), doc(P_o[size >> one])):-
   delayed_doc(P,P_o).
roll_station(ok,          normal, doc(P[size >> one]), doc(P_o[size >> one])):-
   transferred_doc(P,P_o).

low_torque(low).
low_torque(none).

transferred_doc(P[arr_time >> A, dep_time >> A], P[dep_time >> A, arr_time >> _New]).
delayed_doc(    P[arr_time >> A, dep_time >> A], P[dep_time >> Dep, arr_time >> _New]) :-
  delayed(A,Dep).

delayed(late_2,late_2).
delayed(late_1,late_2).
delayed(normal,late_1).
delayed(early_1,normal).
delayed(early_2,early_1).
```

Figure 2: A sample component model: Takeaway Roll Station

Diagnostic Algorithm Concretely, given a symptom, all hypotheses W consistent with this symptom can be computed. A hypothesis $w \in W$ is the triple $\langle p_w, N_w, M_w \rangle$ describing a specific fault, where p_w is the probability of hypothesis w, N_w a set of probe points x with their values v, and M_w a set of components c with their modes[5] m. p_w is defined by $p_w = \prod_{(c=m) \in M_w} p_{c=m}$, where $p_{c=m}$ is the prior probability that component c is in mode m. Probe points x have domain $V_x = \{v \mid w \in W \wedge (x = v) \in N_w\}$.

Given the hypotheses consistent with the symptom, we want to rank the probes according to how well they can distinguish between the hypotheses, and lead to a diagnosis. Currently, we use a standard Shannon entropy calculation [3]. When used in an iterative algorithm that chooses probe points, accepts their values, and removes inconsistent hypotheses, this correspond to a one-step lookahead [2]. Given set W and a probe point x, the expected entropy is defined by

[5] A hypothesis contains all component modes, not just the faulty one.

$$H_e(x, W) = \sum_{v \in V_x} p(W_{x=v})\ H(W_{n(x=v)})$$

where $W_{x=v}$ are the hypotheses consistent with x being v, $W_{n(x=v)}$ is the respective normalized set, $p(W)$ is the total probability of the hypotheses W, and $H(W)$ is the entropy of hypotheses W defined by $H(W) = -\sum_{w \in W} p_w \ln p_w$.

Based on W and X, X_r is a list of probe points ranked with respect to their expected entropy. The algorithm is parameterized such that subsets of W ("leading hypotheses" based on p_w) and X ("relevant probe points" based on V_x) are used to compute X_r.

We have also implemented a variant of this diagnostic algorithm that takes into account the hierarchy of the model's structural description. Essentially, instead of using hypotheses W, an abstraction A_l of W at subsystem level l is used to rank probe points. Diagnosis starts at an abstract level and adaptively proceeds to a lower level only when no discrimination among hypotheses can be performed at this level. The idea of this variation is that initially those probe points are preferred that rule out entire subsystems.

Interactive Diagnosis In interactive diagnosis, the algorithm starts from the symptom and computes an initial set of hypotheses W with a ranked probe list X_r. The technician chooses one of the top-ranked probes $x \in X_r$, makes a measurement $x = v$, and reports the observed value back to the system. The diagnostic algorithm then removes all hypotheses inconsistent with this observation from W, recomputes X_r, and iterates. The algorithm halts if only one hypothesis is left, indicating the faulty component.

RAP Generation With a similar algorithm, RAPs can be generated by automating probe selection and anticipating all possible probe values. Instead of having the user choose probe points, the first probe point x in X_r is taken, and then all possible values are "envisioned" instead of reacting just to the one measured by the user. Thus, the generator spans, at each chosen probe point, a diagnostic subtree with $|V_x|$ branches, one branch for each possible value in V_x. For each branch of the subtree, the generator proceeds as in the interactive diagnostic algorithm, removing inconsistent hypotheses and ranking the remaining probe points, before generating the next subtree.

This procedure results in one diagnostic tree per fault code, where internal nodes are probe points, branches are transitions to subsequent probe points based on possible probe values, and leaves are diagnoses.

When producing RAPs from the diagnostic trees, a range of formatting options are available. For example, it is useful to order the branches of each node such that shorter paths come first. Also, similar measurements of the same types of components can be summarized, especially if they appear in a degenerate subtree which simply checks each component in a suspect set. Another formatting option is to add *focus* information, which is a statement that tells the user on which subsystem all further hypotheses of a diagnostic subtree will focus (see below). All these operations result in skeletal RAPs, which are basically annotated, ordered diagnostic trees. Figure 3 lists a typical skeletal RAP (excerpt).

RAP generated for fault code *a069*:

Measure probe po3.
Is the value of po3 normal?
Yes (All suspects are now in 'chain-5-3'.)
    Measure probes [cm,cn,cp,co] in 'chain-5-3'.
    Is the value of any of them float?
    Yes Diagnoses:
        cm: cm in 'chain-5-3' is open
        cn: cn in 'chain-5-3' is open
        cp: cp in 'chain-5-3' is open
        co: co in 'chain-5-3' is open
    No (co is v5dc, cp is dc-com, cn is dc-com, cm is blocked)
        Diagnosis: doc-to-platen in 'chain-5-3' is obstructed
No (po3 is none)
    (All suspects are now in 'chain-5-4'.)
    Measure probe reg-finger-position in 'chain-5-4'.
    Is the value of reg-finger-position in?
    Yes Diagnosis: pwb-5-4 in 'chain-5-4' is bad
    No (reg-finger-position is out)
        Measure probes [at,as] in 'chain-5-4'.
        Is the value of any of them float?
        Yes Diagnoses:
            at: at in 'chain-5-4' is open
            as: as in 'chain-5-4' is open
        No (as is dc-com, at is v24dc)
            Diagnoses:
            gear in 'chain-5-4:doc-drive-assembly' is bad
            drive-rolls in 'chain-5-4:doc-drive-assembly' is bad
            clutch in 'chain-5-4:doc-drive-assembly' is stuck-open

Figure 3: A skeletal RAP

5 Evaluation and Future Work

Coverage of RAPs. The evaluation of coverage is mostly very encouraging — the generated RAPs cover the diagnoses of existing RAPs fairly well. Missing diagnoses can often be traced back to inadvertent omissions in the model. In a few cases, the generated RAPs also show diagnoses that are missing from the existing RAPs.

The existing RAPs also often summarize a series of diagnoses for similar components, and contain intermediate, general hints (abstract diagnoses). Both features can be achieved in the generated RAPs. However, the existing RAPs are clearly richer in their explanations, and also contain "catch-all" diagnoses and references to other RAPs, to be applied when all else fails.

Observations obviously missing from the generated RAPs are those about "non-systemic" events (e.g., loose or burned objects), missing components, and observations that are functions of the device running in diagnostic mode. The existing RAPs also emphasize a different set of observations, which may be due to a different probe cost model.

One area not addressed by the generated RAPs is descriptive documentation. The existing

RAPs contain background information on symptoms, components, functions, and general servicing information strategically placed throughout the procedures. This information is meant to outline the context and point to further explanations, or to related RAPs. Often, RAPs also contain schematic diagrams and references to the descriptive documentation (another thick binder). This could be somewhat improved by using keywords and an appropriately structured database of canned texts, but the human analysts clearly add important value to the decision trees.

Simple modeling techniques suffice. For diagnostic models of mechatronic systems, the 80/20 rule seems to apply — simple models can go a long way.

In retrospect, while the "critical event" modeling style we adopted considerably simplified the task of generating the model, it may not have been as useful as we thought. To understand what counted as deviation from normality for a particular device, we had to understand what counted as abnormal for devices farther downstream: this required envisioning the component's behavior in context, and characterizing the histories of the signals in and out. We were helped by the fact that the subsystem has one focus of interest, namely the sheet moving through the system, but this often proved awkward.

We are exploring the relative merits of different modeling styles in terms of both simulation speed and naturalness of the description. This includes using discrete event-based models of devices in either a traditional event-driven simulator or a temporal constraint system. It is possible that the smooth integration of event-driven simulation ideas in this context will require the use of certain non-monotonic features, such as "safe defaults". We are also exploring whether it is possible to automatically derive critical events from a simulation, and hence generate the critical event model.

Nevertheless, by and large, we feel we understand the technical issues involved in designing modeling languages to allow the representation of models of electro-mechanical systems at the level of detail necessary for constructing most RAPs. Simple constraint languages, perhaps augmented with real-time features (as we are currently investigating) seem to be adequate.

Perhaps the most suprising lesson seems to be: *General mechanisms for reasoning about space, time, spatial and temporal interactions may not be necessary for useful diagnostic models of real-world devices.* Clearly, this will have to be further substantiated – at least in the reprographics domain – as we work on other subsystems of a reprographic engine. But to date we have been surprised by how far we could go with extremely simple representations.

Sometimes simple inference techniques suffice. Currently, very little control is exercised over the search tree; nevertheless under a single-fault assumption the entire fault tree for a single symptom can be generated in a few seconds on a SparcStation. For larger diagnosis problem, it may well be necessary to focus on a few probable diagnosis (as in, for example, Sherlock [4]), as opposed to exploring all diagnoses simultaneously. Regardless, a better conceptual understanding is needed of combining probabilistic information with constraint-based computation.

Understanding the larger organizational setting is crucial. An unexpected lesson we are learning is that the development, delivery and deployment of knowledge- or model-based diagnostic systems is seriously affected by larger organizational dynamics. In the case of Xerox, diagnostics involves several different groups of people — product and design engineers, service analysts, service technicians, management, researchers developing computational diagnostic systems — who may not share a common background about field service and diagnosis. An appreciation of the organizational dynamics between these groups is essential to understanding the appropriate role of technology in the work-place.

When we started the project we were by and large unaware of the crucial role played by *service analysts*. These are a group of people — distinct from the engineers who work on developing products — charged with assimilating all the sources of information (if any) about the given product to produce documentation to be used by service technicians. As we ploughed through the various sources of documentation about the RDH — principles of operation, Block Schematic Diagrams, RAPs, parts lists, system and module operation descriptions — we began to gain a better understanding of the complex assimilation task performed by service analysts (who manually produce RAPs currently), and of the tools that may be of use in aiding them in their task. (In some cases, service analysts may not even have access to some of the documentation — they may just have a "Box from Heaven" whose modes of correct and incorrect functioning they try to infer by literally "pulling wires" and observing resultant symptoms.)

Equally, in order to make a diagnostic system useful for *service technicians*, a series of additional issues on top of modeling and diagnostic algorithms have to be addressed. Among these are explanation, learning and adaptation, graceful degradation, and integration with complementary techniques [6].

We realized that tools need to be integrated into actual work practice, rather than causing "task intereference"; tools should be a time saving practice for work *already* being done rather than appear to be additional work. This requires studying current work practice and working with practitioners to evolve new work methods.

In particular, it is important to present model-based diagnosis technology as an incremental build on the existing work practice. To do this, the new technology must support all the deliverables the old technology delivered that are still considered essential:

Network models (pathways of interaction) should be built. We've seen that analysts already build Block Schematic Diagrams (BSD's), they are kept up to date, and are used by field service technicians for deductive diagnosis. Guidelines for building BSD's need to be refined (e.g. to capture all of the pathways of interaction we would like).

In addition, service analysts are also responsible for producing a "Principles od Operation" document. How can model-based reasoning techniques be used to aid in the task of generating such documents?

Component models should extend FMEA-like models. Failure Mode Effects Analysis (FMEA), Fault Insertion and Failure Analysis are three current practices that develop a qualitative understanding of the machine behavior. The modes of each component are identified, and the resulting effects are captured. We should understand what additional benefits are provided by the more explicit and detailed compositional models we have been developing, and how such a modeling activity can be introduced as an extension of current practice.

'Just-In-Time" modeling should be supported. In subsequent projects we have found that service analysts develop a more accurate knowledge of the machine over time through envisionment, tests where bad modes are inserted into a machine and behaviors observed, and through feedback from field experience. In addition, components of the machine are constantly being changed and/or updated. This makes model development an evolving process, with analysts recording their models right before the moment they are needed. Modeling is far from being the one-time process that design engineers engage in and hand-off to customers downstream. What kinds of model-analysis and maintenance tools need to be developed to aid in this task?

Future work These and many other related questions are being taken up in our continuing work on understanding the engineering and product development process within Xerox. We have been developing a Xerox-wide modeling and simulation infra-structure, a methodology for representing and reasoning with real-time and hybrid systems and for using constraint-based models to generate control software, and a suite of tools to enhance current work-practice in smart service (model-based diagnosis, FAST/FMEA tools, descriptive documentation, "tips" data-base, . . .). We expect to report on these activities in subsequent papers.

Acknowledgement

The work reported here has benefited immeasurably from interactions with Johan de Kleer, Brian Williams and Olivier Raiman. Special thanks are also due to Bob Easterly, and Ken Kahn for valuable discussions and comments.

References

[1] Johan de Kleer, John S. Brown, "A Qualitative Physics Based on Confluences", *Artificial Intelligence*, vol. 24, 1984; also in: [9], pp. 88–126.

[2] Johan de Kleer, Olivier Raiman, Mark H. Shirley, "One Step Lookahead is Pretty Good", in: *Proc. Second Int. Workshop on Principles of Diagnosis*, Milano, Italy, October 1991, pp. 136–142.

[3] Johan de Kleer, Brian C. Williams, "Diagnosing Multiple Faults", in: *Artificial Intelligence*, 32, 1987.

[4] Johan de Kleer, Brian C. Williams, "Diagnosis with Behavioral Modes", in: *Proc. 11th IJCAI*, 1989, pp. 1324–1330.

[5] Ken D. Forbus, "Qualitative Physics: Past, Present and Future", in: H. E. Shrobe and the American Association for Artificial Intelligence (Eds.), *Exploring Artificial Intelligence*, Morgan Kaufmann, 1988, pp. 239–296; also in: [9], pp. 11–39.

[6] Markus P. J. Fromherz, Mark H. Shirley, "Supporting Service Technicians: Model-Based Diagnosis in Context", in: *Proc. Workshop on AI in Service and Support at AAAI'93*, Washington, DC, July 1993, pp. 59-69.

[7] Walter Hamscher, "Temporally Coarse Representation of Behavior for Model-Based Troubleshooting of Digital Circuits", in: *Proc. 11th IJCAI*, Detroit, MI, August 1989.

[8] Vijay A. Saraswat, *Concurrent Constraint Programming Languages*, MIT Press, 1993.

[9] Daniel S. Weld, Johan de Kleer (eds.), *Readings in Qualitative Reasoning about Physical Systems*, Morgan Kaufmann, 1990.