Automating Model Acquisition by Fault Knowledge Re-use, DR, the Diagnostic Remodeler Algorithm

Abu-Hakima, Suhayya

Automating Model Acquisition by Fault Knowledge Re-Use, DR, the Diagnostic Remodeler Algorithm

NRC 38315

Suhayya Abu-Hakima
Knowledge Systems Laboratory
Institute for Information Technology
Building M-50, Montreal Road
National Research Council of Canada
Ottawa, Canada K1A 0R6
internet: suhayya@ai.iit.nrc.ca
tel: (613) 991-1231
fax: (613) 952-7151

Abstract

This paper addresses the problem of automated model acquisition through the re-use of fault knowledge. The Diagnostic Remodeler (DR) algorithm has been implemented for the automated generation of behavioural component models with an explicit representation of function by re-using fault-based knowledge. DR re-uses as its first application the fault-based knowledge of the Jet Engine Troubleshooting Assistant (JETA). DR extracts a model of the Main Fuel System using real-world engine fault knowledge and two types of background knowledge as input: device dependent and device independent background knowledge. To demonstrate DR's generality, it has also been applied to a coffee maker fault knowledge base to extract the component models of a full coffee device.

Introduction

Fault-based reasoning (FBR) is used in many diagnostic systems. Knowledge in FBR is largely based on maintenance manuals and interviews with experts intended to capture heuristic knowledge about the maintenance and repair of a device or process. The maintenance and repair is directed at keeping a device functioning in a predictable manner. The knowledge in these systems is often represented as hand-coded rules or frames which are organized into troubleshooting hierarchies. At the top level of the hierarchy is the general knowledge representing a problem with a device. This general problem is refined systematically until the leaf nodes of the hierarchy which represent physical repairs to the device are reached. Once these repairs are achieved by a human technician some diagnostic systems re-test to confirm that the symptoms and diagnosed faults are cleared through backtracking in the hierarchy.

Model-based reasoning (MBR) for diagnosis concentrates on reasoning about the expected and correct functioning of a device. A device is modelled based on its components and their expected behaviour [Hamscher and Struss 90]. Such models range from quantitative ones to qualitative ones and all attempt to approximate device behaviour as accurately as possible. Once a device model is stabilized then a device's observed behaviour can be predicted from the model. If a discrepancy in behaviour is detected then possible fault candidates are generated based on assumptions that describe correct model behaviour. Sequential diagnosis is used to choose observations, augment a prediction for the candidate faults and update the list of candidates until a dominant candidate is found.

In MBR there are many conflicting definitions for models. They range from causal models represented as semantic networks with links specifying the relations between component nodes to full blown numerical simulations for complex systems and processes that have taken decades to perfect. Generating models is a key problem in MBR. Some researchers generate causal models, others generate models with structure and behaviour while others generate functional models for devices. Knowledge in models has thus far been hand-coded by experts that understand device component behaviour and function.

In the first phase of the DR algorithm fault-based knowledge from a real-world troubleshooting system is re-used to extract component-connection relations. In the second phase of the algorithm, the extracted components are matched against background knowledge to derive specific behaviours. In addition, any gaps in the fault knowledge component-connection links are tagged. These gaps can be used to uncover inconsistencies in the fault knowledge. Fault-based knowledge from the Jet Engine Troubleshooting Assistant [Halasz et al. 92] is re-used to demonstrate how DR can successfully extract "black box" models of the components that make up the main fuel system of a jet engine. In a second application, DR is used to extract the component models of a coffee maker device from its troubleshooting knowledge in conjunction with some additional background knowledge. The second example is used to illustrate the generality of the DR algorithm and its potential wide applicability for the automated extraction of component models through re-use of fault knowledge.

Hypothesis of Diagnostic Remodeler (DR) Algorithm

Humans use failure-driven reasoning for successful device diagnosis and repair. As humans reason about diagnosis and repair they build primitive mental models of how a device functions and fails. The hypothesis for the Diagnostic Remodeler algorithm is that knowledge of failure and repair embodied in most well-structured diagnostic knowledge-based systems can be used to derive black box component models of a device. DR extracts component models that represent structure, behaviour and function from fault and background knowledge.

A great deal of effort is expended hand-coding complex knowledge bases for diagnostic fault-based reasoning (see DX-93 paper by [Abu-Hakima 93] and [Abu-Hakima 94a; 94b] for more background). The artifacts these diagnostic systems are developed for are often expensive machines which have been designed and continuously modified so that no existing accurate schematic or design of their behaviour or resultant function remains. The J85-CAN-15 is a jet engine which is the first application of the Jet Engine Troubleshooting Assistant, JETA [Halasz et al. 92]. JETA is a typical well-structured fault-based reasoning system similar to MDX [Chandrasekaran et al. 79].

The J85-CAN-15 engine was designed in the 1950's and has easily had at least one modification a year since its launch. As a result of modifications and stresses of daily use (flying in the arctic and flying in desert heat) the jet engine is a very different device than was originally designed and sometimes displays inexplicable behaviour. No existing design schematics can completely capture the engines's behaviour or completely predict its function. It is also a very difficult device to diagnose. For these reasons a tool such as JETA was developed. As is typical with FBR systems, JETA does not diagnose novel faults. Learning the device component model, its behaviour and functionality using the FBR knowledge provides the technician with a DR model that can be used as input to an off-the-shelf model-based diagnostic tool.

DR Algorithm Approach

If we follow the de Kleer [de Kleer and Williams 87] approach to MBR which represents a device and its function as a set of components with behaviour. A device can be diagnosed by assuming a faulty component and enumerating the behavioural states that the fault propagates in the remainder of the device [Davis 84; Hamscher and Struss 90; Struss 89]. This is compared to the behaviour that a technician is observing in attempting to isolate a problem. MBR for diagnosis can detect novel faults since the behaviour of the device is the basis of its knowledge representation and reasoning. Fault-based reasoning uses the faults in the functioning of a device rather than its actual behaviour, hence FBR cannot detect novel faults. However, MBR can lead to a combinatorial explosion in producing a diagnosis for complex systems (for example, an aircraft engine) and it does not lend itself to causal explanation [Struss and Dressler 89].

I have implemented the DR algorithm (introduced in [Abu-Hakima 93]) intended to address the automated generation of a model of a device by the re-use of its fault knowledge. This implies the automated generation of MBR knowledge from FBR knowledge.

Method

DR is an algorithm that takes as input the fault knowledge of a device. It is also necessary to take as input some background knowledge related to the device to attempt to learn its full component behaviour and function. DR initially extracts from the fault knowledge base all references to device components related to a subsystem of interest. Given these components the algorithm backtracks through a diagnostic hierarchy of nodes to generate hypotheses for component connectivity. To establish component connectivity, DR examines symptomatic or parametric knowledge that activates the diagnostic nodes. As confirmed by DR's successful results, as one moves upward in a typical well-structured troubleshooting hierarchy, the knowledge at the higher level nodes is more general and describes higher level problems with device operation. As one moves down in the hierarchy and is closer to the terminal nodes that represent device components, the knowledge is more symptomatic in nature. Symptomatic knowledge is knowledge of specific component parameter failures which can be re-used to generate hypotheses about parameters or behaviours attached to specific components.

DR Algorithm Steps

Two phases clearly divide the operation of the DR algorithm. In the first phase, an existing well-structured knowledge base (e.g. one that diagnoses a complex electromechanical system) is used as input to DR. Two types of background knowledge, device dependent and device independent knowledge are used as second phase inputs to DR. Device independent background knowledge is in a component library and is general in nature. It includes general knowledge that a pump delivers some liquid from a source to a sink and needs a control signal (such as pressure for example) to increase or decrease the flow of liquid. The pump library component model also includes some knowledge about feedback control in moderating the flow of a liquid to a source based on the level of the liquid at the sink. The device dependent knowledge includes the specific details on the input and output (I/O) parameters for different device control modes.

The objective of the DR algorithm is to discover and refine a component behavioural model with explicit function. In the most general sense, the algorithm must identify the components of the device, generate links between those components, and generate hypotheses for the behaviour and function of the components.

To achieve this the DR algorithm must perform five steps:

1. *identify the terminal nodes in the diagnostic hierarchy*
   -these represent component nodes that have no child or sibling refinements
2. *identify the component nodes in diagnostic hierarchy related to the subsystem to be modelled (if required)*
   -perform a pattern match with known name or its derivatives (possibly acronyms) that match subsystem
3. *identify the parents and siblings of the nodes*
   -backtrack from terminal to parent nodes and tag
   -tag shared parents of a node
   -tag siblings of a parent
4. *extract relations (behaviours) between sibling nodes*
   -cluster nodes related by parental nodes
   -movement from the terminal nodes to parent node represents symptomatic information (parameters)
5. *match device model against background knowledge and output gaps for verification to the user*
   -map out the identified components of the subsystem
   -relate the components through shared parameters
   -match derived component model with device dependent knowledge to derive parameter I/O behaviours
   -match derived component model with library component model to extract function and uncover gaps

DR Re-Use of Fault-Based Knowledge

To achieve the knowledge-rich modelling proposed as the output for DR, one requires the use of a well-structured and explicit knowledge representation that can adequately represent diagnostic causality. This is achieved by extracting a model of the connections between the components in the subsystem to be modelled. These connections are further used to extract the variables (for example engine speed, fuel flow, etc.) that typify the behaviour between components.

In typical troubleshooting systems, frames are used since they offer a great deal of flexibility in constructing and reasoning about knowledge. DR uses four of the frame slots in a typical troubleshooting system to determine component connections. The slots used are the *node name*, the *node type*, the *child node of*, and the *child node ranking*. Replace node types are the terminal nodes first identified for a specific subsystem. The subsystem is identified through the *node name* itself. The *child node of* is used to determine the parent of a terminal (component) node. The *child node ranking* is used to determine the siblings of a terminal node. The parent node as mentioned earlier represents symptomatic or parametric knowledge between sibling nodes.

DR Background Knowledge

Two types of background knowledge, device dependent and independent, are used by the second phase of the DR algorithm to tag the direction and the positive or negative I/O behaviours between components.

Device dependent background knowledge is used to identify the type of a component (for example a pump, a filter, a variable control, a vessel, a source, etc.) and any specifics about inputs or outputs related to operational modes. The traditional approach used in modelling feedback in engineering, requires that both the modes of component operation, and their respective I/O parameters that act as behavioural control variables in a particular mode be explicitly identified.

Thus, for the main fuel control (MFC) component of JETA device dependent knowledge identifies that the MFC is a control with 7 fuel scheduling modes that vary from acceleration to deceleration with a variety of speeds in between. For each mode there are key parameters that represent component behaviours. They include engine speed (N), pilot demanded speed (Nd), throttle position or power lever angle (PLA), compressor inlet temperature (T2), fuel flow (Wf), compressor discharge pressure (P3) and inlet guide vanes (IGV) which indicate air bleed valve positions. In the excerpt of device dependent background knowledge below each of the MFC modes has a specific set of behaviours represented as lists of input-output pairs. Below are both the general, and the MFC-specific expressions for device dependent background knowledge.

```
% glossary(KB,Component, ProperName,[Component-
Type,for,[Modes]], [InputOutputPairs]).

% main fuel control terms from JETA's Glossary Frames
and J85 Control
% Parameters/Modes
glossary('JETA','MFC',main_fuel_control,
   [control,for,
    [steady_speed_control,speed_cutback_control,
    acceleration_fuel_limit_control,
    deceleration_fuel_limit_control,
    variable_geometry_scheduling,
    proportional_speed_control,fru_fuel_selection]],
    [[['N','PLA+'],['Nd+','WF/P3+']], [['N','T2_limit'],['Nd-']],
   [['N+','T2'],['WF/P3+']], [['N-'],['WF/P3-','WF_min']],
     [['N','T2'],['IGV','bleed valve positions']],
     [['N','PLA'],['WF/P3']],
     [['WF/P3','P3'],['WF']]]).
```

Device Independent Background Knowledge is the second type of background knowledge and forms a re-usable component library. For each of the components the function of the component is first represented. Function here implies, the purpose of the device component as defined by Sticklen and his colleagues [Sticklen et al. 88]. In addition, the inputs and the outputs of the component are made explicit. In the case of a regulated component that has a control signal, a regulation parameter is identified. Finally the behaviour function that maps the inputs and outputs of the component is described. In the case of a proportional relation (increasing input and increasing output, or decreasing input and decreasing output) a behaviour is identified. More complex components which have complex behavioural relations dependent on specific modes are also tagged. In the case of the main fuel control with its 7 modes of operation that reflect it as a component that has feedback, a piecewise linear behaviour is extracted. This behaviour is a set of behaviours that represent each mode of MFC operation as either proportional or inverse proportional. For a pump, the device independent component model is:

```
component(pump,Pump_name,Fluid,Control,_,F,I,O,R,B):-
   F = function(Pump_name, delivers(Fluid)),
   I = input(Pump_name,fluid(Fluid)),
   O = output(Pump_name,fluid(Fluid)),
   R = regulator(Pump_name,Control),
   behaviour_proportional(Fluid,Control,Behaviour),
   B = behaviour(for(Pump_name),
  behaviour_is_proportional(Fluid,Control,Behaviour)).
```

For a filter (for example a fuel filter or a coffee filter), the device independent component model is:

```
component(filter,Filter_name,Fluid,Control,_,F,I,O,R,B):-
   F = function(Filter_name, filters(Fluid)),
   I = input(Filter_name,fluid(Fluid)),
   O = output(Filter_name,fluid(Fluid)),
   R = regulator(Filter_name,none),
   behaviour_proportional(Fluid,Control,Behaviour),
   B = behaviour(for(Filter_name),
  behaviour_is_proportional(Fluid,Control,Behaviour)).
```

A control component with variable number of inputs, outputs and control variables has piecewise-linear behaviour:

```
component(control,Name,Ins,Outs,Modes,F,I,O,R,B):-
   Outputs = [Main_Output|Outs],
   F = function(Control_name,controls(Main_Output)),
   I = input(Control_name,control(Inputs)),
   O = output(Control_name,control(Outputs)),
   R = regulator(Control_name,regulation_control(Inputs)),
   typify(Inputs,Ouputs,Modes,
  Control_var_list,Behaviours_list),
   B = behaviour(Control_name,
     behaviour_is_piecewise_linear(Control_var_list,
     Behaviours_list)).
```

DR Aircraft Engine Results &Device Model

An analysis of the JETA fault knowledge shows layers of knowledge represented as a directed network which can be reduced to leaves of diagnostic trees. The topmost layer is an entry point to jet engine faults and subsequent layers organize the faults into various branches. The second layer is phases of engine operation and its branches lead to various symptomatic nodes labelled as snags. These snags in turn are refinable down to repair and replacement nodes which represent the terminal nodes of the diagnostic hierarchy[1]. If one examines the knowledge encoded in these terminal nodes more closely one discovers that they represent faults directly on physical engine components. These physical component fault nodes can be grouped into those affecting one of thirteen subsystems by their nomenclature. One can follow the five steps of the DR algorithm to discover the behavioural and functional component model for the main fuel system of the jet engine.

Step 1: One can identify 9 replace nodes through the JETA node frame slot 'node-type'.

Step 2: If one takes a specific subsystem, the MFS (Main Fuel System), one can extract the names of 3 fuel system replacement nodes by pattern matching with the node nomenclature *N-MFS-XXX (this is an internal representation that was used by the knowledge engineer to distinguish between nodes):

1. main fuel control (MFC)
2. overspeed governor for MFC (OSG)
3. main fuel pump supplying MFC (MFP)

Step 3:For each of the 3replacement nodes parents connecting sibling nodes can be extracted.

- MFC and MFP nodes share parent *fuel flow loss*
- OSG shares with MFC *engine speed hang-up* parent
- MFC shares *fuel flow loss* parent with fuel nozzles, FN
- pressurizing and drain valve (PDV) shares *low fuel flow* parent with FN

Step 4:A causal topological network can be the basis for hypothesized component-behaviour relations. Sibling nodes are clustered based on shared parent links. Example DR output relations that form part of the network include:

```
[main_fuel_nozzles,
is_a([nozzle,for,[fuel_flow_control]]),
and_is_connected_to(main_fuel_control),
with_connectivity_parameter(
     [measured_rpm_engine_speed,
     single_spool_engine_speed,weight_of_fuel_flow])],

[[main_fuel_control,
is_a([control,for,
     [steady_speed_control,speed_cutback_control,
     acceleration_fuel_limit_control,
     deceleration_fuel_limit_control,
     variable_geometry_scheduling,
     proportional_speed_control,
     fru_fuel_selection]]),
and_is_connected_to(main_fuel_pump),
with_connectivity_parameter([weight_of_fuel_flow])],

[main_fuel_nozzles,
is_a([nozzle,for,[fuel_flow_control]]),
and_is_connected_to(pressurizing_and_drain_valve),
with_connectivity_parameter(fuel_pump_inlet_pressure[])],
```

1. The diagnostic hierarchy is sometimes referred to as a network since it includes relations that are not directly inherited that allow the JETA reasoner to jump around between nodes thus forming more of a network than a hierarchy.

Step 5: Step 4 output is matched against device independent/dependent background knowledge and gaps identified. In the case of inconsistencies, in phase 1 of the DR algorithm parameters which are not explicitly related to components through background knowledge may point to inaccuracies that should be corrected. The complete component model for the main fuel nozzles (FN) with the identified gaps is:

```
[function(main_fuel_nozzles,flow_control(WF+)),
    input(main_fuel_nozzles,flow(WF+)),
    output(main_fuel_nozzles,flow(WF+)),
    regulator(main_fuel_nozzles,regulation_control(N+)),
    behaviour(main_fuel_nozzles,
            behaviour_is_proportional(WF+,N+,
            [increase_in(N+),increases(WF+),
            decrease_in(N+),decreases(WF+)]))],

[[main_fuel_nozzles,
    [[gap_for_mode,fuel_flow_control,
            extracted,EGT,input,[N+,WF],output,[WF+]],
    [gap_for_mode,fuel_flow_control,
            extracted,N,input,[N+,WF],output,[WF+]],[]]]],
```

Note that EGT is exhaust gas temperature and is an inconsistent link. The parameters engine speed (N) and fuel flow (Wf) are expected and the sign on N is missing as expected.

The partial view of the extracted MFS subsystem is shown in Figure 1.

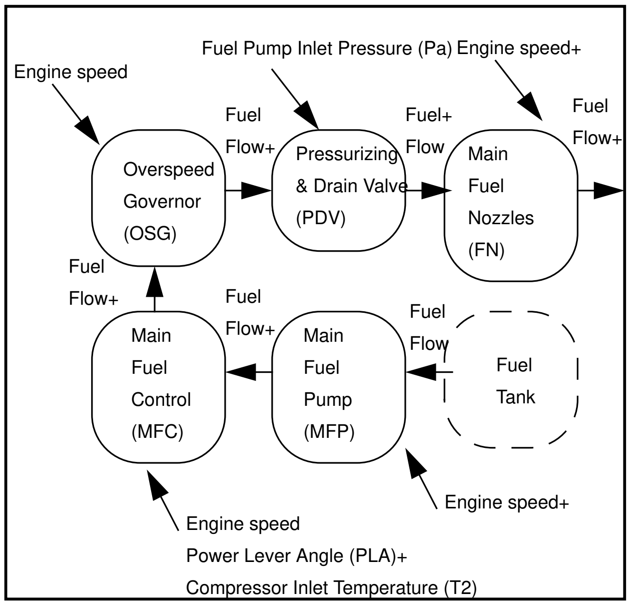

Figure 3: Main Fuel System model extracted by DR

Note that the main fuel pump and main fuel control filters extracted by DR were omitted to simplify the diagram. Thus, DR succeeds in extracting the 7 components (5 shown excluding the 2 filters and the fuel tank) and their respective connections in Phase 1. In the second phase the device dependent and device independent background knowledge is used to derive the direction and relations between the extracted parameters. Any gaps between JETA and the background knowledge are highlighted so that the fault-based knowledge can be made consistent or modified.

DR Coffee Maker Results & Device Model

To test the generality of the DR algorithm and relax some of its assumptions, I generated a 30-node knowledge base for the diagnosis of a coffee maker (a very different device than an aircraft engine). The coffee maker device had a variety of terminal nodes (not only replace types). DR selected all terminal nodes and assumed that they represented device components. Then, as before, parental nodes were used to identify sibling nodes and connections between them. Only 3 of the node slots of the frames of fault-based knowledge were used, the *node name*, the *child node of* and the *child node ranking* slots. From these slots the 5 steps (with step 1 relaxed) of the DR algorithm were used to generate the model in Figure 2.

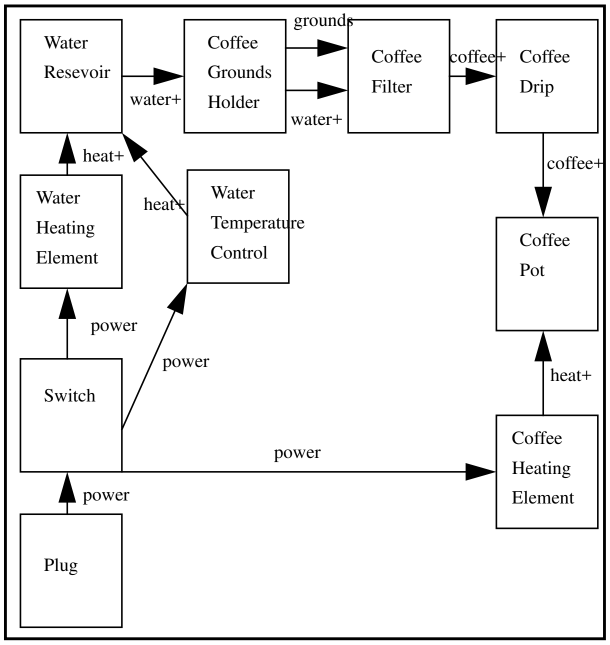

Figure 4: Coffee Maker device model as extracted by DR

A regulator, a switch, a heater, a holder and a filter were the device independent component models added to the library for background knowledge. In addition, 10 expressions that represented the device dependent background knowledge giving the type of component and the input/output behaviour parameters were used by DR. Thus, it was possible to successfully generate a component behaviour model for a full device (rather than only a subsystem) with explicit function and behaviour descriptions for each of the coffee maker components. To provide the reader with some detail, below are the DR generated component models for 2 of the 10 components, specifically for the coffee drip and the water temperature heating control.

```
[function(coffee drip,regulates(coffee+)),
       input(coffee drip,coffee+),output(coffee drip,coffee+),
      regulator(coffee drip,coffee+),
       behaviour(for(coffee drip),
behaviour_is_proportional(
      [increase_in(coffee+),increases(coffee+),
      decrease_in(coffee+),decreases(coffee+)])),

[function(water temperature heat control,regulates(heat+)),
      input(water temperature heat control,heat+),
      output(water temperature heat control,water+),
      regulator(water temperature heat control,heat+),
      behaviour(for(water temperature heat control),
      behaviour_is_proportional(
                  [increase_in(heat+),increases(water+),
                  decrease_in(heat+),decreases(water+)])),...
```

Discussion

There are several issues that I intended to answer in the implementation of the DR algorithm. The first is what is the exact form of the automatically acquired model when some or no background knowledge is used. If no background knowledge is used, is the model much more than a causal rather than a component behaviour model with explicit representation of function? The answer here is that a minimum amount of device dependent knowledge is used to map the fault-based syntax to more meaningful text as shown in the sample output for modelling JETA's main fuel system in DR step 4. If no device independent background knowledge (component library knowledge) is used, then the extraction of gaps between the fault-based encoded knowledge and a general one is not possible. Using no background knowledge, it is possible to extract a component-to-component model with explicit parametric links representing connections in FBR knowledge. Extracting the directions and relationships on these behavioural paths requires generalized device independent background knowledge.

The DR algorithm has been implemented as a general algorithm useful in generating models for devices through the re-use of fault knowledge. Its FBR knowledge used as input makes it specific to generating a model for a particular device. Its device independent background knowledge is re-usable across domains as different as a jet engine MFS is from a coffee maker. I have demonstrated this through the use of the DR algorithm for modelling two very different devices, a subsystem of an aircraft engine and a full coffee maker device. The DR algorithm requires a highly structured frame-based FBR knowledge base. One key question is what criteria will allow it to extract a component model from rule versus frame-based knowledge?

Conclusions

This paper addresses the problem of automated model acquisition for diagnosis. The DR algorithm automates the generation of component models with an explicit representation of behaviour and function. The DR algorithm re-uses FBR knowledge with two types of background knowledge to generate component models of a device. The ratio of background to fault knowledge (measured in lines of code) used by DR for the jet engine MFS was 14% (~170 to 1200) [Abu-Hakima 94a]. This measure helps illustrate that for a small additional investment in background knowledge, black box models for complex devices can be generated by DR through fault knowledge re-use.

References

Abu-Hakima, S. [1994a], Automated model acquisition through fault knowledge re-use, DR, the Diagnostic Remodeler Algorithm, PhD thesis, Carleton University, Ottawa, Canada (1994).

Abu-Hakima, S. [1994b], Diagnostic techniques in knowledge-based systems: a review of approaches, applications and issues. 100 p. NRC ERA (1994).

Abu-Hakima, S. [1993], Automatic Knowledge Acquisition in Diagnosis. Proceedings of DX-93, Fourth International Workshop on Principles of Diagnosis. Aberystwyth, Wales. 236-250. (1993), NRC #35111.

Althoff, K-D., Maurer, F., and Rehbold, R. [1990],"Multiple knowledge acquisition strategies in MOLTKE." *Current trends in knowledge acquisition*, Published by IOS, Amsterdam, Netherlands. pp. 21-40, (1990).

Boose, J.H. [1991], "Knowledge acquisition tools, methods and mediating representations", *Knowledge Acquisition for Knowledge-Based Systems*, Edited by H. Motoda, R. Mizoguchi, J. Boose and B. Gaines. Published by IOS Press, the Netherlands. pp. 25-62, (1991).

Chandrasekaran, B., Gomez, F., Mittal, S. and Smith, J.W. [1979], "An approach to medical diagnosis based on conceptual structures". Proceedings of the Joint Conference on Artificial Intelligence, 134-142.(1979).

Davis, R. [1984], "Diagnostic reasoning based on structure and behaviour", Artificial Intelligence, Vol. 24, pp. 347-410 (1984).

de Kleer, J., Williams, B.C. [1987], "Diagnosing Multiple Faults", *Artificial Intelligence*, vol. 32, (1987).

Halasz, M., Davidson, P., Abu-Hakima, S., and Phan, S. [1992], JETA: A Knowledge-based Approach to Aircraft Gas Turbine Engine Maintenance. Journal of Applied Intelligence, 2, pp. 25-46, Kluwer Academic Publishers, NRC 31832, (1992).

Hamscher, W. and Struss, P. [1990], "Model-Based Diagnosis", *AAAI-90 Tutorial Notes*, Received at the eighth national conference of artificial intelligence, Boston, Massachusetts. July 29, 1990. pp. 1-179, (1990).

Malin, J.T., and Liefker, D.B. [1991], "Representing functions/procedures and processes/structures for analysis of effects of failures on functions and operations", Proceedings of 1991 NASA Goddard Conference on Space Applications of Artificial Intelligence, Greenbelt, Maryland, pp. 141-151, (May 13-15 1991).

Sticklen, J., Chandrasekaran, B., and Bond, W.E. [1988], "Distributed causal reasoning for knowledge acquisition: a functional approach to device understanding", 3rd AAAI sponsored Knowledge Acquisition Workshop for Knowledge-Based Systems, Banff, Canada, pp. 34-1 to 34-18, (1988).

Struss, P. [1989], "New Techniques in Model-based Diagnosis", *Proceedings of Knowledge-based Computer Systems*, Bombay, India, (11-13 December 1989).

Struss, P., Dressler, O. [1989], "Physical Negation - Integrating Fault Models into the General Diagnostic Engine", *Proceedings of the 11th International Joint Conference on Artificial Intelligence (IJCAI-89)*, Detroit, MI, (20-25 August 1989)