Problem-Solving Methods for Diagnosis and their Role in Knowledge Acquisition*

Richard Benjamins

Laboratoire de Recherche en Informatique, University of Paris-Sud
Bat. 490, 91405 Orsay Cedex, France
e-mail: richard@lri.lri.fr
Tel: +33-1-69416495, Fax: +33-1-69416586

Abstract

Diagnosis is a popular topic in Artificial Intelligence. Recently, we presented a knowledge-level analysis of diagnosis covering a large scope of different approaches and systems. Such analysis gives us a better understanding of diagnosis in terms of the goals it comprises and the ways to achieve these goals. In the knowledge acquisition community, libraries with reusable knowledge components are becoming increasingly important. Such libraries facilitate the knowledge acquisition process in the sense that a problem-solving strategy can be constructed from ready made parts rather than to build it up from scratch. The result of the knowledge-level analysis of diagnosis can serve as such a library. The aim of this paper is, first, to present the collection of problem-solving methods and, second, to show some experiments that explore the usefulness of the library in knowledge acquisition. In order to do the experiments, it is necessary to associate problem-solving methods with suitability criteria that reflect the conditions under which problem-solving methods are applicable.

Keywords: diagnosis, knowledge acquisition, reusable components, construction of problem-solving models.

1 Introduction

In the literature, several approaches to diagnosis have been put forward (de Kleer & Williams, 1987, Davis, 1984, Genesereth, 1984, Console & Torasso, 1990b). A knowledge-level analysis (Newell, 1982) of these approaches identifies the relevant knowledge: the goals to be achieved and the ways to achieve these goals. In this article, we present such an analysis for diagnosis. In (Benjamins & Jansweijer, 1994), we claim that the analysis can be considered as part of a competence theory of diagnosis that is able to generate diagnostic problem-solving strategies.

Modeling approaches for knowledge acquisition such as KADS (Breuker *et al.*, 1987, Wielinga *et al.*, 1992a), Component of Expertise (KRESTA) (Steels, 1990, Steels, 1993), Task-structures (Chandrasekaran *et al.*, 1992), Spark, Burn, Firefighter (SBF) (Klinker

*To appear in the International Journal of Expert Systems: Research & Applications, 1995

et al., 1991), PROTÉGÉ-II (Puerta *et al.*, 1992), MIKE (Angele *et al.*, 1993), TASK (Pierret-Golbreich, 1994), and Problem-Solving Methods (McDermott, 1988) stress the importance of building libraries with reusable modeling components. Examples of library ingredients would include generic tasks, inference structures, problem-solving methods, applicability conditions, etc. The knowledge acquisition process is supported by a library because it allows the configuration of a problem solver from reusable components rather than to build it up from scratch.

The aim of this paper is twofold. First, we present the competence theory of diagnosis. Second, we consider the theory as a library and discuss some experiments to explore how the library supports the knowledge acquisition process. We are concerned with the construction of problem-solving strategies (Benjamins, 1994). As it turns out, we need so-called suitability criteria of methods that characterize the methods' applicability.

In Section 2, we describe a framework to model the diagnosis task. In Section 3, we present the knowledge-level analysis of diagnosis. In Section 4, we consider the analysis' result as a library and line out how it supports automatic knowledge acquisition. In Section 5, we discuss the results, and, in Section 6, we relate our work on method-selection to that of others. Finally, Section 7 concludes the paper.

2 Modeling framework

Our modeling framework reflects a convergence of current modeling approaches. Its principal elements are tasks, problem-solving methods, and primitive inferences (see Figure 1).

Task A task has a goal and is characterized by the type of input it receives and the type of output it produces. A task is a specification of *what* needs to be achieved. For example, the goal of diagnosis is to find a solution that explains the initial and the additional observations. A task can be decomposed into subtasks (by a problem-solving method).

Problem-Solving method A problem-solving method (or method, PSM) defines a way *how* the goal of a task can be achieved. It has inputs and outputs and decomposes a task into subtasks and/or primitive inferences. In addition, a method specifies the data flow between its constituents. Control knowledge determines the execution order and iterations of the subtasks and inferences of a PSM. Control knowledge can be specified in advance, if known, or can be opportunistically determined at run time depending on the dynamic problem-solving situation.

Primitive inference A primitive inference (or inference) defines a reasoning step that can be carried out using domain knowledge to achieve a goal. It has a body that specifies how the goal can be achieved using domain knowledge. Inferences form the actual building blocks of a problem solver. When all tasks are decomposed, through PSMs, into inferences, the corresponding data-flow structure forms an **inference structure** (Schreiber *et al.*, 1993). If control knowledge is also specified, we get a problem-solving strategy.

Task-method structure Figure 1 illustrates the relations between tasks, PSMs, and inferences. A task may be realized by several methods, each of which consists of subtasks and/or inferences. Figure 1 is a tangled hierarchy because the same inferences, tasks, and

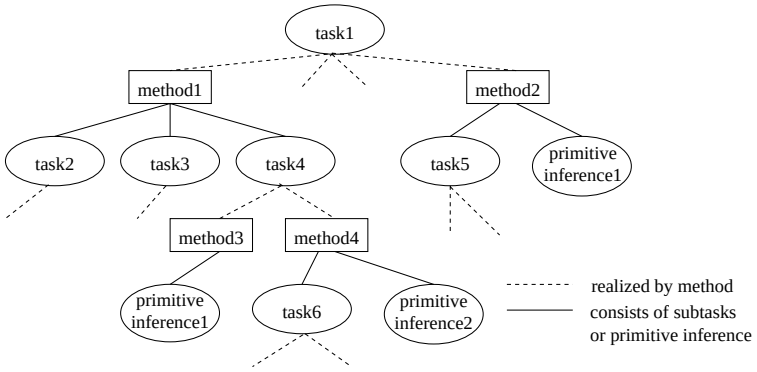

Figure 1: The relations between tasks, methods, and inferences. Dashed lines denote that methods are alternatives for realizing tasks. Solid lines decompose a method into subtasks and/or inferences.

methods can appear at several places (e.g., inference1 in Figure 1); in other words, they are reusable.

Most modeling approaches to Knowledge Acquisition agree on the basic notions underlying the terms described above. There may, however, be slight differences in their precise definitions, in particular, concerning the primitive inferences. In this article, inferences are oriented towards diagnosis. They are a more specific version of the inferences in CommonKADS(Wielinga *et al.*, 1992b, Breuker & van de Velde, 1994), and they are the correspondent of the mechanisms in PROTÉGÉ-II (Puerta *et al.*, 1992), DIDS (Runkel & Birmingham, 1993) and SBF (Klinker *et al.*, 1991), and of the solution methods in KREST (Steels, 1993). A difference is that mechanisms and solution methods are operational procedures, while inferences are not.

3 Diagnosis

We view diagnosis as the task of identifying the cause of a fault that manifests itself through some observed behavior. In the work described here, diagnosis is conceived as consisting of three subtasks, namely (1) *symptom detection*, finding out whether the complaints are indeed symptoms, where a symptom (or abnormality observation) is defined as an observation that deviates from its expectation; (2) *hypothesis generation*, based on the symptoms, generating possible causes, also taking into account the non-symptoms (normality observations); and (3) *hypothesis discrimination*, discriminating between the generated hypotheses, based on additional observations. This decomposition of the diagnosis task (which we refer to as the "prime diagnostic method" (PDM)) is motivated by the work of Davis and Hamscher (Davis & Hamscher, 1988). We distinguish between initial and additional observations because additional observations are often a cost-determining factor in practical diagnosis: to minimize such costs, we must know what reasoning types are involved.

The input of the prime diagnostic method (Figure 2) is a set of initial observations, which are also the input of the symptom detection task. This task decides whether observations are abnormality or normality observations. Abnormality and normality obser-

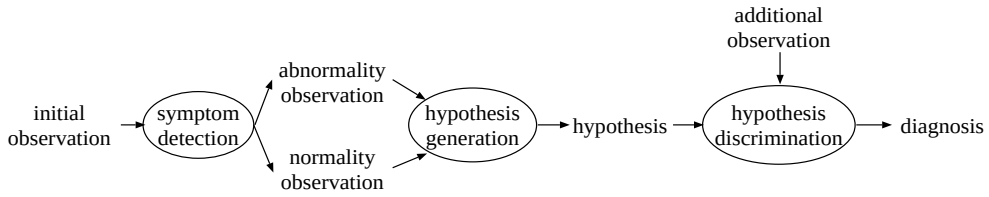

Figure 2: Data flow diagram of the prime diagnostic method. Ellipses represent tasks.

vations are input for the hypothesis generation task. The task generates hypotheses in a hypothesis set, which is the input for the hypothesis discrimination task. Here, additional observations are taken into account. The output of the discrimination task, which is also the PDM output, is a set of diagnoses of which each diagnosis explains both the initial and additional observations.

3.1 A knowledge-level analysis of diagnosis

Based on analyses of the reasoning process of service engineers (Benjamins & Jansweijer, 1989) and on related literature, we identified 38 PSMs and 14 tasks relevant for diagnosis. We present the library following the basic decomposition of diagnosis into its three subtasks: symptom detection (Figure 3), hypothesis generation (Figure 4), and hypothesis discrimination (Figure 5). To facilitate reading, we will refer to the corresponding figure in which the subtasks appear (Figure 3, 4 or 5), after the introduction of each new subtask. PSMs are discussed under the subtask they can realize.

The symptom detection task (Figure 3) There are several methods and subtasks for detecting the normality or abnormality of observations. Figure 3 shows the different methods and how the compare method might be decomposed into subtasks and inferences. Methods are represented by rectangles and tasks by ellipses. Existing diagnostic systems or references to relevant literature are also included (surrounded by dashed lines). The **compare method** generates an expected value for the input observation and then compares the observation and the expectation. This yields a datum containing one of two results: equal or not equal. If the observation and the expectation are judged equal, the observation is considered a normality observation. Otherwise, the observation is considered an abnormality observation (symptom). The **classify method** assumes the availability of domain classification knowledge; through this an initial observation is classified as normal or abnormal. The **user method** relies on a knowledgeable user to determine the status of an observation (e.g., INTERNIST-I (Miller *et al.*, 1982)).

The generate expectation task (Figure 3) Two methods can be used to generate an expectation for an observation. The **lookup method** consults a data base where all relevant expected values are stored (Benjamins & Abu-Hanna, 1990, Steels, 1989). The **prediction method** computes expected values based on a simulation of a device model. Systems which apply this method on structural and behavioral device models include GDE (de Kleer & Williams, 1987) and its extensions (Struss & Dressler, 1989, de Kleer & Williams, 1989, Hamscher, 1988), HT (Davis, 1984) and DART (Genesereth, 1984).

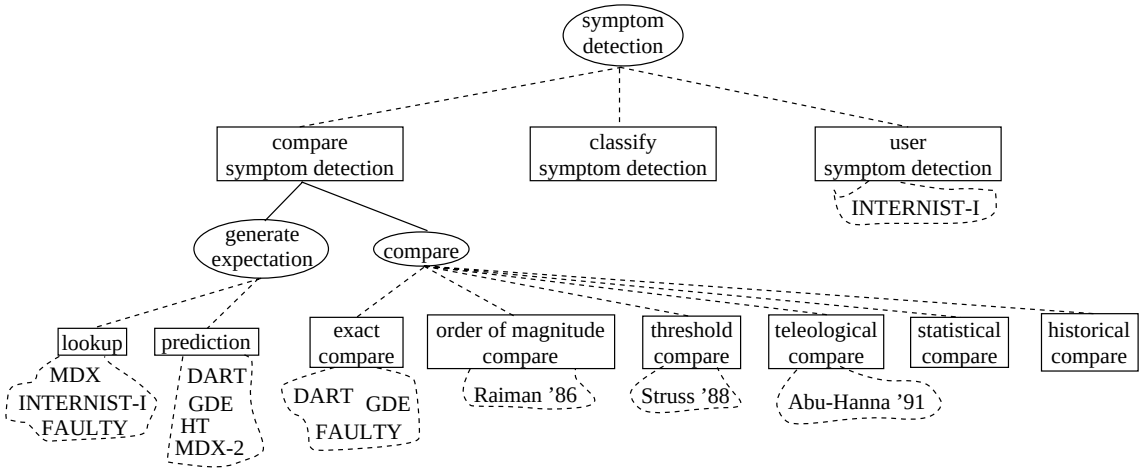

Figure 3: A partial task-method decomposition for the symptom detection task. Rectangles represent methods and ellipses represent tasks. Dashed lines indicate that methods are alternatives for achieving the task goal. Solid lines decompose a method into its subtasks. Note that not all methods are decomposed into their subtasks and primitive inferences.

The compare task (Figure 3) This task decides whether an observation and expectation are equal or not. We have identified several methods for the compare task, reflecting different precision of comparison or the fact that data must be seen in a broader context.

The **exact compare method** directly compares an observation and an expectation and is appropriate in domains where values are exact. This method is used in many diagnostic systems, such as GDE, HT, DART, and FAULTY (Benjamins & Abu-Hanna, 1990). When there are tolerances in system behavior, other methods are needed. In the **order of magnitude compare method**, an observation and expectation that differ but are in the same order of magnitude are considered equal (Raiman, 1986). The **threshold compare method** checks whether the difference between an observation and an expectation exceeds a certain threshold (Struss, 1988). The **teleological compare method** performs a teleological abstraction (according to its function or its purpose) from an observation before comparing it to the expectation in order to prevent false alarms (Abu-Hanna *et al.*, 1991); an abnormality observation (symptom) is thus a discrepancy at the teleological level. In the **statistical compare method**, an expectation is expressed in terms of statistical information (e.g., a normal distribution) and is then compared with the observation. The **historical compare method** uses historical data about a particular device to decide whether a difference between an observation and its expectation should be considered as a discrepancy.

The hypothesis generation task (Figure 4) This task generates hypotheses that explain the set of initial (abnormality and normality) observations. The normality observations can yield extra discriminatory power to the hypothesis generation process. Figure 4 shows an overview of methods and tasks relevant for hypothesis generation.

The **compiled method** for hypothesis generation exploits associations between symptoms and causes and is sometimes followed by a probability filter. This method is often used in domains where the understanding of underlying mechanisms is only partial, such as in medical diagnosis. Examples of this method can be found in MDX for associating symp-

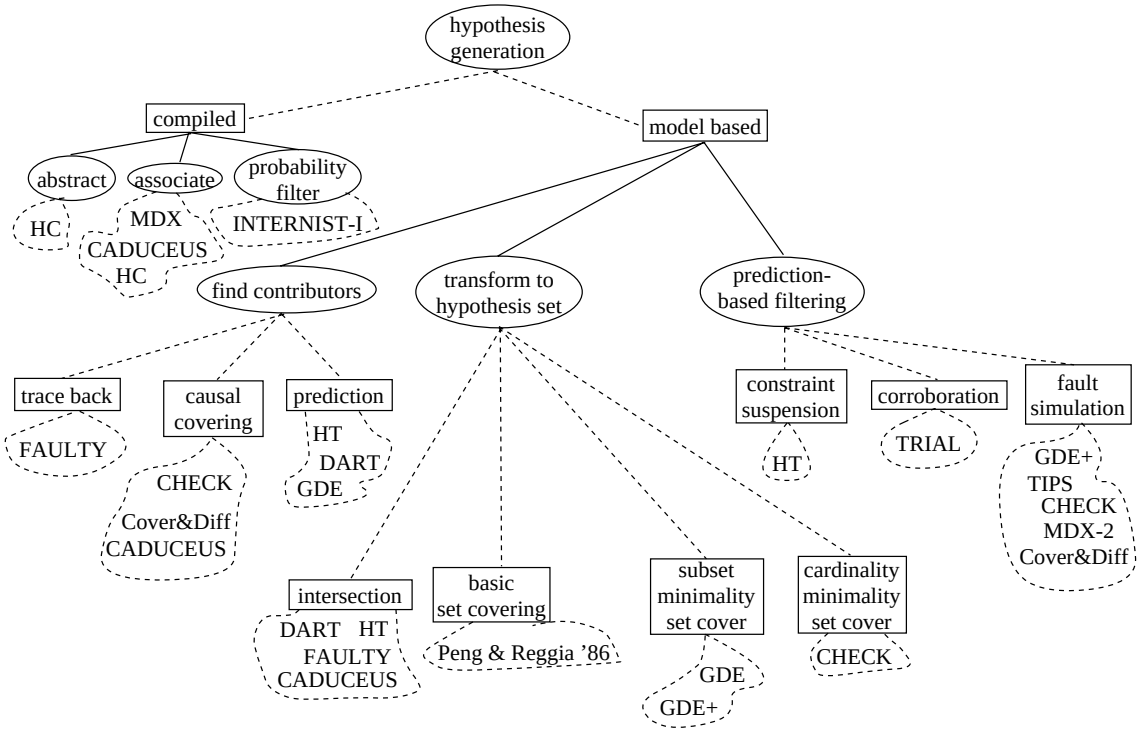

Figure 4: A partial task-method decomposition for the hypothesis generation task.

toms with categories in the classification hierarchy (Josephson & Josephson, 1994) and in Clancey's Heuristic Classification (HC) for matching (Clancey, 1985). Clancey points out that sometimes *abstracted* observations (such as "qualitative" and "definitional" abstractions and abstractions by "generalization") are used in place of *raw* observations to generate hypotheses. In INTERNIST-I, a probability filter is applied to the hypotheses generated, and CADUCEUS (Pople, 1982) uses "planning links" to associate observations to categories in classification hierarchies and "constrictor links" to associate observations to states in a causal network. Because the compiled method is not further worked out in this paper, some diagnostic systems in Figure 4 appear under the tasks or inferences they carry out instead of under the methods.

The **model-based method** for hypothesis generation consists of three subtasks:

- Find a set of model entities (such as components, states, functions) that contribute to an abnormality observation (this is the equivalent of a conflict in GDE (de Kleer & Williams, 1987) and a conflict set in (Reiter, 1987).) A contributor set has at least one broken element.
- Transform the contributor sets into a hypothesis set in which every element is a possible explanation for the observations.
- Use prediction-based filtering to further discard inconsistent hypotheses from the hypothesis set using only initial observations. Filtering is thus considered part of hypothesis generation, called hypothesis testing in (Davis & Hamscher, 1988). Filtering is an optional subtask.

The find contributors task (Figure 4) The **trace back method** finds the contributors to an abnormality observation by tracing backward in a domain model that represents correct behavior (exploiting the network representation of the model). The method is applied in FAULTY. The **causal covering method** operates on a causal model and has an abductive nature. The causal model's network representation determines whether contributors should be generated incrementally (as in Cover & Differentiate (Eshelman, 1988)) or all at once (as in CHECK (Console & Torasso, 1990a)). The trace back and the causal covering method are similar. The latter can be seen as a more specific version of the former (Aben, 1995). In the **prediction** method, the contributor sets are a product of a simulation process. During simulation, the justifications for computed values are recorded. If an inconsistency is found between the simulation's result and the actual observations, the contributors to this inconsistency can be found in the stored justifications. This is the method used in, among others, DART and HT. Inconsistencies do not necessarily occur between observations and predictions alone. If there is knowledge that derives input behavior from a behavioral rule and output behavior (so-called inference rules (Davis, 1984)), inconsistencies can arise when different values are predicted for the same point (de Kleer & Williams, 1987). An assumption-based truth maintenance system gives a direct implementation for this method.

The transform to hypothesis set task (Figure 4) One way to transform the contributor sets into a hypothesis set is to apply the **set-covering method**. It generates all hypotheses that have a non-empty intersection with any of the contributor sets. Thus, each hypothesis covers all contributor sets. This method assumes complete independence of causes; that is, an individual cause explains a set of observations regardless of what other causes are being considered. The set-cover method itself does not apply a parsimony criterion to prune the hypothesis set; it yields all possible hypotheses (including multiple faults) that are consistent with the observations so far. The set-cover method can be strengthened by three different parsimonious criteria which are increasingly stronger (Aben, 1995). The **cardinality minimality** method prefers hypotheses with low cardinality (Console *et al.*, 1989). The **subset minimality** method (de Kleer & Williams, 1987) discards hypotheses that are supersets of other hypotheses, thereby assuming that fault behavior is not constrained. That is, an incorrect model entity can manifest *any* behavior (Hamscher, 1991), an assumption which rarely holds in medical domains because there domain models often represent fault behavior. In the **intersection method**, all contributor sets are intersected, and the (non-empty) intersection yields the hypotheses. If an intersection contains more than one element, each element is considered as a separate hypothesis (assuming single faults). The intersection method is applied in FAULTY, DART, HT, and CADUCEUS, among others.

The prediction based filtering task (Figure 4) This task uses a prediction process using only initial observations to further exclude inconsistent hypotheses. The **constraint suspension method** (Davis, 1984) suspends for each hypothesis in the hypothesis set the behavior of the corresponding model entity (or entities) (by revoking the corresponding simulation and inference rules). The remaining model is simulated taking into account the initial observations. If the result of the simulation process is consistent, the hypothesis is valid and remains in the hypothesis set. Otherwise, the hypothesis cannot consistently explain the initial observations and is discarded. When an observed value matches its

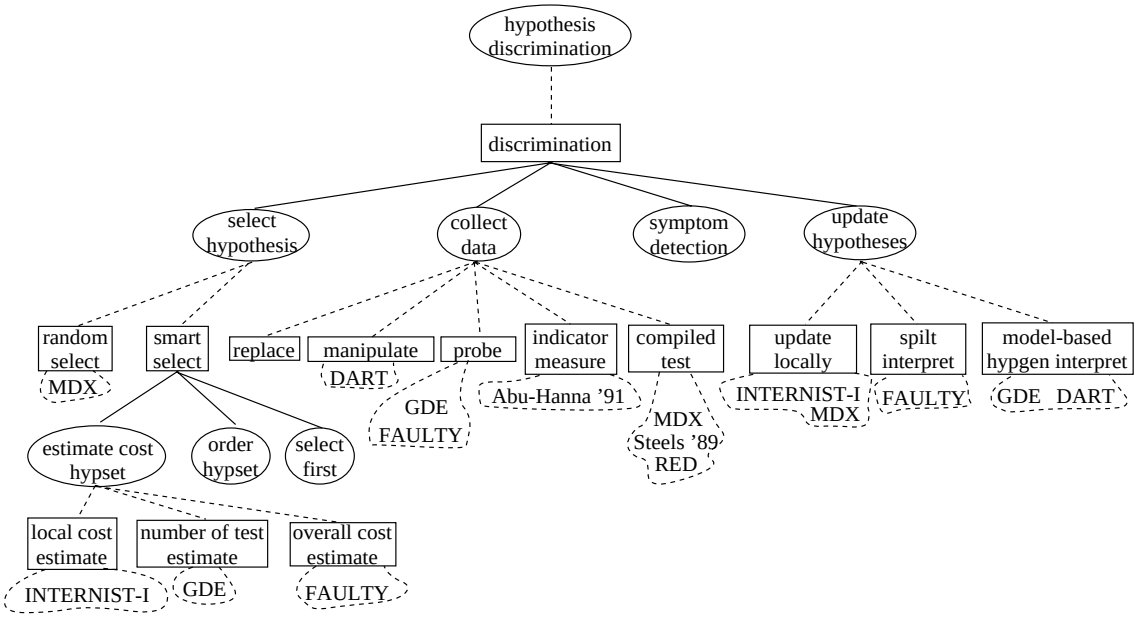

Figure 5: A partial task-method decomposition for the hypothesis discrimination task.

expectation, corroboration occurs. The **corroboration method** applies the idea that any hypothesis (model entity) involved in a corroboration must be innocent and can therefore be discarded from the hypothesis set. The method is used in TRIAL (Raiman, 1989). There are two ways to decide on the involvement of a hypothesis in a corroboration: a detailed simulation or a dependency analysis. Using the notion of corroboration to prune the hypothesis set assumes that fault masking does not occur. The **fault simulation method** instantiates a fault model corresponding to one of the hypotheses. If the behavior predicted from this fault model is inconsistent with the observations, the hypothesis can be discarded (assuming only one fault model for every hypothesis). The fault simulation method is used in CHECK, Cover & Differentiate, and in NEOMYCIN's (Clancey, 1993) "forward reason". TIPS (Punch, 1989) applies causal reasoning and MDX-2 (Josephson & Josephson, 1994) applies "Functional Reasoning" (Sticklen *et al.*, 1989) to simulate the consequences of a fault model. In (Struss & Dressler, 1989), fault models are applied to extend GDE to GDE+ to enrich it with the notion of what is possible and impossible from a physical point of view (as opposed to only considering a logical point of view).

The hypothesis discrimination task (Figure 5) In this task, additional observations are gathered and interpreted to further discriminate between the hypotheses. The output of the discrimination task is a set of diagnoses, each explaining the initial and additional observations. Figure 5 gives a partial decomposition of this task. The **discrimination method** consists of four subtasks: select hypothesis, collect data, symptom detection (to decide on the (ab)normality of observations) (Orsvärn, 1994), and update hypothesis set. The symptom detection task has been described earlier in this article.

The hypothesis selection task (Figure 5) In this task, a hypothesis that requires additional data is selected. This can be done in several ways. The **random select method**

is a simple approach and is useful when relations between hypotheses are not modeled. For example in an establish-and-refine strategy (Chandrasekaran, 1986), the categories at one level of the classification hierarchy represent unrelated hypotheses; each hypothesis has to be evaluated. A more refined method is to select a hypothesis according to some criterion. In the **smart select method**, every hypothesis in the set is associated with a cost estimation for testing the whole hypothesis set starting from that specific hypothesis. This estimation can be based on different types of knowledge depending on the domain. For example, it can be based on **local costs** (e.g., reachability of measuring points, a priori probability of hypotheses, risk of tests) or on the expected **number of tests** to be executed or on a combination of these two (**overall costs**). The hypotheses are ordered according to their cost estimation and data collection begins with the first hypothesis in the ordered list. INTERNIST-I applies the local method using the evoking strength, the frequency and the importance of the observations with respect to the hypothesis. GDE uses a method that minimizes the number of tests to be performed, also considering failure probabilities. FAULTY uses a method to minimize the overall cost of performing tests.

The collect data task (Figure 5) This task acquires additional observations. The **compiled test method** requires that for each hypothesis, knowledge be available about test procedures (Steels, 1989). The parts of "establish" and "hypothesis matching" of MDX and RED1 and 2 (Josephson & Josephson, 1994) that are concerned with obtaining additional observations also apply the compiled method. Each node (hypothesis) in the classification hierarchy is a specialist that knows how to establish or reject itself by determining how well it fits the observations (The refine step of establish & refine we consider as taking a more detailed or refined view on the domain knowledge, to which the prime diagnostic method (PDM) can be applied recursively. The same holds for the sub-classification task in CADUCEUS.) The **probe method** obtains an additional observation through a probe at some measuring point. The observations are obtained at the components that correspond to a hypothesis. A requirement is that these components are accessible for observations. This is more easily satisfied in technical than in medical domains. The method is applied in several device diagnostic systems including GDE, FAULTY, and DYNAMIS (Chittaro *et al.*, 1989). When measuring points are difficult to reach, the **indicator-based measuring method** provides a way out. An indicator is an alternative observation which, however partial, inexact, or indirect, gives an indication of the status of a hypothesis. (Abu-Hanna *et al.*, 1991) sketch a method to derive such indicators in a model-based way. Another method to collect additional data is the **manipulating method**, which forces inputs on the device and observes the result. With respect to the hypothesis we want to verify, we need to deduce a discriminating input vector. Basically, what needs to be done is to determine what kind of *internal* inputs are needed to test the hypothesis and then reason backwards to deduce what *external* inputs are required to enforce the internal inputs. Then, a forward simulation is started to generate an expectation at the device output(s). The manipulation method faces troubles when, while deducing a test input vector, suspected components are encountered. In such cases, one cannot be sure that the applied input vector establishes the intended result. This method is used in DART. In the **replace** method, a suspected component is replaced by a correct one. In this method, the diagnostic action (collect additional data) and repair can coincide.

The update hypothesis task (Figure 5) Once we know whether an additional observation is normal or abnormal (as decided by the symptom detection task (see earlier)), it has to be interpreted with respect to the hypothesis set so that it explains the initial and additional observations. As we will see, this task might involve the same reasoning as in the hypothesis generation task. This similarity is not surprising because, instead of initial observations (used in hypothesis generation), additional observations are now used to update the hypothesis set. A simple method to update the hypothesis set is to interpret the test result locally (**update locally**). That is, the hypothesis is either discarded or kept depending on the acquired datum. This approach is useful when hypotheses are unrelated. MDX applies this method to conclude an establish step. INTERNIST-I also uses this method. If the hypotheses in the hypothesis set are in some way related to each other and the single fault assumptions hold, one can apply the **split interpret method**. Here, the hypothesis set is split into two parts, and one is discarded depending on the status of the additional observation. Falsification of the tested hypothesis falsifies all hypotheses contributing to the place where the datum was obtained (falsification here means that the observations are according to the expectations). Verification, on the other hand, enables discarding all hypotheses that did *not* contribute. This method is used in FAULTY. Another method to interpret additional data is to start the **model-based hypothesis generation** process again, this time with the addition of the new observations. In many diagnostic systems, the interpretation of additional data is done by iterating the same basic steps: generation and discrimination (e.g., GDE, DART).

4 Knowledge acquisition support

In (Benjamins & Jansweijer, 1994), we claim that the collection of PSMs and tasks form part of a "competence theory of diagnosis" that is able to generate many different diagnostic problem-solving strategies. We will back up this claim with some experiments in which we generate and analyze inference structures for diagnosis (strategy = inference structure + control knowledge, see Section 2). The collection of PSMs, tasks, and inferences is considered as a library of reusable knowledge components that can support the knowledge acquisition process. Examples of the kind of support we mean include *given some characterization of the domain at hand, what are the possible inference structures to solve diagnostic problems*, and, *given a certain inference structure, what conditions should be satisfied to apply it successfully.* To have a real inference structure, we would have to include the inputs and outputs of the inferences (Schreiber *et al.*, 1993). However, in this article, we are more interested in the reasoning steps than in their IO-data and leave the inputs and outputs out.

In order to do our knowledge acquisition experiments, we need to be able to express the *requirements* of our PSMs and the *characteristics* of our application domain that we want to diagnose.

4.1 Requirements of PSMs

Suitability criteria of a method refer to properties that have to be true in order to apply the method to a task. They are local requirements that refer to features of the method itself and not to other PSMs.

According to the interaction hypothesis (Chandrasekaran & Johnson, 1993), reasoning knowledge cannot be completely specified without considering the domain knowledge it reasons with. Suitability criteria try to capture the most general domain characteristics a PSM needs to reason with. Thus, a PSM depends on domain knowledge but refers to it in such a way that it is as reusable as possible across domains.

We distinguish three types of criteria, corresponding to relevant categories of knowledge, to consider in knowledge acquisition.

- **Epistemological criteria** specify the type of domain knowledge that has to be available in order to make the PSM work. For example, a prediction method requires that the entities (e.g., components) in the device model have simulation rules (behavioral constraints) that relate inputs to outputs, in such a way that behavioral consequences can be derived.
- **Environmental criteria** refer to environmental requirements that have to be met before a PSM is applicable. For instance, a method for testing hypotheses by measuring requires that the device is accessible for measurements and that measuring tools are available.
- **Assumption criteria** specify underlying assumptions of the method (e.g., the single fault or non-intermittency assumption).

Table 1 gives an overview of the suitability criteria currently included. The criteria are statements that should be true before a method can be applied to a task. For instance, the criterion "causal model" means that it has to be possible to construct a causal model.

The epistemological criteria bear resemblance to the "method ontology" in PROTÉGÉ-II (Gennari *et al.*, 1994), that defines the data requirements of the methods. Suitability criteria are also reminiscent of task features (Aamodt *et al.*, 1992). Finally, our criteria are related to the set of criteria for choosing alternative methods in (Chandrasekaran, 1990), the "sponsors" in TIPS (Punch, 1989), and to the applicability conditions in (Vanwelkenhuysen & Rademakers, 1990). However, our criteria refer mostly to static properties and are therefore relevant for knowledge acquisition, whereas the work described in (Chandrasekaran, 1990, Punch, 1989, Vanwelkenhuysen & Rademakers, 1990) is more concerned with dynamic properties that change during the problem-solving process. They are more useful in flexible reasoning rather than in knowledge acquisition.

Organization of criteria Analogous to the task-method decomposition, criteria can be hierarchically organized according to their generality. Lower level criteria are more specific versions of the higher level ones. Figure 6 shows an example of such an organization. The figure illustrates that the criteria "simulation rules," "inference rules," and "fault simulation rules" are more specific than the criterion "dependencies in model," which is more specific than the criterion "device model." Such an organization of criteria enables us to make the following deductions: if C_x is true and $more\_specific(C_x, C_y)$, then C_y is true, and if C_y is false, then C_x is false. For example, if there are simulation rules, then we can derive that there are dependencies in the model. And, if there are no dependencies in the model, we can derive that there are also no simulation rules. The hierarchical organization is used to evaluate criteria concerning their truth value.

Epistemological criteria	**Explanation**
abstraction knowledge	knowledge to abstract (raw) data
causal model	causal device model
component hypotheses	hypotheses are components
correct device model	device model represents correct behavior
cost info	info about test costs
dependencies in model	model entities are in some way related to each other
device model	any device model
empirical associations	associations between symptoms and hypotheses
expected value database	expected values are pre-stored
expected values obtainable	expectations can be obtained (regardless how)
fault behavior not constrained	faulty component can manifest any behavior
fault simulation rules	rules that describe component behavior when it is broken
historical info	previous problems that a particular device has had
hypotheses unrelated	hypotheses in hypothesis set not related
hypotheses related	hypotheses are related to each other
inference rules	rules to calculate input based on outputs and inputs
knowledge about teleology	purpose of behavior is known (for abstractions)
knowledge for classifying symptoms	classification knowledge (e.g., a hierarchy)
local cost info	local test cost information (such as failure rates)
no net fanout structure	no device model fanout in direction of inputs
probability information	likelihood of hypotheses
simulation rules	rules to compute outputs from inputs
statistical info	statistics about device behavior
tests associated to hypotheses	test procedures to test hypotheses
threshold info	when two values are considered different or equal
Environmental criteria	
(im)precise values	result of (non)tolerances in device behavior
additional observations	possibility to gather extra information during diagnosis
components replaceable	components are replaceable (for testing)
device accessible	device can be unfastened (for testing)
device stable in time	a particular device does not radically change its global behavior
failure rates equal	components are equally likely to fail
measuring points reachable	points to probe can be accessed
measuring tools	test equipment
reachability equal	measuring points or components are equally reachable
user knowledgeable symptoms	users knows what is normal and not
Assumption criteria	
complete association set	no symptom—hypothesis association is missing
complete expected value database	all relevant expectations are available
complete fault model	no fault model is missing
exhaustivity assumption	each symptom has at least one state that covers it
hypotheses can be generated	all possible causes can be generated
independence of causes	cause explains observation(s) regardless of other causes
no fault masking	if there is a fault, it is manifested
non-intermittency assumption	device behaves consistently during diagnostic session
single fault assumption	one fault explains the observations

Table 1: Suitability criteria of problem-solving methods for diagnosis along with a short explanation.

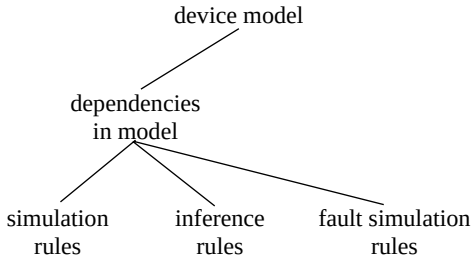

Figure 6: Example organization of suitability criteria according to their generality.

4.2 Characterizing the domain

Suitability criteria form the connection between problem-solving methods and the application or problem domain. The former require characteristics of the application domain while the latter can be described as having certain characteristics. In the real world, they may be expressed in different terms depending on the point of view. For the sake of simplicity, in the experiments we will use the same terms for both. That is, an application domain is described by characteristics using the terminology of suitability criteria. In the following, we illustrate how some domain characteristics can be expressed using criteria terms. The experiments in Section 4.4 should be read in a similar manner.

A particular domain has certain characteristics. For example, a cam-corder (a video camera and recorder in one device) can be described by the following characteristics (Benjamins & Abu-Hanna, 1990). The device can be modeled in terms of its constituting components. Due the complexity of the behavior of the components, it is difficult to construct a simulation model. However, dependencies between components can be specified. These characteristics can be expressed by the criteria "dependencies in model" (and thus also by "device model," see Section 4.1) and "component hypotheses" (the entities in the device model are components). Because there is no simulation model, expected values of outputs and measuring points are taken from a database, expressed by the statement "expected value database." A cam-corder can be unfastened when additional observations are required; thus it has the characteristic "device accessible." Measuring points are known ("measuring points reachable" and "local cost info"), and suspected components can be replaced by correct ones ("components replaceable"). Most of the observations are done with an oscilloscope ("measuring tools") which presents a video-signal: a roughly defined pattern on the scope ("imprecise values") which is correct or incorrect according to some threshold ("threshold info"). Most of the faults turn out to be single faults ("single fault assumption").

4.3 Representation of the library

The library is represented as a quadruple $< D, T, PI, PSM >$. T denotes the set of tasks relevant for diagnosis D. PI stands for the set of primitive inferences. PSM is the set of problem-solving methods that decompose tasks into subtasks or inferences. Suitability criteria are defined for each method in PSM. In terms of a grammar, D would be the start symbol, T the non-terminals, PI the terminals, and PSM the rewrite rules.

We use the Definite Clause Grammar (DCG) of Prolog to represent our library. Because DCG allows for normal Prolog code within the grammar, we can easily include references to

the suitability criteria. Figure 7 shows some rewrite rules. In the actual implementation, the grammar includes additional variables to build up the method tree, but these are left out for clarity. The suitability criteria of the methods are represented separately from the grammar. Figure 8 shows some representations of suitability criteria.

```
% The prime diagnostic method for the diagnosis task
        diagnosis −→
                {criteria(prime_diagnostic_method)},
                        symptom_detection,
                        hypothesis_generation,
                        hypothesis_discrimination.
% Methods for the hypothesis generation task
        hypothesis_generation −→
                {criteria(compiled_hypothesis_generation_method)},
                        [abstract],
                        [associate],
                        [probability_filter].
        hypothesis_generation −→
                {criteria(model_based_hypothesis_generation_method)},
                        find_contributors,
                        transform_to_hypothesis_set,
                        prediction_based_filtering.
```

Figure 7: Some DCG grammar rules of the library. The term at the left side of the arrow in a grammar rule represents a task. The part at the right side of the arrow represents a method. It consists of a reference to the method's suitability criteria and a set of subtasks and/or inferences. The inferences appear between squared brackets. Alternative methods for the same task are represented by separate rules.

```
epistemological(model_based_hypothesis_generation_method, device_model).
epistemological(constraint_suspension_method, simulation_rules).
epistemological(constraint_suspension_method, inference_rules).
assumption(corroboration_method, no_fault_masking).
environmental(probing_method, device_accessible).
```

Figure 8: Some examples of PSM suitability criteria indexed by their type.

4.4 Experiments

We present four experiments to illustrate the usefulness of the library for knowledge acquisition. In the following, when we write requirements of a method, we mean the suitability criteria to be satisfied before the method is applicable.

We have implemented a small Prolog program called TINA (**T**ool **in** **A**cquisition) that consults the grammar and performs the experiments. The word "experiment" should be understood as a tentative way to explore the possibilities of the library rather than as a thorough approach to testing it.

Experiment 1 — Given a particular domain, generate the applicable inference structures. The aim of this experiment is to evaluate whether the library supports the generation of interesting diagnostic inference structures given a particular domain. We take the domain of the cam-corder, which can be described by the characteristics shown in Table 2.

Epistemological	**Environmental**
component hypotheses	additional observations
cost info	components replaceable
dependencies in model	device accessible
device model	imprecise values
local cost info	measuring points reachable
threshold info	measuring tools
Assumption	
complete expected value database	hypotheses can be generated
no fault masking	non-intermittency assumption
single fault assumption	

Table 2: Characterizing the cam-corder domain.

If we provide TINA with these domain characteristics, it generates six possible inference structures, that is, inference structures generated by PSMs whose criteria match the domain characteristics. One of the six inference structures is shown in Figure 9 as a method tree (containing only methods and inferences). The leaves of the tree represent the inferences (in Helvetica) that, when connected through their inputs and outputs, would yield the inference structure. Look at the right part of Figure 9, that is, at the inferences, to get a global idea of the diagnostic problem-solving process suggested (a complete strategy needs control knowledge). To understand the method tree, refer to Section 3. We assume that if within an inference structure a task is performed more than once, it will be realized by the same method (e.g. the compare symptom detection method in Figure 9).

The other five inference structures imply the substitution of a PSM with another applicable PSM. Instead of the probing method, a suspected component could be replaced with a correct one, or the indicator based method could be used. The criteria of both these methods are met. The split interpret method can be replaced by the model-based hypothesis generation method, but now with the additional observation unified with the initial observations.

Thus, by providing TINA with the characteristics of an application domain (in terms of the suitability criteria), the library supports the generation of applicable inference structures. It is difficult to "prove" that the generated inference structures are the best ones, but we can demonstrate that some inference structures *not* generated are not the best. For example, the generated inference structure selects a hypothesis to be tested based on the *overall* cost of testing hypotheses and not on the *number* of expected tests. In diagnosis, hypothesis selection based on the number of expected tests only makes sense if the reachability of the measuring points and the failure rates of components are more or less equal. This is not the case in the camcorder.

Experiment 2 — Allow relaxation of some suitability criteria. It is possible that the requirements of the domain are too constrained, and no applicable inference structure

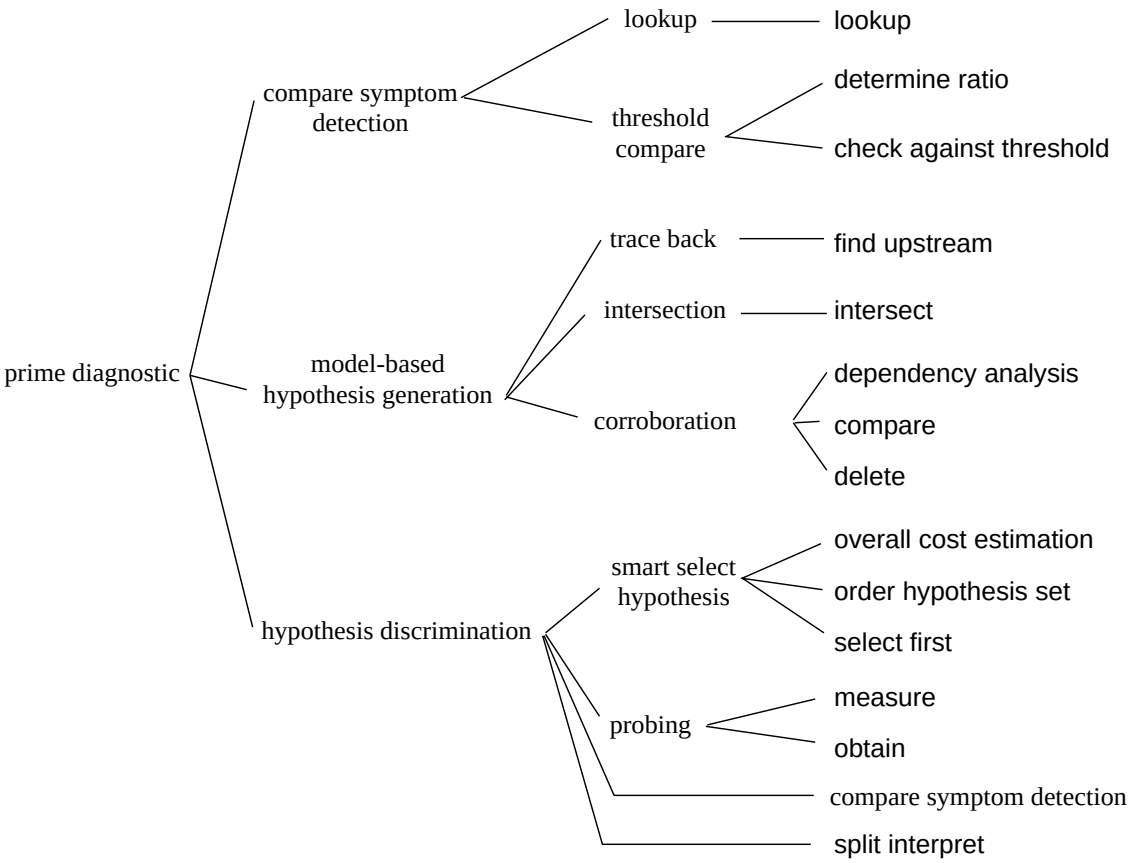

Figure 9: The method tree corresponding to the inference structure in the camcorder domain. The leaves are the inferences (in Helvetica).

can be generated. To deal with such a situation, we added to TINA the possibility to relax some criteria of PSMs according to the knowledge engineer's insight. The relaxation strategy proposed here is rather ad hoc, but it demonstrates the basic idea.

To continue the example of the camcorder, suppose we remove the following domain characteristics: "complete expected value database" and "threshold info" (implying that the lookup and the threshold compare method will not be applicable anymore). If we now ask TINA to generate the applicable inference structures, it responds with "no inference structures applicable in this situation." TINA can start the following dialogue with the knowledge engineer to relax some constraints:

```
How many methods of each inference structure can have unfulfilled criteria?
(type a number or "don't care.")
|: 2.

What criteria type might be relaxed?
1  epistemological
2  assumption
3  environmental
4  don't care

Select option : 1.
```

```
How many failed epistemological criteria do you allow for each method?
Note that the other criteria types should be without problems!
(type a number or "don't care.")
|: 1.
```

The first question ("How many methods ...") allows the knowledge engineer to specify that some methods that compose the inference structure may have unsatisfied criteria (in this case two). The next question asks which type of criterion might be relaxed. The reason behind this is that in some domains not all domain characteristics are unchangeable. For instance, some assumptions might be relaxed, or domain knowledge that is not directly available could, with additional effort, be supplied. They are soft characteristics. Other characteristics might be inherent to the domain and can not be changed (often environmental characteristics). For example, if a device can not be unfastened and, therefore, is not accessible for measurements, there is not much that can be done about it. It is a hard characteristic. TINA lets you choose from three options that reflect the three different types of criteria. The not chosen ones are thus considered as hard characteristics. In the example, we chose to relax some epistemological requirements. The next question asks how many of them can be relaxed within *each* method (one in the example).

TINA generates under this relaxation several applicable inference structures. The output consists of two parts: the inference structure itself (method tree) and the methods whose criteria are not all satisfied. Among the inference structures suggested are (of course) the six inference structures generated in experiment 1. For example, TINA generates the method tree shown in Figure 9 along with the following epistemological requirements not yet satisfied:

PSM	Requirement
compare symptom detection method	expected values obtainable
threshold compare method	threshold info

This output means that the inference structure of Figure 9 is applicable if extra domain knowledge ("expected values obtainable" and "threshold info") can be provided. Expected values are needed to compare observations with their expectations. Note that the requirement reported resides in "expected values obtainable," while we retracted the domain characteristic "complete expected value database." However, the latter is a more specific version of the former (see Section 4.1).

A slightly different inference structure (from the one in Figure 9) suggests the classify symptom detection method instead of the compare method. But then its epistemological requirement has to be taken care of. It is up to the knowledge engineer to decide whether this knowledge can be supplied or not.

PSM	Requirement
classify method	knowledge for classifying symptoms

The first two experiments take as starting points a characterization of a specific domain and suggest applicable inference structures. The next two experiments deal with the opposite situation: start with an inference structure, analyze it, and generate requirements.

Experiment 3 — Provide an inference structure and obtain the corresponding requirements. Another knowledge acquisition situation could be that there exists an idea of what inference structure to use to solve a diagnostic problem. TINA can then be used to generate its requirements. It is up to the knowledge engineer to see whether they can be met. The given inference structure has to be a legal one according to the grammar.

Using the methods, tasks, and inferences from the library, we have modeled several strategies described in the literature (Benjamins, 1993), such as DART, GDE, GDE+ (Struss & Dressler, 1989), CHECK, and FAULTY. As an example of a diagnostic inference structure, we input a combination of GDE/GDE+ and show the requirements generated. The corresponding method tree is shown in Figure 10.

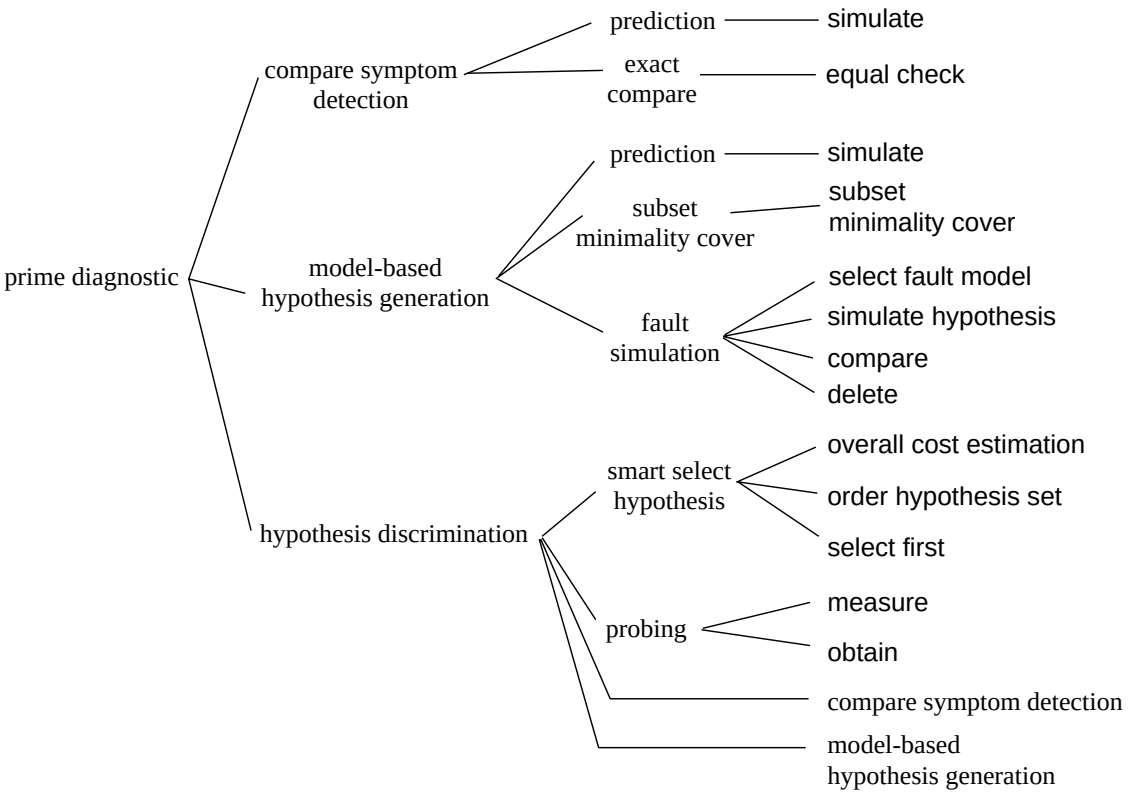

Figure 10: The method tree corresponding to GDE/GDE+. The leaves are the inferences (in Helvetica).

To detect symptoms, the compare symptom detection method is used, which is further broken down into the prediction method and the exact compare method. To generate hypotheses, GDE uses the model-based hypothesis generation method consisting of the prediction method for finding contributors, the subset minimality cover method for transforming the contributor sets to a hypothesis set, and the fault simulation method for filtering the hypothesis set (GDE+). To realize the hypothesis discrimination task, GDE applies the following methods. It selects a hypothesis to test based on an estimation of the overall cost (number of tests and failure rates). The probing method is used to collect additional data about the hypothesis. To interpret the data, the model-based hypothesis generation method is used again. This reuse means that the collected data are processed

in the same manner as the initial data.

TINA generates the requirements for the inference structure as shown in Table 3. "Ep" stands for epistemological, "As" for assumption, and "En" for environmental requirements. We will quickly explain some of the requirements. The PDM assumes that hypotheses can

PSM	Requirement	Type
prime diagnostic method	hypotheses can be generated additional observations	As En
compare symptom detection method	expected values obtainable	Ep
prediction method	simulation rules	Ep
exact compare method	precise values	En
model-based hypothesis generation method	device model non-intermittency assumption	Ep As
prediction method	simulation rules	Ep
subset minimality transform method	fault behavior not constrained independence of causes	Ep As
fault simulation method	fault simulation rules complete fault model	Ep As
hypothesis discrimination method	non-intermittency assumption	As
smart select hypothesis method	cost info	Ep
estimation based on overall cost method	hypotheses related cost info	Ep Ep
probing method	device accessible measuring tools	En En
model-based hypothesis generation method	device model non-intermittency assumption	Ep As

Table 3: Requirements for the PSMs of the inference structure of GDE/GDE+.

be generated, or, more precisely, there should be no hypothesis that would be considered as a possible cause if it can not be generated by the hypothesis generator. An environmental requirement is that additional observations must be obtainable. In order to select a next measurement, GDE minimizes the overall cost of the tests to perform. This minimization requires that the hypotheses in the set be related to each other and that cost information be available.

Experiment 4 — Provide a partial inference structure and obtain its possible completions along with the requirements. This experiment is an extension of the previous one, but here the knowledge engineer has only an idea about the parts of the inference structure to solve the problem. Instead of providing TINA with a complete structure, a partial one is given, partial in the sense that some inferences are left unspecified. The restriction for a method to be applicable (apart from its suitability criteria) is that the number of inferences that constitute the possible completing method(s) must be equal to the number of unspecified inferences. In the partial (simple) "inference structure" shown below there are three unspecified inferences. If we feed this partial structure to TINA, it suggests to complete it with the compiled method for generating hypotheses.

```
user_judgment
UNSPECIFIED_INFERENCE1
UNSPECIFIED_INFERENCE2
```

```
UNSPECIFIED_INFERENCE3
select_random
compiled_test
user_judgment
delete
```

The output of TINA is given in Figure 11 (the completions are in Courier font) and in Table 4.

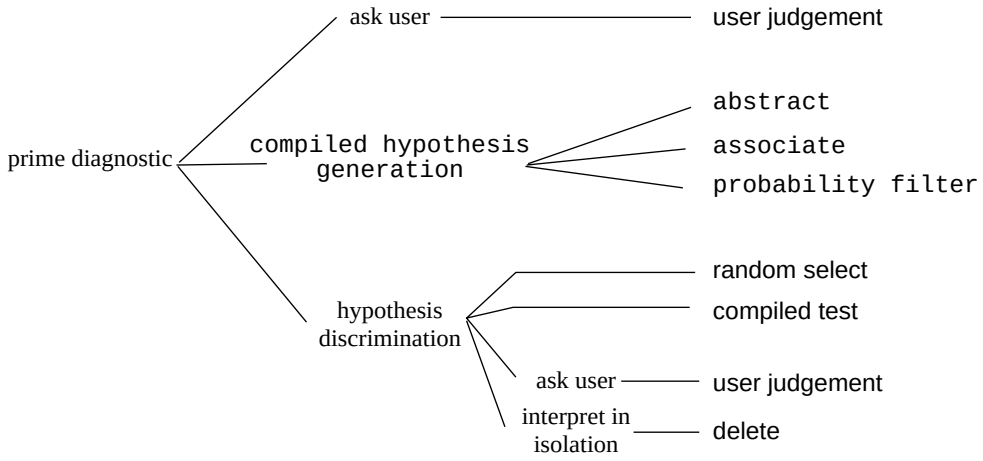

Figure 11: The method tree corresponding to the completed inference structure. The leaves are inferences (in Helvetica) and completions are in Courier font.

PSM	Requirement	Type
prime diagnostic method	hypotheses can be generated	As
	additional observations	En
ask user method	user knowledgeable symptoms	En
compiled hypothesis generation method	empirical associations	Ep
	probability information	Ep
	abstraction knowledge	Ep
	complete association set	As
hypothesis discrimination method	non-intermittency assumption	As
random select hypothesis method	hypotheses unrelated	Ep
compiled test method	tests associated to hypotheses	Ep
interpret in isolation method	hypotheses unrelated	Ep

Table 4: Requirements for the completed inference structure generated by TINA.

TINA thus finds a legal completion of the partial inference structure along with the requirements to be met (Table 4). This experiment deals with a very simple structure. The user is asked to decide on the (ab)normality of the initial observations. Observations may be abstracted and associated to hypotheses (compiled method), possibly followed by a probability filter. This yields a hypothesis set which is exhaustively tested by the compiled test method, in which a test procedure is explicitly associated to each hypothesis. Test results are interpreted in isolation, that is, not considering the impact of the result

on other hypotheses. This makes sense if the hypotheses in the set are unrelated to one another.

Depending on the number of unspecified inferences, TINA generates many or few inference structures. In this experiment, three of eight inferences were unspecified. In another experiment, we supplied TINA with a more complex partial inference structure consisting of sixteen inferences, leaving five of them unspecified. Eight possible completions were generated.

5 Discussion

With the four experiments, we have demonstrated that the library is useful in various ways for constructing and analyzing diagnostic inference structures. Of course, the "goodness" of an inference structure depends on the quality of the library components, in particular on the methods and the criteria. We do not claim that the library presented here is complete. For instance, case-based and Bayesian approaches to diagnosis are not considered. However, we do claim that the library covers many strategies used in the mainstream approaches to diagnosing faults in technical devices and some strategies in the medical domain (Benjamins & Jansweijer, 1994). With respect to the completeness of the suitability criteria set, we have to be more cautious. This is a rather new area and more research is needed.

Generation versus analysis of inference structures. The power of the current library lies in (1) generating inference structures in restricted domains, (2) analyzing inference structures and generating their domain requirements, and (3) completing partial inference structures. It does not lie in the generation of all inference structures that one could think of. For experimenting with the generation of inference structures, the library would need to include knowledge about *global* constraints and relations between the methods, for example, constraints that some method can only be used if another method is also used (mutual dependency (Breuker & van de Velde, 1994)) or that two methods are mutually exclusive. Such constraints limit the number of possible combinations of methods and thus the number of inference structures. An example of a global relation is that if the prediction method is used to generate expectations as part of the symptom detection task, then the find contributors task should also use the prediction method. Once global relations are included, less inference structures will be generated; it would then be interesting to analyze these structures in detail.

Underlying assumption. If in a particular inference structure different methods use different domain models, we assume that the different domain models are compatible with each other. For example, to combine a method that uses a causal model with a method that reasons with a component model requires that these models "know" about each other's content. Often, this is not an optimal situation because once a certain domain model is constructed (which can be quite a difficult task), preferably this model is used in as many methods as possible. This problem could be dealt with by adding a general global constraint which states that methods should, if possible, use the same domain model. However, in some diagnostic domains, the use of multiple domain models might be a necessary feature, for example, when reasoning in one model leads easily to an impasse

(Abu-Hanna, 1994, Abu-Hanna & Jansweijer, 1994)).

Top-down versus bottom-up acquisition. Using the library for inference structure analysis and construction combines top-down and bottom-up acquisition. Top-down acquisition analyzes an inference structure and generates its requirements, while bottom-up works from domain characteristics to inference structures. As the experiments show, there are variations possible, moving the emphasis towards bottom-up (experiments 1 and 2) or towards top-down (experiments 3 and 4).

Generalizing from the experimental results. What do the experimental results mean in general for constructing diagnostic inference structures with a library? Let L be the library. C_D denotes the set of characteristics (in terms of the suitability criteria) that defines an application domain D. The set IS_D represents all inference structures that are applicable in D. We could write this as $\mathrm{L}(C_D) = IS_D$; thus, the applicable inference structures are a function of the domain characteristics, where the function is defined by the library. If we could define typical problem domains D in terms of their characteristics C, then we could generate the sets of applicable inference structures IS_D in D using the library L. This would be a valuable extension because *clusters* of inference structures can be represented for *types* of domains. We have made a first attempt to define some typical domains including empirical, medical, and simple and complex technical domains (thus: $C_{empirical}$, $C_{technical}$, etc.). L then defines the generative relation between C_D and IS_D, thus, for example, $\mathrm{L}(C_{empirical}) = IS_{empirical}$. The quality of such relations depends on the completeness and correctness of the library and the suitability criteria. Since these can not yet be guaranteed, it is too early to make such general conclusions. However, it indicates an interesting direction for further work.

Possible extensions of TINA. TINA demonstrates some possibilities of the library represented in the grammar for knowledge acquisition. Extensions are, however, possible. If one is willing to represent some meta-knowledge about primitive inferences or tasks in the library, interesting features could be realized. For instance, if we include knowledge about which inferences interact with the user (e.g. so-called "transfer tasks" in KADS (Schreiber *et al.*, 1993)), then we could generate inference structures that minimize user interaction. In diagnostic applications, this distinction is important because getting information (i.e. providing the diagnostic system with more observations) can be a cost determining factor.

6 Related work

A significant part of the work presented here, concerns the selection of PSMs from the library. Previous work on PROTÉGÉ-II (Puerta *et al.*, 1992) focussed more on constructing PSMs from mechanisms, as is the case in DIDS, rather than on selecting them from a library. More recent work on PROTÉGÉ-II (Gennari *et al.*, 1994) also aims at selecting PSMs. A so-called *method ontology* specifies the data requirements of a method, which can be used to index the methods. The method ontology relates to our epistemological criteria.

In the CommonKADS library (Breuker & van de Velde, 1994), the reusable components are associated with "component features" expressed as questions (e.g., "Is there a

hierarchical model?"). There are different types of component features defining the dependencies between the library components (e.g., mutual dependent, mutual exclusive), which relate to our suitability criteria and the global constraints and relations between PSMs that we hinted at in Section 5. However, no substantial work has been performed to automate PSM selection from the library.

In KRESTT (Steels, 1993, Goossens, 1995), "feature structures" are used to enable the selection of methods from the library. The input and output roles of the method function as constraints on the method's applicability, and are similar to our epistemological criteria.

Instead of contrasting the method-selection process described here with that of the main approaches to Knowledge Acquisition, we rather prefer to emphasize its complementary aspect. All approaches aim at method selection, but none of them has presented yet a complex application. One of the reasons for this could be that method *requirements* are a relatively new issue. Since our modeling framework builds on a convergence of current approaches, it should not be too difficult to incorporate the work presented here in the respective approaches (although some terminological mapping has to be done). For example, in (Orsvärn, 1995), CommonKADS is used to select methods from the library presented here.

Considering tasks and subtasks in terms of a grammar is not a new idea. It is applied in KEW (van Heijst *et al.*, 1992) among others, in which the construction of an inference structure is depicted as a gradual refinement process in which the knowledge engineer interactively decides which rewrite rule to apply next, thereby considering the conditions of each rewrite rule (and possibly acquiring the domain knowledge). The result of a KEW session is one particular inference structure that satisfies the constraints at hand. We have implemented our library in the KEW formalism (Benjamins, 1993), and it supports the construction of diagnostic inference structures. The difference with TINA is that TINA can work in two directions: generation and analysis. Moreover, it generates not one inference structure, but all of them. It is then up to the knowledge engineer to select the most suitable one.

7 Conclusions

In the first part of this paper, we presented an in-depth analysis of the diagnosis task. This analysis provides a better insight into diagnostic problem solving and existing diagnostic systems in terms of their comprising tasks, problem-solving methods, and inferences. The analysis is performed using an informal, but structured, modeling framework. A formal account of part of the work presented here can be found in (Aben, 1995). In the second part of the paper, we demonstrated how the result of our analysis can be viewed as a library with reusable diagnostic components. We have shown some ways in which the library can be used in Knowledge Acquisition by exploiting the so-called suitability criteria of methods. In particular, we have demonstrated the following possibilities: (1) Given a particular domain, generate the applicable inference structures. (2) In case no applicable inference structures can be found, allow relaxation of some suitability criteria and generate the legal inference structures along with their unfulfilled criteria. (3) Given a particular inference structure, obtain the corresponding requirements. (4) Given a partial inference structure, obtain its possible completions along with the requirements. The quality of the inference structures and of the requirements generated depends on the correctness and completeness of the methods and the suitability criteria. These can not yet be guaranteed,

but strategies described in the mainstream approaches to technical diagnosis were covered. More research has to be conducted to assure reasonable completeness of the suitability criteria.

Another line of further work should investigate global relations between methods. The criteria we work with reflect only local features of methods. We have briefly discussed the importance of such global relations for generating inference structures.

Acknowledgment

Ameen Abu-Hanna, Wouter Jansweijer and Leliane Nunes de Barros are acknowledged for their contribution. Tina Patón is thanked for proofreading the paper. Richard Benjamins works with an institutional research fellowship of the Human Capital & Mobility program, financed by the Commission of the European Communities.

References

Aamodt, A., Benus, B., Duursma, C., Tomlinson, C., Schrooten, R., & Van de Velde, W. (1992). Task features and their use in CommonKADS. Technical Report KADS-II/T1.5/VUB/TR/014/1.0, Free University of Brussels & University of Amsterdam & Lloyd's Register.

Aben, M. (1995). *Formal Methods in Knowledge Engineering.* PhD thesis, University of Amsterdam, Amsterdam.

Abu-Hanna, A. (1994). *Multiple domain models in diagnostic reasoning.* PhD thesis, University of Amsterdam, Amsterdam.

Abu-Hanna, A., Benjamins, V. R., & Jansweijer, W. N. H. (1991). Device understanding and modeling for diagnosis. *IEEE-Expert*, 6(2):26–32.

Abu-Hanna, A. & Jansweijer, W. N. H. (1994). Modeling domain knowledge using explicit conceptualization. *IEEE-Expert*, 9(5):54–64.

Angele, J., Fensel, D., Landes, D., Neubert, S., & Studer, R. (1993). Model-based and incremental knowledge engineering: the MIKE approach. In J. Cuena (Ed.), *Knowledge Oriented Software Design, IFIP Transactions A-27*, Amsterdam: Elsevier.

Benjamins, V. R. (1994). On a role of problem solving methods in knowledge acquisition: experiments with diagnostic strategies. In L. Steels, A. T. Schreiber, & W. van de Velde (Eds.), *Lecture Notes in Artificial Intelligence, 867, 8th European Knowledge Acquisition Workshop, EKAW-94*, pp. 137–157, Berlin, Germany: Springer-Verlag.

Benjamins, V. R. (1993). *Problem Solving Methods for Diagnosis.* PhD thesis, University of Amsterdam, Amsterdam, The Netherlands.

Benjamins, V. R. & Abu-Hanna, A. (1990). FAULTY: A shell for diagnosing complex technical systems. Technical Report SKBS/A2/90-1, SWI, University of Amsterdam, Amsterdam.

Benjamins, V. R. & Jansweijer, W. N. H. (1994). Toward a competence theory of diagnosis. *IEEE-Expert*, 9(5):43–52.

Benjamins, V. R. & Jansweijer, W. N. H. (1989). Models in diagnostic reasoning (Dutch). In L. Siklossy & R. Bakker (Eds.), *Proc. of the Dutch Conference on Artificial Intelligence*, pp. 87–94, Enschede.

Breuker, J. & van de Velde, W. (Eds.), (1994). *CommonKADS Library for Expertise Modeling.* Amsterdam, The Netherlands: IOS Press.

Breuker, J. A., Wielinga, B. J., van Someren, M., de Hoog, R., Schreiber, A. T., de Greef, P., Bredeweg, B., Wielemaker, J., Billault, J. P., Davoodi, M., &

HAYWARD, S. A. (1987). Model Driven Knowledge Acquisition: Interpretation Models. ESPRIT Project P1098 Deliverable D1 (task A1), University of Amsterdam and STL Ltd.

CHANDRASEKARAN, B. (1990). Design problem solving: A task analysis. *AI Magazine*, 11:59–71.

CHANDRASEKARAN, B. (1986). Generic tasks in knowledge based reasoning: High level building blocks for expert system design. *IEEE Expert*, 1(3):23–30.

CHANDRASEKARAN, B. & JOHNSON, T. R. (1993). Generic tasks and task structures: History, critique and new directions. In J. M. David, J. P. Krivine & R. Simmons (Eds.), *Second Generation Expert Systems*. Springer Verlag.

CHANDRASEKARAN, B., JOHNSON, T. R., & SMITH, J. W. (1992). Task-structure analysis for knowledge modeling. *Communications of the ACM*, 35(9):124–137.

CHITTARO, L., COSTANTINI, C., GUIDA, G., TASSO, C., & TOPPANO, E. (1989). Diagnosis based on cooperation of multiple knowledge sources. *Proc. of the specialized conference on second generation expert systems Avignon 1989*, pp. 19 – 33.

CLANCEY, W. J. (1993). Acquiring, representing, and evaluating a competence model of diagnostic strategy. In B. G. Buchanan & D. C. Wilkins (Eds.), *Readings in Knowledge Acquisition and Learning: Automating the Construction and Improvement of Expert Systems*, pp. 178–215. San Mateo, CA: Morgan Kaufmann.

CLANCEY, W. J. (1985). Heuristic classification. *Artificial Intelligence*, 27:289–350.

CONSOLE, L., DUPRÉ, D. T., & TORASSO, P. (1989). A theory of diagnosis for incomplete causal models. In *Proc 11th IJCAI*, pp. 1311–1317, Detroit.

CONSOLE, L. & TORASSO, P. (1990a). Hypothetical reasoning in causal models. *Int. J. of Intelligent Systems*, 5(1):83–124.

CONSOLE, L. & TORASSO, P. (1990b). Integrating models of the correct behaviour into abductive diagnosis. In L. C. Aiello (Ed.), *Proc. ECAI–90*, pp. 160–166, London: ECCAI, Pitman.

DAVIS, R. (1984). Diagnostic reasoning based on structure and behavior. *Artificial Intelligence*, 24:347–410.

DAVIS, R. & HAMSCHER, W. C. (1988). Model-based reasoning: Troubleshooting. In H. E. Shrobe (Ed.), *Exploring Artificial Intelligence*, pp. 297–346. San Mateo, CA: Morgan Kaufmann.

DE KLEER, J. & WILLIAMS, B. (1989). Diagnosis with behavioral modes. In *Proc 11th IJCAI*, pp. 1324–1330, Detroit.

DE KLEER, J. & WILLIAMS, B. (1987). Diagnosing multiple faults. *Artificial Intelligence*, 32:97–130.

ESHELMAN, L. (1988). MOLE: A knowledge-acquisition tool for cover-and-differentiate systems. In S. Marcus (Ed.), *Automating Knowledge Acquisition for Expert Systems*, pp. 37–80. Boston: Kluwer.

GENESERETH, M. R. (1984). The use of design descriptions in automated diagnosis. *Artificial Intelligence*, 24:411–436.

GENNARI, J., TU, S., ROTENFLUH, T., & MUSEN, M. (1994). Mapping domains to methods in support of reuse. *International Journal of Human-Computer Studies*, 41:399–424.

GOOSSENS, L. (1995). From ComMet to KresT. In B. R. Gaines & M. A. Musen (Eds.), *Proceedings of the 8th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop*, pp. 41.1–41.20, Banff, Canada: SRDG Publications, University of Calgary.

HAMSCHER, W. C. (1991). Principles of diagnosis: Current trends and a report on the first international workshop. *AI-magazine*, pp. 15–37.

HAMSCHER, W. C. (1988). *Model-Based Troubleshouting of Digital Systems*. PhD thesis, MIT, Cambridge, MA.

JOSEPHSON, J. & JOSEPHSON, S. (Eds.), (1994). *Abductive Inference, Computation, Philosophy, Technology*. Cambridge: Cambridge University Press.

KLINKER, G., BHOLA, C., DALLEMAGNE, G., MARQUES, D., & MCDERMOTT, J. (1991). Usable and reusable programming constructs. *Knowledge Acquisition*, 3:117–136.

MCDERMOTT, J. (1988). Preliminary steps towards a taxonomy of problem-solving methods. In S. Marcus (Ed.), *Automating Knowledge Acquisition for Expert Systems*, pp. 225–255. Boston:

Kluwer.

Miller, R. A., Pople, H. E., & Myers, J. D. (1982). INTERNIST-I: A general computer-based diagnostic consultant for general internal medicine. *New England Journal of Medicine*, 307:468–476. [Also in: *Readings in Medical Artificial Intelligence: The First Decade*, W. J. Clancey and E. H. Shortliffe (Eds.), Reading, MA: Addison Wesley, 1984].

Newell, A. (1982). The knowledge level. *Artificial Intelligence*, 18:87–127.

Orsvärn, K. (1995). Case-study with Benjamins' library of diagnosis methods. In U. Gappa & H. Voss (Eds.), *Proc. of The Knowledge Engineering Forum 95*, pp. 83–94, Sankt Augustin, Germany: GMD.

Orsvärn, K. (1994). Towards problem solving methods for sequential diagnosis. Technical Report KADS-II/M2.3/TR/SICS/001/1.0, SICS.

Pierret-Golbreich, C. (1994). TASK model: a framework for the design of models of expertise and their operationalization. In B. R. Gaines & M. A. Musen (Eds.), *Proceedings of the 8th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop*, pp. 37.1-37.22, Banff, Canada: SRDG Publications, University of Calgary.

Pople, H. (1982). Heuristic methods for imposing structure on ill-structured problems: The structuring in medical diagnosis. In P. Szolovits, (Ed.), *Artificial Intelligence in Medicine*, pp. 119–190, Boulder CO: Westview Press.

Puerta, A., Egar, J., Tu, S., & Musen, M. (1992). A multiple-method shell for the automatic generation of knowledge acquisition tools. *Knowledge Acquisition*, 4:171–196.

Punch, W. (1989). *A Diagnosis System Using a Task Integrated Problem Solver Architecture (TIPS), Including Causal Reasoning.* PhD thesis, The Ohio State University, Ohio.

Raiman, O. (1989). Diagnosis as a trial: The alibi principle. In *Proc. of Model Based Diagnosis International Workshop*, pp. III-1–III-10, Paris: IBM France, Scientific Center.

Raiman, O. (1986). Order of magnitude reasoning. In *Proc. of the AAAI*, pp. 100–104, San Mateo, CA: Morgan Kaufmann.

Reiter, R. (1987). A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–96.

Runkel, J. T. & Birmingham, W. P. (1993). Knowledge acquisition in the small: building knowledge-acquisition tools from pieces. *Knowledge Acquisition*, 5:221–243.

Schreiber, A. T., Wielinga, B. J., & Breuker, J. A. (Eds.) (1993). *KADS: A Principled Approach to Knowledge-Based System Development*, volume 11 of *Knowledge-Based Systems Book Series*. London, Academic Press.

Steels, L. (1993). The componential framework and its role in reusability. In J. M. David, J. P. Krivine & R. Simmons (Eds.), *Second Generation Expert Systems*, pp. 273–298. Berlin, Heidelberg, Germany: Springer-Verlag.

Steels, L. (1990). Components of expertise. *AI Magazine*, 11(2):28–49.

Steels, L. (1989). Diagnosis with a functional-fault model. *Applied Artificial Intelligence*, 3(2-3):129–153.

Sticklen, J., Chandrasekaran, B., & Bond, W. (1989). Applying a functional approach for model based reasoning. In *Proc. of (IJCAI) Workshop on Model Based Reasoning*, pp. 165–176, Detroit.

Struss, P. (1988). Extentions to ATMS-based diagnosis. In J. Gero (Ed.), *Artificial Intelligence in Engineering: Diagnosis and Learning.* Southampton: Elsevier Science.

Struss, P. & Dressler, O. (1989). Physical negation – integrating fault models into the general diagnostic engine. In *Proc 11th IJCAI*, pp. 1318–1323, Detroit.

van Heijst, G., Terpstra, P., Wielinga, B. J., & Shadbolt, N. (1992). Using generalised directive models in knowledge acquisition. In T. Wetter, K. D. Althoff, J. Boose, B. Gaines, M. Linster & F. Schmalhofer (Eds.), *Current Developments in Knowledge Acquisition: EKAW-92*, Berlin, Germany: Springer-Verlag.

Vanwelkenhuysen, J. & Rademakers, P. (1990). Mapping knowledge-level analysis onto a computational framework. In L. Aiello (Ed.), *Proc. ECAI–90*, pp. 681–686, London: Pitman.

Wielinga, B. J., Schreiber, A. T., & Breuker, J. A. (1992a). KADS: A modelling approach to knowledge engineering. *Knowledge Acquisition*, 4(1):5–53. Special issue 'The KADS ap-

proach to knowledge engineering'. [Reprinted in: B. G. Buchanan & D. C. Wilkins (1992), *Readings in Knowledge Acquisition and Learning*, pp. 92-116, San Mateo, CA: Morgan Kaufmann].

Wielinga, B. J., Van de Velde, W., Schreiber, A. T., & Akkermans, J. M. (1992b). The CommonKADS framework for knowledge modelling. In B. R. Gaines, M. A. Musen & J. H. Boose (Eds.), *Proc. 7th Banff Knowledge Acquisition Workshop*, volume 2, pp. 31.1–31.29, Banff, Canada: SRDG Publications, University of Calgary.