The Role of Assumptions in Knowledge Engineering

Dieter Fensel[1] and V. Richard Benjamins[2]

[1] University of Karlsruhe, Institute AIFB, 76128 Karlsruhe, Germany, dieter.fensel@aifb.uni-karlsruhe.de

[2] Artificial Intelligence Research Institute (IIIA), Spanish Council for Scientific Research (CSIC), Campus UAB, 08193 Bellaterra, Barcelona, Spain, richard@iiia.csic.es, http://www.iiia.csic.es/~richard
&
Dept. of Social Science Informatics (SWI), University of Amsterdam, Roetersstraat 15, 1018 WB Amsterdam, The Netherlands, richard@swi.psy.uva.nl, http://www.swi.psy.uva.nl/usr/richard/home.html

Abstract. Problem-solving methods are means to describe the inference process of knowledge-based systems. During the last years, a number of these problem-solving methods have been identified that can be reused for building new systems. However, problem-solving methods require specific types of domain knowledge and introduce specific restrictions on the tasks that can be solved by them. These requirements and restrictions are *assumptions* that play a key role in reusing problem-solving methods, in acquiring domain knowledge, and in defining the problem that can be tackled by the knowledge-based systems. In the paper, we discuss the different roles, assumptions play in the development process of knowledge-based systems and provide a survey of assumptions used by diagnostic problem solving. We show how such assumptions introduce target and bias for goal-driven machine learning and knowledge discovery techniques.

1 INTRODUCTION

During the last years, Problem-solving methods (PSMs) have become quite successful in describing the reasoning behavior of knowledge-based systems ([Chandrasekaran, 1986], [Marcus, 1988], [Puppe, 1993], [Schreiber et al., 1993], [Schreiber et al., 1994], [Eriksson et al., 1995], [Steels, 1990], [Terpstra et al., 1993], [Angele et al., 1996]). On the one hand, PSMs refine generic inference strategies and search methods to task and domain-specific circumstances. On the other hand, they are not designed for one specific application problem. Instead, they are usable for a family of similar problems: Similar in terms of the goals that should be achieved and similar in the type of knowledge that is required as resource for the reasoning process. Libraries of PSMs are described in [Benjamins, 1995], [Breuker & Van de Velde, 1994], [Chandrasekaran et al., 1992], [Motta & Zdrahal, 1996], and [Puppe, 1993].

One of the first problem-solving methods for knowledge-based systems (KBSs) is *heuristic classification* (see Figure 1). [Clancey, 1985] identified it as a generic reasoning pattern of several expert systems applied to different problems. It consists of three main inference steps:

- a *data abstraction* step that abstracts concrete values like “body-temperature = 39.2 degree Celsius“ to the value “high fever“;

- a *heuristic match* step that uses these abstract descriptions to heuristically establish some possible solution classes.
- a *refinement* step that should find a final solution by discrimination.

Each of the inference step requires specific knowledge types as resource. A data abstraction step can only be performed if hierarchical knowledge over data is available, and a refinement step of solutions can only be done if hierarchical knowledge over solution classes is available.

Describing PSMs by their inference steps, knowledge types, and inference structures that determine the data and knowledge flow between the inferences, has become a common style in knowledge engineering. Examples for diagnosis are provided by [Benjamins, 1995] and for planning by [Barros et al., 1997]. Basically, these descriptions decompose the entire inference process into more elementary sub steps.

[Van de Velde, 1988] and [Akkermans et al., 1993] proposed the description of the *competence* of a PSM in extension to their decompositional descriptions. Such competence descriptions define the goals that can be achieved by a PSM independent from *how* these goals are achieved. Thus, such competence descriptions resemble the idea of functional specifications from software engineering for PSMs. A functional specification of a software artefact describes *what* the software system does without referring to the way how it achieves its functionality [Fensel, 1995c]. Examples of such competence descriptions can be found in [Fensel & Groenboom, 1997], [Fensel & Schönegge, 1997b], and [Fensel & Schönegge, submitted]

However, establishing the competence of a PSM requires the definition of a control flow that defines the execution order of the inference steps of a PSM [Fensel et al., to appear] and a notion of the functionality that is provided by the domain knowledge. The competence definition of a PSM like heuristic classification critically depends on the “competence“ of its hierarchical and heuristic match knowledge. Statements about the absolute or relative correctness and completeness of the method can only be done in terms of assumptions over the absolute or relative correctness and completeness of the domain knowledge. These assumptions are therefore more precise characterizations of the knowledge types and competence of a PSM. In consequence, current work on PSM pays much more attention to

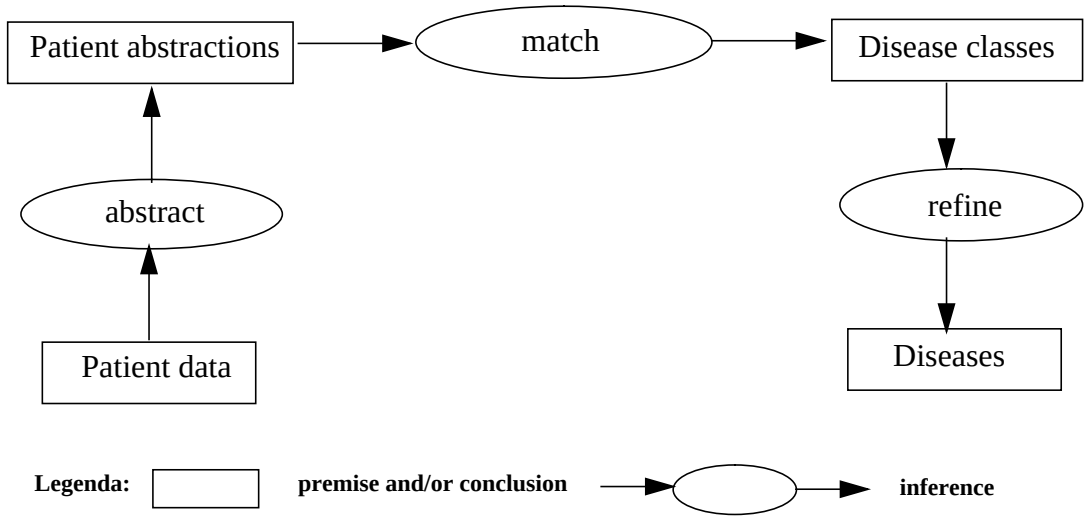

Fig. 1 Heuristic Classification [Clancey, 1985].

these assumptions ([Benjamins & Aben, 1997], [Fensel, 1995a], [Fensel et al., 1996], [Wielinga et al., 1995], [Benjamins & Pierret-Golbreich, 1996], [Benjamins et al., 1996], [Fensel & Benjamins, 1996], [Fensel & Straatman, to appear], [O´Hara & Shadbolt, 1996], [Motta & Zdrahal, 1996], [Breuker, 1997], [Fensel & Schönegge, submitted]).

In this paper, we will take a closer look at assumptions of PSMs. In Section 2, we introduce a general framework for specifying KBSs at a conceptual level that takes into account the important role of assumptions. Also, we sketch the twofold role assumptions can play and express the relationship between these two roles as the law of conservation of assumptions (cf. [Benjamins et al., 1996]). In Section 3, we provide an extensive survey on assumptions used in diagnostic problem solving. This survey provides the empirical base for our argument and delivers numerous illustrations for our point. In Section 4, we describe the role assumptions play in knowledge acquisition. We describe methods for assumption verification, assumption identification, and knowledge acquisition guided by assumptions and discuss the role that existing verification, machine learning, and knowledge discovery techniques can play in these processes. Finally, we provide conclusions and future work.

2 THE LAW OF CONSERVATION OF ASSUMPTIONS

Mostly, papers on problem-solving methods focus on the description of reasoning strategies and discuss their underlying assumptions as a side aspect. We take a complementary point of view and focus on these underlying assumptions as they play important roles:

- Assumptions are necessary to characterise the precise competence of a problem-solving method in terms of the tasks that can be solved by it, and in terms of the domain knowledge that is required by it.
- Assumptions are necessary to enable tractable problem solving and economic system development of complex problems. First, assumptions reduce the worst-case or average-case complexity of computation [Fensel & Straatman, to appear]. Second, assumptions may reduce the costs of the system development process through simplifying the problem that must be solved by the system [Fensel, 1997b].
- Finally, assumptions have to be made to ensure a proper interaction of the problem solver with its environment.

In the following, we will first discuss the different elements of a description of a KBS and second we will sketch their proper relationships and the process of deriving them.

2.1 The Four Elements in Specifying KBSs

In [Fensel & Groenboom, 1997], we provided different aspects of a specification of knowledge-based system which are related by assumptions (see Figure 2): a *task definition* defines the problem to be solved by the KBS; the *PSM* defines the reasoning process of the knowledge-based system; and a *domain model* describes the domain knowledge of the knowledge-based system. Each of these three elements is described independently to enable the reuse of task descriptions in different domains, the reuse of PSMs for different tasks and domains, and the reuse of domain knowledge for different tasks and PSMs. Therefore, a fourth element of a specification of a KBS is an *adapter* that is necessary to adjust the three

other (reusable) parts to each other and to the specific application problem.

The *task definition* specifies the goals that should be achieved in order to solve a given problem, which are functionally specified as an input-output relation. A task definition also defines assumptions about the domain knowledge. Already such a simple task like the selection of the maximal element of a set of elements requires a preference relation as domain knowledge. Assumptions are used to define the requirements on such a relation (e.g. transitivity, symmetry, etc.).

The reasoning of a knowledge-based system can be described by a *problem-solving method*. A PSM consists of three parts. First, a definition of the functionality defines the *competence* of the PSM independent of its realisation. Second, an *operational description* defines the dynamic reasoning process. Such an operational description describes how the competence can be achieved in terms of the reasoning steps and their dynamic interaction (i.e., the knowledge and control flow). The third part of a PSM concerns *assumptions* about the domain knowledge. Each inference step requires a specific type of domain knowledge with specific characteristics.

The description of the *domain model* introduces the domain knowledge as it is required by the PSM and the task definition. Three elements are needed to define a domain model. First, a description of properties of the domain knowledge at a meta-level. The *meta-knowledge* characterises properties of the domain knowledge. It is the counterpart of the assumptions on domain knowledge made by the other parts of a KBS specification: assumptions made about domain knowledge by these parts, must be stated as properties of the domain knowledge. The second element of a domain model concerns the *domain knowledge* and *case data* necessary to define the task in the given application domain, and necessary to carry out the inference steps of the chosen problem-solving method. The third element is formed by *external*

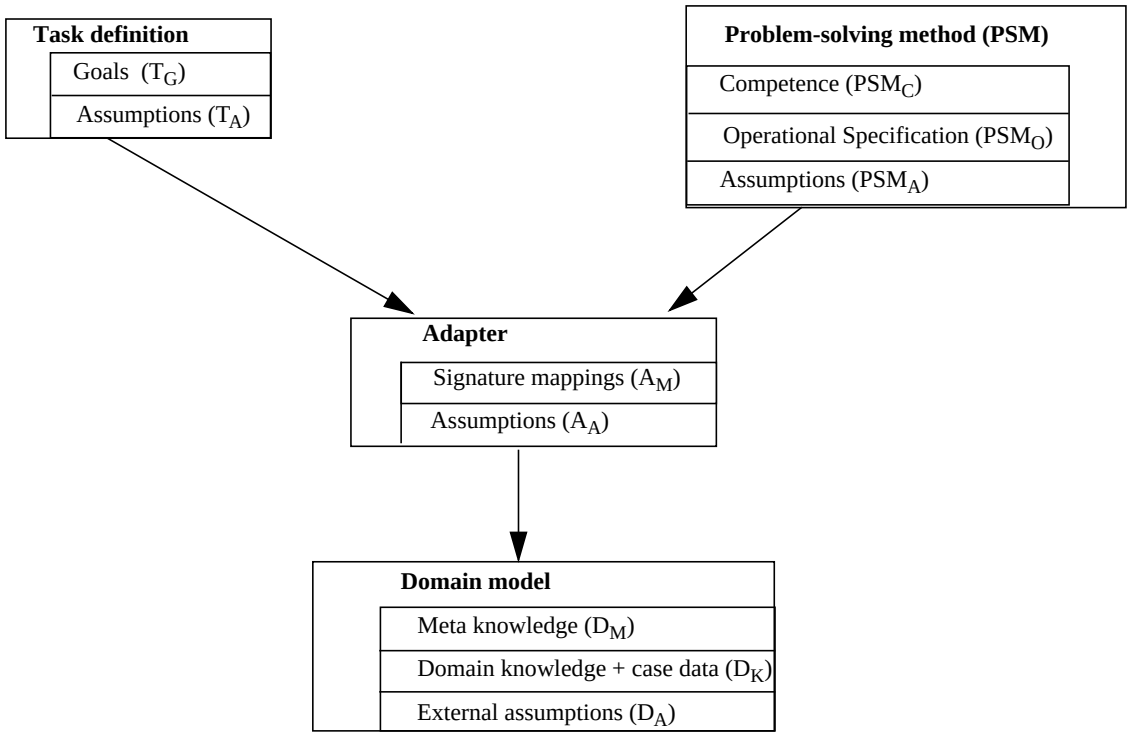

Fig. 2 The different elements of a specification of a knowledge-based systems.

assumptions that link the domain knowledge with the actual domain. These external assumptions capture the implicit and explicit assumptions a modeler made while building a domain model of the real world.

The description of an *adapter* maps the different terminologies of task definition, PSM, and domain model onto each other, collects the assumptions of task and PSM and may introduce further assumptions that have to be made to relate the competence of a PSM with the functionality as it is introduced by the task definition. Because it relates the three other parts of a specification together and establishes their relationship in a way that meets the specific application problem, they can be described independently and selected from libraries. The consistent combination and adaptation of the three different components to the specific aspects of the given application (because they should be reusable they need to abstract from specific aspects of application problems) must be provided by the adapter.

2.2 The Law of Conservation of Assumptions

When establishing the proper relationship between PSM and task, one usually requires *correctness* and *completeness* of the PSM relative to the goals of the task:

- *Correctness* requires that each output that is derived by the PSM also fulfils the goal of the task:
  $$\forall i,o\ (\mathrm{PSM_A}(i) \wedge \mathrm{PSM_C}(i,o) \rightarrow \mathrm{TASK_A}(i) \wedge \mathrm{TASK_G}(i,o))$$
  simplified:
  $$\forall i,o\ (\mathrm{PSM}(i,o) \rightarrow \mathrm{TASK}(i,o))$$
- *Completeness* requires that the PSM provides an output for each input that leads to a fulfilled goal of the task:
  $$\forall i\ (\mathrm{TASK_A}(i) \wedge \exists o_1\ \mathrm{TASK_G}(i,o_1) \rightarrow \mathrm{PSM_A}(i) \wedge \exists o_2\ \mathrm{PSM_C}(i,o_2))$$
  simplified:
  $$\forall i\ (\exists o_1\ \mathrm{TASK}(i,o_1) \rightarrow \exists o_2\ \mathrm{PSM}(i,o_2))$$
  It is not necessarily the same output because the task may not be a function (i.e., several output are possible).

However, a perfect match is unrealistic in many cases. In general, most problems tackled with KBSs are inherently complex and intractable (see e.g. [Bylander, 1991], [Bylander et al., 1991], and [Nebel, 1996]).[1] A PSM has to describe not just a realization of the functionality, but one which takes into account the constraints of the reasoning process and the complexity of the task. The way PSMs achieve efficient realization of functionality is by making *assumptions* [Fensel & Straatman, to appear]. These assumptions put restrictions on the context of the PSM, such as the domain knowledge and the possible inputs of the method or the precise definition of the goal to be achieved when applying the PSM. Notice that such assumptions can work in two directions to achieve this result. First, they can restrict the complexity of the problem, that is, weaken the task definition in such a way that the PSM competence is sufficient to realize the task. Second, they can strengthen the competence of the PSM by assuming (extra) domain knowledge.

- *Weakening*: Reducing the desired functionality of the system and reducing therefore the complexity of the problem by introducing assumptions about the precise task definition.

---

1. Exceptions are classification problems which have often known polynomial time complexity (see [Goel et al., 1987]).

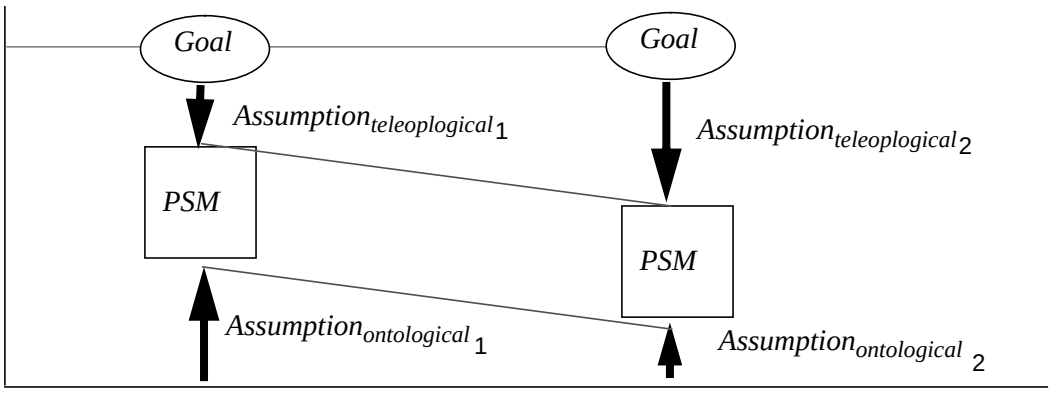

Fig. 3 The two effects of assumptions.

An example of this type of change is to no longer require an optimal solution, but only an acceptable one, or to make the single-fault assumption in model-based diagnosis.

- *Strengthening*: Introducing assumptions about the domain knowledge (or the user of the system) which reduces the functionality or the complexity of the part of the problem that is solved by the PSM. In terms of complexity analysis, the domain knowledge or the user of the system is used as an oracle that solves complex parts of the problem. These requirements therefore strengthen the functionality of the method.

Both strategies are complementary. Informally:

$$\text{TASK} - \text{Assumption}_{\text{weakening}} = \text{PSM} + \text{Assumption}_{\text{strengthening}}$$

That is, the sum of both types of assumptions may be constant. Decreasing the strength of one assumptions type can be compensated by increasing the strength of the other type (see Figure 3), i.e.

$$\text{TASK} - \text{PSM} = \Delta = \text{Assumption}_{\text{weakening}} + \text{Assumption}_{\text{strengthening}}$$

This is called the *law of conservation of assumptions* in [Benjamins et al., 1996]. More formally, both types of assumptions appear at different places in the implications that define the relationship between PSM and task:

- *Adapted Correctness*
  $$\forall i,o\ (\text{Assumption}_{\text{strengthening}} \wedge \text{PSM}(i,o) \rightarrow (\neg\text{Assumption}_{\text{weakening}} \vee \text{TASK}(i,o)))$$
- *Adapted Completeness*
  $$\forall i\ (\exists o_1\ \text{TASK}(i,o_1) \wedge \text{Assumption}_{\text{weakening}} \rightarrow (\neg\text{Assumption}_{\text{strengthening}} \vee \exists o_2\ \text{PSM}(i,o_2))$$

Recalling that an implication is true if the premise is false or if the premise and the conclusion are true, this twofold impact can be explained easily. Assumptions weaken the implication by either strengthening the premise or by weakening its conclusion.[2]

The first type of assumptions is used to weaken the goal which must be achieved by the PSM and the second type of assumption is used to improve the effect which can be achieved by the method by requiring external sources for its reasoning process. Therefore, we will call the first type teleological assumptions (i.e., $Assumptions_{teleological}$,) and the second type ontological assumptions (i.e., $Assumptions_{ontological}$,). Both types of assumptions serve the same purpose of closing the gap between the PSM and the task goal which should be

2. A formula α is weaker than a formula β iff every model of β is also a model of α, i.e. $\beta \models \alpha$ and $\models \beta \rightarrow \alpha$.

achieved by it. On the other hand, both types achieve this through a move in quite the opposite direction (see Figure 3).

In the second case of Figure 3, the PSM makes less assumptions about available domain knowledge. This must be compensated by stronger teleological assumptions, i.e. by decreasing the actual goal which can be achieved by the method. These relationships make it natural to view the sum of the effects of both types of assumptions as constant. The *role* of the two different types of assumptions (i.e., the direction of their influence) remains different. Ontological assumptions are required to define the functionality of a PSM, i.e. they *extend* the effect which can be achieved by the operational specification of a PSM. Teleological assumptions are required to close the gap between this functionality of a PSM and a given goal. They have to *weaken* the goal in cases where the final goal is beyond the scope of the functionality of the PSM.

Besides their different direction, both types of assumptions have something in common which leads to the natural question whether they are interchangeable. The composed outcome of their joined effort is constant. The question arises whether and how the weakening or strengthening of ontological assumptions can be compensated by strengthening or weakening the teleological assumptions and vice versa. This question is quite essential for the *applicability* of a PSM for a given task and domain. The knowledge requirements of a PSM can be weakened or strengthened according to

- the available domain knowledge,
- the effort which is required to derive and to model further knowledge, and
- the (teleological) assumptions which can be made to reduce the goal which must be achieved.

Teleological assumptions have to be made if the (ontological) assumptions about available domain knowledge cannot be satisfied to an extent that would enable the achievement of the goals as they are originally specified. The applicability problem for PSMs is therefore essentially a question of the relationships between these two different types of assumptions. We will illustrate this by an example taken from the area of diagnosis with component models.

> Component-based diagnosis with multiple faults is in the worst case exponential in the number of components ([Bylander et al., 1991]). Every element of the power-set of the set of annotated components is a possible hypothesis. If one is not interested in problem-solving in principle but in practice, further assumptions have to be introduced that either decrease the worst-case, or at least the average-case behavior. A drastic way to reduce the complexity of the diagnostic task is achieved by the *single-fault* or *N-fault* assumption (SFA) [Davis, 1984], which reduces the complexity to polynomial in the number of components. If the single-fault assumption holds, the incorrect behavior of the device is completely explainable by one failing component. Interestingly, the same assumption can either be interpreted as a requirement on domain knowledge or as a restriction of the delivered functionality. The SFA defines either strong requirements on the provided domain knowledge, or significantly restricts the diagnostic problems that can correctly be handled by the diagnostic system.
>
> - If the SFA has to be satisfied by the *domain knowledge*, then each possible fault has to be represented as a single entity. In principle this causes complexity problems for

the domain knowledge as each fault combination (combination of faulty components) has to be represented. However, additional domain knowledge could be used to restrict the exponential growth. [Davis, 1984] discusses an example of a representation change where a 4-fault case (i.e., 15 different combinations of faults) is transformed into a single fault. A chip with four ports can cause faults on each port. When we know that the individual ports never fail, but only the chip as a whole, a fault on four ports can be represented as one fault of the chip. Even without such a representation change, we do not necessarily have to represent all possible fault combinations. We could, for example, exclude all combinations that are not possible or likely in the specific domain (expert knowledge).

- Instead of formulating the requirement above on the domain knowledge, one can also weaken the *task definition* by this assumption. This means that the competence of the PSM meets the task definition under the assumption that only single faults occur. That is, only in cases where a single fault occurs, the method works correctly and complete.

It turns out that the same assumption can either be viewed as a requirement on domain knowledge or as a restriction of the goal of the task. Therefore, it is not an internal property of an assumption that decides its status, instead it is the functional role it plays during system development or problem solving that creates this distinction. Formulating it as a requirement asks for strong effort in acquiring domain knowledge during system development, and formulating it as a restriction asks for additional external effort during problem solving if the given case does not fulfil the restrictions and cannot be handled properly by the limited system.

3 ASSUMPTIONS IN DIAGNOSTIC PROBLEM SOLVING

The first diagnostic systems built were heuristic systems, in the sense that they contained compiled knowledge which linked symptoms directly to hypotheses (usually through rules). With these systems, only foreseen symptoms can be diagnosed, and heuristic knowledge that links symptoms with possible faults needs to be available. One of the main principles underlying model-based diagnosis [Davis, 1984] is the use of a domain model (called Structure, Behavior, Function (SBF) models in [Chandrasekaran, 1991]). Heuristic knowledge that links symptoms with causes is no longer necessarily in these systems. The domain model is used for predicting the desired device behavior, which is then compared to the observed behavior. A discrepancy indicates a symptom. General reasoning techniques such as constraint satisfaction or truth maintenance can be used to derive diagnoses that explain the actual behavior of the device using its model. Because the reasoning part is represented separately from domain knowledge, it can be reused for different domains. This paradigm of model-based diagnosis gave rise to the development of general approaches to diagnosis, such as “constraint suspension“ [Davis, 1984], DART [Genesereth, 1984], GDE [de Kleer & Williams, 1987], and several extensions to GDE (GDE+ [Struss & Dressler, 1989], Sherlock [de Kleer & Williams, 1989]).

In this section, we will focus on assumptions underlying these approaches to diagnostic problem solving. First, we discuss assumptions that are necessary to relate the task definition

of a diagnostic system with its real-world environment (see Section 3.1). That is, assumptions on the available case data, the required domain knowledge and the problem type. Second, we discuss assumptions introduced to reduce the complexity of the reasoning process necessary to execute the diagnostic task (see Section 3.2). Such assumptions are introduced to either change the worst-case complexity or the average-case behavior of problem solving. Third, we sketch further assumptions that are related to the appropriate *interaction* of the problem solver with its environment (see Section 3.3).

3.1 Assumptions Necessary to Define the Diagnostic Task

In model-based diagnosis (cf. [de Kleer et al., 1992]), the definition of the task of the KBS requires a *system description* of the device under consideration and a *set of observations*, where some indicate *normal* and other *abnormal* behavior. The goal of the task is to find a *diagnosis* that, together with the system description, *explains* the observations. In the following, we discuss four different aspects of such a task definition and show the assumptions related to each of them. The four aspects are: identifying abnormalities, identifying causes of these abnormalities, defining hypotheses, and defining diagnoses.

3.1.1 Identifying Abnormalities

Identification of abnormal behavior is necessary before a diagnostic process can be started to find explanations for the abnormalities. This identification task requires three kinds of knowledge, of which two are related to the type of input, and one to the interpretation of possible discrepancies (see [Benjamins, 1993]):

- *observations* of the behavior of the device must be provided to the diagnostic reasoner;
- a *behavioral description* of the device must be provided to the diagnostic reasoner;
- knowledge concerning the (*im*)*preciseness* of the observations and the behavioral description as well as *comparison knowledge* (thresholds, etc.) are necessary to decide whether a discrepancy is significant. Other required knowledge concerns the interpretation of *missing values*, and whether an observation can have several values (i.e., its value type).

Relevant assumptions state that the two types of inputs (i.e., observations amd behavioral descriptions) need to be *reliable*. Otherwise, the discrepancy could be explained by a measuring fault or a modelling fault. In other words, these assumptions guarantee that if a prediction yields a different behavior than the observed behavior of the artefact, then the artefact has a defect [Davis & Hamscher, 1988].

These assumptions are also necessary for the meta-level decision whether a diagnosis problem is given at all (i.e., whether there is an abnormality in system behavior). This decision relies on a further assumption: the *no design error assumption* [Davis, 1984] which says that if no fault occurs, then the device must be able to achieve the desired behavior. In other words, the discrepancy must be the result of a faulty situation where some parts of the system are defect. It cannot be the result of a situation where the system works correctly, but cannot achieve the desired functionality because it is not designed for this. If this assumption does not hold, one has a design problem and not a diagnostic problem.

3.1.2 Identifying Causes

Another purpose of the system description is the identification of possible causes of faulty behavior. This cause-identification knowledge must be *reliable* [Davis & Hamscher, 1988], or, in other words, the knowledge used in model-based diagnosis is assumed to be a correct and complete description of the artefact. Correct and complete in the sense, that it enables the derivation of correct and complete diagnoses if discrepancies appear.[3] In accordance with different types of device models and diagnostic methods, these assumptions wear different clothes. In the following, we restrict our attention to component-oriented device models that describe a device in terms of components, their behaviors (a functional description), and their connections.[4] The set of all possible hypotheses is the power-set of the set of annotated components

$$\{\ mode_{11}\ (c_1),\ \ mode_{12}\ (c_1), ...,\ \ mode_{nm_n}\ (c_n)\},$$

where the annotation $mode_{ji}(c_j)$ describes that the *j*-th component is in mode *i*. [Davis, 1984] has pointed out that one should be aware of the underlying assumptions for such a diagnostic approach and listed a number of them.

First, the *localised failure of function* assumption: the device must be decomposable in well-defined and localised entities (i.e., components) that can be treated as causes of faulty behavior. Second, these *components have a functional description* that provides the (correct) output for their possible inputs. If this functional description is local, that is, it does not refer to the functioning of the whole device, the *no function in structure* assumption [de Kleer & Brown, 1984] is satisfied. Several diagnostic methods also expect the reverse of the functional descriptions, thus, rules that *derive the expected input from the provided output* called "inference rules" in [Davis, 1984]. If only correct functional descriptions are available, fault behavior is defined as any other behavior than the correct one. Fault behavior of components can be constrained by including fault models, that is, *functional descriptions of the components in case they are broken* (cf. [de Kleer & Williams, 1989], [Struss & Dressler, 1989]). If one assumes that these functional descriptions are complete (the *complete fault knowledge* assumption), then components can be considered innocent if none of their fault descriptions is consistent with the observed faulty behavior. A result of using fault models is that all kinds of non-specified—and physically impossible—behaviors of a component are excluded as diagnosis. For example, using fault models, it becomes impossible to conclude that the fault *"one of two light bulbs is not working"* is explained by a defect battery that does not provide power and a defect lamp that lights without electricity (cf. [Struss & Dressler, 1989]).

Further assumptions that are related to the functional descriptions of components are the *no fault masking* and the *non intermittency* assumption. The former assumption states that the defect of an individual or composite component, or of the entire device must be visible by changed outputs (cf. [Davis & Hamscher, 1988], [Raiman, 1992]). According to the latter assumption, a component that gets identical inputs at different points of time, must produce

---

3. A typical problem of diagnosis without knowledge about fault models (i.e., incomplete knowledge) is that the reasoner provides, in addition to the right diagnoses, also wrong diagnoses. The result is complete but not correct because the provided domain knowledge is not complete.

4. It is a critical modelling decision what to view as a component and which types of interactions are represented (cf. [Davis, 1984]). Several points of view are possible to decide what is regarded as being a component. Different levels of physical representations result in different entities; the independent entities that are used in the manufacturing process of the artefact could be used as components; or functional unities of the artefact could be seen as components.

identical outputs. In other words, the output is a function of the input (cf. [Raiman et al., 1991]). [Raiman et al., 1991] argue that intermittency results from incomplete input specifications of components, but that it is impossible to get rid of it (it is impossible to represent all required additional inputs in a complete way).

A third assumption underlying many diagnostic approaches is the *no faults in structure* assumption (cf. [Davis & Hamscher, 1988]) that manifests itself in different variants according to the particular domain. The assumption states that the interactions of the components are correctly modelled and that they are complete. This assumption gives rise to three different classes of more specific assumptions. First, the *no broken interaction* assumption states that connections between the components work correctly (e.g. no wires between components are broken).[5] If this is too strong, the assumption can be weakened by representing the connections themselves as components too. Second, the *no unexpected directions* assumption (or existence of a causal pathway assumption, [Davis, 1984]) states that the directions of the interactions are correctly modelled and are complete. For example, a light bulb gets power from a battery and there is no interaction in the opposite direction. The *no hidden interactions* assumption (cf. [Böttcher, 1996]) assumes that there are no non-represented interactions (i.e., closed-world assumptions on connections). A bridge fault [Davis, 1984] is an example of a violation of this assumption in the electronic domain. Electronic devices whose components unintendedly interact through heat exchange, is another example [Böttcher, 1996]. In the worst case, all potential unintended interaction paths between components are represented [Preist & Welhalm, 1990]. The no hidden interactions assumption is critical since most models (like design models of the device) describe correctly working devices and unexpected interactions are therefore precisely not mentioned. A refinement of this assumptions is that there are no *assembly errors* (i.e., every individual component works correctly but they have been wired up incorrectly).

3.1.3 Defining Hypotheses

In addition to knowledge that is required to identify a discrepancy and knowledge that provides hypotheses used to explain these discrepancies, one requires further knowledge to decide which type of explanation is required. [Console & Torasso, 1992] distinguish two types of explanations: weak explanations, that are *consistent* with the observations (no contradiction can be derived from the union of the device model, the observations, and the hypothesis), and strong explanations, that *imply* the observations (the observations can be derived from the device model and the hypothesis). Both types of explanation can be combined by dividing observations in two classes: observations that need to be explained by deriving them from a hypothesis, and observations that need only be consistent with the hypothesis. In this case one requires *knowledge that allows to divide the set of observations*. The decision which type of explanation to use, can only be made based on assumptions about the environment in which the KBS is used.

3.1.4 Defining Diagnoses

Having established observations, hypotheses and an explanatory relation that relates hypotheses with observations, one must establish the notion of *diagnosis*. Not each

---

5. It is possible to represent the interactions between components as possible hypotheses but this leads to new problems (see 3.1.5).

hypothesis that correctly explains all observations needs to be a desired diagnosis. One could accept only *parsimonious* hypotheses as *diagnoses* (cf. [Bylander et al., 1991]). A hypothesis or explanation H is parsimonious if *H* is an explanation and there exists no other hypothesis *H'* that also is an explanation and *H'* < *H*. One has to make assumptions about the desired diagnosis (cf. [McIlraith, 1994]) in order to define the partial ordering (<) on hypotheses. For example, whether the diagnostic task is concerned with finding all components that are necessarily fault to explain the system behavior, or whether it is concerned with finding all components that are necessarily correct to explain the system behavior. In the first case, we aim at economy in repair, whereas in safety critical applications (e.g., nuclear power plants) one should obviously choose for the second case.

As shown by [McIlraith, 1994], the assumptions about the type of explanation relation (i.e., consistency versus derivability) and about the explanations (i.e., definition of parsimony) make also strong commitments on the domain knowledge (the device model) that is used to describe the system. If we ask for a consistent explanation with minimal sets of faulty components (i.e., $H_1 < H_2$ if H_1 assumes less components as being fault than H_2), we need knowledge that constrains the normal behavior of components. Otherwise we would simply derive all components as correct. If we ask for a consistent explanation with minimal sets of correct components (i.e., $H_1 < H_2$ if H_1 assumes less components as being correct than H_2), we need knowledge that constrains the abnormal behavior of components. Otherwise we would simply derive all components as faulty.

The definition of parsimonious hypotheses introduces a *preference* on hypotheses. This could be extended by defining further preferences on diagnoses to select one optimal one (e.g., by introducing assumptions related to the probability of faults). Again, knowledge about preferences must be available to define a preference function and a corresponding ordering.

3.1.5 Summary

Figure 4 summarises the assumptions that are discussed above and groups them according to their purpose. All these assumptions are necessary to relate the definition of the functionality of the diagnostic system with the diagnostic problem (i.e., the task) to be solved and with the domain knowledge that is required to define the task. Table 1 provides an explanation of the assumptions along with the role they play (function), the domain they are about (case data, domain knowledge or task), and some references where they are discussed in more detail.

Table 1: Effect Assumptions in component-oriented diagnosis (cd = case data, dk = domain knowledge, t = task).

name	explanation	is about	function	some references
existence of observations	observations must be provided to the diagnostic system	cd	It is necessary for detecting discrepancies.	[Benjamins, 1993]
reliability of observations	The provided observations must be reliable.	cd	It is necessary for assuming that the discrepancy must be explained by a diagnosis.	[Benjamins, 1993], [Davis & Hamscher, 1988]
existence of a behavioral description	The desired system behavior must be known to the diagnostic reasoner.	dk	It is necessary for detecting discrepancies.	[Benjamins, 1993]
reliability of behavioral description	The description of the system must be reliable.	dk	It is necessary for assuming that the discrepancy must be explained by a diagnosis.	[Benjamins, 1993], [Davis & Hamscher, 1988]

Table 1: Effect Assumptions in component-oriented diagnosis (cd = case data, dk = domain knowledge, t = task).

name	explanation	is about	function	some references
existence of knowledge to identify discrepances	Knowledge is required to compare the observations with the behavioral description.	dk	It is necessary for interpreting discrepancies.	[Benjamins, 1993]
reliability of the discrepancy identification knowledge	The knowledge used to detect discrepancies must be reliable.	dk	It is necessary for interpreting discrepancies correctly.	[Benjamins, 1993]
no design error	The discrepancy between expected and actual behavior does not result from the (incorrect) design of the device.	t	The behavioral discrepancy is a fault and not just an impossibility.	[Davis, 1984]
existence of a set of components	The device can be decomposed into a set of components.	dk	The entire device can be decomposed into smaller units that constitute the device.	[Davis, 1984], [Davis & Hamscher, 1988]
localized failure of function, no function in structure	Faulty components can be identified as causes.	dk	The reasons for faulty behavior do not have to be constructed but can be selected from a finite set.	[Davis, 1984], [de Kleer & Brown, 1984]
existence of a set of annotations (i.e., of component modes)	Components could have several behavioral modes that need to be provided.	dk	The diagnostic reasoner can select from the behavioral modes provided for each component.	[Struss & Dressler, 1989], [de Kleer et al., 1992]
completeness of the set of annotations = complete fault knowledge	All possible modes of the components are known.	dk	It is used to infer the mode of a component if all other behaviors do not (even not partially) explain the fault.	[Struss & Dressler, 1989], [de Kleer et al., 1992]
existence of input-output descriptions of the components	This knowledge defines the input-output behavior of the components.	dk	The behavioral description of the components is required to detect their faulty behavior and to derive the overall behavior of the complete device.	[de Kleer & Williams, 1987], [Davis & Hamscher, 1988]
existence of output-input descriptions of the components	This knowledge defines the output-input relation of the components.	dk	This knowledge can be used to derive additional discrepancies.	[Davis, 1984], [Raiman, 1989]
existence of functional descriptions of faulty behavior of components	This knowledge defines the input-output behavior of the components in case they are broken.	dk	The behavioral description of the components is required to identify different possible faults of a component.	[de Kleer & Williams, 1987], [Struss & Dressler, 1989]
complete behavioral descriptions (complete fault models)	All possible behaviors of a component are modelled by its functional description.	dk	It is used to completely constrain the possible behavior of a component.	[de Kleer & Williams, 1987], [Struss & Dressler, 1989]
no fault masking	A fault of a component is visible in its behavior and in the behavior of the entire device.	cd & dk	It is necessary for detecting faulty components.	[Davis, 1984], [Davis & Hamscher, 1988], [Raiman, 1992]
non intermittency	The output of a component is a function of the input (e.g., the behavior does not change over time).	cd	It is necessary for interpreting the discrepancy between an observation and an output of a behavioral description of a component.	[Davis, 1984], [Raiman et al., 1991]

Table 1: Effect Assumptions in component-oriented diagnosis (cd = case data, dk = domain knowledge, t = task).

name	explanation	is about	function	some references
existence of a model of the component interactions	It assumes that the possible interactions between components are known to the reasoner.	dk	This model is required to derive the overall behavior of the system and the local inputs of components from the local outputs of the components.	[Davis, 1984], [Davis & Hamscher, 1988]
no fault in structure assumption	Faulty components are the only causes.	dk	Only components need to be treated as possible causes for the faulty behavior.	[Davis, 1984], [Davis & Hamscher, 1988]
no broken interactions	The interactions work properly, i.e., the connections work properly.	dk	Only components need to be treated as possible causes for the faulty behavior and the interaction model describes the real interactions.	[Davis, 1984], [Davis & Hamscher, 1988]
no unexpected direction	The direction of the interaction is as represented.	dk	Only components need to be treated as possible causes for the faulty behavior and the interaction model describes the real interactions.	[Davis, 1984]
no hidden interactions (closed world assumption)	There are no interactions that are not represented in the model.	dk	Only components need to be treated as possible causes for the faulty behavior and the interaction model describes the real interactions.	[Davis, 1984], [Böttcher, 1996]
no assembly error	The components are not wired incorrectly.	dk	Only components need to be treated as possible causes for the faulty behavior and the interaction model describes the real interactions.	[Davis & Hamscher, 1988], [Böttcher, 1996]
type of explanation relation (type of hypotheses)	Need an observation be consistent with the hypothesis or must it be derivable from it.	dk & t	The problem solving is either constraints satisfaction or abductive inference.	[Console & Torasso, 1992], [de Kleer et al., 1992], [ten Teije & van Harmelen, 1996]
classification of observations	It introduces an distinction between observations that describes normal and abnormal behavior.	dk	In abductive inference only the abnormal behavior must be explained.	[Console & Torasso, 1992]
type of explanation (type of diagnosis)	Should the set of fault components contain all components that need to be fault or that could be fault.	dk & t	The diagnosis is used for an economic repair process versus it is used for safety-critical monitoring.	[McIlraith, 1994]
preference knowledge on diagnoses	It defines preferences between diagnoses.	dk	Necessary for selecting the diagnoses with high preferences.	[de Kleer & Williams, 1987], [Davis & Hamscher, 1988]

All these assumptions are necessary to relate a model of the device with the actual device under concern. “There is no such thing as an assumption-free representation. Every model, every representation contains simplifying assumptions“ [Davis & Hamscher, 1988]. If the assumptions are too strong, one could consider weakening them.[6] However, this raises

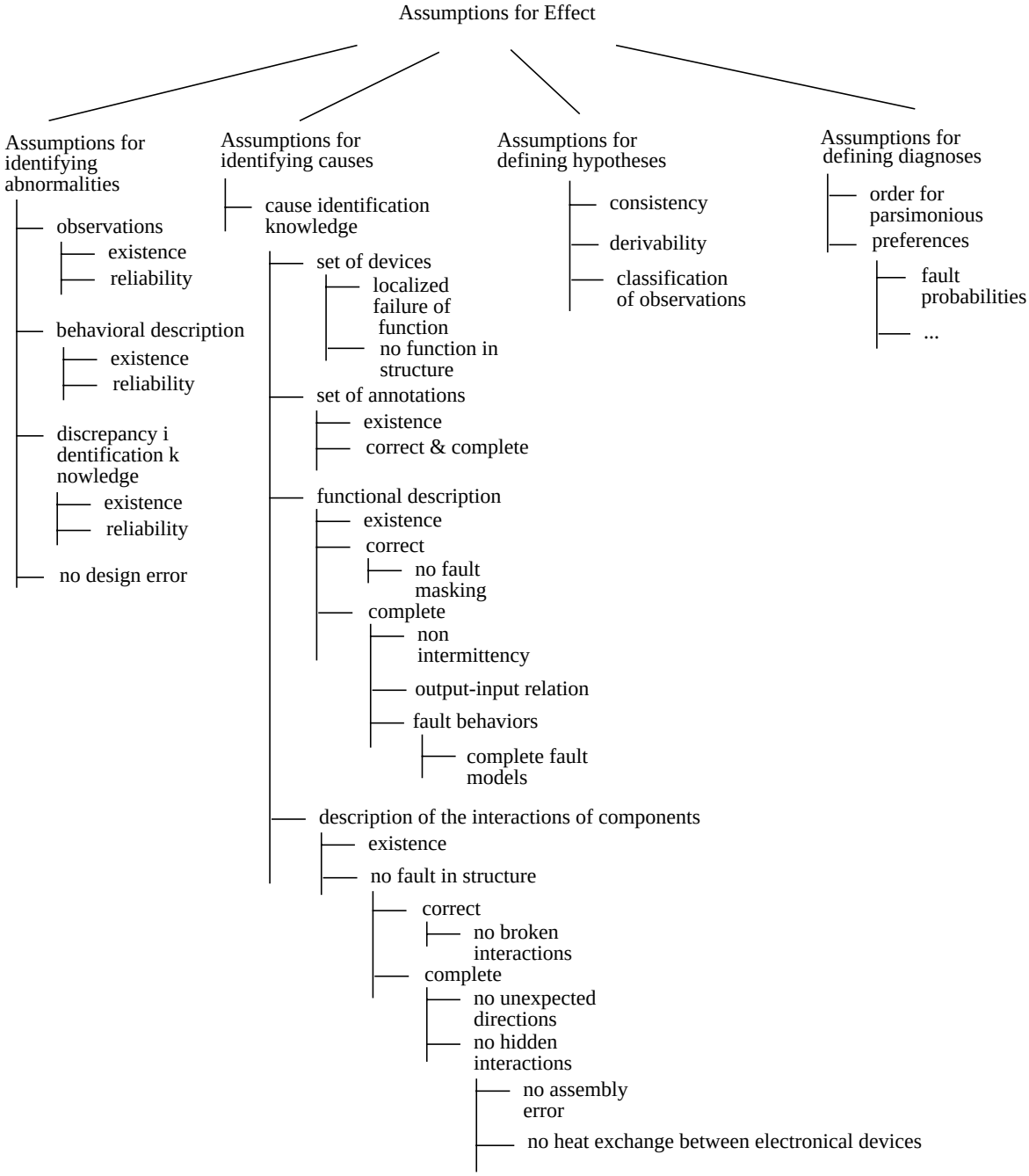

Fig. 4 Assumptions for Effect.

another problem in model-based diagnosis, namely its *high complexity or intractability*. This will be discussed in the following section.

---

6. For example, they can be weakened by representing all desired interactions as components (e.g., wires) that could fail; by representing additional possibilities of interactions (e.g., electronical devices can interact via heat exchange) [Böttcher, 1996]; by representing all potential unintended interaction paths between components [Preist & Welhalm, 1990]; by representing additional inputs to get rid of intermittency [Raiman et al., 1991]. Each of these weakenings significantly increases the computational complexity of the problem-solving process.

3.2 Assumptions Necessary to Define an Efficient Problem Solver

Besides the assumptions that are necessary to define the diagnostic task, further assumptions are necessary because of the complexity of model-based diagnosis. Component-based diagnosis is in the worst case exponential in the number of annotated components ([Bylander et al., 1991]). Every element of the power-set of the set of annotated components is a possible hypothesis. As we are not interested in problem-solving in principle but in practice, further assumptions have to be introduced that either decrease the worst-case, or at least the average-case behavior.

3.2.1 Reducing the Worst-Case Complexity: The Single-Fault Assumption

A drastic way to reduce the complexity of the diagnostic task is achieved by the *single-fault* or *N-fault* assumption [Davis, 1984], which reduces the complexity to polynomial in the number of components. If the single-fault assumption holds, the incorrect behavior of the device is completely explainable by one failing component. As already mentioned in section 2.2, this assumption defines either strong requirements on the provided domain knowledge, or significantly restricts the diagnostic problems that can correctly be handled by the diagnostic system. In the first case, each possible fault has to be represented as a single entity. In the second case, the methods works only in cases where a single fault occurs.

3.2.2 Reducing the Average-Case behavior: The Minimality Assumption of GDE

As the single-fault assumption might be too strong an assumption for several applications, either as a requirement on the domain knowledge or as a restriction on the task, [Reiter, 1987] and [de Kleer & Williams, 1987] provide approaches able to deal with multiple faults. However, this re-introduces the complexity problems of MBD. To deal with this problem, GDE [de Kleer & Williams, 1987] exploits the *minimality assumption*, which reduces, in practical cases, the exponential worst case behavior to a complexity that grows with the square of the number of components. In GDE, this assumptions helps reducing the complexity in two ways. First, a conflict is a set of components that cannot work correctly given the provided domain knowledge and the observed behavior. Under the minimality assumption, each super-set of a conflict is also a conflict and all conflicts can be represented by minimal conflicts. Second, a hypothesis contains at least one component of each conflict. Every super-set of such a hypothesis is again a hypothesis. Therefore, diagnoses can be represented by minimal diagnoses. The minimality assumption requires that diagnoses are independent or monotonic (see [Bylander et al., 1991]): a diagnosis that assumes more components as being faulty, explains more observations.

A drastic way to ensure that the minimality assumption holds, is to neglect any knowledge about the behavior of faulty components. Thus, any behavior that is not correct is considered as fault. A disadvantage of this is that physical rules may be violated (i.e., existing knowledge about faulty behavior). We already mentioned the example provided in [Struss & Dressler, 1989], where a fault (one of two bulbs does not light) is explained by a broken battery that does not provide power and a broken bulb that lights without power. Knowledge about how components behave when they are faulty (called fault models) could be used to constrain the set of diagnoses derived by the system. On the other hand, it increases the complexity of the task. If for one component m possible fault behaviors are provided, this leads to $m+1$ possible states instead of two (correct and fault). The maximum number of candidates increases from

2^n to $(m+1)^n$.

A similar extension of GDE that includes fault models, is the Sherlock system (cf. [de Kleer & Williams, 1989]). With fault models, it is no longer guaranteed that every super-set of the faulty components that constitute the diagnosis, is also a diagnosis, and therefore the minimality assumption as such cannot be exploited. In Sherlock, a diagnosis does not only contain fault components (and implicitly assumes that all other, not mentioned, components are correct), but it contains a set of components assumed to work correctly and a set of components assumed to be fault. A conflict is now a set of some correct and fault components that is inconsistent with the provided domain knowledge and the observations. In order to accommodate to this situation, [de Kleer et al., 1992] extend the concept of minimal diagnoses to kernel diagnoses and characterise the conditions under which the minimality assumption still holds. The kernel diagnoses are given by the prime implicants of the minimal conflicts. Moreover, the minimal sets of kernel diagnoses sufficient to cover every diagnosis correspond to the irredundant sets of prime implicants[7] of all minimal conflicts. These extensions cause drastic additional effort, because there can be exponentially more kernel diagnoses than minimal diagnoses, and finding irredundant sets of prime implicants is NP-hard. Therefore, [de Kleer et al., 1992] characterise two assumptions under which the kernel diagnoses are identical to the minimal diagnoses. The kernel diagnoses are identical to the minimal diagnoses if all conflicts contain only fault components. In this case, there is again only one irredundant set of minimal diagnoses (the set containing all minimal diagnoses). The two assumptions that can ensure these properties are the *ignorance of abnormal behavior* assumption and the *limited knowledge of abnormal behavior* assumption.

The ignorance of abnormal behavior assumption excludes knowledge about faulty behavior and thus characterises the original situation of GDE. The limited knowledge of abnormal behavior assumption states that the knowledge of abnormal behavior does not rule out any diagnosis indicating a set of faulty components, if there exist a valid diagnosis indicating a subset of them as faulty components, and if the additionat components assumed faulty are not inconsistent with the observations and the system description.[8] The latter assumption is a refinement of the former, that is, the truth of the ignorance of abnormal behavior assumption implies the truth of the limited knowledge of abnormal behavior assumption.

A similar type of assumption is used by [Bylander et al., 1991] to characterise different complexity classes of component-based diagnosis. In general, finding one or all diagnoses is intractable. The *independent* and *monotonic* assumption, which have the same effect as the limited knowledge of abnormal behavior assumption, require that each super-set of a diagnosis indicating a set of faulty components is also a diagnosis.[9] In this case, the worst-case complexity of finding one minimal diagnosis grows polynomially with the square of the number of components. However, the task of finding all minimal diagnoses is still NP-hard in the number of components. This corresponds to the fact that the minimality assumption of GDE (i.e., the ignorance of abnormal behavior and limited knowledge of abnormal behavior assumptions), that searches for all diagnoses, does not change the worst-case but only the

---

7. See [McCluskey, 1956]. An implicant is a conjunction of positive and negative literals. Without fault models, minimal hypotheses contain only negative literals ($\neg ok(c_i)$). In the case of fault models we have positive and negative literals ($ok(c_i)$ and $\neg ok(c_i)$) in the hypotheses. Therefore, minimality cannot be simply defined by set inclusion of the literals of a conjunction.

8. [McIlraith, 1994] generalizes these assumptions for the dual case of diagnosing a minimal set of components proven to be correct and applies these assumptions for characterizing minimal abductive diagnoses.

9. More precisely, the explanatory power of a hypothesis increases monotonously by adding fault or correct components.

average-case behavior of the diagnostic reasoner.

3.2.3 Search Guidance

The complexity of component-based diagnosis (especially when working with fault models) requires further assumptions that enable efficient reasoning for practical cases (cf. [Struss, 1992], [Böttcher & Dressler, 1994]). Again, these assumptions do not change the worst case complexity but should reduce the necessary effort in practical cases. A well-known notion to increase efficiency is a reasoning focus. Defining a focus for the reasoning process can be achieved by exploiting hierarchies or probability information. The *hierarchically-layered device-model* assumption assumes the existence of hierarchically layered models that allow step wise refinement of diagnosis to reduce the complexity of the diagnostic process (cf. the complexity analysis of hierarchical structures of [Goel et al., 1987]). The large number of components at the lowest level of refinement is replaced by a small number of components at a higher level. Only the relevant parts of the model are refined during the problem-solving process. The *hierarchically-layered behavioral-model* assumption assumes the existence of more abstract descriptions of the behavior that can improve the efficiency because reasoning can be performed at a more coarse grained, and thus simpler, level (cf. [Abu-Hanna, 1994]). The *existence of probabilities* assumption assumes knowledge about the probability of faults that can be used to guide the search process for diagnoses by focusing on faults with high probabilities. Usually, these probabilities introduce new assumptions (e.g., the *components fail independently* assumption [de Kleer & Williams, 1989]).

All these knowledge types and their related assumptions rely on further assumptions concerning the utility of this search control knowledge. For example, the hierarchically-layered device model improves only the search process when the faults are not distributed in a way that enforces the problem solver to expand each abstract component descriptions to their lowest levels. It significantly improves the search process if the problem solver need to refine only one abstract component description at each level.

3.2.4 Summary

Figure 5 summarises the assumptions and groups them according to their purpose. All these assumptions are introduced to reduce the computational effort required to solve the problem. Table 2 provides an explanation of the assumptions along with the role they play (function), the domain they are about (case data, domain knowledge or task), and some references where they are discussed in more detail.

Table 2: Efficiency Assumptions in component-oriented diagnosis (cd = case data, dk = domain knowledge, t = task).

name	explanation	is about	function	some references
single fault (SFA), *N*-fault	There is one or there are at most *N* faults.	dk or t	It polynomializes the worst-case complexity for finding one or all diagnoses.	[Davis, 1984]

Table 2: Efficiency Assumptions in component-oriented diagnosis (cd = case data, dk = domain knowledge, t = task).

name	explanation	is about	function	some references
minimality	Sets of hypotheses can be represented by one minimal hypothesis.	dk	It polynomializes the average-case behavior for finding all diagnoses.	[Reiter, 1987], [de Kleer & Williams, 1987], [Bylander et al., 1991], [de Kleer et al., 1992]
ignorance of abnormal behavior	No knowledge that constrains possible faulty behavior is provided.	dk	It polynomializes the average-case behavior for finding all diagnoses.	[de Kleer & Williams, 1987], [de Kleer et al., 1992]
independency	The explanatory power of a diagnosis is the union of the explanatory power of its elements.	dk	It polynomializes the worst-case complexity for finding one diagnoses.	[Bylander et al., 1991]
monotonicity	The explanatory power of a diagnosis increases monotonously with its size.	dk	It polynomializes the worst-case complexity for finding one diagnoses.	[Bylander et al., 1991]
limited knowledge of abnormal behavior	Valid diagnoses do not become invalid by adding further correct or fault components to it.	dk	It polynomializes the average-case behavior for finding all diagnoses.	[de Kleer et al., 1992]
existence of search control knowledge	This knowledge is used to guide the search process for diagnoses.	dk	It improves the average-case behavior for finding all diagnoses.	[Struss, 1992], [Böttcher & Dressler, 1994]
existence of a hierarchically-layered device-model	The device model is hierarchically structured.	dk	The hierarchical structure of the device focuses the search process.	[Goel et al., 1987], [Struss, 1992], [Böttcher & Dressler, 1994]
existence of a hierarchically-layered behavioral-model	The behavioral description of the system is hierarchically structured.	dk	Abstract descriptions of the behavior should reduce the search effort.	[Struss, 1992], [Abu-Hanna, 1994], [Böttcher & Dressler, 1994]
existence of probabilities	Faults are annotated by their probability.	dk	Probabilities of faults focus the search process.	[de Kleer & Williams, 1987], [Struss, 1992], [Böttcher & Dressler, 1994]
fault probabilities are independent	Each fault appears independently from other possible faults.	cd & dk	It is used in computing probabilities for hypotheses.	[de Kleer & Williams, 1989]

3.3 Assumptions in System-Environment Interaction

Until now, we have sketched a diagnostic problem solver working in batch mode. In receives some observables as input and tries to efficiently derive a number of hypotheses that can explain the fault behaviors. However, this is not a very reaslistic scenario especially in the case where hypothesis *discrimination* becomes necessary. In general, hypothesis discrimination becomes necessary if the number of hypotheses found, exceeds the desired number (cf. [Davis & Hamscher, 1988]). Additional observations must be provided as the initial observations were not strong enough to discriminate between existing hypotheses.

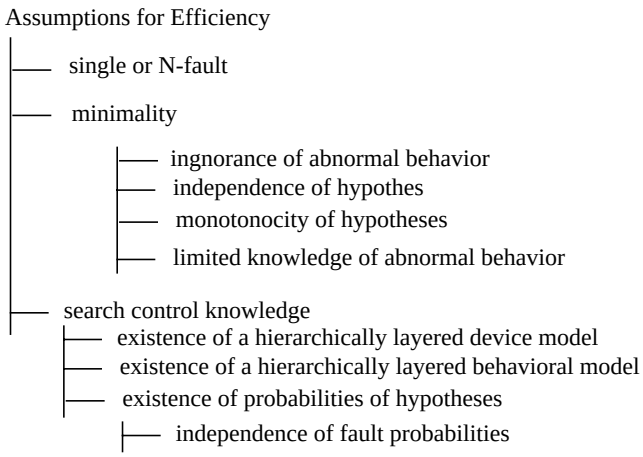

Fig. 5 Assumptions for Efficiency.

Assumptions related to this activity will be discusses now (see Table 3). First, it must be possible to obtain *additional observations*. Examples of more specific versions of this assumption are: can the device be unfastened, are measuring points reachable, can components be replaced easily to test behavior, can new input be provided to the device, etc. Second, assumptions can be made about the *utility of additional observations*. One can assume cost information of additional measurements and knowledge about their discriminatory power (i.e., knowledge about dependencies between hypotheses) to optimise their selection. GDE uses minimal entropy as a measure to mminimize the expected number of tests (= additional observations). FAULTY [Abu-Hanna et al., 1991] minimizes the estimated number of tests based on a variety of balanced global factors. Again, NP-hard problems arise if one tries to optimise these decisions. Therefore, assumptions concerning *heuristic knowledge* that guide this process are necessary.

Table 3: Interaction Assumptions in component-oriented diagnosis (cd = case data, dk = domain knowledge, t = task).

name	explanation	is about	function	some references
Possibility of additional observations	What are further possible observations.	dk	It is necessary to get further information for hypotheses discrimination.	[Davis & Hamscher, 1988], [Benjamins, 1993]
Utility of additional observations	How useful are these observations (information gain versus costs).	dk	It necessary for optimal decisions during hypotheses discrimination.	[de Kleer & Williams, 1987], [Davis & Hamscher, 1988], [Benjamins, 1993]
heuristic search knowledge	This knowledge is used to guide the search process for optimal selection of further observations.	dk	It necessary for efficiently making sub-optimal decisions.	[de Kleer & Williams, 1987], [Davis & Hamscher, 1988], [Benjamins, 1993]

All these assumptions are necessary to optimise the cooperation of the diagnostic system with

its environment. In principle, one could assume that all observations that are possible are provided to the system before it starts its diagnostic reasoning. However, collecting observations is often a major cost-determining factor. Therefore, assumptions are introduced concerning the efficiency of gathering information with minimal costs.

4 ASSUMPTIONS AS GUIDELINES FOR THE KNOWLEDGE ACQUISITION PROCESS

Software architectures have received increasing interest by the software engineering community to enhance the system development process and the level of software reuse (cf. [Garlan and D. Perry, 1995], [Shaw & Garlan, 1996]). In this paper, we have presented an architecture for describing KBSs. We showed the essential role assumptions play in this architecture to ensure that the different parts of a KBS specification stand in proper relationships to each other and to ensure the adequate relationship of the overall specification with its environment. We expect that dealing with these assumptions will become the backbone of the knowledge engineering process.

In general, one can distinguish three different activities in dealing with assumptions resembling the different reasoning styles of deduction, abduction, and induction: validation and verification of assumptions, searching and constructing assumptions, and constructing knowledge to fulfil assumptions.

4.1 Validation and Verification of Assumptions

Verification and validation of assumptions is an important part of developing correct reasoning systems. In [Fensel & Schönegge, 1997b], we adapted the Karlsruhe Interactive Verifier (KIV) [Reif, 1995] for verifying architectural specifications of KBSs. The KIV system is an advanced tool for the construction of provably correct software. It supports the entire design process starting from formal specifications and ending with verified code. An essential part of KIV is a tactical theorem prover that interactively supports the verification of first-order specifications with specifications in dynamic logic. Originally designed for the development of procedural programs, we describe in [Fensel & Schönegge, 1997b] how it can be used for the purpose of verifying KBSs. The main activity in adapting KIV is to refine the generic module concept of KIV for the specific conceptual model used to specify KBSs and to develop a method that enables systematic bokkeeping of assumptions, which are used to relate the different parts of a specification.

Existing work on verifying KBSs (cf. [Lydiard, 1992], [Plant & Preece, 1996]) is focused at specific representation formalisms (usually production rules and KL-ONE like formalisms) and prove rather abstract properties of KBSs (so-called *anomalies*). One the one hand, these approaches make very strong (meta-)assumptions by assuming a specific representation formalisms for describing the KBS. On the other hand, they do not make any (meta-) assumptions about the architecture (i.e., the general KBS structure) that can be used to describe a reasoning system. In consequence, most of these approaches are situated at a different level of generality than our approach (cf. [Newell, 1982]).

4.2 Searching and Constructing Assumptions

Verifying assumptions requires that one is aware of these assumptions. This raises the natural question of how one gets aware of assumptions. Most commonly, an assumption is noticed in case it is no longer valid and causes a system error. Clearly, this is a very dangerous and costly method. In [Fensel & Schönegge, 1997a], [Fensel & Schönegge, submitted], we show how assumptions that are necessary to close gaps between different elements of a specification can be found using *failed* attempts to prove their proper relationship. In other words, we try to prove that a PSM achieves a goal and the assumptions appear as gaps in the proof process. The analysis of partial proofs gives hints for the construction of possible counter examples and for repairing the proof by introducing further assumptions. These assumptions are the *missing pieces* in proving the correctness of the specification. Verifying these specification is therefore a way to detect underlying hidden assumptions. Again, we could apply the *interactive* theorem prover of KIV as tool support. It returns with open goals that it cannot prove but their assertion would be sufficient to complete the proof. These open goals are assumptions that are sufficient (but not necessarily minimal) to establish the correct relationships. As opposed to verification, here one does not start a proof with the goal to prove correctness. Instead, one starts an impossible proof and views the proof process as a search and construction process of assumptions. In the following we will refer to this method for detecting assumptions as *inverse verification*.[10] We will take an example from [Fensel & Schönegge, 1997a], [Fensel & Schönegge, submitted] to illustrate our point. Let us assume a task definition that asks for a complete and parsimonious explanation for a set of observables:

task *complete and parsimonious explanation*
$goal(x) \leftrightarrow complete(x) \wedge parsimonious(x)$
$complete(x) \leftrightarrow expl(x) = observables$
$parsimonious(x) \leftrightarrow \neg\exists x' \, (x' \subset x \wedge expl(x) \subseteq expl(x'))$
end

That is, an explanation has to explain all observables and no smaller explanation may exists that has the same or larger explanatory power. We further assume a PSM that has the competence to find complete and local-parsimonious explanations based on a local search algorithm adapted to diagnosis, i.e.:

PSM competence *complete and local-parsimonious explanation*
$output(x) \leftrightarrow complete(x) \wedge local\text{-}parsimonious(x)$
$local\text{-}parsimonious(x) \leftrightarrow \neg \, \exists x', y \, (y = x \setminus \{x'\} \wedge expl(x) \subseteq expl(x'))$
end

That is, the method finds complete explanations that cannot be further minimized by deleting one hypothesis from them. Using the interactive theorem prover of KIV, we find the following assumption that is necessary to close the gap between the goal of the task and the competence of the PSM:

Monotonic abduction assumption
$y \subset x \rightarrow expl(y) \subseteq expl(x)$
end

10. Another method is used in this paper where Section 3 provides a survey of literature on model-based diagnosis where assumptions are usually discussed as a side aspect of introduced reasoning strategies.

This assumption that we constructed with inverse verification was already mentioned in Section 3.2 as “monotonicity“ assumption. It is used by many approaches for model-based diagnosis to reduce the computational effort in diagnosis. It assumes that a smaller set of hypothesis always can explain less observations, i.e. extending a set of hypotheses is a straightforward way to enrich the explanatory power of the hypotheses set.

Clearly, inverse verification states a typical abductive problem:

> “The problem of performing deduction of new facts from a set of axioms is well-studied and understood. An equality important but far less explored problem is the derivation of hypotheses to explain observed events. In formal terms this involves finding an *assumption* that, together with some axioms, implies a given formula.“ [Cox & Pietrzykowski, 1986]

[de Kleer, 1986] describes a truth-maintenance system (ATMS) that could in principle applied to our problem. Actually most of the approaches to model-based diagnosis we discussed in section 3 use adaptations of this technique. However, applying this technique introduces two strong (meta-)assumptions:

- All the assumptions required to solve the gap between the goals of the task and the competence of the PSM must already be known and provided to the system.
- The system needs to know the impacts of the assumptions, i.e. their influence on the truth of the formulas describing competence of the problem-solving method and the goals of the task.

If this complete knowledge is available establishing the proper set of assumptions boils down to select a minimal set of assumptions and a bookkeeping mechanism like ATMS can process this task. When such a complete set of assumptions does not exist, finding assumptions is rather a constructive activity.

Constructive approaches to derive such assumptions (also called weakest preconditions [Dijkstra, 1975]) can be found in program debugging with inductive techniques (cf. [Shapiro, 1982], [Lloyd, 1987]), explanation-based learning (cf. [Minton et al., 1989], [Minton, 1995]) or more general in inductive logic programming ([Muggleton & Buntine, 1988], [Muggleton & De Raedt, 1994]). However, these approaches achieve automatization by making strong (meta-)assumption about the syntactical structure of the representation formalisms of the components, about the representations of the “error“, and about the way an error can be fixed. Usually, Prolog or Horn logic is the assumed representation formalism and errors or counter-examples are represented by a set of input-output tuples or a finite set of ground literals. Modification is done by changing the internal specification of a component. In this scenario, error detection boils down to backtrack a resolution-based derivation tree for a “wrong“ literal. In extension to the scope of these techniques, we have to aim for new formulas (i.e., an assumption may be represented by a complex first-order formula) and our “counter-examples“ are not represented by a set of ground literals but by a complex first-order specification. Again most of the mentioned approaches do not regard architectural descriptions of the entire reasoning system. An exception form approaches to explanation-based learning that use explicit architecture axioms [Minton, 1995].

4.3 Constructing Knowledge to Fulfil Assumptions

Up to now, we have discussed assumptions as input of the verification process and as outcome of the inverse verification process. But what to do if some assumption is proven to be necessary without being fulfilled? That is, inverse verification has shown they are required to enable proper relationships between the components, and verification has proven that the responsible component does not fulfil the assumption. Usually these assumptions formulate requirements on domain knowledge that is not (yet) available. Therefore, such derived but violated assumptions define goals for manual and automatic knowledge acquisition techniques that can make use of these explicit goals. Using such assumptions as goals for machine learning, knowledge discovery, and data mining techniques therefore introduces interesting links to recent work on *goal*-driven learning [Ram & Leake, 1995b] and knowledge discovery approaches [Mark, 1996] that reflect the task environment the knowledge should be used in (see [Engels, 1996], [Engels et al., submitted]). Two basic principles are shared with these approaches:

- the use of an architecture that structures problem solving and learning ([Ram et al., 1995])
- the explicit notion of goals (or target concepts) that guide the problem-solving and learning process [Ram & Leake, 1995a].

Most of the learning effort is viewed to be triggered from knowledge gaps and failures [Ram et al., 1995]. Assumptions are explicit notions of what is required as knowledge by the reasoner. In that sense they allow the application of learning techniques during the development and design process of the systems. Instead of manifesting itself as a runtime error, a knowledge gap and failure is made explicit from the beginning. The notion of goals and target concepts for selecting and guiding learning techniques enables these techniques to deal with goals and failures indicated by assumptions, for example:

- **complete fault knowledge.** Using explanation-based or conceptual clustering techniques to generalize cases of misbehavior of the device that cannot be explained by the diagnostic problem solver, can lead to a realistic task definition that specifies the cases that can/cannot be solved by the problem solver.
- **Existence of input-output description of components.** Scientific discovery techniques can be used to derive functional representations from observing the behavior of the components.
- **Single-fault assumption.** Learning techniques that modify the representation formalism of the device can derive a representation where multi-faults are represented by single entities.
- **Existence of heuristic search-control knowledge.** Explanation-based learning techniques are designed to learn control rules that should improve system performance. Clustering techniques can be used to establish hierarchical structured system description.

5 CONCLUSIONS

[Fensel & Groenboom, 1997] introduced an architecture for the description of knowledge-based systems that decomposes its specification into four different parts: a task, specifying

the reasoning goals of the system; a PSM, specifying its reasoning behavior; a domain model that provides the required domain knowledge; and an adapter establishing the proper relationship between the different parts and enable reusability of the other parts. This architecture generalizes existing approaches as the *model of expertise* of CommonKADS [Schreiber et al., 1994] for purpose of reusability of the different elements of the model. This architecture focuses the attention on the different types of assumptions that have to be made to establish a consistent and correct system model. It is essential to know the underlying assumptions of a reasoning system in order to know when it is applicable. Moreover, assumptions are good ways to characterise systems and they can be used to guide the acquisition process of domain knowledge. They define the type of knowledge and its properties as they are required by the reasoner. In this paper, we have dealt with three aspects related to these assumptions:

- We showed the complementary role assumptions play to restrict the complexity of the task or the required competence of the domain knowledge used as resource for the reasoning process. We called this the law of conservation of assumptions [Benjamins et al., 1996].
- We provided an extensive survey on assumptions based on work on model-based diagnosis. This survey provides an empirical base of our argument.
- Finally, we sketched the role of assumptions in the development *process* of a reasoning system. They define obligations for verification, goals for inverse verification, and target concepts for manual and automatic knowledge construction techniques.

Currently, we apply our ideas to PSMs from the area of design problem solving [Fensel et al., 1997], planning [Barros et al., 1997] and develop more systematic support in explicating the context of knowledge components and in adapting them to changed context [Fensel & Schönegge, submitted]. This should enable the reuse of knowledge components and knowledge-based reasoners in heterogeneous environments [Benjamins, 1997], [Fensel, 1997a].

Acknowledgment. We thank Claudia Böttcher, Bert Bredeweg, Joost Breuker, Kees de Koning, Remco Straatman, Annette ten Teije, and Frank van Harmelen for helpful comments.

Richard Benjamins was partially supported by the Netherlands Computer Science Research Foundation with financial support from the Netherlands Organisation for Scientific Research (NWO), and by the European Commission through a Marie Curie Research Grant (TMR).

REFERENCES

[Abu-Hanna, 1994] A. Abu-Hanna: Multiple Domain Models in Diagnostic Reasoning, PhD thesis, University of Amsterdam, 1994.

[Abu-Hanna et al., 1991] A. Abu-Hanna, V. R. Benjamins and W. N. H. Jansweijer: Device Understanding and Modeling for Diagnosis, *IEEE-Expert*, 6(2):26—32, 1991.

[Akkermans et al., 1993] J. M. Akkermans, B. Wielinga, and A. TH. Schreiber: Steps in Constructing Problem-Solving Methods. In N. Aussenac et al. (eds.), *Knowledge-Acquisition for Knowledge-Based Systems*, LNAI 723, Springer-Verlag, Berlin, 1993.

[Angele et al., 1996] J. Angele, D. Fensel, and R. Studer: Domain and Task Modelling in MIKE. In A. Sutcliffe et al. (eds.), *Domain Knowledge for Interactive System Design*, Chapman & Hall, 1996.

[Barros et al., 1997] Barros, L. Nunes de, J. Hendler, and V. R. Benjamins: Par-KAP: A Knowledge Acquisition Tool for Building Practical Planning System. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI ´97)*, Nagoya, Japan, August 1997.

[Benjamins, 1993] R. Benjamins: *Problem-Solving Methods for Diagnosis*, PhD Thesis, University of Amsterdam, Amsterdam, the Netherlands, 1993.

[Benjamins, 1995] V. R. Benjamins: Problem Solving Methods for Diagnosis And Their Role in Knowledge Acquisition, *International Journal of Expert Systems: Research and Application*, 8(2):93—120, 1995.

[Benjamins, 1997] R. Benjamins: Problem-Solving Methods in Cyberspace. In *Proceeedings of the Workshop on Problem-Solving Methods for Knowledge-based Systems (W26)* of the *Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97)*, NAGOYA, Japan, August 23-29, 1997.

[Benjamins & Aben, 1997] R. Benjamins and M. Aben: Structure-Preserving Knowledge-Based System Development through Reusable Libraries: a Case Study in Diagnosis, *International Journal on Human-Computer Studies (IJHCS)*, 47(2):223—258, 1997.

[Benjamins et al., 1996] R. Benjamins, D. Fensel, and R. Straatman: Assumptions of Problem-Solving Methods and Their Role in Knowledge Engineering. In *Proceedings of the 12. European Conference on Artificial Intelligence (ECAI-96)*, Budapest, August 12-16, 1996.

[Benjamins & Pierret-Golbreich, 1996] R. Benjamins and C. Pierret-Golbreich: Assumptions of Problem-Solving Methods. In N. Shadbolt et al. (eds.), *Advances in Knowledge Acquisition*, Lecture Notes in Artificial Intelligence (LNAI), no 1076, Springer-Verlag, Berlin, 1996.

[Böttcher, 1996] C. Böttcher: No Faults in Structure? - How to Diagnose Hidden Interactions, 1996.

[Böttcher & Dressler, 1994] C. Böttcher and O. Dressler: A Framework For Controlling Model-Based Diagnosis Systems with Multiple Actions, *Annals of Mathematics and Artificial Intelligence*, 11:241-261, 1994.

[Breuker, 1997] J. Breuker: Problems in Indexing Problem Solving Methods. In *Proceeedings of the Workshop on Problem-Solving Methods for Knowledge-based Systems (W26)* of the *Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97)*, NAGOYA, Japan, August 23-29, 1997.

[Breuker & Van de Velde, 1994] J. Breuker and W. Van de Velde (eds.): *The CommonKADS Library for Expertise Modelling*, IOS Press, Amsterdam, The Netherlands, 1994.

[Bylander, 1991] T. Bylander: Complexity Results for Planning. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91)*, Sydney, Australia, August 1991.

[Bylander & Chandrasekaran, 1988] T. Bylander and B. Chandrasekaran: Generic Tasks in Knowledge-Based Reasoning. The Right Level of Abstraction for Knowledge Acquisition. In B. Gaines et al. (eds.): *Knowledge Acquisition for Knowledge-Based Systems*, vol I, pp. 65—77, Academic Press, London, 1988.

[Bylander et al., 1991] T. Bylander, D. Allemang, M. C. Tanner, and J. R. Josephson: The Computational Complexity of Abduction, *Artificial Intelligence*, 49, pages 25—60, 1991.

[Chandrasekaran, 1986] B. Chandrasekaran: Generic Tasks in Knowledge-based Reasoning: High-level Building Blocks for Expert System Design. *IEEE Expert*, 1(3): 23—30, 1986.

[Chandrasekaran, 1991] B. Chandrasekaran: Models versus rules, deep versus compiled, content

versus form, *IEEE-Expert*, 6(2): 75--79.

[Chandrasekaran et al., 1992] B. Chandrasekaran, T.R. Johnson, and J. W. Smith: Task Structure Analysis for Knowledge Modeling, *Communications of the ACM*, 35(9): 124—137, 1992.

[Clancey, 1985] W.J. Clancey: Heuristic Classification, *Artificial Intelligence*, 27:289—350, 1985.

[Console & Torasso, 1992] L. Console and P. Torasso: A Spectrum of Logical Definitions of Model-Based Diagnosis. In W. Hamscher et al. (eds.), *Readings in Model-based Diagnosis*, Morgan Kaufman Publ., San Mateo, CA, 1992.

[Cox & Pietrzykowski, 1986] P. T. Cox and T. Pietrzykowski: Causes of Events: Their Computation and Application. In *Proceedings of the 8th International Conference on Automated Deduction*, Oxford, England, July 27 - August 1, LNCS 230, Springer-Verlag, 1986.

[Davis, 1984] R. Davis: Diagnostic Reasoning Based on Structure and Behavior, *Artificial Intelligence*, 24: 347-410, 1984.

[Davis & Hamscher, 1988] R. Davis and W. Hamscher: Model-based Reasoning: Troubleshooting. In H. E. Shrobe (ed.), *Exploring AI: Survey Talks from the National Conference on AI*, Morgen Kaufman, San Mateo, CA, 1988.

[Dijkstra, 1975] E. W. Dijkstra: Guarded Commands, Nondeterminacy, and Formal Derivation of Programs, *Communication of the ACM*, 18:453-457, 1975.

[Engels, 1996] R. Engels: Planning Tasks for Knowledge Discovery in Databases; Performing Task-Oriented User-Guidance. In *Proceedings of the 2nd International Conference on Knowledge Discovery in Databases (KDD-96)*, 1996.

[Engels et al., submitted] R. Engels, G. Lindner, and R. Studer: Process Guidance for Knowledge Discovery, submitted.

[Eriksson et al., 1995] H. Eriksson, Y. Shahar, S. W. Tu, A. R. Puerta, and M. A. Musen: Task Modeling with Reusable Problem-Solving Methods, *Artificial Intelligence*, 79(2):293—326, 1995.

[Fensel, 1995a] D. Fensel: Assumptions and Limitations of a Problem-Solving Method: A Case Study. In *Proceedings of the 9th Banff Knowledge Acquisition for Knowledge-Based System Workshop (KAW´95)*, Banff, Canada, January 26 - February 3, 1995.

[Fensel, 1995c] D. Fensel: Formal Specification Languages in Knowledge and Software Engineering, *The Knowledge Engineering Review*, 10(4), 1995.

[Fensel, 1997a] D. Fensel: An Ontology-based Broker: Making Problem-Solving Method Reuse Work. In *Proceeedings of the Workshop on Problem-Solving Methods for Knowledge-based Systems (W26)* of the *Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97)*, NAGOYA, Japan, August 23-29, 1997.

[Fensel, 1997b] D. Fensel: The Tower-of-Adapter Method for Developing and Reusing Problem-Solving Methods. In E. Plaza et al. (eds.), *Knowledge Acquisition, Modeling and Management*, Lecture Notes in Artificial Intelligence (LNAI), 1319, Springer-Verlag, 1997.

[Fensel & Benjamins, 1996] D. Fensel and R. Benjamins: Assumptions in Model-Based Diagnosis. In *Proceedings of the 10th Banff Knowledge Acquisition for Knowledge-Based System Workshop (KAW´95)*, Banff, Canada, November 9 - 14, 1996.

[Fensel et al., 1996] D. Fensel, A. Schönegge, R. Groenboom, and B. Wielinga: Specification and Verification of Knowledge-Based Systems. In *Proceedings of the 10th Banff Knowledge Acquisition for Knowledge-Based System Workshop (KAW´96)*, Banff, Canada, November 9-14, 1996.

[Fensel et al., 1997] D. Fensel, E. Motta, S. Decker, and Z. Zdrahal: Using Ontologies For Defining Tasks, Problem-Solving Methods and Their Mappings. In E. Plaza et al. (eds.), *Knowledge Acquisition, Modeling and Management*, Lecture Notes in Artificial Intelligence (LNAI), 1319,

Springer-Verlag, 1997.

[Fensel & Groenboom, 1997] D. Fensel and R. Groenboom: Specifying Knowledge-Based Systems with Reusable Components. In *Proceedings of the 9th International Conference on Software Engineering & Knowledge Engineering (SEKE-97)*, Madrid, Spain, June 18-20, 1997.

[Fensel & Schönegge, 1997a] D. Fensel and A. Schönegge: Assumption Hunting as Development Method for Knowledge-Based Systems. In *Proceeedings of the Workshop on Problem-Solving Methods for Knowledge-based Systems at the 15th International Joint Conference on AI (IJCAI-97)*, Nagoya, Japan, August 23, 1997.

[Fensel & Schönegge, 1997b] D. Fensel and A. Schönegge: Specifying and Verifying Knowledge-Based Systems with KIV. In Proceedings of the *12th IEEE International Conference on Automated Software Engineering (ASEC-97)*, Incline Village, Nevada, November 1997.

[Fensel & Schönegge, submitted] D. Fensel and A. Schönegge: Methods to Solve the Context Dependency Problem of Problem-Solving Methods, submitted (available via http:// www.aifb.uni-karlsruhe.de/~dfe.

[Fensel & Straatman, to appear] D. Fensel und R. Straatman: The Essence of Problem-Solving Methods: Making Assumptions to Gain Efficiency, to appear in *The International Journal of Human Computer Studies (IJHCS)*.

[Fensel et al., to appear] D. Fensel, R. Groenboom, and G. R. Renardel de Lavalette: MCL: Specifying the Reasoning of Knowledge-based Systems, to appear in *Data and Knowledge Engineering (DKE)*.

[Garlan and D. Perry, 1995] D. Garlan and D. Perry (eds.), Special Issue on Software Architecture, *IEEE Transactions onSoftware Engineering*, 21(4), 1995.

[Genesereth, 1984] M. R. Genesereth: The Use of Design Descriptions in Automated Diagnosis, *Artificial Intelligence (AI)*, 24:411-436, 1984.

[Goel et al., 1987] A. Goel, N. Soundararajan, and B. Chandrasekaran: Complexity in Classificatory Reasoning. In *Proceedings of the 6th National Conference on Artificial Intelligence (AAAI-87)*, Seattle, Washington, July 13-17, 1987.

[O´Hara & Shadbolt, 1996] K. O'Hara and N. Shadbolt: The Thin End of the Wedge: Efficiency and the Generalized Directive Model Methodology. In N. Shadbolt (eds.), *Advances in Knowledge Acquisition*, LNAI 1076, Springer-Verlag, Berlin, 1996.

[Harel, 1984] D. Harel: Dynamic Logic. In D. Gabby et al. (eds.), *Handbook of Philosophical Logic, vol. II*, Extensions of Classical Logic, Publishing Company, Dordrecht (NL), 1984.

[de Kleer, 1986] J. de Kleer: An Assumption-based TMS, *Artificial Intelligence*, 28, 1986.

[de Kleer & Brown, 1984] J. de Kleer and J. S. Brown: A Qualitative Physics Based on Confluences, *Artificial Intelligence*, 24:7-83, 1984.

[de Kleer et al., 1992] J. de Kleer, A. K. Mackworth, and R. Reiter: Characterizing Diagnoses and Systems, *Artificial Intelligence*, 56, 1992.

[de Kleer & Williams, 1987] J. de Kleer and B. C. Williams: Diagnosing Multiple Faults, *Artificial Intelligence*, 32:97-130, 1987.

[de Kleer & Williams, 1989] J. de Kleer and B. C. Williams: Diagnosis with Behavioral Modes. In *Proceedings of the 11th International Joint Conference on AI (IJCAI-89)*, Detroit, MI, 1989.

[Lloyd, 1987] J. W. Lloyd: Declarative Error Diagnosis, *New Generation Computing*, 5:133—154, 1987.

[Lydiard, 1992] T. J. Lydiard: Overview of Current Practice and Research Initiatives for the Verification and Validation of KBS, *The Knowledge Engineering Review*, 7(2):101—113, 1992.

[Marcus, 1988] S. Marcus (ed.). *Automating Knowledge Acquisition for Experts Systems*, Kluwer

Academic Publisher, Boston, 1988.

[Marcus et al., 1988] S. Marcus, J. Stout, and J. McDermott VT: An Expert Elevator Designer That Uses Knowledge-based Backtracking, *AI Magazine*, 9(1):95—111, 1988.

[Mark, 1996] B. Mark: Special Issue on Data-Mining, *IEEE-Expert*, 11(5), 1996.

[McCluskey, 1956] E. J. McCluskey: Minimizing of Boolean Functions, *Bell Systems Technology Journal*, 35(5):1417-1444, 1956.

[McIlraith, 1994] S. McIlraith: Further Contribution to Characterizing Diagnosis, *Annals of Mathematics and AI*, special issues on model-based diagnosis, 11(1-4), 1994.

[Minton, 1995] S. Minton: Quantitative Results Concerning the Utility of Explanation-Based Learning. In [Ram & Leake, 1995b].

[Minton et al., 1989] S. Minton, S. Carbonell, C. Knoblock, D. R. Kuokka, O. Etzioni, and Y. Gil: Explanation-based Learning: A Problem Solving Perspective, *Artificial Intelligence*, 40:63—118, 1989.

[Motta & Zdrahal, 1996] E. Motta and Z. Zdrahal: Parametric Design Problem Solving. In *Proceedings of the 10h Banff Knowledge Acquisition for Knowledge-Based System Workshop (KAW´96)*, Banff, Canada, November 9-15, 1996.

[Muggleton & Buntine, 1988] S. Muggleton and W. Buntine: Machine Invention of First-Order Predicates by Inverting Resolution. In *Proceedings of the 5th International Conference on Machine Learning (ICML-88)*, Michigan, US, 1988.

[Muggleton & De Raedt, 1994] S. Muggleton and L. De Raedt: Inductive Logic Programming: Theory and Methods, *Journal of Logic Programming*, 19/20:629—679, 1994.

[Nebel, 1996] B. Nebel: Artificial intelligence: A Computational Perspective. To appear in G. Brewka (ed.), *Essentials in Knowledge Representation*.

[Nejdl et al., 1995] W. Nejdl, P. Froehlich and M. Schroeder: A Formal Framework For Representing Diagnosis Strategies in Model-Based Diagnosis Systems. In *Proceedings of the 14th International Joint Conference on AI (IJCAI-95)*, Montreal, Canada, August 20-25, 1995.

[Newell, 1982] A. Newell: The Knowledge Level, *Artificial Intelligence*, 18:87—127, 1982.

[Plant & Preece, 1996] R. Plant and A. D. Preece (eds.): Special Issue on Verification and Validation, *International Journal on Human-Computer Studies*, 44, 1996.

[Preist & Welhalm, 1990] C. Preist and B. Welham: Modelling Bridge Faults fo Diagnosis in Electronic Circuits. In *Proceedings of the 1st International Workshop on Principles of Diagnosis*, Stanford, 1990.

[Puppe, 1993] F. Puppe: *Systematic Introduction to Expert Systems: Knowledge Representation and Problem-Solving Methods*, Springer-Verlag, Berlin, 1993.

[Raiman, 1989] O. Raiman: Diagnosis as a Trial. In *Proceedings of the Model Based Diagnosis International Workshop*, Paris, 1989.

[Raiman, 1992] O. Raiman: The Alibi Principle. In W. Hamscher et al. (eds.), Readings in model-based diagnosis, Morgan Kaufmann Publ., San Mateo, CA, 1992.

[Raiman et al., 1991] O. Raiman, J. de Kleer, V. Saraswat, M. Shirley: Characterizing Non-Intermittent Faults. In *Proceedings of the 9th National Conference on AI (AAAI-91)*, Anaheim, CA, July 14-19, 1991.

[Ram et al., 1995] A. Ram, M. T. Cox, and S. Narayanan: Goal-Driven Learning in Multistrategy Reasoning and Learning Systems. In [Ram & Leake, 1995b].

[Ram & Leake, 1995a] A. Ram and D. B. Leake: Learning, Goals, and Learning Goals. In [Ram & Leake, 1995b].

[Ram & Leake, 1995b] A. Ram and D. B. Leake: *Goal-Driven Learning*, The MIT Press, 1995.

[Reif, 1995] W. Reif: The KIV Approach to Software Engineering. In M. Broy and S. Jähnichen

(eds.): *Methods, Languages, and Tools for the Construction of Correct Software,* Lecture Notes in Computer Science (LNCS), no 1009, Springer-Verlag, Berlin, 1995.

[Reiter, 1987] R. Reiter: A Theory of Diagnosis from First Principles, *Artificial Intelligence*, 32:57-95,1987.

[Schreiber et al., 1993]A.T. Schreiber, B.J. Wielinga, and J. A. Breuker (eds.): *KADS: A Principled Approach to Knowledge-Based System Development, vol 11 of Knowledge-Based Systems Book Series,* Academic Press, London, 1993.

[Schreiber et al., 1994] A.T. Schreiber, B. Wielinga, J. M. Akkermans, W. Van De Velde, and R. de Hoog: CommonKADS. A Comprehensive Methodology for KBS Development, *IEEE Expert*, 9(6):28—37, 1994.

[Shapiro, 1982] E. Y. Shapiro: *Algorithimc Program Debugging*, The MIT Press, 1982.

[Shaw & Garlan, 1996] M. Shaw and D. Garlan: S*oftware Architectures. Perspectives on an Emerging Discipline*, Prentice-Hall, 1996.

[Steels, 1990] L. Steels: Components of Expertise, *AI Magazine*, 11(2), 1990.

[Sticklen et al., 1989] J. Sticklen, B. Chandrasekaran, and W.E. Bond: Applying a Functional Approach for Model Based Reasoning, Proc. of (IJCAI) Workshop on Model Based Reasoning, Detroit, 1989,

[Struss, 1992] P. Struss: Diagnosis as a Process. In W. Hamscher etal. (eds.), *Readings in Model-based Diagnosis,* Morgan Kaufman Publ., San Mateo, CA, 1992.

[Struss & Dressler, 1989] P. Struss and O. Dressler: "Physical Negation"—Integrating Fault Models into the General Diagnostic Engine. In *Proceedings of the 11th International Joint Conference on AI (IJCAI-89)*, Detroit, MI, 1989.

[ten Teije & van Harmelen, 1996] A. ten Teije and F. Van Harmelen: Using reflection techniques for flexible problem solving (with examples from diagnosis), *Future Generation Computer Systems*, 12:217—234, 1996.

[Terpstra et al., 1993] P. Terpstra, G. van Heijst, B. Wielinga, and N. Shadbolt: Knowledge Acquisition Support Through Generalised Directive Models. In M. David et al. (eds.): *Second Generation Expert Systems*, Springer-Verlag, 1993.

[Van de Velde, 1988] W. van de Velde: Inference Structure as a Basis for Problem Solving. In *Proceedings of the 8th European Conference on Artificial Intelligence (ECAI-88)*, Munich, August 1-5, 1988.

[van Heijst et al., 1992] G. van Heijst, P. Terpstra, B. J. Wielinga and N. Shadbolt: Using Generalised Directive Models in Knowledge Acquisition. In T. Wetter et al. (eds.), *Current Developments in Knowledge Acquisition*, LNAI, Springer-Verlag, 1992.

[Wielinga et al., 1995] B. Wielinga, J. M: Akkermans, and A. TH. Schreiber: A Formal Analysis of Parametric Design Problem Solving. In *Proceedings of the 9th Banff Knowledge Acquisition for Knowledge-Based System Workshop (KAW´95)*, Banff, Canada, January 26 - February 3, 1995.