Consistency-Based Problem Solving for Environmental Decision Support

Ulrich Heller*

*Model-Based Systems and Qualitative Reasoning Group of the Technical University of Munich,
Orleansstrasse 34, 81667 München, Germany*

&

Peter Struss

*Model-Based Systems and Qualitative Reasoning Group of the Technical University of Munich,
Orleansstrasse 34, 81667 München, Germany
OCC'M Software GmbH, Gleißentalstrasse 22, 82041 Deisenhofen, Germany*

Abstract: *Complex systems do not always behave as we would like them to. With the complexity of the system, be it a water treatment plant, an ecological system or a technical device, the tasks of situation assessment (finding out what is the actual state of the system) and therapy recognition (finding out what can be done to influence it in a desirable direction again) require more and more complex reasoning. This paper proposes a general approach to computational support for these tasks, namely consistency-based problem solving. Building upon research in model-based systems and, more specifically, consistency-based diagnosis, we have developed a revision and generalization of traditional (component-oriented) theories and techniques of diagnosis from first principles. Our approach is both more general in terms of the class of problems to be addressed and more specific by proposing and exploiting a structured representation of system and domain knowledge. A motivating example from the domain of water treatment will facilitate the presentation of the theory of consistency-based problem solving and the description of an implemented reasoning system, the Generalized Diagnosis Engine, G^+DE, that performs this kind of reasoning for a wide range of problem domains.*

To whom correspondence should be addressed. E-mail: heller@in.tum.de.

1 INTRODUCTION: CONSISTENCY-BASED PROBLEM SOLVING

With increasing understanding of the complexity of environmental systems and increasing human intervention and exploitation, we are confronted with increasingly difficult environmental problems. Ecosystems do not always behave as we would like them to. Things "go wrong."

Taking the perspective of natural resources management and adopting the goal to protect a certain balance of natural processes, which is also the precondition of human life, we have to employ diagnostic reasoning: we have to *assess the situation*, potentially finding hidden causes for unexpected behavior, so that we can *find therapies*, which might involve complex interventions.

On an abstract level, this is quite similar to problem solving in the technical domain, for example, for managers of chemical plants or technicians in an automobile workshop. Indeed, computer support for diagnostic tasks has been developed and refined, mostly tailored for dealing with technical devices, and it appears desirable to extend

its scope in order to produce useful tools for environmental decision support systems.

Powerful support to situation assessment and therapy recognition requires exploiting the available knowledge about the respective domain of systems. This means a computer system has to represent this knowledge in a formal way and make it the subject of manipulation by an inference system. *Model-based systems* are an approach, where both tasks mentioned above are viewed as a *search for a suitable model* of the respective physical system:

- *Situation assessment* aims at a model that explains the available *observations* about the system's behavior.
- *Therapy recognition* tries to identify a model of the system under therapy that is compliant with what is considered a *desired behavior* or, at least, developing in the right direction. Besides, this model is to be "reachable" in a sense to be defined below.

For performing this search in a systematic way, we take the following perspective: Our knowledge about the respective domain is a collection of knowledge fragments that concern the ontology (objects in this domain, their possible relationships and interactions) and the phenomena that can occur (human interventions and natural processes, their conditions and effects). Combinations of these fragments span the space for the search. A point in this space represents a certain model hypothesis, and this needs to be checked for compliance with a set of criteria that are external to the model: they have to agree with the given observations or the goal the system is supposed to accomplish. This means we need a formal, usually mathematical, model and an algorithm that uses this model to derive more information about the behavior of the system and to check this against the respective criteria. In this paper, we will focus on the consistency-based paradigm, that is, finding a model that is consistent with given criterion. It is worth mentioning that there exist approaches to abductive diagnostic reasoning, aiming at solutions with higher explanatory power. (For a discussion of the differences and intermediate forms see Console and Torasso, 1991.)

Thus, we need a software component that performs *model-based prediction and consistency checking* (Figure 1). The basic question is then which model hypotheses are to be checked in this way. Since the search space is usually large, they are not generated by arbitrary guesses, and do not have to be. Very often, we have a reasonable starting point for the search which is assumed to be not too far away from the solution.

In situation assessment, we have certain observations about the actual system, the objects that constitute it, their configuration, and measurements about certain quantities, and we also have certain expectations about what usually holds for the system, at least in the undisturbed case. The disturbance has changed the state and potentially the
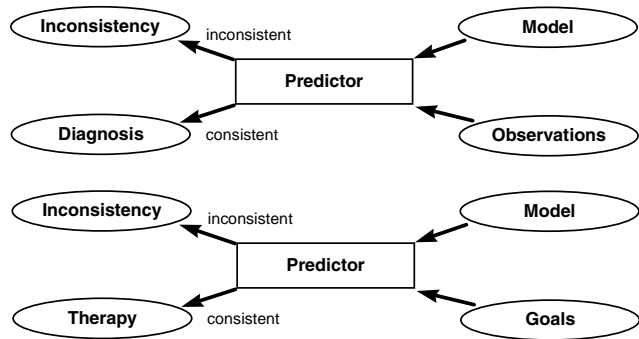

Fig. 1. Prediction and consistency check for diagnosis (top) and therapy recognition (bottom).

structure of the system, but not in an arbitrary and global way. This is why we expect that we can establish a model of the disturbed system by applying a *restricted set of local modifications* to the initial hypothesis. The candidates for such modifications are selected from the representation of our domain knowledge, which lists what can happen to the system (e.g., component failures in technical diagnosis, diseases in the medical domain, toxic substances, biological and chemical processes in ecology).

For therapy recognition, the natural starting point is the disturbed system, and performing the real therapy task will only be feasible if it can be carried out by a limited number of interventions that take the system to a healthy state. Therefore, the two tasks can be seen as an integral whole, with the result of situation assessment being the starting point for therapy recognition. In both tasks, we are looking for models of a system that emerge from an initial one as *minimal revisions*. As a result, a second software component is needed for model-based problem solving: *model revision*, which generates revised model hypotheses from those found inconsistent with the respective criteria using a library or *domain theory* (i.e., a repository of potential model fragments), which has to include possible disturbances and interventions.

This establishes the overall process of a so-called consistency-based problem solver as depicted in Figure 2. At this stage, we have not imposed particular restrictions on the form and content of the models or the predictor. However, a closer look reveals certain requirements.

First, we have to reflect the nature of model hypotheses and their generation by revision: in human reasoning and in model-based systems, it involves concepts of the physical systems and principles in the domain, such as types of objects and their properties, the structure of systems, faults, diseases, and human actions. In other words, the resulting model hypothesis is stated at a *conceptual* level, while the model required for prediction and consistency checking has to be a *mathematical behavior model* stated in terms of (differential) equations, constraints or some other
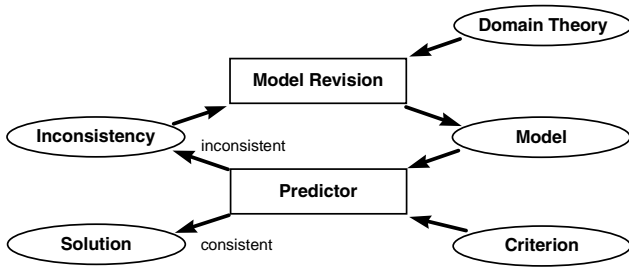
Fig. 2. The basic cycle of a consistency-based problem solver.

formalism—model revision and prediction happen at different modeling levels (see Figure 3).

Bridging this gap is usually done manually, but has to be automated in order to perform an automated model search. This sets the requirement for another software component: a *model composer*, which automatically generates an executable mathematical model from a conceptual specification of a physical system using the domain theory. (Figure 4, a general treatment of automated model composition, can be found in Nayak [1995], for instance.)

This establishes a strong requirement on the modeling system:

- *Conceptual modeling*: The model fragments in the domain theory need an explicit representation of the conceptual modeling level.
- *Compositional modeling*: The fragments of the mathematical behavior model need an associated conceptual description of the conditions under which they are valid and have to be included in the model, which forms the basis for an algorithmic solution to the composition of the overall behavior model.

Another requirement emerges from both the nature of the task and the kind of physical systems we consider in this paper. First, the notion of a disturbance implies a *significant* deviation from what is considered normal or desirable, rather than a small numerical difference in certain quan-
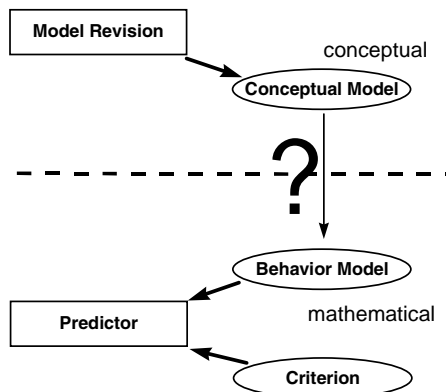

Fig. 3. The gap between the revision of the model at conceptual level and prediction based on a mathematical model.
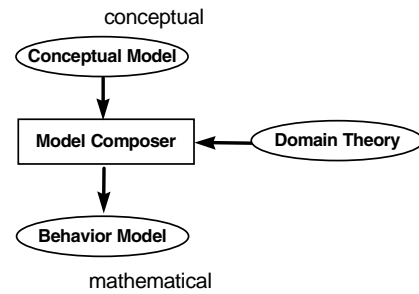

Fig. 4. Bridging the gap by automated model composition.

tities. Second, this is even more apparent for natural and, in particular, ecological systems whose state can often be characterized only by partial information and in qualitative terms, one of the reasons being their spatial and temporal extension, which does not allow for complete information. Third, our knowledge about the relevant phenomena and processes is restricted and lacking numerical precision. This results in the necessity for *qualitative modeling*, which allows us to perform prediction and consistency checking even with only partial specification of models, observations, and goals.

Artificial intelligence, and more specifically, the research areas of qualitative modeling and model-based diagnosis have produced theoretical foundations, methods, and software systems that address the goals and requirements of the approach to environmental decision support as outlined above. However, at the current stage, both the theory and practical solutions are confined to the special case of finding component faults in artifacts with a fixed structure. While this provides some coverage for many industrial applications, it is too restrictive for others, like process industries, and even more when natural systems are the subject or part of it, as in water treatment. In this paper, we present an extension to the theory of consistency-based diagnosis and therapy recognition and G$^+$DE, the Generalized Diagnosis Engine, an implementation of this theory, that provide solutions to these problem domains.

In the next section, a motivating example is used to illustrate how situation assessment and therapy recognition can be applied to a simplified real scenario from the water treatment domain. In Section 3, the fundamental theories and algorithmic implementations of consistency-based diagnosis for the case of technical devices are presented, as well as their inherent limitations (and inadequacy for the example scenario). Section 4 introduces a more general modeling approach, which is a common abstraction of both component-oriented and process-oriented paradigms. Section 5 shortly characterizes the class of solution we would like to obtain, and Section 6 gives an overview of an implemented reasoning system that is able to do so for a large class of models. Finally, some related work, limitations, and perspectives are discussed.
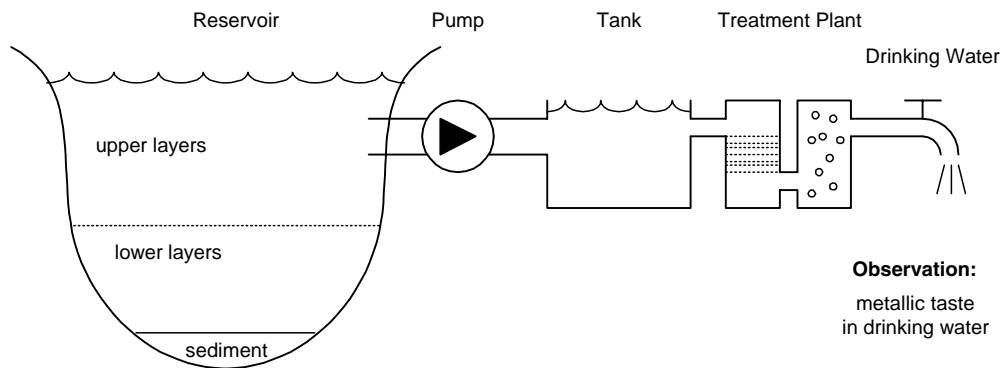
Fig. 5. A simple water treatment scenario.

2 AN EXAMPLE FROM THE WATER TREATMENT DOMAIN

In the following, we present a simplified scenario from a collaborative project in the domain of hydro-ecology and water treatment carried out with partners in Porto Alegre, Brazil. The municipal department of water and sewage in Porto Alegre, DMAE, faces challenges in securing the city's water supply from lakes and reservoirs threatened by an increasing load of organic pollution.

The destabilized ecological equilibrium of the small Lomba do Sabão leads to unexpected effects for drinking water generated from raw water captured there: In hot summer days, a distinctly unpleasant metallic taste was observed in the processed drinking water. Analysis of water samples confirmed a high concentration of dissolved iron—above legal and tolerable levels. However, there was no known source of iron—neither in the treatment plant (corroded pipes could be ruled out) nor in the ecosystem itself. The situation is shown in a simple diagram in Figure 5.

Discussions between environmental experts resulted in a surprising hypothesis emerging as the most likely explanation: There is a high amount of solid iron in the sediment of the reservoir, which was unknown so far. The pH of the bottom layers (hypolimnion) of the Lomba do Sabão typically

lies in a medium range, and this has almost completely prevented the redissolving of iron into the water body. Suddenly, however, the pH has been significantly lowered, the most likely cause being a local algal bloom triggered by excessive accumulations of nutrients. The unexpected acidic conditions that affected the lower water layers have activated a chemical redissolving process, thus enriching the water with high concentrations of dissolved iron. The iron, ascending to the surface layers, was captured with the raw water intake, and the treatment plant was unable to handle the unexpected high concentrations of iron received and an excessive amount remained in the drinking water, being perceived as an undesirable taste.

After confirming this situation assessment (depicted in Figure 6), experts discussed possible countermeasures in order to ensure a clean water supply. Certainly, removal of the newly discovered sedimental iron was not an option. Viable alternatives included adding an oxidation agent in the early stages of the treatment process, so that dissolved iron could be removed from the captured water. Also, the application of calcium carbonate as a means of raising the pH of the water body by artificial alkalization was considered. For future occurrences of local algal blooms, algaecides were discussed as a preventive measure, but mainly existing long-term plans for the prevention of eutrophica-
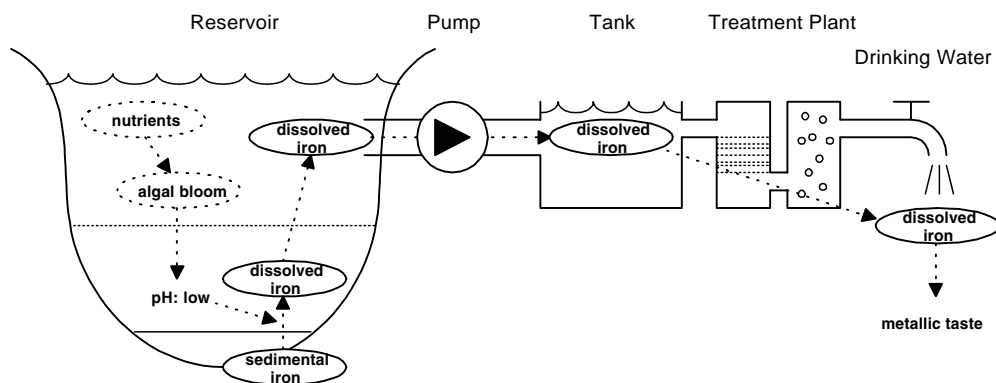

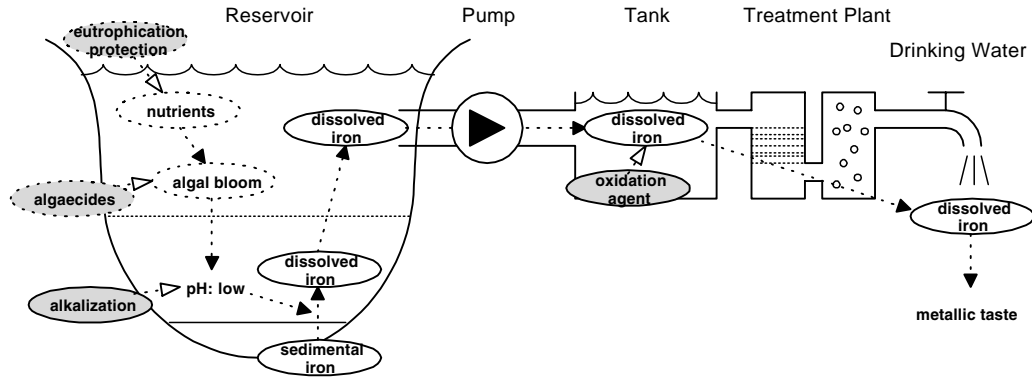**Fig. 6.** Situation assessment in the example scenario.

Fig. 7. Available therapy options for the scenario.

tion gained additional support. In the end, the application of an oxidation agent seemed to provide both minimum side effects and the advantage of immediate relief. All alternatives are shown in Figure 7.

When designing computational support for the discussed problem, successful applications of consistency-based problem solving in component-oriented diagnosis come to mind. Historically, important theoretical foundations and efficient algorithms have been developed within this restricted problem domain. As our approach to generalizing both the theory and the implementations builds upon fundamental work in this area, we will give a short overview of consistency-based diagnosis in the next section.

3 CONSISTENCY-BASED DIAGNOSIS

3.1 The theory

In Section 1, we introduced an approach to model-based situation assessment and therapy recognition in an informal way. Now, we will formulate this approach, consistency-based problem solving, in a formal way and characterize the work that has been carried out in this area so far, in terms of theory and implementation. The goal of this step is to identify the utility and the limitations of the state of the art with respect to the requirements in our domain.

If we consider our model of the system under consideration (this model might feature a complex inner structure as presented in Section 4) as a logical theory, MODEL, then a solution to the tasks involves three basic procedures:

- *Behavior prediction*, i.e., deriving conclusions about variables from the behavior model:

$$\text{MODEL} \vdash ?$$

- *Consistency check*, i.e., identifying whether or not the model contradicts a certain criterion:

$$\text{MODEL}_1 \cup \text{CRITERION} \vdash^? \bot$$

- *Model revision*, i.e., moving from an inconsistent model to a consistent one:

$$\text{MODEL}_1 \cup \text{CRITERION} \vdash \bot$$
$$\Rightarrow \text{MODEL}_2 \cup \text{CRITERION} \nvdash \bot .$$

The abstract CRITERION varies during problem solving: it is given by observations when situation assessment is performed and by goals when the therapy is to be determined. As pointed out in the introduction, the sources (and the selection criteria) for the revision step are quite different, and it suggests itself to cascade the two tasks of situation assessment and therapy recognition.

Most of the previous work has addressed the task of consistency-based diagnosis (Dressler and Struss, 1996) from the following perspective:

- The entities relevant to diagnosis are *components* $C_i \in$ COMPS, which can be associated with a set of different behavior modes $\{\text{mode}_i(C_i)\}$: the correct one, $\text{ok}(C_i)$, and at least one *fault mode* (possibly with unspecified behavior).
- A system to be diagnosed consists of a given set of such components that interact in a way determined by the *fixed structure* of the system (its blueprint) and are to be scrutinized for faulty behavior.
- The result of the diagnosis is *an assignment of actual behavior modes* to all these components.
- The criterion for a proper diagnosis candidate is that the respective mode assignment *is consistent* with a set of observations OBS of the actual system behavior.

If we summarize the domain theory and the description of the structure of the particular system and its parameters as the *system description* SD as in Reiter (1987), then diagnosis starts from a point where assuming that all components behave correctly contradicts the observations and the system description:

$$\text{SD} \cup \text{OBS} \cup \{\text{ok}(C_i) \mid C_i \in \text{COMPS}\} \vdash \bot .$$

The diagnosis procedure is then organized as a search for revised mode assignments to the components that eliminate inconsistency:

$$SD \cup OBS \cup \{mode_i(C_i)|C_i \in COMPS\} \nvdash \perp.$$

This is the basis for implemented systems that have started to become part of industrial applications in various domains, for instance, cars (Sachenbacher et al., 1998), gas turbines (Travé-Massuyès and Milne, 1997), and space crafts (Pell et al., 1996). Researchers in model-based diagnosis have also tackled problems in less rigid domains like software debugging and configuration (e.g., Crow and Rushby, 1991; Stumptner and Wotawa, 1998). It is worthwhile exploring whether these systems could form the basis for applications in areas like water treatment and environmental decision support.

3.2 Revisiting the example

However, if we revisit the example introduced in the previous section, we notice that diagnosis of devices made up of components in a fixed structure is much simpler than solving problems in water treatment plants or ecological systems. Although components such as pumps, valves, and pipes are involved in water treatment, it is not their malfunctioning that is causing the problem. Also it is not the case that any of the chemical, mechanical, or biological processes involved do not perform well. We summarize in what respect the classical theories and systems of consistency-based diagnosis are too narrow and, as a result, fail to provide a solution to many diagnostic problems in the environmental domain, but also in technical applications (e.g., process industries):

- We could not call iron redissolving or algal blooms a fault. Natural processes do not break or fail like components. This means: The relevant constituents of the system *do not necessarily have fault modes.*
- It is not the case that one of the constituents of our original system description can be blamed for the inconsistency with the observations. The reason is *additional, unanticipated* constituents, namely sedimental iron and the algal bloom, which we were not aware of. This means: A revision of the system *description cannot be confined to a set of given constituents* (components).
- Also, for finding an appropriate treatment, changing the "mode" (or existence) of the given constituents (e.g., by replacement of a broken component) is not the issue. One has to find actions that, again, *expand the entire system* (e.g., by introducing algicides or an oxidation agent).
- There are no "failures of nature." Metallic taste is not a fault, even though we might want to avoid it. A certain phenomenon may be perfectly consistent with the observations, while inconsistencies arise *only with our*

goals and intentions. This means: The classical diagnosis task, based on inconsistencies between the model and the observations, is to be explicitly split into situation assessment (using the observations) and therapy recognition (taking the goals into account).

As a consequence, a more general theory of consistency-based problem solving is needed. We attempt to contribute to this goal by proposing a revision and extension to consistency-based diagnosis that preserves the principled approach while expanding the scope of the underlying modeling paradigms and diagnosis tasks and algorithms. At the same time, we would like to re-use as much as possible from advanced consistency-based diagnosis techniques, so the next subsection is dedicated to these.

3.3 Conflict-driven and dependency-based diagnosis

A closer look at the existing algorithms and implementations reveals what we can expect to preserve and exploit in our generalized approach. Obviously, the central question to answer in model revision is what to revise if a particular model is found inconsistent. As long as the only result is that the entire model is inconsistent, revisions appear quite arbitrary. However, usually it is possible to localize the source of an inconsistency to some degree. If the starter of the car does not work, we do not suspect the windshield wipers, because they do not interfere with this function, and in the water treatment example, potential parts of the reservoir that do not contribute to the water captured for drinking water supply do not have to be inspected. This means that the starting point for revisions will often be a subset of the entire model. In component-oriented diagnosis, the system identifies sets of models of correctly working components that together are inconsistent with the observations, so-called *conflicts*. Such a conflict states that at least one of the involved components is not in accordance with its model of correct behavior. Hence, every valid diagnosis has to revise correctness assumptions for one component out of each known conflict. For instance, if two conflicts

$$\{\neg(ok(C_1)), \neg(ok(C_2))\}, \{\neg(ok(C_1)), \neg(ok(C_3))\}$$

have been obtained, then two diagnoses can be obtained that are minimal (w. r. t. set inclusion):

$$\{\neg(ok(C_1))\}, \{\neg(ok(C_2)), \neg(ok(C_3))\}.$$

They specify two revisions of the model by switching the assignment of the behavior mode to components. This way, the revision works in a focused fashion, driven by conflicts (Figure 8).

If we combine model revision with the elements to a solution as outlined in the introduction, namely prediction/consistency check and model composition, we obtain what is displayed in Figure 9. Again, this reveals a gap:
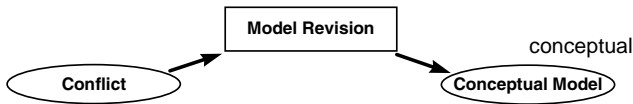
Fig. 8. Conflict-driven model revision.

while the contradiction shows up in the *mathematical* model, the model revision step requires conflicts (i.e., statements about the *conceptual* view of the system), for example, in terms of component behavior modes.

The intuitive solution is that the system has to keep track of what conceptual model units underlie a detected discrepancy. The conceptual units were associated with behavior model fragments in the composition step, and obviously there is no chance to obtain conflicts unless these associations are recorded. Secondly, the system has to record the dependencies of the discrepancy on the various predictions that contributed to it which in turn are derived from behavior model fragments. This way, the problem solver will be able to identify the conceptual units that gave rise to a discrepancy, that is, a conflict (Figure 10).

In many systems, both tasks, dependency recording and conflict generation, are performed by a general purpose tool, called the assumption-based truth-maintenance system (ATMS; de Kleer, 1986) or variants of it, as described in de Kleer and Williams (1987), Struss and Dressler (1989), Dressler and Struss (1996). This architecture is generally referred to as the general diagnostic engine (GDE). There are, of course, a number of diagnosis algorithms that exploit specific characteristics of the class of devices under consideration to achieve better performance (e.g., Fattah and Dechter, 1995; Stumptner and Wotawa, 1997), but we focus on the general concept here.

Identifying sets of conflicting hypotheses about the conceptual system model as the starting point for generating model revisions is the general foundation of consistency-based diagnosis. It is only its use in component-oriented diagnosis that is very special. We will build on this general foundation, which will allow us to even re-use its
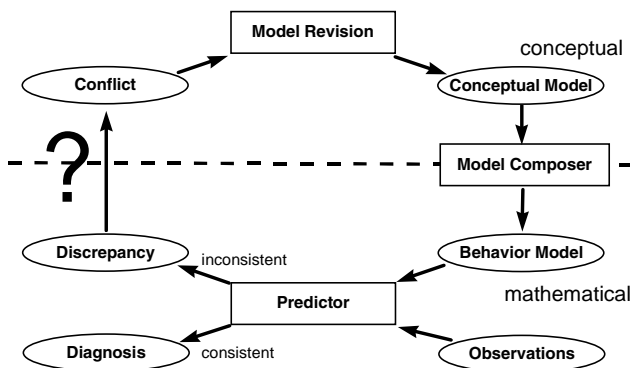

Fig. 9. The gap between discrepancies and conflicts.

implementation, namely prediction/consistency checking and conflict generation. Obviously, we need a different modeling formalism including model composition. This is presented in the next section. Since our example showed clearly that simply switching behavior modes of components does not suffice, a more general kind of model revision is required and described thereafter. Finally, a reasoning system for calculating these revisions is presented, that builds upon the mentioned GDE algorithm.

4 THE MODELING APPROACH

We follow the principles of structure-to-behavior reasoning and compositional modeling and provide a generalization of both component-based and process-based modeling paradigms. For this, we adopt a relevant part of the qualitative process theory (Forbus, 1984). For a concise formalization of our modeling approach in first-order logic, please refer to (Heller, 2001). According to this view, the system model (or system description) consists of two parts: the domain theory and a situation description. The diagram in Figure 11 provides an overview and also indicates the specificity of each section. We briefly discuss each part in the following.

4.1 Domain theory

The domain theory captures what we know about the domain, that is, all systems of a certain class (e.g., hydrological ecosystems or water treatment plants). We distinguish *structural elements* (objects and relations) from *behavior constituents* (which might be processes or other model fragments). The ontology consists of the following:

- *Object types*, which occur in structural descriptions, for instance, types of components in a device (resistor, broken wire), spatially distinguished entities (layers of a water body, pipes, tanks), etc. Object types can be structured hierarchically.
- *Relations* for characterizing "configurations" of objects. Examples are spatial relationships (contained-in, below), connectivity of components, etc. Some important properties of relations (like uniqueness) can be specified.
- *Quantities* as the basic elements for behavioral descriptions. Multiple quantity types (e.g., with different domains) can be defined and objects of a given type can be supplied with a number of associated quantities with given roles (e.g., the resistance of a resistor, the concentration of dissolved iron in a water tank, etc.).

The domain theory also has to provide a vocabulary for behavior descriptions and the inferences that derive behavioral constituents from a structural description. It introduces *behavior constituent types*. These are physical phenomena, which are considered to contribute to the behavior of the
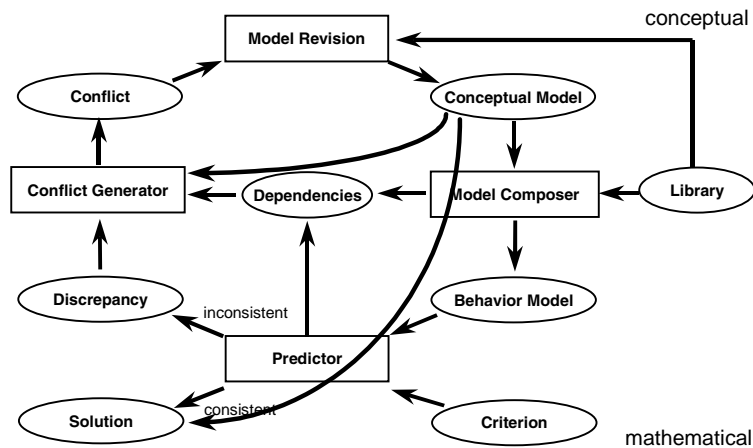
Fig. 10. The conflict generator based on dependencies.

overall system. They can represent basic component laws (Ohm's Law, logical-or) or processes like in the qualitative process theory (QPT; Forbus, 1984), or partial behavior models like in Bredeweg (1991). They occur deterministically under certain conditions, and their occurrence generates particular effects.

See the following diagram (Figure 12) for an example of a behavior constituent type definition. The visual elements are actually employed by the graphical user interface of the G$^+$DE prototype described in Section 6.

The diagram elements have to be interpreted as follows:

- Structural Conditions: assertions about the existence of relations and objects. In the example: solid iron has to be present ("fixed-in") the sediment, which is a solid layer below a water layer. The boxes with the double frames

Fig. 11. Proposed structure of system models.

denote object templates (i.e., a role name and an object type). The ellipses represent relation templates.
- Quantity Conditions: statements about values of quantities. In the example: the pH of the water layer has to be below a certain threshold (upper right corner, the hexagon contains a constraint designator "LESS-THAN-x") and the concentration of solid iron in the sediment has to be positive (lower left corner).
- Structural Effects: creation or possibly even elimination of objects and relations. In the example: Dissolved iron will be present in the water layer (the structural elements with the star symbol denote structural effects). But note that we encode implications ("there is also ...") rather than temporal changes ("at a later time there will be ...")—see below.
- Quantity Effects: can be expressed as restrictions on variables. In the example: The rate of the redissolving process (a helper quantity) will increase with the sedimental iron concentration and decrease with pH (here simplified in the qualitative constraint DIV). We also allow for partially specified effects in the form of influences as used in QPT. In our case, the concentration of the dissolved iron will increase proportionally to the process rate, while the sedimental iron will be depleted (small boxes with signs).

The abstract form of a behavior constituent type can then be written as

$$StructuralConditions \wedge QuantityConditions$$
$$\Rightarrow StructuralEffects \wedge QuantityEffects$$

More precisely, we state that for each constellation of objects satisfying the structural and quantity conditions, an instance of the behavior constituent is occurring and imposes the respective effects on the constellation.
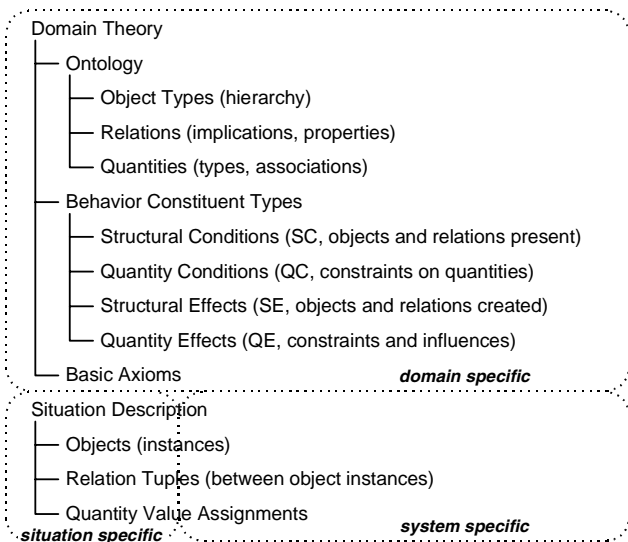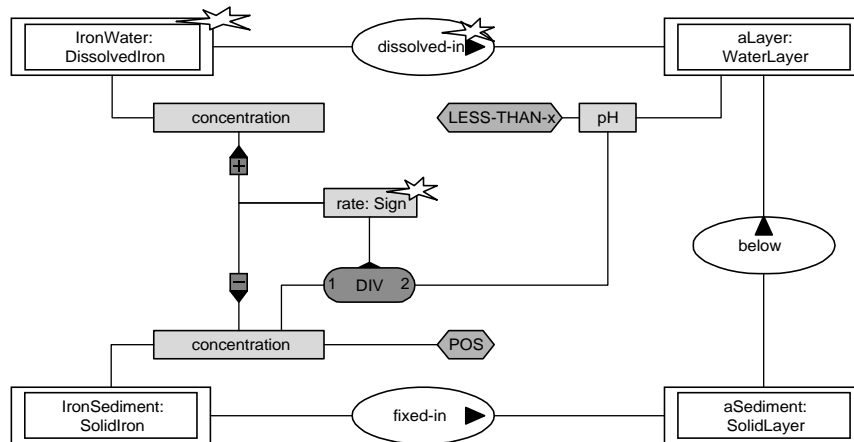
Fig. 12. Example behavior constituent type "iron redissolving."

Additionally, we include a section for the fundamental laws that determine the mechanisms of model formation, how influences combine, and prediction over time (continuity, integration, etc.). These "basic axioms" cannot be specified arbitrarily by the modeler but rather represent the domain-independent laws like the one of behavior constituent occurrence stated just above.

At this point, we make almost no commitment with respect to the quantity domains (symbolic, qualitative, real, etc.), the formalism for specifying the quantity effects (constraints, differential equations, etc.), and the expressiveness of structural conditions and effects (e.g., nonexistence of certain objects as condition or destruction of objects as a structural effect). In general, conditions and effects are assumed to be local and compositional. Some requirements for model formation and prediction will be discussed in Section 6.

4.2 Situation description

A particular system under consideration is characterized by its *object structure*, that is, instances of the object types and individual tuples of object relations (for instance, the components and the connection structure of a device). In the following, we will refer to both objects and relation tuples as structural elements.

A particular situation of the system is characterized by *quantity value assignments*. Dependent on the task and context, they may represent actual measurements (e.g., an increased amount of iron in the drinking water), specification of goals (a certain amount of iron), mere hypotheses, etc.

This way of modeling (like QPT which can be regarded as a specialization of it) allows for dynamic changes in the set of active processes and, thus, distinguishes from approaches that represent systems as a predefined sequence of processes that are then considered to possibly fail very

much like components (see e.g. Guckenbiehl et al., 1999). Unlike the QPT-based work of Collins (1993), our approach fully supports structural effects and, hence, facilitates diagnosis with respect to changes in the (object-related) structure (e.g., due to unanticipated objects or interactions).

5 CHARACTERIZING SOLUTIONS

It becomes clear that the structure of the system description, as discussed in the previous section allows for a flexibility that is far beyond a fixed circuit diagram with components that might occasionally break. Rather, the dynamics of processes suddenly taking effect and ceasing again, as well as structure being created represent a significantly higher degree of freedom.

Correspondingly, a diagnosis task does not simply aim at a mode assignment that is consistent with the observations—which can be interpreted as situation assessment and therapy at the same time, since replacing components is usually the only action available and a system with all components o.k. will behave according to the (implicit) goals associated with it.

As we have argued before, the general case requires mostly two separate steps in finding a system description that adequately represents the observed behavior (i.e., it is at least consistent with it) and then looking for one that is consistent with the (external) goals. Still, for both steps we can employ consistency-based revision techniques as presented above. But note two important differences. First, the search space for a revised system description is not a finite set of mode assignments, but rather spans structural changes (additions and removals) alongside value assignments to quantities associated with the currently valid set of structural elements. Secondly, the set of allowable revisions is different for the two tasks mentioned above: for situation assessment, structural changes have to be *plausible*,

in the sense that the revisions are acceptable explanations for the observed behavior (e.g., the introduction of sedimental iron is acceptable, while the appearance of a second reservoir is not). For therapy recognition, the criterion is whether a change is *achievable*, that is, whether it can be brought about by intentional intervention (of course, every possible action will have to be represented in the domain theory, e.g., in the form of introducible oxidation agents).

The latter issue brings us to what we consider *revisable* in the initial system structure:

- We allow for user-defined assumptions to qualify quantity assignments (such as mode assignments, which was the only "revisable" in the classical component-oriented case). Thus, they are marked *changeable*, as a different value might be determined to be the only consistent one.
- Assumptions can also be used for the existence of structural elements. These assumed elements can be called *removable*, as their existence might lead to a conflict.
- Finally, certain object types can be named as being *introducible* to allow the addition of objects of the respective type to the system model. This provides the most important means for controlling the problem-solving task, since a more restrictive set of introducibles requires "deepening" the search for causes. In the example, one could accept that dissolved iron can appear in drinking water and, thus, end up with a very simple explanation of the metallic taste. But excluding dissolved iron from the set of introducibles will trigger the search for sedimental iron, as shown in Section 2. Certainly, more sophisticated means of defining what is introducible can be imagined (such as confining where an object of a certain type can be introduced), but we leave that for future research.

On the basis of these defeasible assumptions as well as introducible elements, one can now define the set of acceptable solutions:

> Definition (Solution): A solution is a *minimal consistent allowable structure in that a maximal set of user-defined assumptions holds*. A structure is defined to be *allowable*, if it contains at least the structural elements specified by the user as nonremovable (without assumption) and all other structural elements are either introducible or a necessary consequence of an occurring behavior constituent (by ways of structural effects).

Obviously, a structure missing necessary structural effects is not consistent with the laws of behavior constituent occurrence; therefore, solutions always include all such effects.

Minimality is understood with respect to set inclusion. Note that we do not use the absolute cardinality of structural elements present in a solution as a criterion for preferring one solution over another, we just exclude "unnecessary" or "superfluous" objects from being

included in a solution. "Candidate ranking" is an entirely different story and is much more complex for the search space spanned here than it is for mode assignments. A discussion of the intricate issues involved is beyond the scope of this paper. In Heller (2001), one can also find a formalization of this characterization in default logic.

6 G⁺DE—THE GENERALIZED DIAGNOSTIC ENGINE

While there exist efficient implementations of the diagnosis task for the component-oriented case (GDE; see above), we have seen that the more general framework as specified within this paper requires a more sophisticated solution. For once, we have to bridge the gap between conceptual models and mathematical ones by model composition, and the much more general modeling language provides additional challenges as pointed out in the last section.

One of the main problems lies in the inherent nonmonotonicity, which is ultimately a consequence of using the mentioned closed-world assumption to calculate quantities, which may have effects on the activity of behavior constituents, which in turn might change quantities or even introduce additional structural elements. This makes implementations difficult and computationally expensive. However, for a large class of domain theories, we have been able to implement a reasoning system, the generalized diagnostic engine (G⁺DE), that employs monotonic reasoning and a form of constraint satisfaction as implemented by a classical GDE module.

See Figure 13 for an overview of the system architecture of G⁺DE. To the left of the diagram, editors are depicted, where the user can graphically specify the domain theory, the system structure, and quantity specifications for the system at hand. The scenario editor will also be used for feedback of the completed structure, of predicted quantity values, and for an interpretation of diagnosis candidates (see below). The following sections will discuss the most important design decisions for the modules calculating these solutions (to the right of the diagram).

6.1 Instantiation and activity

In the following, we consider the case where we have only positive structural effects, that is, objects (and relations) are only generated, but not destroyed, and only positive structural conditions (i.e., no conditions in terms of nonexistence of objects or relations). If we assume, furthermore, that we can determine the identity of any generated object (e.g., using some relation like spatial locatedness by "contained-in"), then we can monotonically construct all objects that can *possibly* be generated by any instance of a behavior constituent, without taking the quantity values into account.
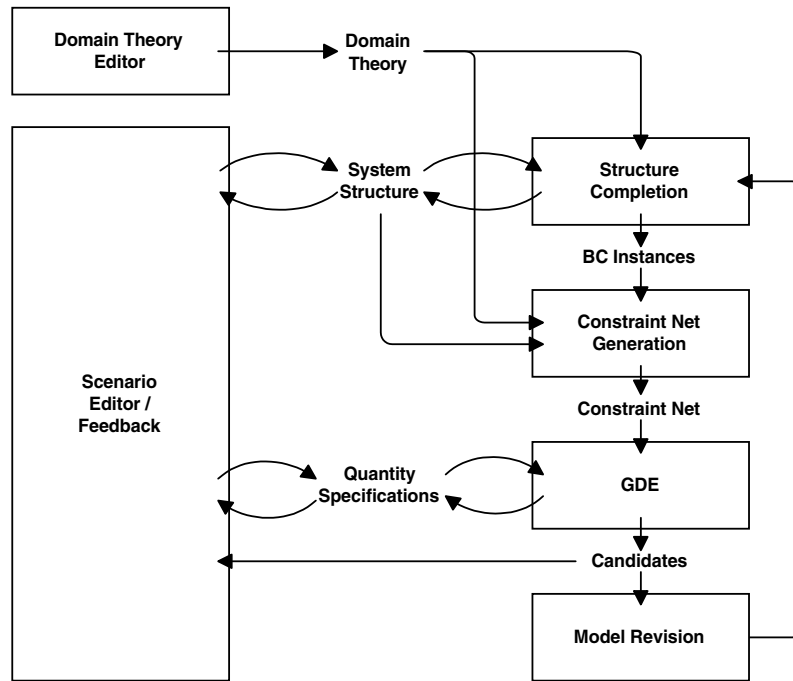
Fig. 13. Abstract architecture of G+DE.

This corresponds to the common distinction between "instantiation" and "activation," as described in the qualitative process theory (Forbus, 1984), which mentions structural effects, but does not specify exactly how they are treated. We decided to use this mechanism to separate the structural conditions and structural effects from the quantity conditions and quantity effects.

Thus, in an instantiation phase, the initially given system structure is completed by all potentially active behavior constituents and their respective structural effects (i.e., for each set of objects matching the structural conditions of a behavior constituent type, we create an instance of a behavior constituent plus all objects and relations created in the associated structural effects). This happens in the module *Structural Completion* in Figure 12.

Together with the *Constraint Net Generation*, this corresponds to the model composition mentioned earlier. The fundamental idea of instantiation versus activation is that later, when we take quantity values into account, instantiated behavior constituents can be determined to be in an active or inactive state depending on their quantity conditions. This breaks the nonmonotonic reasoning cycle described above, or rather prepares a model for a constraint satisfaction algorithm to find all solutions defined in the nonmonotonic framework.

Guard variables (representing the activity of behavior constituents) and constraints ensure that only structural effects of active behavior constituents become effective—which can in turn trigger further behavior constituents.

Thus, since structural elements that have been introduced might not actually be created by active behavior constituents, they are guarded by a boolean variable, representing whether they are actually present or not. To be precise, we also use a secondary set of guard variable for behavior constituents that indicates whether all their structural effects are actually (instead of just potentially) fulfilled. Similarly, the quantity conditions can be compiled into a set of constraints so that the activity of a behavior constituent can be made dependent of the values of specified object quantities. The quantity effects require a slightly more complex solution, since their effect is also enabled or disabled by the activity of the behavior constituent they belong to. Conditional constraints will ensure this. A simple construction with an intermediate variable will also disable all influences generated by the quantity effects.

Using the example (iron redissolving) from above, Figure 14 shows the generated fragment of the constraint network for a single instance, including all guard variables. All described constraints and influences will have to be created for each instantiated behavior constituent. Furthermore, influences (there are still some denoted in Figure 14) will have to be "resolved": For the complete model, the net effect of all specified influences can be described as a constraint, usually an additive one. Here, the assumption, that the known model is complete is actually used, and this is why we record these important "closed-world assumptions" (CWAs; see below for exact semantics). All of this
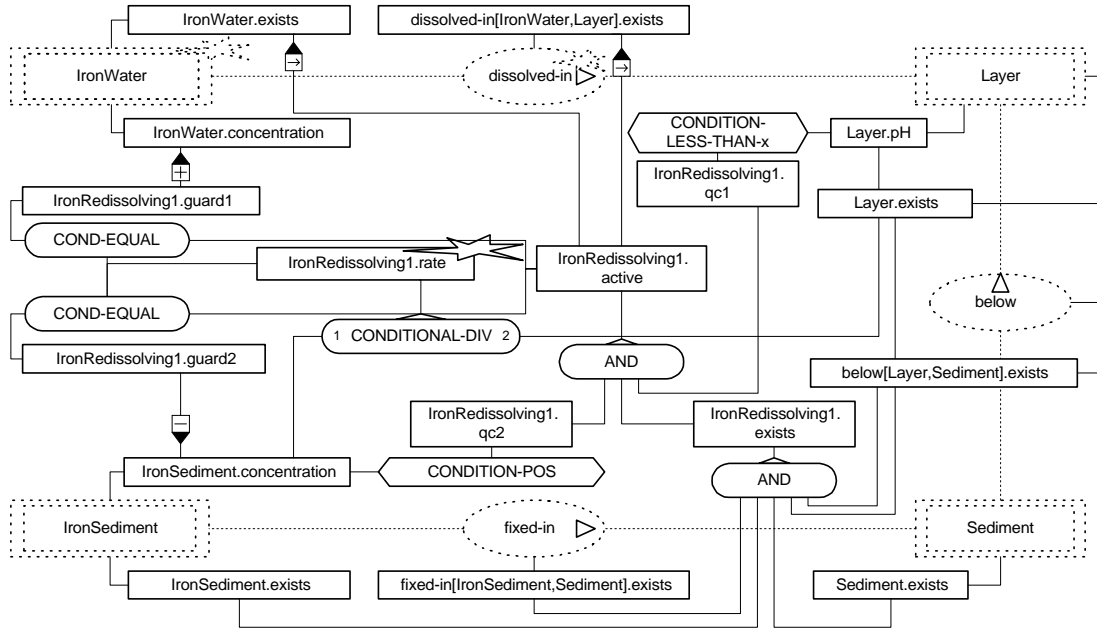
Fig. 14. Example constraint network fragment for an instance of iron redissolving.

happens in the *Constraint Net Generation* module, yielding a constraint model that can be used for prediction and consistency checking.

For this task, we employ a flexible implementation of a classical *General Diagnostic Engine*, which does not only provide predictions, but also minimal diagnostic candidates in terms of assumptions occurring in conflicts; this is a step beyond simple conflict detection, as also minimal "hitting sets" for multiple conflicts are calculated. Assumption tracking or dependency recording, as described in Section 3.3 is handled partly in model composition, as structural assumptions ("removable elements") are included in the model, but the assignment of their existence variable to "true" is labeled with the respective assumption, as are usual ("changeable") quantity assignments. Thus, all user-defined assumptions are included in the quantity specifications. Only closed-world assumptions are associated with certain constraints (from influence resolution, see above), and the predictor is equipped to record them, whenever the constraint is used, as this is part of classical consistency-based diagnosis.

Note that prediction in the general sense will now also "switch" on or off behavior constituents and make structural elements appear or disappear. This information will be made available in the form of quantity specifications as well.

Finally, the GDE module will be able to determine whether the provided model was consistent, which means that there is a solution to the given tasks within the instantiated search space. The precise extension (in terms of objects, relations, and quantity values) can be extracted from the quantity specifications.

6.2 Model revision

In all other cases, a set of *candidates* (i.e., minimal sets of assumptions) to be retracted are generated. While this is simple for the propositions concerning structure and values marked as retractable, the real challenge lies in the closed-world assumptions recorded during resolution and collected whenever a summary resolution constraint appears in a inconsistency.

We mentioned above that we use this assumption about the completeness of the known model whenever we collect all (known) influences on a given variable. So we can label the particular occurrence of the assumption with this variable. Now, putting it precisely, the conjunction of all these "local" CWAs represents the *assumption that there is a (minimal) solution within the generated search space*. Conversely, the negation of a single CWA represents the necessity for an additional influence on the given variable, which will require additional structural elements to be part of a solution.

Now, it is important to mention that we take the domain theory to be fixed and complete. This means that any phenomenon providing an additional influence will have to be described there in the form a behavior constituent. Since we also have information on the type of quantity and even the type of object it is associated with, we have a reasonably good starting position for a backward search through the domain theory.

It is the task of the *Model Revision* module to find a minimal structural augmentation that will instantiate a behavior constituent generating the required additional influence. Of course, such a hypothesis has to be tested once more by producing the complete model (including all potential side effects of the revision), and checking it against the quantity specifications. This is indicated by the backward arrow in Figure 12. There are two motivations for a loop here: On the one hand, many different revisions might have to be assessed until a consistent one is discovered. On the other hand, an inconsistent revision might simply be "not yet" complete and require further completion by extending the model.

A complicated issue is ensuring the *minimality* of augmentations: it is not easy to focus the search, so that a smaller "deeper" set of causes bringing about the same changes and, thus, removing the inconsistency, can be excluded. In any practical application, one will probably have to rely on additional (maybe heuristic) knowledge about how to select useful model revisions.

7 DISCUSSION AND PERSPECTIVES

The described approach to model-based problem solving builds upon the strong logical foundations of component-oriented consistency-based diagnosis, while overcoming its specific conditions and limitations. This should help to expand the scope of applicability of model-based support for different domains and tasks.

Collins (1993) is also aiming at extending classical diagnosis to (QPT) process models. He proposes an analogous distribution of the global closed-world assumption and is using an abductive backchaining algorithm to generate explanations implying the violation of the local closed-world assumption. But since he is not considering structural effects of processes, structural revisions are taken as primitives.

The presented G$^+$DE architecture has been implemented in a prototypical fashion. The architecture (see the overview in Figure 13) consists of a set of flexible reasoning modules with a powerful graphical user interface. We have successfully applied it to small scenarios from the water treatment domain, including the example described in Section 2, as well as a number of tasks from medical theory validation (secondary hypertension phenomena) and problems in electrical circuits that are beyond simple component-oriented diagnosis, such as thermal interaction and exploitation of structural aggregation. See Heller (2001) for an overview. Of course, a number of optimizations for G$^+$DE, especially in guiding and focussing the search for revisions, will be necessary for real-world applications.

Future work also concerns the task of therapy recognition. Currently, goals are formalized mainly in the form of "deviations" (i.e., qualitative expressions stating that the desired value of a certain quantity lies above or below the current value). When allowing only achievable actions as introducible candidates, the diagnostic algorithm can provide simple therapy proposals.

REFERENCES

Bredeweg, B. (1991), *Expertise in Qualitative Prediction of Behaviour*, Doctoral thesis, Faculty of Psychology, University of Amsterdam.

Collins, J. W. (1993), *Process-based Diagnosis: An Approach to Understanding Novel Failures*, Doctoral thesis, Institute for the Learning Sciences, Northwestern University.

Console, L. & Torasso, P. (1991), A spectrum of logical definitions of model-based diagnosis, *Computational Intelligence*, **7**(3), 133–41.

Crow, J. & Rushby, J. (1991), Model-based reconfiguration: toward an integration with diagnosis, Proceedings of AAAI-91.

de Kleer, J. (1986), An assumption-based TMS, *Artificial Intelligence* **28** (2), 127–62.

de Kleer, J. & Williams, B. C. (1987), Diagnosing multiple faults, *Artificial Intelligence*, **32**, 97–130.

Dressler, O. & Struss, P. (1996), The consistency-based approach to automated diagnosis of technical devices, In Brewka, G. (ed.), *Principles of Knowledge Representation*, CSLI Publications, Stanford, pp. 267–311.

Fattah, Y. & Dechter, R. (1995), Diagnosing tree-decomposable circuits, Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95).

Forbus, K. (1984), Qualitative process theory. *Artificial Intelligence*, **24** (1–3), 85–168.

Guckenbiehl, T., Milde, H., Neumann, B. & Struss, P. (1999), Meeting re-use requirements of real-life diagnosis applications, Puppe, F. (ed.), XPS99: Knowledge-based Systems. Lecture Notes in *Artificial Intelligence* **1570**, Springer Verlag, Berlin, pp. 90–100.

Heller, U. (2001), *Process-oriented Consistency-based Diagnosis—Theory, Implementation and Applications*, PhD thesis, Department of Computer Science, Technical University of Munich.

Nayak, P. (1995), Automated modeling of physical systems, Lecture Notes in *Computer Science*, **1003**, Springer.

Pell, B., Bernard, D., Chien, S., Gat, E., Muscettola, N., Nayak, P., Wagner, M. & Williams, B. (1996), A remote agent prototype for spacecraft autonomy, In Proceedings of the SPIE Conference on Optical Science, Engineering, and Instrumentation.

Reiter, R. (1980), A logic for default reasoning, *Artificial Intelligence*, **13**, 81–132.

Reiter, R. (1987), A theory of diagnosis from first principles, *Artificial Intelligence*, **32** (1), 57–96.

Sachenbacher, M., Malik, A. & Struss, P. (1998), From electrics to emissions: experiences in applying model-based diagnosis to real problems in real cars, Proceedings of the Ninth International Workshop on Principles of Diagnosis (DX-98), Cape Cod.

Struss, P. & Dressler, O. (1989), Physical negation: integrating fault models into the general diagnostic engine, International Joint Conference on Artificial Intelligence (IJCAI-89), Detroit.

Stumptner, M. & Wotawa, F. (1997), Diagnosing tree-structured systems, Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97).

Stumptner, M. & Wotawa, F. (1998), Debugging functional programs, Proceedings of the Ninth International Workshop on Principles of Diagnosis (DX-98), Cape Cod.

Travé-Massuyès, L. & Milne, R. (1997), Gas-turbine condition monitoring using qualitative model-based diagnosis, IEEE Expert **12** (3), 22–31.