On Classification and Modeling Issues in Distributed Model-based Diagnosis

Franz Wotawa and Ingo Pill∗

Institute for Software Technology, Graz University of Technology, Inffeldgasse 16b/II, 8010 Graz, Austria
E-mail: {wotawa,ipill}@ist.tugraz.at

With model-based diagnosis, diagnoses for occurring faults can be directly computed from a given system model and actual observations about system behavior. Model-based diagnosis has been successfully accommodated to several purposes, including the diagnosis of space probes and configuration knowledge bases. Recent research includes extensions for distributed systems, motivated by the ever-growing system complexity and inherently distributed domains like service-oriented architectures. Previous work in this context lacks however a detailed analysis and classification approach that considers essential underlying issues like diagnosis architecture, utilized models, and abstract requirements that might stem from the application domain. In this paper, we will show an analysis of distributed system diagnosis and a characterization in the three dimensions mentioned.

Keywords: model-based diagnosis, distributed diagnosis

1. INTRODUCTION

Model-based diagnosis [9,11,31] is a diagnosis approach that allows deriving possible root causes for certain misbehavior from a system’s model and actual observations. The concept of model-based diagnosis has gained a lot of attention during the past decades, and several applications have been reported. These include software debugging [24,25], as well as the diagnosis of space probes [39,30], cars [22,23], and configuration knowledge bases [12]. In “classical” model-based diagnosis, a diagnosis algorithm, for example Reiter’s hitting set algorithm [31,15] or GDE [11], takes available observations about actual system behavior as well as a system model encoding the system’s specification, and derives possible diagnoses with respect to optimization criteria like diagnosis probability and size. With the classic approach, the system model as well as all observations have to be stored and processed centralized, even in the case of distributed systems with subsystems connected via communication channels.

The drawbacks of such a centralized approach have been the subject of academic discussion, e.g. [37]. For instance, as the (single) state space is usually aggregated from those of its subsystems, the considered space is substantially larger than when considering components locally. Since, in the worst case, diagnosis computation is exponential in the size of considered components, this is a severe computational issue. Inherently, also the robustness of the diagnostic reasoning process is weaker. A failure in the centralized diagnoser leads to a complete crash of the diagnostic reasoning. Regarding maintenance and component reuse, a centralized approach might suffer from poor scalability. Changes in the structure might require the centralized diagnoser itself to be changed, where it is even possible that the whole diagnoser has to be rebuilt. However, depending on the situation, a centralized diagnoser might not always be a bad choice. In case the system is not physically distributed, or if there are special requirements like storing all failure-related data in one place, there might be no reason to use a distributed diagnosis approach.

Considering related research, the term “distributed diagnosis” is overloaded and refers to very different scenarios. It might be used for instance to refer to the diagnosis of a system that comprises independent but interconnected subsystems, where each one has a corresponding local diagnoser. While from an abstract point of view the application seems to be very close to the diagnosis of a team of collaborating robots, there are subtle but essential differences. In fact, these differences are so severe that distributed diagnosis algorithms and systems that suit one domain, cannot be used for the other; but let us discuss an example for such an issue in detail.

In the scenario of a distributed system with channels for subsystem communication, original observa-
tions about system and environment attributes are most likely to be local to the corresponding subsystems (but are communicated afterwards). Hence, even in the case of distributing an observation by communication, it could not conflict with other (original) measurements of the same or related (physical) attribute like temperature. A corresponding application area with such characteristics could be that of industrial transportation systems as diagnosed centralized in [28].

This scenario is different from that of a collaborating team of autonomous robots that are equipped with sensors for the same or related physical attributes. In this case, one robot might measure ambient temperature to be too high, while its neighbor measures it to be in nominal range. Independent consideration by each robot might lead to disparate decisions and behavior (where in turn such disagreements might affect overall performance). Thus when aggregating local diagnoses for a system-wide one, deductions might stem from incompatible local diagnoses.

Consequently, an essential difference between the two scenarios is that in one case there can be no contradictions between available (possibly communicated) observations, while in the other, local observations might contradict each other. In the latter case, thus inconsistency of local observations has to be taken into account in order to handle such cases correctly. This illustrates the need for a deep analysis and classification of distributed diagnosis, as well as the necessity of defining the term distributed diagnosis and its aspects in detail.

Besides those already mentioned, there is a multitude of applications for distributed diagnosis, including settings that make use of service-oriented architectures (SOAs). SOAs aggregate numerous instances of web services, brokers, message busses, mediators, monitors, and other components that might be physically distributed over vast distances. In such settings a distributed diagnosis approach has significant advantages in respect of intellectual property disclosure and dynamic reconfiguration of services. While a company might be willing to provide an interface offering diagnostic data for a service, due to business and security reasons they might not want to disclose all the internal details necessary for customers to develop effective models on their own. Dynamic reconfiguration of services can be accommodated by corresponding exchanges of local diagnosers.

Another application is suggested by our example of autonomous robot teams. Collaborating autonomous systems might make use of distributed and dynamic knowledge bases [33,35]. Considering, for instance, data communication bandwidth and robustness, a distributed approach for the diagnosis of the underlying distributed knowledge base might be of advantage in such settings.

With the growing complexity of related systems, distributed diagnosis is also of interest for home automation systems, a setting which, due to its familiarity, we will use for our explanatory examples in Section 5.

Despite the varying needs for diagnostic reasoning in different distributed settings, to the best of our knowledge, there is no systematic study regarding an analysis and classification of approaches simultaneously considering essential features like architecture, utilized models, and abstract application related issues. The motivation of this paper is to discuss previous work in the domain of distributed diagnosis, the analysis of related modeling issues, and outlining a characterization of distributed diagnosis systems with respect to those essential aspects.

The remainder of this document is structured as follows. In Section 2, we discuss related research, followed by an analysis of relevant modeling issues presented in Section 3. Section 4 contains our analysis of the distributed diagnosis problem itself and related definitions. In Section 5, we discuss our characterization of distributed diagnosis systems, while in Section 6 we draw our conclusion and give directions for future work.

2. RELATED RESEARCH

In [20], Kurien et al. introduced a basic formalism for distributed diagnosis that is based on information interchange between local diagnosers where the intersection between the local component sets is empty. Local diagnoses with their corresponding value assignments are communicated to negotiate diagnoses that all local diagnosers agree on. In this process, inconsistencies between diagnoses from different local diagnosers are used to reduce the set of possible assignments. Thus, their approach can be classified as a distributed and decentralized approach that is based on local diagnosis systems interacting via shared connections and sharing global observations. The content of Kurien et al.’s paper is most closely to the one discussed in this paper. Similar work includes Wonham et al. [37] who use local automata models to generate local diagnosers that communicate for diagnosis re-
finement. Daigle et al. [8] tackle the task of distributed diagnosis in the context of mobile robots based on a bond graph modeling framework while aiming at minimizing the amount of remote measurements that make each subsystem locally diagnosable.

Other work in the domain of mobile robot diagnosis, specifically in the context of mobile robot interaction for achieving a common goal, includes [18]. In this paper, Kalech et al. describe an approach using distributed algorithms for the localization of faults in the team coordination.

Some colleagues outlined distributed diagnosis in the context of discrete event systems [21]. In [3], Baroni et al. extended their previous work on diagnosis of active systems [2] (communicating components that are idling, except when triggered by some incoming event) to the distributed case. Guillou et al. introduced in [16] the use of chronicles for distributed diagnosis, based on previous work of the same group [7]. The latter paper shows how incremental and decentralized diagnosis can be implemented effectively. Most recently, Ribot et al. discussed in [32] the use of design requirements that enable the diagnosability of discrete event systems. This work is of particular interest regarding practical applications, because it addresses the issue of how to construct system models so that the system can be diagnosed afterwards. In [34] the author analyzes and classifies distributed systems in respect of their local, decentral and central diagnosability. Our characterization complements this classification in that we characterize the diagnosis framework rather than the system.

Practical issues like enabling diagnosis under real-time requirements have also been discussed in literature. In [5], Chung and Barrett presented the distributed on-line diagnosis of spacecrafts under real-time constraints. Their approach combines model-based diagnosis with rule-based systems, where the underlying idea is to compile models into rules that can be used on-line. The advantage is that in this case, diagnosis time can be estimated and guaranteed, so that these rules can be used efficiently in critical real-time systems.

Summarizing, we see that there is a lot of research in the context of distributed diagnosis. However, to the best of our knowledge there is no detailed analysis and classification approach for general distributed diagnosis that simultaneously considers all the dimensions proposed in the introduction. While previous work is mainly focused on their particular problem domain, we are interested in classifying and addressing distributed diagnosis in general. In our reasoning, we follow very closely Reiter’s model-based diagnosis theory [31], with the objective of extending it to the distributed case. For early work in this respect, we refer the interested reader to [40] where we discussed challenges regarding distributed diagnosis and an extension of Reiter’s theory.

3. MODELING

In this section, we discuss modeling issues relevant for model-based diagnosis. Although there is more than one approach for system behavior modeling, for practical reasons, most of them rely on the well known component-connection modeling paradigm (CCMP). With CCMP the user is required to define a system’s structure as well as it’s behavior. System behavior is defined by that of its (sub-)components and ports. Ports provide the necessary connections for information exchange between components. To ease maintenance, usually components with the same behavior are defined as instances of a component class that might also be reused across various projects. Parameterization and inheritance provide convenient ways to deal with similar but unequal component instances.

According to Reiter’s original theory of diagnosis [31], only the correct behavior of a component has to be given in a corresponding model. Such an approach comes with the advantage of not requiring knowledge about possible faults and their consequences. And indeed, there are many applications where either no fault models are available, or where there are too many possible fault modes. However, as Struss et al. indicate in [36], there are cases where, if consistency-based diagnosis considers only definitions of correct behavior, this leads to the computation of diagnoses with no representation in the physical world. An example would be a bulb that is lit without power. Hence, although correct with respect to the model, such deceptive diagnoses are incorrect when considering reality. In order to solve this problem, Struss et al. suggested to introduce fault models. Console et al. discussed in [6] the close relationship between abduction and deduction. In particular, using [6] we can state that consistency-based diagnosis using fault models is equivalent to abduction-based diagnosis with respect to the computed diagnosis candidates. That is, an abduction-based diagnosis engine and a consistency-based one would compute the same root causes when using fault models in the consistency-based case. De
Kleer and colleagues [10] extended the formal definition of consistency-based diagnosis in order to be capable of handling fault modes, and also discussed some examples showing the influence of fault modes on the computation of diagnoses. For instance, we have that the property that all supersets of diagnoses are diagnoses themselves is no longer valid when using fault modes.

Motivated by the impact on computational complexity that is entailed by introducing fault models (affecting the base of the exponential term(s) for the amount of possible diagnoses), Gottlob et al. suggested to add physical impossibilities to models [14]. A physical impossibility is a behavioral rule that has to be fulfilled in all circumstances, and describes the relationship between connection values and the status of components. For example, let us consider an electronic circuit that comprises a battery and two parallel bulbs. If we observe that one bulb is lit while the other is not, we are able to conclude that (a) the battery works correctly and that (b) the bulb not lit is broken. This results from the fact that the battery has to be working, as without power no bulb could be lit. Lacking corresponding fault models, we can use physical impossibilities to obtain the same effect. We state that it cannot be the case that the battery is not working (or not providing voltage) if a bulb is lit. From a more general perspective, physical impossibilities are similar to invariants, e.g. loop invariants or class invariants as used in verification. We follow this more general term of invariants and consider physical impossibilities as system invariants. Such system invariants have only little influence on the computational complexity, in the sense that they increase the number of sentences but do not affect the number of variables or their domains, but contradict CCMP as they are not assigned to a component but to the system itself.

An extension to CCMP is the concept of hierarchical models. A model is called hierarchical, if a component model itself is implemented using CCMP. Hence, if CCMP is used recursively to aggregate a system from components, subcomponents, ..., as well as necessary connections. Note that thus a distributed model is not necessarily hierarchical. Igor Mozetič was the first who published the idea of hierarchical models [27]. Autio and Reiter [1] introduced a formal definition of hierarchical models including results regarding diagnosis capabilities. It is worth noting that hierarchical models are used for two reasons. The first is related to the modeling process itself. Using hierarchical models, components can be modeled from available simpler components, taking advantage of all the possibilities entailed with modularization and related libraries. Hence, modeling becomes easier and more convenient. The other reason concerns computational issues. On the top level, diagnosis has to consider only a small set of components, due to the abstraction performed when subsuming several components to form one hierarchical component. Since, in the worst case, the computation of diagnoses is exponential in the number of components, structuring systems in a hierarchical way saves computation time.

Academic literature offers some modeling languages for model-based diagnosis. In [13,17] the authors introduce the modeling language AD2L that is based on CCMP, and allows describing system models including fault models and system invariants. More recently, Provan and Wang [29,38] suggested a benchmark generator and later on a language for sharing models to be used for performance evaluation of diagnosis algorithms.

In distributed diagnosis, (local) models follow the same principles like models used in the non-distributed case. Hence, the models most likely rely on CCMP and (not necessarily) implement fault models, invariants, or hierarchical components. Having a closer look at the underlying formal modeling methods, we see that they vary from discrete event systems [21] to logic [31] and bond graphs [8]. The reason for choosing a certain format might stem from the requirements regarding temporal aspects. If temporal reasoning is not necessary at all, even simple horn-clause propositional logic might be sufficient.

4. DISTRIBUTED DIAGNOSIS

In this section, we will introduce several definitions used in our discussion of classification criteria for distributed diagnosis.

In contrast to centralized approaches using a global system model, all distributed diagnosis systems have in common that they make use of local models and local diagnosis engines. Specifically, all these approaches aim at computing global diagnoses from local ones without relying on a (single) global model. Hence, we define a distributed diagnosis system as a system aggregating local diagnosis engines (or local diagnosers), using local models and (not necessarily local) observations for the computation of local diagnoses. Subsequently, these local diagnoses are used to derive global ones. As discussed in Section 2, an analysis and clas-
sification of distributed systems in respect of their local, decentral and central diagnosability can be found in [34]. In Figure 1, we depict the architecture of a spatially distributed diagnosis system. It comprises two or more local diagnosis systems, each of which has a diagnosis engine $DE_i$, a model $SD_i$ of the underlying (sub-)system, and a set of observations $OBS_i$. The latter represents grounded facts that are obtained from sensor information after filtering and symbol grounding. It is worth noting that, due to sensing failures and sensor noise, sensed observations might not reflect reality. However, they are the only information available for diagnostic reasoning, and sensor models [26] can help to deal with corresponding errors. An essential part of the discussed distributed diagnosis architecture is the global diagnosis backbone. This backbone is used for combining the local diagnoses and might be used also for communicating observations between the local diagnosis engines (if required). Note that in general, there is no requirement to have a global diagnosis engine (or global diagnoser) as backbone. Depending on the specific diagnosis problem and related (local/decentral/central) diagnosability issues, computing global diagnoses might also be possible via communicating local diagnoses among the local diagnosis engines. In this case, the global diagnosis backbone in Fig. 1 might be reduced to a communication backbone for the exchange of diagnosis results.

**Definition 1 (Diagnosis System)** A diagnosis system $DS$ is a tuple $(SD, COMP, CONN, PE, \Psi)$ where $SD$ is a set of logical sentences describing the system's behavior, $COMP$ is the set of components, $CONN$ the set of connections, $PE$ the set of physical entities, and $\Psi$ is a function mapping connections to their corresponding physical entities. For simplicity, we assume the presence of functions $\text{comp}(DS)$ and $\text{conn}(DS)$ that, given a diagnosis system, return the corresponding sets of components and connections respectively.

Extending Reiter’s formalism, we introduce the sets $CONN$ and $PE$ in order to accommodate distributed systems. While $CONN$ is the set of connections a component has with the environment, the set $PE$ contains the environmental (physical) entities that the components might be connected to. The relation between these sets is defined by function $\Psi$. The reason for separating connections from physical entities is that this allows us to differentiate between entities at model level and real (e.g. physical) system entities. This separation is necessary, so that we can reason about scenarios, where two system parts measure the same physical entity (e.g. ambient temperature), but do this in a different way. Hence, in this case there might be inconsistencies in the observations that have to be dealt with.

According to [31], a diagnosis problem comprises a diagnosis system and a set of observations. While we follow Reiter’s definition in principle, we give a slightly modified definition of an actual diagnosis, in order to be self-compliant with Definition 1. Note that, like Reiter, we use the assumptions $AB(C)$ and $\neg AB(C)$ to refer to abnormal and normal behavior of some component $C \in COMP$.

**Definition 2 (Diagnosis)** Let $(SD, COMP, CONN, PE, \Psi)$ be a diagnosis system, and $OBS$ be a set of observations. A set $\Delta \subseteq COMP$ is a diagnosis iff $SD \cup OBS \cup \{\neg AB(C) | C \in COMP \setminus \Delta \} \cup \{AB(C) | C \in \Delta \}$ is satisfiable.

In the context of this paper, an observation is an assignment of a value to a connection. In practice, such observations are either stated by some user, or are derived from sensors via symbol grounding. For computing the diagnoses, it is assumed that model and observations are correct and reflect reality. As without further knowledge the incorrectness of observations cannot be stated, this is a useful assumption. However, as we will see later, this might not be the case for distributed diagnosis systems where there might be more...
than one measurement of the same physical attribute. Note that our definition can be easily extended to include fault modes, such that there are more than the two standard modes $AB()$ and $\neg AB()$ for a component (or entirely different ones).

For our distributed scenario, we formally define a distributed diagnosis system to be a set of local diagnosis systems.

**Definition 3 (Distributed Diagnosis System (DDS))**

Let $(SD, COMP, CONN, PE, \Psi)$ be a global diagnosis system. A set $\{DS_i = (SD, COMP_i, CONN_i, PE_i, \Psi) | i \in \{1, \ldots, n\}\}$ is a distributed diagnosis system (DDS) comprising $n$ local diagnosis systems iff the following conditions are satisfied:

1. $\forall i = 1 : SD_i \subseteq SD$
2. $(\bigcup_{i=1}^{n} COMP_i) = COMP$
3. $(\bigcup_{i=1}^{n} CONN_i) = CONN$

Note that the global diagnosis system in this definition does not refer to the global diagnosis backbone as of Figure 1, but is introduced solely as reference for defining completeness and correctness (see Def. 5). It is worth noting that our definition allows $SD$ to contain more data than the union of the individual descriptions, and also that we use a global set of physical entities for the local systems on purpose. The latter emphasizes the fact that some $p \in PE$ might be shared, a situation that might lead to problems with observations as discussed before. Furthermore, for obvious reasons this set is equal to the set of physical entities for the global diagnosis system. The choice whether $\psi$ is a global function or split into individual functions $\psi_i$ for each local system is one of personal taste.

In order to apply a diagnosis algorithm to a distributed diagnosis problem, we first have to define this problem. We state a distributed diagnosis problem as follows:

**Definition 4 (Distributed Diagnosis Problem)** A distributed diagnosis problem comprises a DDS $\{DS_i | i \in \{1, \ldots, n\}\}$ and a set of observations $\{OBS_i | i \in \{1, \ldots, n\}\}$ where $OBS_i$ are the observations for the corresponding local diagnosis system $DS_i$.

One characterization of distributed diagnosis is that the local diagnosis systems from a DDS, and their corresponding observations, are used to compute local diagnoses following Definition 2. Subsequently, these local diagnoses are combined to obtain global ones. The correctness and completeness of such a distributed diagnosis algorithm (using a DDS and defining the “backbone”) depends on the integration mechanism utilized. A DDS for our letter shoot system example in [28] could consist, for instance, of six local diagnosers, one for each component. In reality, a tube segment like component 2 could be very long, containing more than one sensor. Thus, in case only 4 out of 5 sensors report a letter passing component 2, local diagnoses could hint, for instance, at one or four incorrect sensor readings. A distributed diagnosis algorithm then has to combine the individual local solutions, considering aspects like component connections. As we do not propose a concrete algorithm but a classification, we do not construct a solution at this point, but define the diagnoses of a centralized approach as reference.

Formally, we are able to define the correctness and completeness of such an algorithm with respect to global and local models as well as global and local observations. Note that in general, due to possible inconsistencies, the set of global observations cannot be obtained by taking the union of local observations. Simple approaches like trying all options, majority vote, or using fault probabilities defined statically by a sensor’s characteristics or dynamically by fault history, as well as more elaborate approaches like [19] provide the means to resolve such inconsistencies [4].

**Definition 5 (Correctness, Completeness)** Let $GDS = (SD, COMP, CONN, PE, \Psi)$ be a global diagnosis system, $DDS = \{DS_i | i \in \{1, \ldots, n\}\}$ a corresponding distributed diagnosis system, $OBS$ the global observations, and $\{OBS_i | i \in \{1, \ldots, n\}\}$ the local observations. A distributed diagnosis algorithm $DD$ is correct iff all computed diagnoses using the DDS and the local observations are also diagnoses of the global diagnosis problem. A distributed diagnosis algorithm $DD$ is complete iff all global diagnoses are computed.

The correctness and completeness of already published distributed diagnosis algorithms usually requires additional assumptions. For example, it is often assumed that DDSs do not share components and that local observations are never in contradiction. While these assumptions are appropriate for several domains, there are applications, like autonomous mobile robot teams, where these assumptions are invalid. Consider, for instance, our example of a team of robots working on the same task. Each robot is perceiving the world via its sensors, and each robot’s diagnosis system is relying on its sensor information to reflect the state of the real world. While the robots are perceiving the same physical entities, due to symbol grounding they might use different corresponding observations. Therefore, they
do not share observations, and as a consequence, local observations might be in contradiction. Moreover, the robots may compute diagnoses that reflect their individually perceived correctness or incorrectness of real world entities. When each robot uses the same model, the intersection of considered components is not empty. Hence, these assumptions are not valid in this application domain, which has consequences on the choice regarding the use of a specific distributed diagnosis algorithm.

In order to let us characterize DDS and thus corresponding diagnosis algorithms in more detail, we further partition the space of DDS into subclasses. Note that we show classification examples at the end of this section, but define all the subclasses and discuss their relation first. Let us start with a DDS where the intersection of the local components is empty.

**Definition 6 (Partitioned DDS)** A distributed diagnosis system \( \{DS_i\}_{i \in \{1, \ldots, n\}} \) is a partitioned DDS if and only if \( i \neq j \rightarrow \text{comp}(DS_i) \cap \text{comp}(DS_j) = \emptyset \).

In a partitioned DDS, the local diagnosis systems do not share any component such that there can be no competing diagnosis results for one and the same component. Obviously however, physical entities can still be shared. Thus, in order to further divide the DDS space, we introduce structural independent DDSs, where even no connections are shared.

**Definition 7 (Structural Independent DDS)** A partitioned DDS \( \{DS_i\}_{i \in \{1, \ldots, n\}} \) is a structural independent DDS if and only if \( i \neq j \rightarrow \text{conn}(DS_i) \cap \text{conn}(DS_j) = \emptyset \).

While this definition does not allow sharing a connection (e.g. an external/unmodelled sensor), it is not strict enough to ensure that local observations are not in contradiction. This stems from the fact that it allows independent sensor measurements of one and the same environmental entity \( p \in PE \) by the local diagnosis systems. To ensure real independence in a distributed setting, we have to apply further restrictions.

**Definition 8 (Independent DDS)** A structural independent DDS \( \{DS_i\}_{i \in \{1, \ldots, n\}} \) is an independent DDS if and only if \( i \neq j \rightarrow \{\Psi(c)\}_{c \in \text{conn}(DS_i)} \cap \{\Psi(d)\}_{d \in \text{conn}(DS_j)} = \emptyset \).

In the case of an independent DDS, it is ensured that local observations cannot intersect. Hence, a simple algorithm that computes local diagnoses and puts them together by computing all possible combinations is correct and complete. For a structural independent DDS, such a simple algorithm can only be used if the local observations are never in contradiction.

A different branch of DDS, which occurs often in practice, is that where a global diagnosis system is partitioned into connected subsystems. Many of today's systems like power supplies or telecommunication networks fall into this category.

**Definition 9 (Connected DDS)** A partitioned DDS \( \{DS_i\}_{i \in \{1, \ldots, n\}} \) is a connected DDS if\( ∀ i, j \in \{1, \ldots, n\} \) we have \( i \neq j \land \text{conn}(DS_i) \cap \text{conn}(DS_j) \neq \emptyset \).

Let DDS be the set of all DDS, PDDS the set of partitioned DDS, SIDSS the set of structural independent DDS, and CDSS the set of connected DDS. Then, from the definitions above, we are able to easily identify the following relationships:

**Corollary 1** The following relationships hold for the different classes of DDS:

- DDS ⊂ PDDS ⊂ SIDSS ⊂ IDSS
- PDDS ⊂ CDSS
- CDSS ∩ SIDSS = ∅

Note that, as the relationships directly follow from the definitions, proofs are omitted.

Now let us consider the defined classes in the context of some concrete application examples in the home automation domain. Today, home automation is an ever-growing area, so that besides traditional applications like air conditioning and heating, nowadays global house automation might also control lighting, jalousies, cleaning and service robots, garden- and houseplant watering, as well as multimedia appliances to name just a few applications. The physical distributedness and obvious heterogeneity of such systems clearly suggest distributed approaches to diagnostic reasoning. With the following examples we will illustrate the earlier described DDS classes and will furthermore show that an actual classification heavily depends on the specifics of the actual scenario and does not simply relate to an application domain.

Assume as first example a simple home automation system for a one-room loft where we have an air conditioner, a multimedia appliance, a single sensor array, and a single remote control.

In case no subsystem considers the remote control as integral part (or no more than one of the subsys-
same physical entity
not an independent DDS as two sensors measure the
temperature is measured with two local sensors), it is
partitioned and structurally independent (the common
each one having its own local (integrated) temperature

Consider now a larger room with two air conditioners,
nario and also the partition of the diagnosis system.

actual classification’s dependence on the specific sce-
nection, and the latter as no two connections aim at
similar as no two connections necessarily share a con-
sion does not apply. In case of networks a clear
restricted autonomous systems, a general DDS or a
connected DDS might be used because further re-
strictions do not apply. In case of networks a clear
separation of components is possible and observa-
tions can be exchanged over the sharing con-
connections only. Hence, a connected DDS might be
sufficient. Once a characterization of diagnosis al-
gorithms according to DDS is available, such a clas-
sification would allow to select correct and complete
algorithms for different problems. How-
however, even when such a classification of algo-
rithms is not available, the characterization into
DDS classes helps to understand the problems
that have to be tackled when developing an algo-

tems), and each sensor belongs only to one subsystem,
then this scenario would fit a partitioned DDS. This
comes from the fact that no component is shared. Fur-
thermore this setting allows for a structural indepen-
dent, and even independent DDS. The former is pos-
sible as no two components necessarily share a con-
nection, and the latter as no two connections aim at
the same environmental entity. Obviously this assumes
that the sensor array offers no redundancy.

The following variations of this problem show an
actual classification’s dependence on the specific sce-
nario and also the partition of the diagnosis system.

Consider now a larger room with two air conditioners,
each one having its own local (integrated) temperature
sensor and local diagnosers. While the DDS then is
partitioned and structurally independent (the common
temperature is measured with two local sensors), it is
not an independent DDS as two sensors measure the
same physical entity \( p \in PE \). In case of a single global
temperature sensor that is an integral part of each lo-
cal model, the DDS would also not be structural in-
dependent (one sensor would be connected to two lo-
cal diagnosers). In that case, if furthermore the system
would contain only the air conditioners and the sensor,
we would have a connected DDS. The influence of the
system partition becomes evident when we consider
that removing the shared sensor (and thus the shared
connection) from the local air conditioner models and
adding its own local diagnoser to the system allows for
an independent DDS again.

The implications of our slight variations of the ini-
tial home automation diagnosis problem perfectly il-
strate that the details of a diagnosis scenario and the
actual diagnoser partition play a significant role in the
classification process. This dependence and the related
impact on the diagnosis process regarding the question
which distributed diagnosis algorithms suit an actual
situation best, furthermore suggest that it is advisable
to consider diagnosis aspects in early design stages.

Designers then can accommodate related issues in their
early decisions and find the optimal balance between
resources needed in the diagnostic process versus hard-
ware installation costs (e.g. in the context of sensor
sharing).

5. CHARACTERIZING DISTRIBUTED
DIAGNOSIS SYSTEMS

After discussing modeling aspects in the context of
model-based diagnosis, introducing the notions of the
distributed diagnosis problems and related distributed
diagnosis systems along several definitions for a sensi-
tible classification, we are now able to characterize dis-
tributed diagnosis. In the following, we will discuss
this characterization along three dimensions: model-
ing, classification of DDS, and used architecture.

- Model: There are many modeling paradigms and
modeling languages used in practice. One
might use discrete event systems, finite state
machines, or a simple non-temporal logic for
describing the behavior of a system. The
underlying models might follow the component-
connection modeling paradigm, with or without
making use of physical impossibilities, fault mod-
els, or modeling hierarchies. The different model-
ling paradigms and styles can be used to character-
ize diagnosis systems and therefore also DDS. In
case of DDS, the choice of the modeling language
and paradigm has an impact on the partitioning of
the local diagnosis models. However, the choice
has no direct influence on the combination of lo-
cal diagnoses in order to obtain global ones, if
the local diagnosis engines only return subsets
of components as diagnosis candidates (as stated
in definition of diagnosis: Def. 2). Nevertheless,
the underlying modeling language and paradigm
have an impact on the choice of the underlying lo-
cal diagnosis algorithm and theorem provers used.
Hence, a characterization of DDS according to the
model is useful to select appropriate local diagno-
sis engines and theorem provers.

- DDS class: A distributed diagnosis problem that
belongs to a certain application area can be char-
acterized according to the DDS classes introduced
in this paper. This characterization heavily de-
pends on the application scenario. In case of dis-
tributed autonomous systems, a general DDS or a
connected DDS might be used because further re-
strictions do not apply. In case of networks a clear
separation of components is possible and observa-
tions can be exchanged over the sharing con-
nections only. Hence, a connected DDS might be
sufficient. Once a characterization of diagnosis al-
gorithms according to DDS is available, such a clas-
sification would allow to select correct and complete
algorithms for different problems. How-
never, even when such a classification of algo-
rithms is not available, the characterization into
DDS classes helps to understand the problems
that have to be tackled when developing an algo-

pendent DDS, someone has to be aware that there might be contradicting observations, which have to be handled in the right way.

- **Architecture:** Regarding the implementation of DDS there are two general architectures one can follow. The centralized distributed diagnosis is characterized by a centralized global diagnosis backbone that takes the local diagnosis results and combines them to form global diagnoses. Such a centralized diagnosis backbone is also capable of distributing observations and performing measurement selection. The advantage is that obtaining global diagnoses is easier from the algorithmic point of view and that communication can be controlled globally, which might lead to less messages necessary to obtain global diagnoses. The disadvantage is that the approach is less robust. In case of a fault in the centralized global diagnosis backbone, there is no way of coming up with a global diagnostic view.

In the decentralized distributed architecture, the local diagnosis engines communicate their results and compute a global view without using a central diagnosis backbone. In this case, the global diagnosis backbone consists of direct connections between the local diagnosis engines. The advantage of this variant is that a decentralized approach is more robust and that faults in one part should have only local influence. However, there is a communication overhead because all necessary information has to be communicated to all connected subsystems.

6. CONCLUSION

In this paper, we discussed modeling aspects in the context of model-based diagnosis, analyzed the distributed diagnosis problem, introduced corresponding definitions, suggest a specific classification of DDS, and propose a characterization of competing approaches regarding essential problem aspects. The three dimensions proposed are modeling, classification of DDS, and used architecture. While a lot of applications in the broad context of distributed diagnosis seem to be similar, we showed by simple examples that there are significant differences that have to be taken into account when selecting an appropriate algorithm for a specific problem. With our characterization, we enable the selection of the right diagnosis algorithm and architecture, given a distributed diagnosis problem in a certain application area. Currently missing is a classification of already published distributed diagnosis algorithms using the classifications given in this paper. However, even without such a classification, the characterization into DDS classes helps in understanding the problems that have to be tackled when developing an algorithm. For example, in the case of a structural independent DDS, we are made aware that possibly contradicting observations have to be taken care of.

Future work will include classifying previous DDS approaches using our classification framework. Furthermore, we are interested in providing algorithms including correctness and completeness proofs, as well as in identifying specific algorithmic areas and aspects that should be covered by future research. An important problem we will tackle with future research is that of solving diagnosis problems for autonomous mobile robots in a distributed way.

Acknowledgements

The work presented in the paper has been funded by the Austrian Science Fund (FWF) under contract numbers P20199-N15 and P22959-N23. The authors would also like to thank the reviewers for their valuable feedback as well as the reviewers from the ECAI 2010 I-Kbet workshop for their valuable comments on an earlier workshop-version of this paper.

References
