Primary and Secondary Plan Diagnosis*

Femke de Jonge and **Nico Roos** and **Cees Witteveen**
Dept. of Computer Science, Universiteit Maastricht, P.O.Box 616, NL-6200 MD Maastricht
fax: +31-43-3884897, {F.deJonge,Roos}@cs.unimaas.nl
Faculty EEMCS, Delft University of Technology, P.O.Box 5031, NL-2600 GA Delft
fax: +31-15-2786632, C.Witteveen@tudelft.nl

Abstract

Diagnosis of plan failures is an important subject in both single- and multi-agent planning. Plan diagnosis may provide information that can improve the way the plan failures are dealt with in three ways: (*i*) it provides information necessary for the adjustment of the current plan or for the development of a new plan, (*ii*) it can be used to point out which equipment and/or agents should be repaired or adjusted so they will not further harm the plan execution, and (*iii*) it can identify the agents responsible for plan execution failures.

We introduce two general types of plan diagnosis: *primary plan diagnosis* identifying the incorrect or failed execution of actions, and *secondary plan diagnosis* that identifies the underlying causes of the faulty actions. Furthermore, three special cases of secondary diagnosis are distinguished, namely *equipment diagnosis*, *environment diagnosis* and *agent diagnosis*.

1 Introduction

In multi-agent planning research there is a tendency to deal with plans that become larger, more detailed and more complex. Clearly, as complexity grows, the vulnerability of plans for failures will grow correspondingly. Taking appropriate measures to such plan failures requires knowledge on their causes. So it is important to be able to detect both the occurrence of failures and to determine the causes of them. Therefore, we consider diagnosis as an integral part of the capabilities of planning agents in single- and multi-agent systems.

In this paper we adapt and extend a classical Model-Based Diagnosis (MBD) approach to the diagnosis of plan execution. The system to be diagnosed consists not only of the plan and its execution, but also of the equipment needed for the execution, the environment in which the plan is executed and the executing agents themselves. Therefore, the agents, the equipment and the environment need also be the subject of diagnosis.

To motivate the need for these different types of diagnosis we distinguished, consider a very simple example in which a pilot agent of an airplane participates in a larger multi-agent system for the Air Traffic Control of an airport. Suppose that the pilot agent is performing a landing procedure and that its plan prescribes the deployment of the landing gear. Unfortunately, the pilot was forced to make a belly landing. Clearly, the plan execution has failed and we wish to apply diagnosis to find out why. A first, but superficial, diagnosis will point out that the agent's action of deploying the landing gear has failed and that the fault mode of this action is "landing gear not locked". We will denote this type of diagnosis as *primary plan diagnosis*; this type of diagnosis focuses on set of fault behaviors of *actions* that explain the differences between the expected and the observed plan execution.

Often, however, it is more interesting to determine the causes behind such faulty action executions. In our example, a faulty sensor may incorrectly indicate that the landing gear is already extended and locked, which led the pilot agent to the belief that the action was successfully executed. We will denote the diagnosis of these underlying causes as *secondary plan diagnosis*. Secondary diagnosis can be viewed as a diagnosis of the primary diagnosis. It informs us about malfunctioning equipment, unexpected environment changes (such as the weather) and faulty agents. As a special type of secondary diagnosis, we are also able to determine the agents *responsible* for the failed execution of some actions. In our example, the pilot agent might be responsible, but so might be the airplane maintenance agent.

In our opinion, diagnosis in general, and secondary diagnosis in particular, enables the agents involved to make specific adjustments to the system or the plan as to manage current plan-execution failures and to avoid new plan-execution failures. These adjustments can be categorized with regard to their benefits to the general system. First of all, diagnosis provides information on how the plan behaves during execution, which might contribute to a failure-free (re)planning. For example, we can imagine that the initial knowledge of how a dynamic environment may influence the plan execution is rather limited. Diagnosis may provide information that expands the knowledge about plan execution. Secondly, a secondary diagnosis can point out which equipment used for the

*This research is supported by the Technology Foundation STW, applied science division of NWO and the technology programme of the Ministry of Economic Affairs (the Netherlands). Project DIT5780: Distributed Model Based Diagnosis and Repair.

plan execution was malfunctioning. Broken equipment then can be fixed to improve future plan execution. Moreover, if the amount of possible repairs is limited, diagnosis can indicate which repair has the most, positive, influence on future plan execution. In this respect, agents can be viewed in the same way as equipment: agents too can malfunction, either because of incorrect beliefs of the agent, or because the agent somehow died (crashed). Secondary diagnosis can also provide the information necessary to recover and adjust agents thereby contributing to a better plan execution. Hence, it can contribute to solving the well known qualification problem [McCarthy, 1977]. Finally, diagnosis can indicate the agents responsible (accountable) for the failures in the plan execution. This information is very interesting when evaluating the system, and can also be used to divide costs of repairs and/or changes in the plan amongst the agents.

To realize the benefits of plan-based diagnosis outlined above, we introduce an object-oriented view to describe plan-execution. Based on this model primary and secondary diagnosis will be defined. The primary plan diagnosis more or less corresponds with the main aspects of diagnosis of plan execution described by Witteveen and Roos [Witteveen *et al.*, 2005; Roos and Witteveen, 2005]. To enable us to apply secondary plan diagnosis, we expand their model such that it is not only possible to analyze the plan execution process, but also the role of the objects that influence the plan execution. The resulting model is specified in section 3 and consists of objects representing the plan and its execution, the equipment that is used for the plan execution, the environmental objects that are somehow involved in the plan, and the agent executing the plan. On this model, we can apply techniques inspired by model-based diagnosis to find the primary diagnosis, as described in subsection 4.1. The secondary diagnosis is presented in subsection 4.2, while subsection 4.3 discusses the agent that are held responsible for the failed actions. But first of all, we will place our approach into perspective by discussing some approaches to plan diagnosis in the following section.

2 Related research

In this section we briefly discuss some other approaches to plan diagnosis.

Birnbaum et al. [Birnbaum *et al.*, 1990] apply MBD to *planning agents* relating planning assumptions made by the agents to *outcomes* of their planning activities. However, they do not consider faults caused by execution failures as a separate source of errors.

de Jonge et al. [de Jonge and Roos, 2004; de Jonge *et al.*, 2005] present an approach that directly applies model-based diagnosis to plan execution. Here, the authors focus on agents each having an individual plan, and on the conflicts that may arise between these plans (e.g. if they require the same resource). Diagnosis is applied to determine those factors that are accountable for *future* conflicts. The authors, however, do not take into account dependencies between health modes of actions and do not consider agents that collaborate to execute a common plan.

Kalech and Kaminka [Kalech and Kaminka, 2003; 2004] apply *social diagnosis* in order to find the cause of an anomalous plan execution. They consider hierarchical plans consisting of so-called *behaviors*. Such plans do not prescribe a (partial) execution order on a set of actions. Instead, based on its observations and beliefs, each agent chooses the appropriate behavior to be executed. Each behavior in turn may consist of primitive actions to be executed, or of a set of other behaviors to choose from. Social diagnosis then addresses the issue of determining what went wrong in the joint execution of such a plan by identifying the disagreeing agents and the causes for their selection of incompatible behaviors (e.g., belief disagreement, communication errors). Although we do not consider hierarchical plans of behaviors, social diagnosis is related to the here proposed agent diagnosis.

Lesser et al. [Carver and Lesser, 2003; Horling *et al.*, 2001] also apply diagnosis to (multi-agent) plans. Their research concentrates on the use of a *causal model* that can help an agent to refine its initial diagnosis of a failing *component* (called a *task*) of a plan. As a consequence of using such a causal model, the agent would be able to generate a new, situation-specific plan that is better suited to pursue its goal. While their approach in its ultimate intentions (establishing anomalies in order to find a suitable plan repair) comes close to our approach, their approach to diagnosis concentrates on specifying the exact causes of the failing of one single *component* (task) of a plan. Diagnosis is based on observations of a component without taking into account the consequences of failures of such a component w.r.t. the remaining plan. In our approach, instead, we are interested in applying MBD-inspired methods to *detect* plan failures. Such failures are based on observations during plan execution and may concern individual components of the plan, but also agent properties. Furthermore, we do not only concentrate on failing components themselves, but also on the consequences of these failures for the future execution of plan elements.

Witteveen et al. [Witteveen *et al.*, 2005; Roos and Witteveen, 2005] show how classical MBD can be applied to plan execution. To illustrate the different aspects of diagnosis discussed in the introduction, below we present an adapted and extended version of their formalization of plan diagnosis. This formalization enables the handling of the approaches of de Jonge et al. [de Jonge and Roos, 2004; de Jonge *et al.*, 2005], Kalech and Kaminka [Kalech and Kaminka, 2003; 2004], and Lesser et al. [Carver and Lesser, 2003; Horling *et al.*, 2001]. The work of Birnbaum et al. [Birnbaum *et al.*, 1990] is not covered by the proposed formalization since it focuses on the planning activity instead of on plan execution.

3 Preliminaries

Objects In [Witteveen *et al.*, 2005] it was shown that by using an object-oriented description of the world instead of a conventional state-based description, it becomes possible to apply classical MBD to plan execution. Here, we will take this approach one step further by also introducing objects for agents executing the plan and for the actions themselves. Hence, we assume a set of objects $\mathcal{O}$ that will be used to describe the plan, the agents, the equipment and the environment.

The objects $\mathcal{O}$ are partitioned into classes or types. We distinguish four general classes, namely: *actions* $\mathcal{A}$, *agents* $\mathcal{A}g$, *equipment* $\mathcal{E}$ and *environment objects* $\mathcal{N}$.

States and partial states Each object in $o \in \mathcal{O}$ is assumed to have a domain D_o of values. The *state* of the objects $\mathcal{O} = \{o_1, ..., o_n\}$ at some time point is described by a tuple $\sigma \in D_{o_1} \times ... \times D_{o_n}$ of values. In particular, the states $\sigma^{\mathcal{A}}$, $\sigma^{\mathcal{A}g}$, $\sigma^{\mathcal{E}}$ and $\sigma^{\mathcal{N}}$ are used to denote the state of the action objects $\mathcal{A}$, the agent objects $\mathcal{A}g$, the equipment objects $\mathcal{E}$ and the environment objects $\mathcal{N}$, respectively.

The state $\sigma^{\mathcal{N}}$ of environment objects $\mathcal{N}$ describes the state of the agents' environment at some point in time. These state descriptions can be the location of an airplane or the availability of a gate.

The states $\sigma^{\mathcal{A}}$, $\sigma^{\mathcal{A}g}$ and $\sigma^{\mathcal{E}}$ of action, agent and equipment objects, respectively, describe the *health modes* of these objects for the purpose of diagnosis [Kleer and Williams, 1989; Struss and Dressler, 1989]. We assume that each of their corresponding domains contains at least (*i*) the value nor to denote that the action, agent and equipment objects behave normally, and (*ii*) the general fault mode ab to denote that the action, agent and equipment objects behave in an unknown and possibly abnormal way. Moreover, the domains may contain several more specific fault modes. For instance, the domain of a 'flight' action may contain a fault mode indicating that the flight is 20 minutes delayed.[1]

It will not always be possible to give a complete state description. Therefore, we introduce a *partial state* as an element $\pi \in D_{o_{i_1}} \times D_{o_{i_2}} \times \ldots \times D_{o_{i_k}}$, where $1 \leq k \leq n$ and $1 \leq i_1 < \ldots < i_k \leq |\mathcal{O}|$. We use $O(\pi)$ to denote the set of objects $\{o_{i_1}, o_{i_2}, \ldots, o_{i_k}\} \subseteq \mathcal{O}$ specified in such a state π. The value of an object $o \in O(\pi)$ in π will be denoted by $\pi(o)$. The value of an object $o \in \mathcal{O}$ not occurring in a partial state π is said to be *unknown* (or unpredictable) in π, denoted by $\perp$. Including $\perp$ in every value domain D_i allows us to consider every partial state π as an element of $D_1 \times D_2 \times \ldots \times D_{|\mathcal{O}|}$.

Partial states can be ordered with respect to their information content: given values d and d', we say that $d \leq d'$ holds iff $d = \perp$ or $d = d'$. The containment relation $\sqsubseteq$ between partial states is the point-wise extension of $\leq$: π is said to be contained in π', denoted by $\pi \sqsubseteq \pi'$, iff $\forall o \in \mathcal{O}[\pi(o) \leq \pi'(o)]$. Given a subset of objects $S \subseteq \mathcal{O}$, two partial states π, π' are said to be *S-equivalent*, denoted by $\pi =_S \pi'$, if for every $o \in S$, $\pi(o) = \pi'(o)$. We define the partial state π restricted to a given set S, denoted by $\pi \restriction S$, as the state $\pi' \sqsubseteq \pi$ such that $O(\pi') = S \cap O(\pi)$.

An important notion for our notion of diagnosis is the *compatibility relation* between partial states. Intuitively, two states π and π' are said to be compatible if they could refer to the same complete state. This means that they do not disagree on the values of objects defined in both states, i.e., for every $o \in \mathcal{O}$ either $\pi(o) = \pi'(o)$ or at least one of the values $\pi(o)$ and $\pi'(o)$ is undefined. So we define π and π' to be compatible, denoted by $\pi \approx \pi'$, iff $\forall o \in \mathcal{O}[\pi(o) \leq \pi'(o)\ or\ \pi'(o) \leq \pi(o)]$. As an easy consequence we have, using the notion of S-equivalent states, $\pi \approx \pi'$ iff $\pi =_{O(\pi) \cap O(\pi')} \pi'$. Finally, if π and π' are compatible states, they can be *merged* into the $\sqsubseteq$-least state $\pi \sqcup \pi'$ containing them both: $\forall o \in \mathcal{O}[\pi \sqcup \pi'(o) = max_{\leq}\{\pi(o), \pi'(o)\}]$.

Goals An (elementary) goal g of an agent specifies a set of states an agent wants to bring about using a plan. Here, we specify each such a goal g as a constraint, that is, a relation over some product $D_{i_1} \times \ldots \times D_{i_k}$ of domains. We say that a goal g is satisfied by a partial state π, denoted by $\pi \models g$, if the relation g contains some tuple (partial state) $(d_{i_1}, d_{i_2}, \ldots d_{i_k})$ such that $(d_{i_1}, d_{i_2}, \ldots d_{i_k}) \sqsubseteq \pi$. We assume each agent a to have a set G_a of such elementary goals $g \in G_a$. We use $\pi \models G_a$ to denote that all goals in G_a hold in π, i.e. for all $g \in G_a$, $\pi \models g$.

Actions and action execution The set $\mathcal{A}$ of action objects, also called *plan steps* is partitioned into subclasses α_i called *action types* or *plan operators*. Through the execution of a specific action object $a \in \mathcal{A}$, the state of environment objects $\mathcal{N}$ and possibly also of equipment $\mathcal{E}$ objects may change. We describe such changes of an action object $a \in \mathcal{A}$ by a (partial) function f^{α} where α is the type of the action (plan operator) a is an instance of:

$$f^{\alpha} : D_a \times D_{ag} \times D_{e_1} \times ... \times D_{e_i} \times D_{n_1} \times ... \times D_{n_j} \rightarrow D_{e'_1} \times ... \times D_{e'_k} \times D_{n'_1} \times ... \times D_{n'_l}$$

where $a \in \alpha \subset \mathcal{A}$ is an action of type α, $ag \in \mathcal{A}g$ is the execution agent, $e_1, ..., e_i \in \mathcal{E}$ are the required equipment objects, $n_1, ..., n_i \in \mathcal{N}$ are the required environment objects, and $\{e'_1, ..., e'_k, n'_1, ..., n'_l\} \subseteq \{e_1, ..., e_i, n_1, ..., n_j\}$ are equipment and environment objects that are changed by the action a. Note that since the values of equipment objects only indicate health modes of these objects we allow for equipment objects in the range of f^{α} in order be able to describe repair and maintenance actions.

To distinguish the different types of parameters in a more clear way, semicolons will be placed between them when they appear in the argument of a function, e.g.:

$$f^{\text{transport}}(driving : \mathcal{A};\ hal : \mathcal{A}g;\ truck : \mathcal{E};\ goods : \mathcal{N}).$$

The objects whose value domains occur in $dom(f^{\alpha})$ will be denoted by

$$dom_O(o_a) = \{o_a, o_{ag}, o_{e_1}, ..., o_{e_i}, o_{n_1}, ..., o_{n_j}\}$$

and, analogously,

$$ran_O(o_a) = \{o_{e'_1}, ..., o_{e'_l}, o_{n'_1}, ..., o_{n'_j}\}.$$

Moreover, we will use $dom_O^{\mathcal{A}g}(o_a)$, $dom_O^{\mathcal{E}}(o_a)$ and $dom_O^{\mathcal{N}}(o_a)$ to denote $\{o_{ag}\}$, $\{o_{e_1}, ..., o_{e_i}\}$ and $\{o_{n_1}, ..., o_{n_j}\}$ respectively. Note that we use the action instance o_a to denote the objects involved in the execution of o_a according to the function f^{α} with $o_a \in \alpha$.

The result of an action may not always be known if, for instance, the action fails or if equipment is malfunctioning.

[1] Note that in a more elaborate approach the value of, for instance, an equipment object may also indicate the location of the equipment. In this paper we only represent the health mode of the equipment.

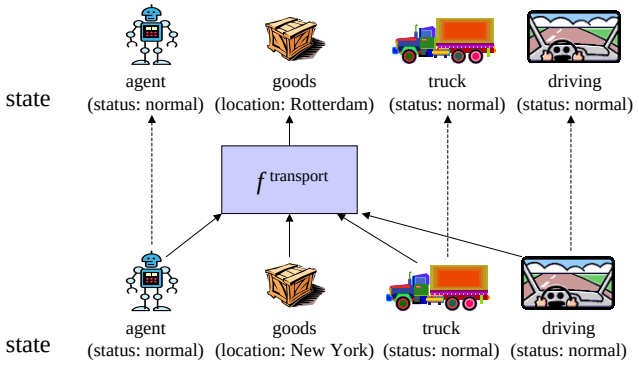

Figure 1: An action and its state transformation.

Therefore we allow that the function associated with an action maps the value of an object o to $\perp$ to denote that the effect of the action on o is unknown. In fact, we only require that the effect of an action is completely specified for all objects in the function's range if the action is executed in normal circumstances. That is, the agent is capable of executing the planned action given the planned equipment.

Figure 1 gives an illustration of the above outlined state transformation as result of the application of a drive action. Note that in this example only the state of the goods is changed as the result of the transport action. In the paper we assume that an object representing equipment only indicates the health state of the equipment. In a more elaborated approach, the health state is only one of the attributes of the object. Another attribute may indicate the location of the object. In Figure 1, the location of the *truck* changes also as result of the *drive* action.

Plans A plan is a tuple $P = \langle A, < \rangle$ where $A \subseteq \mathcal{A}$ is a subset of plan steps (action objects) that need to be executed and $<$ is a partial order defined on $A \times A$ where $a < a'$ indicates that the plan step a must finish before the plan step a' may start. Note that each plan step $a \in A$ occurs exactly once in the plan P, while there may be several plan steps that belong to the same action type. We will denote the *transitive reduction* of $<$ by $\ll$, i.e., $\ll$ is the smallest sub-relation of $<$ such that the transitive closure $\ll^+$ of $\ll$ equals $<$.

We assume that if in a plan P two action instances a and a' are independent, in principle they may be executed concurrently. This means that the precedence relation $<$ at least should capture all resource dependencies that would prohibit concurrent execution of actions. Therefore, we assume $<$ to satisfy the following *concurrency requirement*:

If $ran_O(a) \cap dom_O(a') \neq \varnothing$ then $a < a'$ or $a' < a$.

That is, for concurrent instances, domains and ranges do not overlap. [2]

[2]Note that since $ran_O(a) \subseteq dom_O(a)$, this requirement excludes overlapping ranges of concurrent actions, but domains of concurrent actions are allowed to overlap as long as the values of the object in the overlapping domains are not affected by the actions.

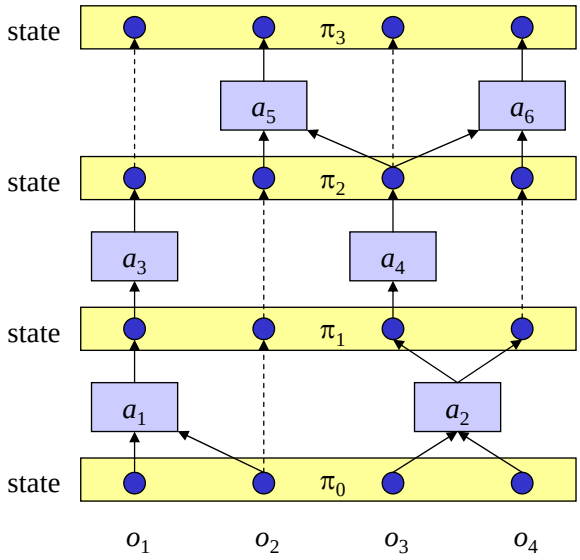

Figure 2: Plans and action instances. Each state characterizes the values of four objects o_1, o_2, o_3 and o_4. States are changed by application of action instances

Example Figure 2 gives an illustration of a plan. Since an action object is applied only once in a plan, for clarity reasons, *we will replace the function describing the behavior of the action by the name of the action*. The arrows relate action to the objects is uses as inputs and the objects it modifies as its outputs. In this plan, the dependency relation is specified as $a_1 \ll a_3$, $a_2 \ll a_4$, $a_4 \ll a_5$, $a_4 \ll a_6$ and $a_1 \ll a_5$. Note that the last dependency has to be included because a_5 changes the value of o_2 needed by a_1. The action a_1 shows that not every object occurring in the domain of an action need to be affected by the action. The actions a_5 and a_6 illustrate that concurrent actions may have overlapping domains. □

Plan execution For simplicity, we will assume that every action in a plan P takes a unit of time to execute. We are allowed to observe the execution of a plan P at discrete times $t = 0, 1, 2, \ldots, k$ where k is the depth of the plan, i.e., the longest $<$-chain of actions occurring in P. Let $depth_P(a)$ be the depth of action a in plan $P = \langle A, < \rangle$. Here, $depth_P(a) = 0$ if $\{a' \mid a' \ll a\} = \varnothing$ and $depth_P(a) = 1 + max\{depth_P(a') \mid a' \ll a\}$, otherwise. If the context is clear, we often will omit the subscript P. We assume that the plan starts to be executed at time $t = 0$ and that concurrency is fully exploited, i.e., if $depth_P(a) = k$, then execution of a has been completed at time $t = k + 1$. Thus, all actions a with $depth_P(a) = 0$ are completed at time $t = 1$ and every action a with $depth_P(a) = k$ will be started at time k and will be completed at time $k + 1$. Note that thanks to the above specified concurrency requirement, concurrent execution of actions having the same depth leads to a well-defined result.

A timed state is a tuple (π, t) where π is a state and $t \geq 0$ a time point. We would like to consider the predicted ef-

fect (time state) (π', t') as the result of executing plan P on a given timed state (π, t). To define this relation in a precise way, we will need the following concepts. First of all, let P_t denote the set of actions a with $depth_P(a) = t$, let $P_{>t} = \bigcup_{t'>t} P_{t'}$, $P_{<t} = \bigcup_{t'<t} P_{t'}$ and $P_{[t,t']} = \bigcup_{k=t}^{t'} P_k$.

Secondly, we say that a plan step a is *enabled* in a state π if $dom_O(a) \subseteq O(\pi)$.

Now we can predict the timed state $(\pi', t+1)$ using the timed state (π, t) and the set P_t of to be executed plan steps as follows:

1. whenever an object o does not occur in the range of an action $a \in P_t$, its value in state π' is the same as its value in π, i.e., $\pi(o) = \pi'(o)$;
2. if the object o occurs in the range of an action $a \in P_t$ that is enabled in π, its value changes according to the function specification, i.e., $\pi'(o) = f^{\alpha}(\pi \restriction dom_O(a))(o)$.

Formally, we say that $(\pi', t+1)$ is (directly) generated by execution of P from (π, t), abbreviated by $(\pi, t) \rightarrow_P (\pi', t+1)$, iff the following conditions hold:

1. $\pi'(o) = f^{\alpha}(\pi \restriction dom_O(a))(o)$ for each $a \in P_t$ such that $a \in \alpha$ and for each $o \in ran_O(\alpha)$.
2. $\pi'(o) = \pi(o)$ for each $o \notin ran_O(P_t)$, that is, the value of any object not occurring in the range of an action in P_t should remain unchanged. Here, $ran_O(P_t)$ is a shorthand for the union of the sets $ran_O(a)$ with $a \in P_t$.

For arbitrary values of $t \leq t'$ we say that (π', t') is (directly or indirectly) generated by execution of P from (π, t), denoted by $(\pi, t) \rightarrow_P^* (\pi', t')$, iff the following conditions hold:

1. if $t = t'$ then $\pi' = \pi$;
2. if $t' = t + 1$ then $(\pi, t) \rightarrow_P (\pi', t')$;
3. if $t' > t+1$ then there must exists some state $(\pi'', t'-1)$ such that $(\pi, t) \rightarrow_P^* (\pi'', t'-1)$ and $(\pi'', t'-1) \rightarrow_P (\pi', t')$.

3.1 Normality assumptions

In the above subsection, we defined the (expected) result of a plan execution given the known states of several objects. In general, we do not know the state of every object. More particularly, we do not know the health mode of the objects affected by an action unless we can directly verify the effect of the action execution. More in general, the results of plan execution are uncertain since we need not know the health mode of actions, agents and equipment. Therefore, to predict the effect of a plan execution, we must make assumptions about the (health) state of actions, agents and equipment. We will simply assume, that actions, agents and equipment are in the state nor, unless we have information stating otherwise. Hence, to a given partial state π we add a set of default assumption δ specifying for actions, agents and equipment that they are executed or behaving normally.

Equivalently, such a set of assumptions δ associated with π specifies a partial state π_δ such that:

- $O(\pi_\delta) \subseteq \mathcal{O} - O(\pi)$,
- for each $o \in O(\pi_\delta)$: $\pi_\delta(o) = nor$.

Note that normally, in the absence of actions that can sabotage equipment, the status of the equipment objects in $O(\pi_\delta) \cap \mathcal{E}$ will not change during plan execution.

Using these assumptions δ, we can define the result of a *normal* execution of a plan P by extending the initial partial state π at time point $t = 0$ with the state π_δ and then considering the timed state (π', t') as the result of executing P on the timed state $(\pi \sqcup \pi_\delta, 0)$. That is, (π', t') is the result of normal plan execution on $(\pi, 0)$ iff $(\pi \sqcup \pi_\delta, 0) \rightarrow_P^* (\pi', t')$.

4 Plan diagnosis

By making (partial) observations at different time points of the ongoing plan execution we may establish that there are discrepancies between the expected and the observed plan execution of the plan. These discrepancies indicate that the results of executing one or more actions differs from the way they were planned. Identifying these actions and, if possible, what went wrong in the actions' execution will be called *primary plan diagnosis*. Actions may fail because external factors such as changes in the environmental conditions (the weather), failing equipment or incorrect beliefs of agents. These external factors are underlying causes which are important for predicting how the remainder of a plan will be executed. The *secondary plan diagnosis* aims at establishing these underlying causes.

4.1 Primary plan diagnosis

In [Witteveen *et al.*, 2005; Roos and Witteveen, 2005], Witteveen et al. describe how plan execution can be diagnosed by viewing action instances of a plan as components of a system and by viewing the input and output objects of an action as in and outputs of a component. This made it possible to apply classical MBD to plan execution. Here, we will use a modified version of the plan diagnosis proposed by Witteveen et al.

Since a plan $P = (A, <)$ is a partial order, actions (plan steps) in A are executed only once. Therefore, we could define a *primary diagnosis* in which the execution of some actions may fail, using the set of default assumption δ. However, for other types of diagnosis such as diagnosis of equipment such an approach does not suffice. One of the reasons is that e.g. equipment may start malfunctioning *during* the execution of some action instance and not as the result of it. In general, there may be quite a number of abnormalities that cannot be attributed to the malfunctioning of an action. So we define the more general notion of a *qualification* κ, consisting of triples (o_j, d, t) each specifying an object o_j, the value d of the object and the time point t at which the object o_j takes this value d.

In case of primary diagnosis, the qualification κ is used to change the value (the health mode) of *plan steps*. Hence, the triples have the form $(a, d, depth(a))$ with $a \in \mathcal{A}$ and $d \in D_a$. Note that the plan diagnosis defined in [Witteveen *et al.*, 2005] is a special case of primary diagnosis where the qualification κ consists of triples $(a, ab, depth(a))$ and where for the general fault mode ab the behavior of the action is unknown.

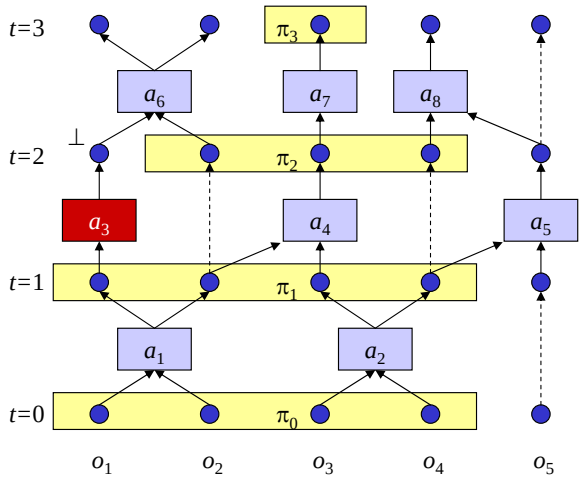

Figure 3: Plan execution with abnormal actions.

Using qualifications, we say that $(\pi', t+1)$ is (directly) generated by execution of P from (π, t) given the qualification κ, abbreviated by $(\pi, t) \rightarrow_{\kappa;P} (\pi', t+1)$, iff the following conditions hold for:

1. $\pi''(o) = d$ for each $o \in \mathcal{O}$ if $(o, d, t) \in \kappa$, else $\pi''(o) = \pi(o)$.
2. $(\pi'', t) \rightarrow_P (\pi', t+1)$.

for some auxiliary state π''.

For arbitrary values of $t \leq t'$ we say that (π', t') is (directly or indirectly) generated by execution of P from (π, t) given the qualification κ, denoted by $(\pi, t) \rightarrow^*_{\kappa;P} (\pi', t')$, iff the following conditions hold:

1. if $t = t'$ then $\pi' = \pi$;
2. if $t' = t+1$ then $(\pi, t) \rightarrow_{\kappa;P} (\pi', t')$;
3. if $t' > t+1$ then there must exists some state $(\pi'', t'-1)$ such that $(\pi, t) \rightarrow^*_{\kappa;P} (\pi'', t'-1)$ and $(\pi'', t'-1) \rightarrow_{\kappa;P} (\pi', t')$.

Example Figure 3 gives an illustration of an execution of a plan. Suppose action a_3 is abnormal and generates a result that is unpredictable ($\perp$). Given the qualification $\kappa = \{(a_3, ab, 1)\}$ and the partially observed state π_0 at time point $t = 0$, we predict the partial states π_i as indicated in Figure 3, where $(\pi_0, t_0) \rightarrow^*_{\kappa;P} (\pi_i, t_i)$ for $i = 1, 2, 3$. Note that since the value of o_1 and of o_5 cannot be predicted at time $t = 2$, the result of action a_6 and of action a_8 cannot be predicted and π_3 contains only the value of o_3. □

Suppose now that we have a (partial) observation $obs(t) = (\pi, t)$ of the state of the world at time t and an observation $obs(t') = (\pi', t')$ at time $t' > t \geq 0$ during the execution of the plan P. We would like to use these observations to infer the health states of the actions occurring in P. Assuming a normal execution of P, we can (partially) predict the state of the world at a time point t' given the observation $obs(t)$: if all actions behave normally, we predict a partial state $\pi'_\varnothing$ at time t' such that $(\pi \sqcup \pi_\delta, t) \rightarrow^*_P (\pi'_\varnothing, t')$. Since we do not require observations to be made systematically, $O(\pi')$ and $O(\pi'_\varnothing)$ might only partially overlap. Therefore, if this assumption holds, the values of the objects that occur in both the predicted state and the observed state at time t' should match, i.e, we should have

$$\pi' \approx \pi'_\varnothing.$$

If this is not the case, the execution of some action instances must have gone wrong and we have to determine an action qualification κ such that the predicted state derived using κ agrees with π'. This is nothing else then a straight-forward extension of the diagnosis concept in MBD [Reiter, 1987; Console and Torasso, 1991] to plan diagnosis:

Definition 1 *Let $P = \langle A, < \rangle$ be a plan with observations $obs(t) = (\pi, t)$ and $obs(t') = (\pi', t')$, where $t < t' \leq depth(P)$ and let the action qualification κ be a set of triples $(a, d, depth(a))$ with $a \in \mathcal{A}$ and $d \in D_a$. Moreover, let $(\pi \sqcup \pi_\delta, t) \rightarrow^*_{\kappa;P} (\pi'_\kappa, t')$ be a derivation assuming an action qualification κ.*

Then κ is said to be a primary plan diagnosis *(action diagnosis) of $\langle P, obs(t), obs(t') \rangle$ iff $\pi' \approx \pi'_\kappa$.*

So in a primary plan diagnosis κ, the observed partial state π' at time t' and the predicted state π'_κ at time t' assuming the action qualification κ agree upon the values of all objects $O(\pi') \cap O(\pi'_\kappa)$ occurring in both states.

Example Consider again Figure 3 and suppose that we did not know that action a_3 was abnormal and that we observed $obs(0) = ((d_1, d_2, d_3, d_4), 0)$ and $obs(3) = ((d'_1, d'_3, d'_5), 3)$. Using the normal plan derivation relation starting with $obs(0)$ we will predict a state $\pi'_\varnothing$ at time $t = 3$ where $\pi'_\varnothing = (d''_1, d''_2, d''_3, \perp_4, \perp_5)$. If everything is ok ($\kappa = \varnothing$), the values of the objects predicted as well as observed at time $t = 3$ should correspond, i.e. we should have $d'_j = d''_j$ for $j = 1, 3$. If, for example, only d'_1 would differ from d''_1, then we could qualify a_6 as abnormal, since then the predicted state at time $t = 3$ using $\kappa = \{(a_6, ab, 2)\}$ would be $\pi'_\kappa = (\perp_1, \perp_2, d''_3, \perp_4, \perp_5)$ and this partial state agrees with the observed state. □

Note that for all objects in $O(\pi') \cap O(\pi'_\kappa)$, the qualification κ provides an *explanation* for the observation π' made at time point t'. Hence, for these objects the qualification provides an *abductive diagnosis* [Console and Torasso, 1990]. For all observed objects in $O(\pi') - O(\pi'_\kappa)$, no value can be predicted given the qualification κ. Hence, by declaring them to be unpredictable, possible conflicts with respect to these objects if a normal execution of all actions is assumed, are resolved. This corresponds with the idea of a *consistency-based diagnosis* [Reiter, 1987].

Diagnosing a sequence of observations In the previous section we described how to diagnose the executions of a plan between two observations at different time points. Here, the observation at the earliest time point corresponds to observed inputs of a system in classical Model-Based Diagnosis while the observations at the latest time point corresponds to the observed outputs in classical Model-Based Diagnosis. During the execution of a plan, however, we may make observations at more than two time points during the execution of the

plan. Unless we observe the complete state of the world at each of these time points, we cannot use successive pairs observations to make the best possible diagnosis of the part of the plan executed between these time points. Hence, we must extend our definition of plan diagnosis to handle sequences of observations.

The use of a sequence of partial observations implies that a diagnosis of the part of a plan executed between time points t_i and t_{i+1} may lead to predictions for the unobserved objects at t_{i+1} that are relevant for diagnosing the part of the plan executed between t_{i+1} and t_{i+2}. Hence, a qualification of the actions executed between two time points t_i and t_{i+1} depends on the qualification of actions executed before t_i.

Definition 2 *Let $P = \langle A, < \rangle$ be a plan with observations $obs(t_1) = (\pi_1, t_1), ..., obs(t_k) = (\pi_k, t_k)$, where $t_1 < t_2 < ... < t_k \leq depth(P)$. Moreover, let κ be an action qualification.*

The action qualification κ is said to be a plan diagnosis of $\langle P, obs(t_1), ..., obs(t_k) \rangle$ iff

- $(\pi_1 \sqcup \pi_\delta, t_1) \rightarrow^*_{\kappa;P} (\pi'_2, t_2)$,
- $(\pi_i \sqcup \pi'_i, t_i) \rightarrow^*_{\kappa;P} (\pi'_{i+1}, t_{i+1})$ *for* $1 < i < k$, *and*
- $\pi_i \approx \pi'_i$ *for* $1 < i \leq k$.

4.2 Secondary plan diagnosis

Actions may fail because of unforeseen (environmental) conditions such as being struck by lightning, malfunctioning equipment or incorrect beliefs of agents. Diagnosing these secondary causes is more difficult since weather, equipment and agents may play a role in the execution of several actions. Moreover, objects such as equipment and weather may go through several unforeseen mode changes.

The above introduced qualification for primary diagnosis can also be used for secondary diagnosis. In fact, we did not use the default assumptions to model qualifications of action in order to have a uniform representation for both failing actions and underlying causes. A *secondary qualification* κ consists of triples (o_j, d, t) where $o_j \in \mathcal{O} - \mathcal{A}$ is an object that changes to the value $d \in D_j$ at time point t. Usually we choose for the time point t the depth $depth(a)$ of the first action instance where change manifests itself. So, for some action a, $t = depth(a)$ and $o_j \in dom_O(a)$.

An object such as an airplane may have several (fault) modes. Between these modes transitions are possible. For example, continuing to drive an overheated engine will cause more severe damage, namely a completely ruined engine. Of course, not every transition between the (fault) modes is valid. For example, a truck with a broken engine cannot become a truck with only a flat tyre without first repairing the truck's engine. Hence, we need *Discrete Event Systems* [Cassandras and Lafortune, 1999] to represent equipment or objects such as the weather.

The specification of the discrete event system consists of the values D_o of an object o, the events $(o, d, t) \in \kappa$ that change the value of the object o, and a *transition function* describing for object o the set of valid transitions. Hence, we assume that for every object $o_j \in \mathcal{O}$ a transition function $tr_j : D_j \rightarrow 2^{D_j}$ has been specified. This transition function describes how the value of an object may change due to, for the agent, unknown events. One of the goals of diagnosis is to determine some of these unknown events.

The values of some objects in the environment may only change due to the execution of actions. For these objects o_j, the transition function is the identity function; i.e.: $tr_j(d) = d$ for every $d \in D_j$. The identity function disallows any change in the object's value that is not the result of an action.

Since the transition function places restriction on the possible transitions of an object, we have to adapt the first item of the specification of $(\pi, t) \rightarrow_{\kappa;P} (\pi', t+1)$.

1. $\pi''(j) = d$ if $(o_j, d, a) \in \kappa$, $d \in tr_j(\pi(j))$ and $t = depth(a)$, else $\pi''(j) = \pi(j)$.
2. $(\pi'', t) \rightarrow_P (\pi', t+1)$

Definition 3 *Let $P = \langle A, < \rangle$ be a plan with observations $obs(t) = (\pi, t)$ and $obs(t') = (\pi', t')$, where $t < t' \leq depth(P)$ and let the action qualification κ be a set of triples (o, d, t) with $o \in \mathcal{O} - \mathcal{A}$ and $d \in D_a$. Moreover, let $(\pi \sqcup \pi_\delta, t) \rightarrow^*_{\kappa;P} (\pi'_\kappa, t')$ be a derivation assuming a qualification κ and the transition functions $tr_j : D_j \rightarrow 2^{D_j}$ for each object $o_j \in \mathcal{O}$.*

Then qualification κ is said to be a secondary plan diagnosis *of $\langle P, obs(t), obs(t') \rangle$ iff $\pi' \approx \pi'_\kappa$.*

The secondary diagnosis can be divided into *agent, equipment* and *environment diagnosis* depending on whether the object o in a triple $(o, d, t) \in \kappa$ belongs to $\mathcal{A}g$, $\mathcal{E}$ or $\mathcal{N}$ respectively.

An interesting special case of secondary diagnosis is *agent diagnosis*. Agents may incorrectly execute an action because of wrong internal beliefs about the agents's environment or about how actions should be executed. One possible cause of such wrong beliefs are incorrect observations of malfunctioning equipment such as sensors. In principle, an agent's incorrect beliefs can be can be modeled using the agent's state. Hence, we need an *agent qualification* (o_j, d, t) with $o_j \in \mathcal{A}g$ describing the incorrect beliefs of an agent that have led to the incorrect execution of actions. This can especially be the case if an action must be achieved by choosing appropriate behaviors. Note that agent diagnosis is closely related to social diagnosis described by Kalech and Kaminka [Kalech and Kaminka, 2003; 2004].

4.3 Applications of diagnosis

As mentioned in the introduction, the information provided by primary and secondary diagnosis can be used to improve the way agents deal with plan failures.

First, to adjust the planning after plan failure, we need an analysis of the expected future execution of the plan and whether the goals will still be reached.Secondary plan diagnosis enables us to the determine which future actions may also be effected by the malfunctioning agents and equipment, and by unforeseen state changes in the environment.

Definition 4 *Let t be the current time point and let κ be a secondary diagnosis of the plan executed sofar. Then the set of future actions that will be effected given the current diagnosis κ is:*

$$\{a \in \mathcal{A} \mid (o_j, d, t') \in \kappa, o_j \in dom_O(a), d \neq nor, depth(a) \geq t'\}$$

Besides identifying the actions that will be effected by agent and equipment failure or by unexpected changes in the environment, we can also determine the goals that can still be reached.

Definition 5 *Let t be the current time point, let π current partial state and let κ be an secondary diagnosis of the plan executed sofar. Moreover, let $(\pi, t) \rightarrow_{\kappa;P} (\pi', depth(P))$. Then the set of goals that can still be realized is given by:*

$$\{g \in G \mid \pi' \models g\}$$

Second, based on the equipment diagnosis, the agents can point out which equipment should be repaired. Moreover, we can view repairs as events that change an equipment object from a failed state into a normal state. Then, we can use definitions 4 and 5 to verify the consequences a certain repair has. This way, agents can consider which repair to choose if repairs are limited (e.g., due to their costs).

Third, it is also important to know the agents responsible for the failures. This information can contribute to negotiation on repairs of plan failure, to division of costs of failed plans or of plan repair, and to avoiding failures of future plans.

As an illustration of different agents that can be responsible for a plan-execution failure, reconsidering the example in the introduction where the agent responsible for the belly landing can be the pilot agent, the maintenance agent, or the airline agent that reduced the maintenance budget.

Here we will present a very simple model of responsibility. We introduce a responsibility function $res : (\mathcal{O} - \mathcal{N}) \rightarrow \mathcal{A}g$ specifying the agent that is responsible for each of the action, agent and equipment objects.

Definition 6 *Let κ be any diagnosis of a plan execution and let $res : (\mathcal{O} - \mathcal{N}) \rightarrow \mathcal{A}g$ be a responsibility function.*

Then for each event $(o, d, t) \in \kappa$, the responsible agent is determined by: $res(o)$.

5 Conclusion

This paper describes a generalization of the model for plan diagnosis as presented in [Witteveen *et al.*, 2005; Roos and Witteveen, 2005]. New in the current approach is (*i*) the introduction of primary and secondary diagnosis, and (*ii*) the introduction of objects representing actions, agents and equipment. The primary diagnosis identifies failed actions and possibly in which way they failed while the secondary diagnosis addresses the causes for action failures. The latter is an improvement over the plan diagnosis presented in [Witteveen *et al.*, 2005; Roos and Witteveen, 2005], where only dependencies between action failures could be described using causal rules. An additional feature of the here proposed approach is that all objects can be modeled as discrete events systems. This enables the description of the unknown dynamic behavior of objects such as equipment over time. The secondary diagnosis then identifies the events behind the state changes of these objects. The results of primary and secondary diagnosis can be used to predict future action failures, to determine the goals that can still be reached and to identify the agents that can be held responsible for plan-execution failures.

References

[Birnbaum *et al.*, 1990] L. Birnbaum, G. Collins, M. Freed, and B. Krulwich. Model-based diagnosis of planning failures. In *AAAI 90*, pages 318–323, 1990.

[Carver and Lesser, 2003] N. Carver and V.R. Lesser. Domain monotonicity and the performance of local solutions strategies for cdps-based distributed sensor interpretation and distributed diagnosis. *Autonomous Agents and Multi-Agent Systems*, 6(1):35–76, 2003.

[Cassandras and Lafortune, 1999] C. G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, 1999.

[Console and Torasso, 1990] L. Console and P. Torasso. Hypothetical reasoning in causal models. *International Journal of Intelligence Systems*, 5:83–124, 1990.

[Console and Torasso, 1991] L. Console and P. Torasso. A spectrum of logical definitions of model-based diagnosis. *Computational Intelligence*, 7:133–141, 1991.

[de Jonge and Roos, 2004] F. de Jonge and N. Roos. Plan-execution health repair in a multi-agent system. In *PlanSIG 2004*, 2004.

[de Jonge *et al.*, 2005] F. de Jonge, N. Roos, and H.J. van den Herik. Keeping plan execution healthy. In *Multi-Agent Systems and Applications IV: CEEMAS 2005, LNCS 3690*, pages 377–387, 2005.

[Horling *et al.*, 2001] Bryan Horling, Brett Benyo, and Victor Lesser. Using Self-Diagnosis to Adapt Organizational Structures. In *Proceedings of the 5th International Conference on Autonomous Agents*, pages 529–536. ACM Press, 2001.

[Kalech and Kaminka, 2003] M. Kalech and G. A. Kaminka. On the design ov social diagnosis algorithms for multi-agent teams. In *IJCAI-03*, pages 370–375, 2003.

[Kalech and Kaminka, 2004] M. Kalech and G. A. Kaminka. Diagnosing a team of agents: Scaling-up. In *AAMAS 2004*, 2004.

[Kleer and Williams, 1989] J. de Kleer and B. C. Williams. Diagnosing with behaviour modes. In *IJCAI 89*, pages 104–109, 1989.

[McCarthy, 1977] John L. McCarthy. Epistemological problems of artificial intelligence. In *IJCAI*, pages 1038–1044, 1977.

[Reiter, 1987] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–95, 1987.

[Roos and Witteveen, 2005] N. Roos and C. Witteveen. Diagnosis of plans and agents. In *Multi-Agent Systems and Applications IV: CEEMAS 2005, LNCS 3690*, pages 357–366, 2005.

[Struss and Dressler, 1989] Peter Struss and Oskar Dressler. "physical negation" integrating fault models into the general diagnostic engine. In *IJCAI*, pages 1318–1323, 1989.

[Witteveen *et al.*, 2005] C. Witteveen, N. Roos, R. van der Krogt, and M. de Weerdt. Diagnosis of single and multi-agent plans. In *AAMAS 2005*, pages 805–812, 2005.