Diagnosis of multi-agent plan execution[*]

Femke de Jonge[1], Nico Roos[1], and Cees Witteveen[2]

[1] Dept of Computer Science, Universiteit Maastricht
P.O.Box 616, NL-6200 MD Maastricht
{f.dejonge,roos}@cs.unimaas.nl

[2] Faculty EEMCS, Delft University of Technology
P.O.Box 5031, NL-2600 GA Delft
witt@ewi.tudelft.nl

Abstract. Diagnosis of plan failures is an important subject in both single- and multi-agent planning. Plan diagnosis can be used to deal with plan failures 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 plan diagnosis are distinguished, namely *agent diagnosis*, *equipment diagnosis* and *environment diagnosis*.

1 Introduction

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

To illustrate the relevance of plan diagnosis, 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, superficial, diagnosis will point out that the agent's action of deploying the landing gear has failed and that the fault mode of

[*] 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.

this action is “landing gear not locked”. We will denote this type of diagnosis as *primary plan diagnosis*. This type of diagnosis focuses on a 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. Primary diagnosis can contribute to plan repair by identifying the failed action and how they failed. Secondary diagnosis can also can contribute to plan repair by pointing out the broken equipment and the misbehaving agents. Plan repair can either plan to fix the broken equipment or to use other equipment. Moreover, information about agents helps to recover and adjust the agents thereby contributing to a better plan execution. Finally, secondary 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.

In this paper we adapt and extend a classical Model-Based Diagnosis (MBD) approach to the diagnosis of plan execution. To enable secondary diagnosis, 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. We introduce an object-oriented description of plans in which objects represent actions, agents, equipment and the environment. The state of these objects may change dynamically by other causes than the execution of planned actions. Primary and secondary diagnosis is used to identify these causes.[3]

To realize the benefits of plan-based diagnosis outlined above, we present in section 3 an adaptation of the object-oriented description of plan execution introduced in [18, 16]. Section 4 shows how this object-oriented plan description enables the formalization of primary and secondary plan diagnosis. But first of all, we will place our approach into perspective by discussing some approaches to plan diagnosis in the following section.

[3] How to implement diagnosis is outside the scope of this paper. For an example of an approach for distributed diagnosis in a multi-agent system, we refer to [15].

2 Related research

To realize plan diagnosis, we adapt and extend the object oriented representation of plans presented in [18, 16]. This representation has been chosen because it is more suited to plan diagnosis than the traditional plan representations such as [10, 2, 9, 17] and it also enables us to extend the classical Model-Based Diagnosis (MBD) approach to the planning domain. Since one of the goals of plan diagnosis is to facilitate plan repair, it might be beneficial to make use of information used to generate a plan and therefore use one of the traditional plan representations. However, we wish to study plan diagnosis independent of any planning approach that is used to generate a plan or that will be used to repair a plan.

Similar to our use of MBD as a starting point of plan diagnosis, Birnbaum et al. [1] apply MBD to *planning agents* relating health states of agents to *outcomes* of their planning activities, but not taking into account faults that can be attributed to actions occurring in a plan as a separate source of errors.

de Jonge et al. [7, 8] present an approach that directly applies MBD 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 [12, 13] 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).

Lesser et al. [3, 11] 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. 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.

Witteveen et al. [18, 16] 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. [7, 8], Kalech and Kaminka [12, 13], and Lesser et al. [3, 11]. The work of Birnbaum et al. [1] is not covered by the proposed formalization since it focuses on the planning activity instead of on plan execution.

3 Plans as systems

Objects In [18] 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 finite 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, four projections of the state σ: $\sigma^{\mathcal{A}}$, $\sigma^{\mathcal{A}g}$, $\sigma^{\mathcal{E}}$ 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 $\mathcal{N}$.

The state $\sigma^{\mathcal{N}}$ of environment objects $\mathcal{N}$ describes the state of the agents' environment at some point in time. For instance, these state descriptions can represent 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 working order of these objects. Their domains consist of the health modes of the action, agent and equipment objects. We assume that each of these 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.[4]

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 $\pi \upharpoonright S$, as the state $\pi' \sqsubseteq \pi$ such that $O(\pi') = S \cap O(\pi)$.

[4] 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.

An important notion for diagnosis is the notion of *compatibility* between partial states. Intuitively, two states π and π' are said to be compatible if there is no essential disagreement about the values assigned to variables in the two states. That is, 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: $\forall o \in \mathcal{O}\ [\pi \sqcup \pi'(o) = max_{\leq}\{\pi(o), \pi'(o)\}]$.

Normality assumptions The health mode of action, agent and equipment objects need not be known explicitly but are usually assumed to be normal: nor. Although it seems reasonable to also assume environment objects such as the weather, to be normal, we cannot make this assumption for every environment object. It makes no sense to assign the state normal to, for instance, a good to be transported. Therefore, we exclude environment objects from our normality assumption. By adding these normality assumptions to a partial state π we create a partial state $\bar{\pi}$ where $\forall o \in \mathcal{O}$ $[\bar{\pi}(o) = nor$ if $\pi(o) = \bot$ and $o \notin \mathcal{N}; \bar{\pi}(o) = \pi(o)$ otherwise$]$.

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 $g \in G_a$, $\pi \models g$.

Action execution Through the execution of a specific action $a \in \mathcal{A}$, the state of environment objects $\mathcal{N}$ and possibly also of equipment objects $\mathcal{E}$ may change. We describe such changes induced by a specific action (also called *plan step*) $a \in \mathcal{A}$ by a (partial) function f^{α} where α is the type of the action (also called *plan operator*) of which a is an instance.

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

where $a \in \alpha \subset \mathcal{A}$ is a specific action of type α, $ag \in \mathcal{A}g$ is the execution agent, $e_1, ..., e_i \in \mathcal{E}$ are the equipment objects required, $n_1, ..., n_i \in \mathcal{N}$ are the environment objects required, 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 equipment objects to occur 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 specifying the 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, likewise $ran_O(o_a) = \{o_{e'_1}, ..., o_{e'_l}, o_{n'_1}, ..., o_{n'_j}\}$.

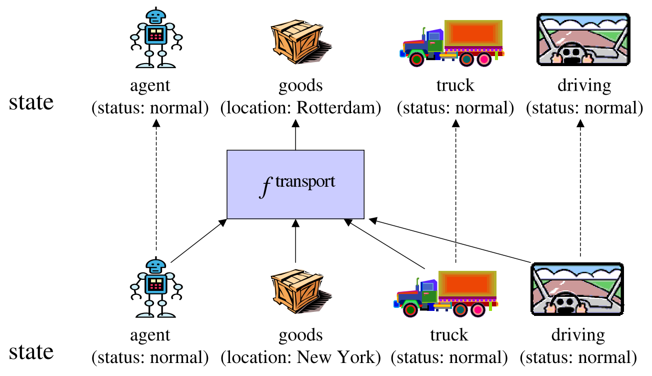

Fig. 1. An action and its state transformation.

The result of an action may not always be known if, for instance, the action fails or if equipment is malfunctioning. Therefore we allow that the function associated with an action maps the value of an object to $\perp$ to denote that the effect of the action on an object is unknown.

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.

Plans A plan is a tuple $P = \langle A, < \rangle$ where $A \subseteq \mathcal{A}$ is a subset of the actions (plan steps) that need to be executed and $<$ is a partial order defined on $A \times A$ where $a < a'$ indicates that the action a must finish before the action a' may start. Note that each action $a \in A$ occurs exactly once in the plan P. 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 actions 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*:

$$\text{If } ran_O(a) \cap dom_O(a') \neq \varnothing \text{ then } a < a' \text{ or } a' < a.$$[5]

Figure 2 gives an illustration of a plan with one abnormally executed action. 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 actions to the objects it uses as inputs and the objects it modifies as its outputs. In this plan, the dependency relation is specified as $a_1 \ll a_3$, $a_1 \ll a_4$, $a_2 \ll a_4$, $a_2 \ll a_5$, $a_4 \ll a_7$, $a_5 \ll a_8$and $a_4 \ll a_6$. Note that the last dependency has to be included because a_6 changes the value of o_2 needed by a_4. The action a_4 shows that not every object occurring in the domain of an action need to be affected by the action.

[5] 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.

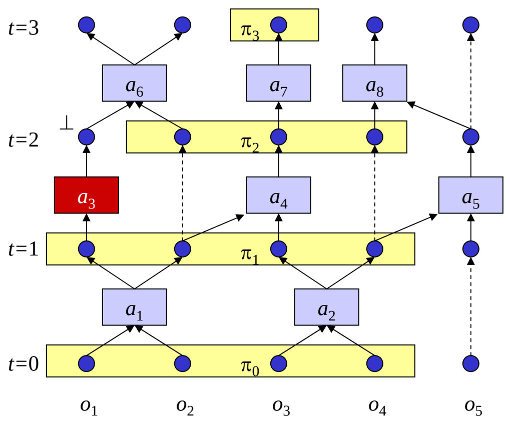

Fig. 2. Plan execution with one abnormal action.

Plan execution For simplicity, we will assume that every action in a plan P takes one time unit 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 effect (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 an action 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 actions. 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 $o \in ran_O(a)$.
2. $\pi'(o) = \pi(o)$ for each $o \notin \bigcup_{a \in P_t} ran_O(a)$, that is, the value of any object not occurring in the range of an action in P_t should remain unchanged.
3. $\pi'(o) = \perp$ otherwise.

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 $(\pi'', t' - 1) \rightarrow_P (\pi', t')$.

Disruptions of plan execution In [18, 16], Witteveen et al. describe how plan execution can be diagnosed by viewing an action of a plan as a component of a system having a *normal* or an *abnormal* behavior, and by viewing the input and output objects of an action as in- and outputs of a component. This view made it possible to apply classical MBD to plan execution. In their view, a diagnosis is a subset $Q \subseteq A$ of abnormally executed actions.

Here, we will use a modified version of the plan diagnosis proposed by Witteveen et al. First of all, 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 $d \in D_{o_j}$ of the object o_j and the time point t at which the object o_j takes this value d. Such a triple might be used to specify a state change at time t of an object o_j to a value $\sigma(o_j) = d$. If the object o_j denotes an action, an agent or equipment, value d will usually indicate some fault mode.

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:

1. For each $o_j \in \mathcal{O}$: $\pi''(o_j) = d$ if $(o_j, d, t) \in \kappa$, and $\pi''(o_j) = \pi(o_j)$ otherwise.
2. $(\pi'', t) \rightarrow_P (\pi', t + 1)$.

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 $(\pi, t) \rightarrow^*_{\kappa;P} (\pi', t')$.

An object such as an airplane may have several (fault) modes. Between these modes transitions are possible. For example, continuing to fly with 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, an airplane with a broken engine cannot become an airplane with only a flat tyre without repairing the engine first. Hence, we need to describe the valid state changes of objects. A *transition function* $tr_j : D_j \rightarrow 2^{D_j}$ will be used to describe the transitions that may occur.

Remark 1. Note that we can view each object o_j as representing a *Discrete Event System* [4]. The triples (o_j, d, t) in a qualification κ describe the unknown *events* that change the state of the object o_j and $tr_j : D_j \rightarrow 2^{D_j}$ is the transition function of the DES. Figure 3 gives an illustration. The goal of diagnosis is to identify these unknown events (o_j, d, t) that have caused the state changes. Also note that actions enforce state changes independent of the transition function tr_j.

The transition function make is possible to judge whether a qualification κ describes valid transitions. It places restrictions on the autonomous state changes (not caused by actions) that may occur. Hence, we must verify whether a qualification κ induces a *sound* derivation given a plan P and the transition functions tr_j. We say that a qualification κ induces a *sound* derivation $(\pi, t) \rightarrow^*_{\kappa;P} (\pi', t')$ iff:

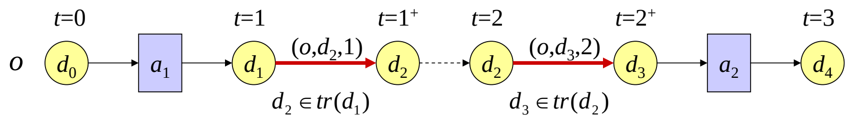

Fig. 3. A Discrete Event System of the object o.

- for no pair of events $(o_j, d, t), (o_j, d', t') \in \kappa$, we have that $t = t'$, and
- for each event $(o_j, d, t') \in \kappa$, if $(\pi, t) \rightarrow^*_{\kappa;P} (\pi', t')$, then $d \in tr^*_j(\pi'(o_j))$.

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. 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 [18, 16], 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 components. Here, we will use a modified version of the plan diagnosis proposed by Witteveen et al. using a qualification κ of events $(a, d, depth(a))$ with $a \in A$. This qualification κ is called an *action qualification*.

Figure 2 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 2, 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, denoted by $\bar{\pi}$, we predict a partial state $\pi'_{\varnothing}$ at time t' such that $(\bar{\pi}, t) \rightarrow^*_P (\pi'_{\varnothing}, t')$. 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 actions 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 to plan diagnosis (cf. [14, 6]).

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 κ induces a* sound *derivation $(\bar{\pi}, t) \rightarrow^{*}_{\kappa;P} (\pi'_{\kappa}, t')$ given the plan P and the transition functions $tr_j : D_j \rightarrow 2^{D_j}$ for each action $a_j \in A$.*

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.

Consider again Figure 2 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 π'_{κ} at time $t = 3$ where $\pi'_{\kappa} = (d''_1, d''_2, d''_3)$. 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 $\pi'_{\kappa} = (d''_3)$ and this partial state agrees with the observed state on the value of o_3.

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* [5]. 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* [14].

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 state changes.

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 $o_j \in dom_O(a)$.

Definition 2. *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 $d \in D_o$. Moreover, let κ induces a* sound *derivation $(\bar{\pi}, t) \rightarrow^*_{\kappa;P} (\pi'_\kappa, t')$ given the plan P and the transition functions $tr_j : D_j \rightarrow 2^{D_j}$ for each object $o_j \in \mathcal{O}$.*

Then the 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. Note that agent diagnosis is related to *social diagnosis* described by Kalech and Kaminka [12, 13] if the agents' health modes are used to describe the agents' incorrect beliefs.

Predicting the future Secondary diagnosis offers an important advantage over primary diagnosis. First, secondary diagnosis enables us to the determine which future actions may also be affected by the malfunctioning agents and equipment, and by unforeseen state changes in the environment.

Definition 3. *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 directly be affected given the current diagnosis κ is:*

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

Second, besides identifying the actions that will be affected, we can also determine the goals that can still be reached.

Definition 4. *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: $\{g \in G \mid \pi' \models g\}$*

Responsible agents Besides knowing the underlying cause of plan execution failures, it is also important to know the agents responsible for the failures. To illustrate this, 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 5. *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 [18, 16]. 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 [18, 16], where only dependencies between action failures could be described using causal rules. An additional feature of the 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 unknown state changes of objects and possibly the agents that can be held responsible for the state changes.

References

1. L. Birnbaum, G. Collins, M. Freed, and B. Krulwich. Model-based diagnosis of planning failures. In *AAAI 90*, pages 318–323, 1990.
2. A. L. Blum and M. L. Furst. Fast planning through planning graph analysis. *Artificial Intelligence*, 90:281–300, 1997.
3. 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.
4. C. G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, 1999.
5. L. Console and P. Torasso. Hypothetical reasoning in causal models. *International Journal of Intelligence Systems*, 5:83–124, 1990.
6. L. Console and P. Torasso. A spectrum of logical definitions of model-based diagnosis. *Computational Intelligence*, 7:133–141, 1991.
7. F. de Jonge and N. Roos. Plan-execution health repair in a multi-agent system. In *PlanSIG 2004*, 2004.
8. 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.
9. D. McDermott et al. The pddl planning domain definition language. In *The AIPS-98 Planning Competition Committee*, 1998.
10. R. E. Fikes and N. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 5:189–208, 1971.
11. 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.
12. M. Kalech and G. A. Kaminka. On the design ov social diagnosis algorithms for multi-agent teams. In *IJCAI-03*, pages 370–375, 2003.
13. M. Kalech and G. A. Kaminka. Diagnosing a team of agents: Scaling-up. In *AAMAS 2004*, 2004.
14. R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–95, 1987.
15. N. Roos, A. ten Teije, and C. Witteveen. A protocol for multi-agent diagnosis with spatially distributed knowledge. In *AAMAS 2003*, pages 655–661, 2003.
16. 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.
17. H. Tonino, A. Bos, M. de Weerdt, and C. Witteveen. Plan coordination by revision in collective agent based systems. *Artificial Intelligence*, 142:121–145, 2002.
18. 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.