Using DESs for Temporal Diagnosis of Multi-agent Plan Execution*

Femke de Jonge, Nico Roos, and Huib Aldewereld

Dept. of Computer Science, Universiteit Maastricht
P.O. Box 616, NL-6200 MD Maastricht
{f.dejonge,roos,h.aldewereld}@micc.unimaas.nl

Abstract. The most common reason for plan repair are the violation of a plan's temporal constraints. Air Traffic Control is an example of an area in which violations of the plan's temporal constraints is rather a rule than an exception. In such domains there is a need for identifying the underlying causes of the constraint violations in order to improve plan repairs and to anticipate future constraint violations. This paper presents a model for identifying the causes of the temporal constraint violations.

1 Introduction

Violation of a plan's temporal constraints is one of the most common problems during plan execution. In air traffic control, for instance, violation of a plan's temporal constraints is rather a rule than an exception requiring constant adaptions of aircraft's plans. Common causes are problems in luggage handling, security issues, no-shows of passengers, unforseen changes in the weather conditions, and so on and so forth. In order to repair plans, accurate information about the cause of the problem is important. It enables planners to come up with better plan repairs, thereby avoiding fire fighting tactics. This requires that we not only identify *primary* cause, that is, failing plan step(s) causing constraint violations, but also the *secondary* cause, that is, failing equipment, unforseen changes in the environment and malfunctioning agents that are responsible for plan step failures.

In order to make a diagnosis of temporal constraint violations, we need a model of a plan's temporal execution. In this paper we will investigate the use of discrete event systems (DES) [1] for this purpose. Section 3 discusses how discrete event systems can be adapted for this purpose, and Sect. 4 discusses how the resulting model of a plan can be used to make predictions. Section 5 defines plan-execution diagnosis for temporal constraint violations. We will argue that diagnosis of temporal constraint violations differs from standard diagnosis using DESs [2,3,4]. Section 6 presents a small example and Sect. 7 concludes the paper. First, to place our approach into perspective, we discuss some some related work.

* 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.

P. Petta et al. (Eds.): MATES 2007, LNAI 4687, pp. 61–72, 2007.
© Springer-Verlag Berlin Heidelberg 2007

2 Related Work

In this section we briefly discuss some other approaches to plan diagnosis and subsequently some approaches to diagnosis using discrete event systems.

Plan Diagnosis. There are several papers addressing different aspects of plan diagnosis. These papers discuss diagnosis of the planning agent [5], diagnosis of a plan consisting of a hierarchy of behaviors [6,7], diagnosis of the execution of a single task of a plan [8,9], and diagnosis of plans consisting of several plan steps (task) [10,11,12,13,14]. None of the papers, however, address the violation of a plan's temporal constraints. This also holds for the approach of de Jonge et al. [15,16], which is closely related to the approach proposed in this paper. de Jonge et al. apply discrete event systems to describe linear plans of individual agents. However, temporal aspects of the agents' plan cannot be modeled. The agents' plans may interact through constraints over the states of plan steps (representing for instance resource constraints). Diagnosis is subsequently applied to identify disruption events causing constraint violations. If disruption events are observable future constraint violations can be predicted and diagnosis is used to propose repair events to avoid these future constraint violations. This paper extends the approach of de Jonge et al. by enabling diagnosis of temporal constraint violations.

Discrete Event Systems. Discrete Event Systems (DES) are a modeling method of real world systems based on finite state machines (FSM) [1]. In a DES a finite set of states describes at some abstraction level the state of a real world system. State changes are caused by events and a transition function specifies the changes triggered by the events. The events are usually observable control events. However, unobservable failure events mays also cause state changes. Diagnosis of a DES aims at identifying the unobserved failure events based on a trace of observable events [2]. Essentially, this is a form of abductive diagnosis. Note that the trace of observable events depends on the state of the system and the transition function. Therefore, a DES is sometimes viewed as a machine accepting a language of observable and unobservable events.

To model a system, usually one starts modeling individual components using Discrete Event Systems. Interactions between the components are described by exchanging events between the components. For diagnosis, these DESs may be combined into a global DES [17]. More recently, methods for diagnosing coupled DESs without first creating a global DES have been proposed [3,4].

3 Modeling Plan Execution

This section formalizes the description of the plan execution to be diagnosed.

The Environment. In a domain such as air traffic control, the environment has an important influence on the intended execution of a plan. Unforseen changes in the state of the environment such as snow on runways or strong headwinds, may influence the temporal execution of a plan. In order to identify this type of influences using diagnosis, we first need to model them.

The deviations in the strength of the aircraft's headwind is a continuous function over time. Obviously, we will never have enough information to identify this function in our diagnostic process. Therefore, we should abstract from the continuous function and use abstract values such as *strong-tailwind*, *tailwind*, *no-wind*, *headwind* and *strong-headwind* instead. These values hold for certain time intervals. Hence, we define the environment by a set of objects $\mathcal{O}^{env}$ where with each object $o \in \mathcal{O}^{env}$ a finite set of possible states S_o is associated.

A state change of one object may cause a state change in another object. For instance heavy snowfall may influence the state of the runway on an airport. If a state change of, for instance, the weather immediately influences the state of the runway, Discrete Event Systems (DES) [1] would be an obvious choice to describe the causal dependencies between objects. However, heavy snow fall does not immediately causes the closure of a runway. Therefore, we introduce Discrete Event Systems that can generate an event after an object has been in a certain state for some period of time. Using such a time delayed event we may change the state of the runway after it has been snowing for a specified period of time. Unfortunately, this does not solve the problem completely. How long it will have to snow before snowfall changes the state of the runway, may depend on the condition of the runway such as the presence of salt, and on other events such as snow removal activities. Therefore, in this example the time delayed event must be generated by runway and not by the weather. Figure 1 gives an illustration.

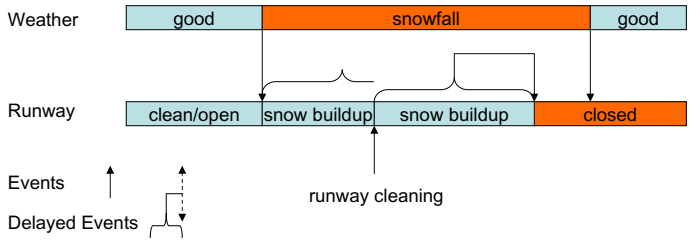

Fig. 1. An application of delayed events

Note that the event generated by the weather causes the state change of the runway from 'clean/open' to 'snow buildup'. The event 'runway cleaning' indicating the removal of snow from the runway, causes a state change from 'snow buildup' to 'snow buildup'. The latter event may seem odd. The effect of a reflexive state transition is that all timers of the time delayed events are reset. Also note that in a more refined model one may also distinguish levels of snow that have been buildup on the runway.

Summarizing the above, an environment object $o \in \mathcal{O}^{env}$ is modeled as a Discrete Event System in which output events can be generated after the object o has been in some state for some specified period of time. This is described by rules of the form:

$$(S_o \times (\mathbb{R} \times \mathbb{R}) \Rightarrow E_o^{out} \times 2^{\mathcal{O}})$$

These rules specify the state to which the rule applies and the time the object must be in this state. Note that for the latter we use an interval $[t_{\min}, t_{\max}] \subseteq (\mathbb{R} \times \mathbb{R})$. The

reason for using a time interval is the following. Since an object's state in the model is an abstract representation of the actual state, we cannot know exactly how long the object must be in a state before the event will occur. We can model this uncertainty using intervals.

Also note that these rules specify the output event that is generated and a set of objects $O^{des} \subseteq \mathcal{O}$ to which the event is sent. The set of objects $\mathcal{O}$ contains the environment objects as well as objects for describing plan steps, equipment and agents.

Plans. Since the execution of a step (action) of a plan may depend on the state of the environment objects and since the state of environment objects may change during the execution, we cannot use a representation in which plan steps are treated as atoms. Therefore, most representations are not suited for our purpose. What we need is a representation that enables us to state that a plan step finishes too late because it started too late or because a delay occurred during execution as a result of unforseen changes in the environment. Hence, we should be able to assign states to plan steps and these states may change during the execution of a plan step. This suggests that we should also use Discrete Event Systems to model plan steps [16]. We therefore introduce a special set of objects $\mathcal{O}^{plan}$ to represent plan steps. Figure 2 gives an illustration.

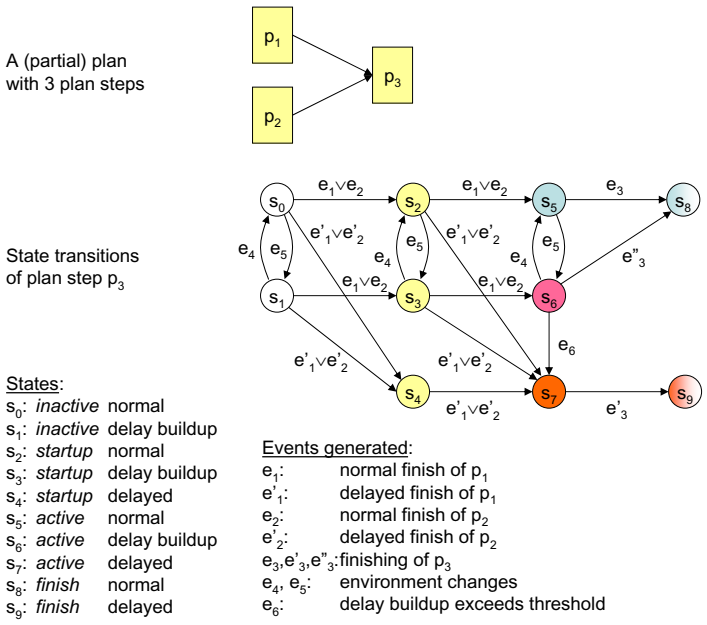

Fig. 2. Modeling a plan step

Summarizing the above, a plan step p is modeled as a Discrete Event System in which the set of state S_p contains:

– a special *initial* state representing the expected initial situation before the start of executing the plan step,

- A set of inactive states representing environment changes that may influence the execution of the plan step,
- a set of startup states representing (*i*) that some but not all preceding plan steps have finished as well as how they have finished, and (*ii*) environment changes that may influence the execution of the plan step,
- a set of active state representing how the plan step is executed, and
- a set of finish states of the plan step.

These special states are all disjunct. Note that no startup states are needed in a linear plan.

Based on the state of a plan step p, at a certain time point an event must be generated that brings the plan step into a finishing state. (One of the events e_3, e_3', e_3'' in the example of Fig. 2.) We therefore need rules for generating events based on the scheduled finishing times and the current state of plan steps.

$$(S \times (\mathbb{R} \times \mathbb{R}) \rightarrow E^{out} \times 2^{\mathcal{O}})$$

So, somewhere in the interval $[t_{\min}, t_{\max}] \subseteq (\mathbb{R} \times \mathbb{R})$ an output event $e \in E^{out}$ is generated and is sent to the objects $O \subseteq \mathcal{O}$. If the object is not in the state specified by the rule during the interval $[t_{\min}, t_{\max}]$, the rule will not generate an event. In this way we can, for instance, specify the finish events of an plan step, as illustrated in Fig. 3. Finally, if the object is in the state specified by a rule and subsequently changes to another state during the interval specified by the rule, then the event is either generated by this rule or is generated by another object or is an external event. In the latter two cases the rule will not generate an event because the state of the object has changed.

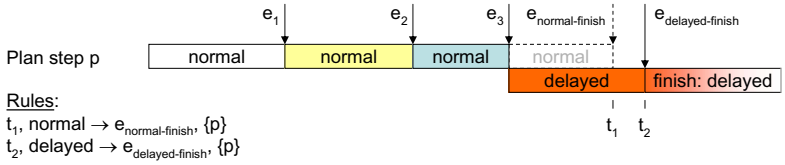

Fig. 3. An application of time generated events

The Discrete Event System. Based on the above described requirements we can give a specification of the objects we will use to describe plan executions by a group of agents in some environment. We assume that the set of objects $\mathcal{O}$ can be partitions into plan steps $\mathcal{O}^{plan}$, agents $\mathcal{O}^{ag}$, equipment $\mathcal{O}^{eq}$ and environment objects $\mathcal{O}^{env}$. Each object $o \in \mathcal{O}$ is described by a Discrete Event System [1].

Definition 1. *An object $o \in \mathcal{O}$ is a Discrete Event System $(S_o, s_o, E_o^{in}, E_o^{out}, \tau_o, \rho_o)$ where*

- *S_o is a set of states,*
- *$s_o \in S_o$ is the initial state at time point 0,*
- *E_o^{in} is the set of events the object may react to,*

- E_o^{out} *is the set of events the object may generate,*
- τ_o *is a set of state transition rules of the form* $(S_o \times E_o^{in} \rightarrow S_o)$*, and*
- ρ_o *is a set consisting of duration generated event rules of the form:*

$$((S_o \times (\mathbb{R} \times \mathbb{R}) \Rightarrow E_o^{out} \times 2^O))$$

and time generated event rules of the form:

$$(S_o \times (\mathbb{R} \times \mathbb{R}) \rightarrow E_o^{out} \times 2^O).$$

Above we did not discuss agent and equipment objects explicitly. Equipment objects do not differ much from environment objects. For equipment we distinguish a special state 'normal' and possibly several states for describing malfunctions. Also for agents we distinguish a state 'normal'. Agents may also have several other health state as well as states indicate beliefs about the environment, which might be incorrect.

Timed Events. To describe the (expected) occurrence of events, we introduce so called *timed events*. A *timed event* is a couple $(e, [t, t'])$ where $e \in E$ is an event and $[t, t'] \subseteq (\mathbb{R} \times \mathbb{R})$ is a time interval in which e occurs. A set of timed events will be denoted by: Π.

Constraints. A plan description normally consists of a set of plan steps with precedence relations between the plan steps. We can distinguish two types of precedence relation, namely those that describe the order of plan steps needed to guarantee the desired effect of a plan and precedence relations that have been added to avoid resource conflicts. Although all precedence relations can be modeled using the above described events, we will use a different description for precedence relations that are added for avoiding resource conflicts. When, for instance, an aircraft is delayed, it may be better to change the planned landing sequence of aircraft so that other aircraft can still arrive as scheduled. Therefore, we will model resource constraints separately as constraints over the states of plan steps requiring the same resource. These constraints can be used to specify for instance that a combination of plane a being delayed and plane b being early is not allowed because both planes are scheduled on the same gate.

Definition 2. *A constraint* ctr *over* n *objects* $(S_{o_i}, s_{o_i}, E_{o_i}^{in}, E_{o_i}^{out}, \tau_{o_i}, \rho_{o_i})$ *is a tuple:*

$$\langle o_1, ..., o_n, AS \rangle$$

with $AS \subseteq S_{o_1} \times ... \times S_{o_n}$*, describing states of objects that are allowed at the same time.*

A set of constraints will be denoted by C*.*

Constraints can be observed to hold or to be violated during the execution of a plan. *Timed constraints* are used to denote the time interval in which the constraint is observed *to hold* or *to be violated*: $(ctr, [t, t'])$ and $(\neg ctr, [t, t'])$, respectively, where $ctr \in C$ is a constraint and $[t, t'] \subseteq (\mathbb{R} \times \mathbb{R})$ is a time interval in which ctr is observed to hold or to be violated. A set of timed constraints will be denoted by $\mathcal{C}$.

The Model. Using the discrete event systems introduced in this section and applying the modeling method discussed, we can now build a model of possible ways of executing a plan. To summarize, we have to:

- describe the environment objects $\mathcal{O}^{env}$, their state transitions, their states at time point 0, and the rules generating events;
- describe the states of each plan step $\mathcal{O}^{plan}$, their state transitions, states of the plan steps at time point 0, and, using information about the schedule of each plan step, the rules generating (finishing) events;
- if necessary, also describe the equipment $\mathcal{O}^{eq}$ and the agents $\mathcal{O}^{ag}$; and
- describe the constraints $\mathcal{C}$ that must hold between states of different agents' plan steps.

Any model has a boundary of what is and what is not modeled. For instance, we may model the state of the weather but not the processes that determine the changes of the weather. Still, based on the weather forecasts we wish to be able to adapt state of the weather object. External events can be used for this purpose. An external event is an input event of an object that is not generated by some object; i.e., it is not an output event of some object and there is no rule generating it. So, $E^{ext} = \bigcup_{o \in \mathcal{O}} E_o^{in} - \bigcup_{o \in \mathcal{O}} E_o^{out}$. The *expected* external timed-events are external events occurring within specific time intervals. We use the set Π^{exp} to denote expected external timed-event.

The objects $\mathcal{O}$, the expected external timed-events Π^{exp} and the constraints C gives us the execution model $M = (\mathcal{O}, \Pi^{exp}, C)$ with $\mathcal{O} = \mathcal{O}^{env} \cup \mathcal{O}^{plan} \cup \mathcal{O}^{eq} \cup \mathcal{O}^{ag}$.

4 Making Predictions

The model proposed in the previous section enables us to simulate, at an abstract level, the normal and the abnormal execution of a plan. Deviations with observations made enables us to formulate the diagnostic problem.

Simulation. An execution model $(\mathcal{O}, \Pi)$ consisting of objects $\mathcal{O}$ and external timed events Π can be used to simulate a multi-agent plan execution starting from the initial state of the plan at time point 0. The history of an object $o \in \mathcal{O}$ describes for each object a sequence of timed events that generate state changes in the object o. Together with each state change of o, the history also describes the timed events generated by the rules of the object o during the current state of o.

Definition 3. *Let $o \in \mathcal{O}$ be an object with DES $(S_o, s_o, E_o^{in}, E_o^{out}, \tau_o, \rho_o)$.*
A history H_o of the object $o \in \mathcal{O}$ is a sequence of the form:

$$H_o = \langle\ (e_0, t_0, s_0, \langle (e_{0,0}, t_{0,0}, O_{0,0}^{des}), ..., (e_{0,m}, t_{0,m_0}, O_{0,m_0}^{des}) \rangle),\ \ldots\ ,$$
$$(e_n, t_n, s_n, \langle (e_{n,0}, t_{n,0}, O_{n,0}^{des}), ..., (e_{n,m_n}, t_{n,m_n}, O_{n,m_n}^{des}) \rangle)\ \ \rangle$$

where

- $e_0 = nill$, $t_0 = 0$ *and* $s_0 = s_o$;
- $t_{i-1} < t_i$ *and* $(s_{i-1}, e_i \rightarrow s_i) \in \tau_o$ *for each* $0 < i \leq n$;
- *for each timed event* $(e_{i,j}, t_{i,j})$ *there is either a rule:*
  - $(s_i, [t, t'] \rightarrow e_{i,j}, O_{i,j}^{des}) \in \rho_o$ *such that* $t_{i,j} \in [t, t'] \cap [t_i, t_{i+1}] \neq \varnothing$, *or*
  - $(s_i, [t, t'] \Rightarrow e_{i,j}, O_{i,j}^{des}) \in \rho_o$ *such that* $t_{i,j} \in [t_i + t, t_i + t'] \cap [t_i, t_{i+1}] \neq \varnothing$

  generating this timed event.

The history of the whole model consists of the history of all the objects. Of course, the history of the objects are not independent of each other. Every event that causes a state transition of an object is an external event, or an event that is *generated* by a rule of a possibly different object. Moreover, an event generated by a rule of an object must be *sent* to every object in the corresponding set O^{des}.

Definition 4. *A history H is the set of histories of all objects in $\mathcal{O}$:*

$$H = \{H_o \mid o \in \mathcal{O}\}$$

where for each object $o \in \mathcal{O}$ and for each element

$$(e_i, t_i, s_i, \langle (e_{i,0}, t_{i,0}, O^{des}_{i,0}), ..., (e_{i,m_i}, t_{i,m_i}, O^{des}_{i,m_i})\rangle) \in H_o,$$

(e_i is generated) *$e_i \in E^{ext}$, or there is a $(e_j, t_j, s_j, \langle ..., (e_{j,k}, t_{j,k}, O^{des}_{j,k}), ...\rangle) \in H_{o'}$ such that $e_i = e_{j,k}$, $t_i = t_{j,k}$ and $o \in O^{des}_{j,k}$, and*

($e_{i,j}$ is sent) *for each $(e_{i,j}, t_{i,j}, O^{des}_{i,j})$ and for each object $o' \in O^{des}_{i,j}$, either $e_{i,j}$ is not applicable in the state of o' at $t_{i,j}$, or there is a $(e_k, t_k, s_k, \langle ...\rangle) \in H_{o'}$ such that $e_{i,j} = e_k$ and $t_{i,j} = t_k$.*

An event e is applicable *in the state s of an object o iff $(s, e \rightarrow s') \in \tau_o$.*

Satisfiability and Consistency of Timed Constraints and Events. Since a history H of the objects $\mathcal{O}$ specifies all the occurring events, we can check whether an (observed) timed event $(e, [t, t'])$ is satisfied by the history H. That is, whether there exists an object $o \in \mathcal{O}$ and a history of that object H_o such that $(e_j, t_j, s_j, \langle ...\rangle) \in H_o$, $e = e_j$ and $t_j \in [t, t']$. Since we can view a history H as a possible description of the world and since we can view a timed event $(e, [t, t'])$ as a *proposition*, we say that the history *satisfies* the timed event: $H \models (e, [t, t'])$.

Similarly we can check whether the timed constraints $(ctr, [t, t'])$ and $(\neg ctr, [t, t'])$ with $ctr = \langle o_1, ..., o_n, AS\rangle \in C$ are *satisfied* by a history H, denoted by $H \models (ctr, [t, t'])$ and $H \models (\neg ctr, [t, t'])$, respectively. So, also the timed constraints are viewed as *propositions*.

An execution model $(\mathcal{O}, \Pi)$ specifies a set of histories because of the uncertainty that results from the use of time intervals. Some of the histories may satisfy a proposition φ; i.e. a timed constraint or a timed event, while others do not. If the propositions describe observations, then as long as there is one history H satisfying the propositions, there is no conflict between the observations and the execution model $(\mathcal{O}, \Pi)$. In other words, the observations are *consistent* with the execution model. If one or more observations are *inconsistent* with the execution model $(\mathcal{O}, \Pi)$, we know that the current set of external timed events Π needs revision. Diagnosis will give us a revised set of external events.

5 Diagnosis and Explanation

Some of the observed timed constraints $\mathcal{C}$ and some of the observed timed events Π^{obs} may not be consistent with the execution model $M = (\mathcal{O}, \Pi^{exp}, C)$. These inconsistencies indicate that the expected external events did not occur as specified by Π^{exp}.[1]

[1] We assume the absence of errors in the description of the objects $\mathcal{O}$.

Hence, we can formulate a *plan execution diagnosis problem*: $(M, \Pi^{obs}, \mathcal{C}^{obs})$. This section defines a diagnosis and an explanation given a plan execution diagnosis problem.

Diagnosis. Diagnosis of plan execution differs from traditional diagnosis of discrete event systems [2,3,4]. Traditional diagnosis of discrete event systems is *abductive diagnosis*. In abductive diagnosis, the model of the plan execution extended with a diagnosis must *satisfy* all observed events and all constraints. As we saw in the previous section, because of the uncertainty in the model of the plan execution, this requirement is too strong. What we need is *consistency-based diagnosis*. Consistency-based diagnosis enables us to identify the set of external timed events that resolve the inconsistencies between the execution model M, the observed timed events Π^{obs} and the observed timed constraints $\mathcal{C}^{obs}$.

Definition 5. *Let $(M, \Pi^{obs}, \mathcal{C}^{obs})$ be a plan execution diagnosis problem where $M = (\mathcal{O}, \Pi^{exp}, C)$ is a model of the intended plan execution. Moreover, let Δ with $\Delta \subseteq \{(e, [t, t']) \mid e \in E^{ext}, 0 \leq t \leq t'\}$ be a* candidate diagnosis.

Δ is a diagnosis *of a plan execution iff $((\mathcal{O}, \Delta), \Pi^{obs}, \mathcal{C}^{obs})$ is consistent; i.e., there is a history H for $(\mathcal{O}, \Delta)$ such that $H \models \Pi^{obs} \cup \mathcal{C}^{obs}$.*

Preference Criteria. There may be several diagnoses Δ according to Definition 5. The quality of these diagnoses need not be the same. Preference criteria are used to select the subset of the diagnoses. Usually, the preference criteria select the most probable diagnoses. A criterium that is often used for diagnoses is preferring diagnoses that minimize the difference with the normal state of affairs. In plan diagnosis this would be the external timed events Π^{exp}.

A difficulty in comparing Π^{exp} and Δ is that there are transitions to the same state s starting from different states that are triggered by different events. For instance, two events causing a transition to a state representing strong winds, one from a state representing no wind and one from the state representing a light breeze. If the expected light breeze did not occur, we should still be able to infer that the change to strong winds did occur. Therefore we will restrict the external events to so called *absolute* events. An absolute event causes a transition to a new state independent of the previous state thereby simplifying comparison of external events.

Definition 6. *An external event $e \in E^{ext}$ of an object $o \in \mathcal{O}$ is an* absolute *event iff for every $s, s' \in S_o$: $\tau_o(s, e) = \tau_o(s', e)$.*

The use of absolute events enables us to determine the difference between the expected timed events Π^{exp} and a diagnosis Δ. The difference consists of two aspects, (i) the unexpected timed events that occurred according to the diagnosis: $\Delta \ominus \Pi^{exp}$, and (ii) the expected timed events that did not occur according to the diagnosis: $\Pi^{exp} \ominus \Delta$. Here, the function $\ominus$ is defined as:

$$(X \ominus Y) = \{(e, [t, t']) \in X \mid \forall (e, [t'', t''']) \in Y : [t, t'] \cap [t'', t'''] = \varnothing\}.$$

We prefer diagnoses Δ that *minimize* the differences with Π^{exp} if the probability that differences with Π^{exp} occur is sufficiently small.

Explanation. In our application domain of air traffic control one often claims that during normal daily operation all relevant events, including the external events, are observable. This does not imply that no constraint violation will occur when agents execute their plans. On the contrary, air traffic controllers are working around the clock to avoid incidents. Clearly, if all external events that have occurred, are observed, then $\{(e,[t,t']) \mid (e,[t,t']) \in \Pi^{obs}, e \in E^{ext}\}$ is a diagnosis. However, such a diagnosis does not give an adequate explanation of an observed constraint violation.

For the purpose of plan repair, distributing cost of a plan repair, improvements of future plans, and so on and so forth, we would like to know which external events are accountable of the observed constrain violation during some time interval. A diagnosis does not provide this information. It only specifies the expected and unexpected external events that occurred without linking them to a specific observed constraint violation. So, given a diagnosis, an *explanation* of an observed constraint violation must specify the presence of unexpected external timed events and the absence of expected timed events causing the constraint violation.[2]

Determining an explanation for a constraint violation is not straight forward. To illustrate this, consider the following example. An aircraft that has a delayed departure may still arrive on time at its destination because of the absence of strong headwinds. However, because no gate is available after landing, the aircraft has a delayed arrival resulting in a constraint violation. The constraint violation could be explained by considering the external event causing the delayed departure while ignoring in the explanation the absence of strong headwinds and the unavailability of a gate after landing. Clearly, this is not a proper explanation of the delayed arrival because the plane landed on time.

How do we determine the external events that explain a proposition (an observed timed event or an observed timed constraint)? First, observe that for every proposition, there is a non-empty set of objects the history of which determine the satisfiability of the proposition. Second, the use of absolute external events implies that we do not have to consider any event changing the state of an object o that occurs before an absolute external event e changing the state of o. We do have to consider every event e' generated by and event rule of an object o' changing the state of o after e. We also have to consider the absent absolute events that where expected to occur after e.

Definition 7. *Let $(M, \Pi^{obs}, \mathcal{C}^{obs})$ be a plan diagnosis problem and let Δ be a diagnosis. Moreover, let $\varphi = (\epsilon, [t_\epsilon, t'_\epsilon])$ be a proposition for which we seek an explanation. Finally, let us view an observation of a timed constraint as a timed event to which we can extend precedence relation $\prec_H$ induced by a history H.*

$(\mathcal{X}^a, \mathcal{X}^p)$ with $\mathcal{X}^p \subseteq (\Delta \ominus \Pi^{exp})$ and $\mathcal{X}^a \subseteq (\Pi^{exp} \ominus \Delta)$ is an explanation *of φ iff*

1. *$\mathcal{X}^p \cup (\Pi^{exp} \ominus \mathcal{X}^a)$ is a preferred diagnosis of the plan execution diagnosis problem $(M, \{\varphi\})$,*
2. *for no $(\mathcal{Y}^p, \mathcal{Y}^a)$ with $\mathcal{Y}^p \subseteq (\Delta \ominus \Pi^{exp})$ and $\mathcal{Y}^a \subseteq (\Pi^{exp} \ominus \Delta)$:*
   $\mathcal{X}^p \cup \mathcal{Y}^p \cup (\Pi^{exp} - \mathcal{X}^a - \mathcal{Y}^a)$ is not *a diagnosis of the plan execution diagnosis problem $(M, \{\varphi\})$.*

Note that $\mathcal{X}^p$ denotes the unexpected external events are present in the diagnosis Δ, and $\mathcal{X}^a$ denotes the expected external events that absent in the diagnosis Δ. Also note that

[2] Here, we use a pragmatic interpretation of the concept 'causes'.

the second requirement in the above definition is needed because of *non-monotonicity* of explanations.

6 Example

This section illustrates the relevance of the model in our application domain, the field of air traffic control, using a small example.

Flight KL1243 to DeGaulle Paris, which is docked at gate E11, is delayed because it has to wait for passengers (the expected off-block event after which the aircraft is to taxi to the runway does not occur at the planned time, but occurs 15 minutes later). After further investigation it becomes apparent that the passengers that KL1243 is waiting for are transfers from flight D845. Flight D845, from Heathrow London, was delayed due to strong headwinds, and only just began de-boarding at gate D21.

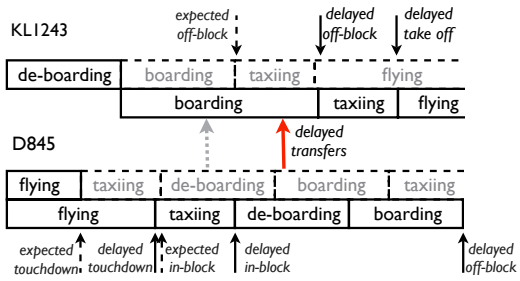

Fig. 4. Example diagnosis

Figure 4 shows the model of the schedule for flights KL1243 and D845. As can be seen, the expected in-block time of D845, which is the start of the de-boarding, was expected before the off-block time of KL1243, but due to (unexpected) weather changes has been delayed. This delay causes a delay in the boarding of KL1243, which is noted by the delay in the occurrence of it's off-block event.

Clearly, diagnosis identifies the explanation: 'strong headwinds: London to Amsterdam'. Note that this diagnosis can be used to predict that other flights from the same direction will probably be delayed as well (until the weather changes).

7 Conclusion

Identifying causes of temporal constraint violations during plan execution is an important issue in many domains, especially in our application domain of air traffic control. Identifying causes of these constraint violations support plan repair and can help improving new plans. In this paper we have investigated whether a plan can be modeled using Discrete Event Systems for the purpose of diagnosing cause of temporal constraint violations. We have shown that plan execution can be modeled using DESs and that a diagnosis can be defined in terms of the presence or absence of external events. Such a diagnosis

describes the unforseen state changes in agents executing the plan, equipment used to execute the plan and the environment in which the plan is executed. Finally, we have shown that explanations for individual constraint violations can be determined.

In future work we will investigate whether we can abstract from time information of events and of event generation rules. Moreover, we will investigate efficient distributed implementations of the model based on approaches proposed in [16] and [4].

References

1. Cassandras, C.G., Lafortune, S.: Introduction to Discrete Event Systems. Kluwer Academic Publishers, Boston, MA (1999)
2. Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., Teneketzis, D.: Diagnosibility of discrete event systems. IEEE Transactions on Automatic Control 40, 1555–1575 (1995)
3. Baroni, P., Lamperti, G., Pogliano, P., Zanella, M.: Diagnosis of large active systems. Artificial Intelligence 110(1), 135–183 (1999)
4. Pencolé, Y., Cordier, M.: A formal framework for the decentralised diagnosis of large scale discrete event systems and its application to telecommunication networks. Artificial Intelligence 164(1–2), 121–170 (2005)
5. Birnbaum, L., Collins, G., Freed, M., Krulwich, B.: Model-based diagnosis of planning failures. In: AAAI 90, pp. 318–323 (1990)
6. Kalech, M., Kaminka, G.A.: On the design of social diagnosis algorithms for multi-agent teams. In: IJCAI-03, pp. 370–375 (2003)
7. Kalech, M., Kaminka, G.A.: Diagnosing a team of agents: Scaling-up. In: AAMAS 2005, pp. 249–255 (2005)
8. Carver, N., Lesser, V.: 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)
9. Horling, B., Benyo, B., Lesser, V.: Using self-diagnosis to adapt organizational structures. In: Proc. 5th Int'l Conf. on Autonomous Agents, pp. 529–536. ACM Press, New York (2001)
10. de Jonge, F., Roos, N., Witteveen, C.: Primary and secondary plan diagnosis. In: DX'06. 17^{th} International Workshop on Principles of Diagnosis, pp. 133–140. Universidad de Valladolid (2006)
11. Jonge, F., Roos, N., Witteveen, C.: Diagnosis of multi-agent plan execution. In: Fischer, K., Timm, I.J., André, E., Zhong, N. (eds.) MATES 2006. LNCS (LNAI), vol. 4196, pp. 86–97. Springer, Heidelberg (2006)
12. Roos, N., Witteveen, C.: Diagnosis of plans and agents. In: Pěchouček, M., Petta, P., Varga, L.Z. (eds.) CEEMAS 2005. LNCS (LNAI), vol. 3690, pp. 357–366. Springer, Heidelberg (2005)
13. Roos, N., Witteveen, C.: Models and methods for plan diagnosis. In: Formal Approaches to Multi-Agent Systems (FAMAS'06), ECAI 2006, Workshop Notes (2006)
14. Witteveen, C., Roos, N., van der Krogt, R., de Weerdt, M.: Diagnosis of single and multi-agent plans. In: AAMAS 2005, pp. 805–812 (2005)
15. de Jonge, F., Roos, N.: Plan-execution health repair in a multi-agent system. In: PlanSIG 2004. Proc. 23rd Annual Workshop of the UK Planning and Scheduling SIG (2004)
16. de Jonge, F., Roos, N., van den Herik, H.: Keeping plan execution healthy. In: Pěchouček, M., Petta, P., Varga, L.Z. (eds.) CEEMAS 2005. LNCS (LNAI), vol. 3690, pp. 377–387. Springer, Heidelberg (2005)
17. Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., Teneketzis, D.: Failure diagnosis using discrete event models. IEEE Transactions on Control Systems Technology 4, 105–124 (1996)