Principles of self-maintenance in an on-board architecture including active diagnosis

Elodie Chanthery*,** and **Yannick Pencolé***
* LAAS-CNRS, University of Toulouse, Toulouse, France
** University of Toulouse, INSA, Toulouse, France
[elodie.chanthery; yannick.pencole]@laas.fr

Abstract

This article presents some principles for self-maintenance in an on-board architecture that embeds on-line *active diagnosis*. The objective of active diagnosis is to find an action plan that refines the diagnosis without radically changing the mission plan. This leads to the definition of a planning problem that relies on an *active diagnoser*. According to a decision criterion, the aim is then to find the best conditional plan to reach a diagnosable region of the active diagnoser. The interactions problems between the result of the planning for diagnosis and the mission planning are then detailed and a solution that uses the execution controller of the on-board architecture is proposed.

1 Introduction

Autonomy is the ability to independently perform decisions and act in a changing environment. One of the most important characteristic of an autonomous system is its ability to take care of itself when performing its mission. This ability, also called self-maintenance, requires on-line diagnosis (fault detection and isolation) and on-line planning/replanning [Chanthery *et al.*, 2005] launched by processes integrated in the on-board architecture of the system.

On-line diagnosis is usually considered as a task that reacts to a flow of observations provided by sensors and returns estimations of the system's state. On-line diagnosis may be compared with a supervision task. The goal is to follow the temporal evolution of a dynamic system. However, this process is often too limited in practice due to a limited number of possible observations.

Off-line diagnosis is focused on the localisation of the fault. The goal is to determine additional information that will refine the diagnosis with minimal cost. This may be compared with a test sequencing problem and can be solved as a post-mortem diagnosis problem. Initially described by [Pattipati and Dontamsetty, 1992] for binary tests, then extended in the framework of the AGENDA method [Olive *et al.*, 2003] for multi-valued tests, the test sequencing problem is an off-line process. The problem consists in finding a test sequence that may isolate each fault in minimizing the tests cost.

This paper proposes to make on-line diagnosis on an autonomous system by taking into account the fact that autonomous systems can act on themselves. One way to improve the performance of the diagnostic process is to use the action capabilities of the system in order to refine the diagnosis in case of ambiguity: this is what is called the *active diagnosis* problem.

This paper is organised as follows. Section 2 presents related works on active diagnosis on dynamical systems. Section 3 presents a formal background about fault diagnosis in a DES. Section 4 proposes an on-board architecture for active diagnosis and a formal characterisation of the active diagnoser. Section 5 explains how the planning problem for active diagnosis can be derived from the active diagnoser. It proposes a planning algorithm, and puts side by side the mission planning and the planning for diagnosis criteria. Section 6 discusses the difficulties involved by the integration of active diagnosis into an on-board architecture including the mission planning process.

2 Related Works

The choice of the set of actions performed on-line for refining the diagnosis may be compared with an on-line test sequencing problem. However, for dynamical systems that are currently performing a mission, the challenge of an active diagnosis process is to propose an admissible sequence of actions (or *plan*) that refines the diagnosis without radically changing the mission plan. The conflicts between the plan for diagnosis and the mission plan have to be taken into account.

To our best knowledge, there are very few works about active diagnosis in dynamical systems. The main contribution on active diagnosis of discrete-event systems is the work of [Sampath *et al.*, 1998]. Active diagnosis is formulated as a supervisory control problem [Ramadge and Wonham, 1989] where the legal language is an "appropriate" regular sublanguage of the regular language of the system. The proposed solution is to design a system controller in such a way that it satisfies specified control objectives and results in a diagnosable controlled system. In other words, the action domain is restricted so that the system always remains diagnosable. This approach seems to be too restrictive for autonomous systems that realize a mission. Indeed, they need to keep all their action capability, even if they intermittently loose their diagnosis capability. In this article, we reuse the idea that consists in combining monitoring, diagnosis capability and controller, without restricting the controller's action capabilities.

In the field of hybrid systems, [Bayoudh *et al.*, 2008] propose to achieve active diagnosis in a hybrid system framework. Starting in a non diagnosable region, the authors use the diagnosability analysis method to determine the sequence of controllable actions to be applied to the system in order to bring it into a diagnosable one. The problem of the search of an active diagnosis plan is formulated as follows: given an uncertain state of the active diagnoser, the active diagnosis problem is to find a controllable path leading to a certain state. The authors propose to solve this problem as a conditional planning problem using an AND-OR graph, but the solving algorithm is not implemented. In particular, the type of tree exploration and the criterion for the exploration are not given. We propose to give the formal definition of active diagnoser and to discuss the planning problem more precisely.

The integration of active diagnosis into production plans is described in [Kuhn *et al.*, 2008] and experimented on a model of an industrial digital printing press. The method is based on an active diagnoser that updates the current system state and stimulates the planner to generate informative plans that may determine faulty actions and achieve production goals as well. The advantage of interleaving diagnosis and planning is that diagnosis is quickly confirmed or invalidated. The objective is to determine faulty actions without explaining the cause of the failure. The repair procedure then consists in exchanging the printer modules according to their failure probability until the system is working properly. Our study focuses on autonomous systems that face hardware/software faults without any possibility of external repairing. The mission planner requires explicit diagnosis that determines the current degraded mode in order to optimally update the plan. Our approach consists in separating diagnosis and mission planning modules, because this is today the most common way to construct an on-board architecture for autonomous systems [Alami *et al.*, 1998].

The problem of diagnosis refinement by action planning has also been raised by [McIlraith, 1995]. The author proposes a formal situation calculus framework for diagnostic problem solving which incorporates a theory of action and change.

3 Fault diagnosis in DES

This section recalls formal background about fault diagnosis in discrete-event systems.

Definition 1 (Model of discrete-event system) *A discrete-event system is modelled as a tuple $G = (X, \Sigma, T, x_0)$ where:*

- *X is a finite set of states;*
- *Σ is a finite set of events;*
- *$T \subseteq X \times \Sigma \times X$ is a finite set of transitions;*
- *x_0 is the initial state of the system.*

A discrete event system generates a regular prefix-closed language $L(G) \subseteq \Sigma^\star$ where $\Sigma^\star$ denotes the Kleene closure of Σ. Each word w of $L(G)$ represents a finite sequence of events occurring on the system from the initial state x_0. The set $\Sigma_f \subseteq \Sigma$ denotes the set of faults of the system. In order to perform diagnosis, it is then required to observe the system with the help of sensors. Sensors implement a function $Obs : \Sigma \rightarrow (\Sigma_o \cup \{\varepsilon\})$ that associates to any event of the system, either an observable event of Σ_o or the empty event ε. This function Obs is called the observation mask in [Jiang and Kumar, 2004]. Let $w \in L(G)$, $Obs(w)$ denotes the observable trace of w and is recursively defined as follows:

$$\begin{aligned} Obs(w) &= \varepsilon \text{ if } w = \varepsilon \\ &= Obs(u).Obs(v) \text{ if } u \in \Sigma, v \in \Sigma^\star \end{aligned}$$

Definition 2 (Observed system) *An* observed system *is represented as a pair (G, Obs) where G is a DES model and Obs is the observation mask.*

To perform diagnosis, it is required to implement a *monitor* that is able to process any possible observations that is produced by the observed system, that is any observable sequence $Obs(w), \forall w \in L(G)$. The observable language of G, denoted $Obs(L(G))$, is thus a subset of $\Sigma_o^\star$.

The classical fault diagnosis problem on DES basically consists in recording observations from the system and providing the set of possible faults whose occurrence is consistent with these observations: given the recorded sequence of observations σ, the occurrence of a fault F is consistent with σ if there exists at least a sequence of events w in the model G ($w \in L(G)$) that contains F and that is such that $Obs(w) = \sigma$. In such a case, the observable sequence $Obs(w)$ is an *observable trace* of F.

Definition 3 (Observable trace) *The set of traces $Traces(F)$ of the fault F is the regular language defined as the set of finite observable sequences:*

$$Traces(F) = \{\sigma \in \Sigma_o^\star, \exists w \in L(G), (F \in w) \wedge (Obs(w) = \sigma)\}.$$

Given the observations σ, a fault F is a possible solution to the diagnosis problem if $\sigma \in Traces(F)$. There may be several solutions as an observable sequence may be an observable trace of several faults. By extension, we also define the traces of the *absence of* F, denoted $Traces(\neg F)$, that gathers the possible observable traces of the system when F has not occurred:

$$Traces(\neg F) = \{\sigma \in \Sigma_o^\star, \exists w \in L(G), (F \notin w) \wedge (Obs(w) = \sigma)\}.$$

From these two sets, it follows:

1. if $\sigma \in Traces(F) \setminus Traces(\neg F)$, the occurrence of F is sure;
2. if $\sigma \in Traces(\neg F) \setminus Traces(F)$, the occurrence of F is impossible;
3. if $\sigma \in Traces(F) \cap Traces(\neg F)$, the occurrence of F is possible.

The set of traces of a fault F is defined as a regular language so it can be represented by a minimal deterministic finite-state machine [Hopcroft *et al.*, 2001] $M(F) = (S, \Sigma_o, \delta, s_0, tag)$ where:

- S is a finite set of states;
- Σ_o is the alphabet of the machine;

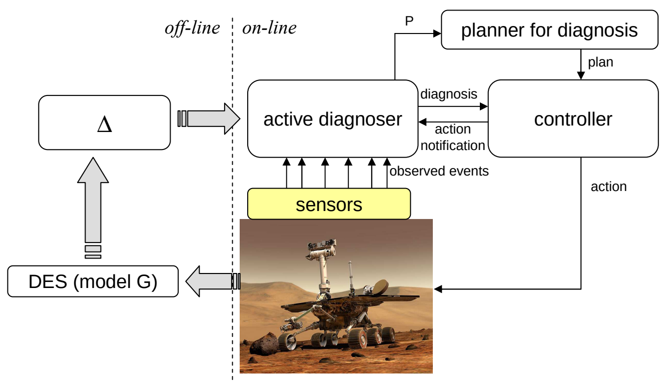

Figure 1: Part of the embedded architecture for active diagnosis.

- $\delta : S \times \Sigma_o \to S$ is the transition function;
- $s_0 \in S$ is the initial state;
- $tag : S \to \{F\text{-}possible, F\text{-}impossible\}$.

By analogy, the set of traces $Traces(\neg F)$ can also be represented by a machine $M(\neg F)$ (see [Chanthery and Pencolé, 2009] for details). Both machines $M(F)$ and $M(\neg F)$ can be used as a monitor of the system and determine at any time whether the current sequence of observations is a trace of F or not. For instance, if $M(F)$ is in state s after observing the sequence σ with $tag(s) = F\text{-}possible$ then $\sigma \in Traces(F)$, if $tag(s) = F\text{-}impossible$ then $\sigma \notin Traces(F)$. The classical diagnoser [Sampath *et al.*, 1996] can be trivially characterised by the synchronous product of the set of machines $M(F), F \in \Sigma_f$ [Pencolé *et al.*, 2006].

4 Active Diagnosis

This section extends the classical framework in order to perform *active diagnosis* on DES. The proposed active diagnoser takes into account the fact that at a given time, it may be possible to act on the system and get a better diagnosis if such an action is performed. As shown in Figure 1, the purpose of the active diagnoser modelled by the machine Δ is firstly to provide a diagnosis for any observable situation (like any other diagnoser) and secondly to provide information about how useful the trigger of an active diagnostic session could be. Furthermore, it provides the input data as a planning problem P to a planner that could plan actions for refining the diagnosis (see Section 5).

4.1 Model extension for active diagnosis

From the system model point of view, an *action* performed by the controller is represented as a controllable event [Ramadge and Wonham, 1989].

Definition 4 (Action) *An action is an event of the system that occurs only if the controller of the system performs the action.*

In the following, we suppose that the controller notifies the diagnoser about the performed action which means that any action is actually fully observed by the diagnoser (Figure 1). Let $\Sigma_a \subseteq \Sigma$ be the set of available actions on the observed system modelled by (G, Obs) then it follows that:

$$\forall a \in \Sigma_a, (a \in \Sigma) \wedge (Obs(a) = a)$$

in other words $\Sigma_a \subseteq \Sigma_o$.

Finally, in order to ensure that the controller is always able to perform an action on the system, the study of the active diagnosis problem is restricted to the subclass of discrete-event systems in which the following hypothesis holds.

Hypothesis 1 *From any state $x \in X$, it is always possible to perform an action $a \in \Sigma_a$ after the occurrence of a finite sequence of reactive events $e \in \Sigma \setminus \Sigma_a$.*

In the case that this hypothesis does not hold, it means that the system can reach a state from which the controller cannot perform any more actions. In this case, the active diagnosis problem has trivially no solution.

For the sake of readability, the definition of the active diagnoser Δ is decomposed into two steps. First, we introduce a specialised active diagnoser $\Delta(F)$ for a given fault F and then present the active diagnoser as the union of these specialised diagnosers.

4.2 Specialised active diagnoser

The challenge of active diagnosis is to refine diagnosis by pruning ambiguities about the presence or the absence of faults at a given time by acting on the system. The definition of the diagnoser $\Delta(F)$ relies on the following facts:

1. if there is an active solution for diagnosing the fault F with certainty then this solution produces an observable trace that necessarily belongs to $Traces(F) \setminus Traces(\neg F)$;
2. if there is an active solution for diagnosing the absence of the fault F with certainty then this solution produces an observable trace that necessarily belongs to $Traces(\neg F) \setminus Traces(F)$.

Let $M(F) = (S_1, \Sigma_o, \delta_1, s_{01}, tag_1)$ denote the machine representing $Traces(F)$ and $M(\neg F) = (S_2, \Sigma_o, \delta_2, s_{02}, tag_2)$ denote the one of $Traces(\neg F)$, the active diagnoser is then obtained by synchronising the machines $M(F)$ and $M(\neg F)$ on the observable events.

Definition 5 *The active diagnoser of F is the complete deterministic finite-state machine $\Delta(F) = (S, \Sigma_o, \delta, s_0, tag)$ where:*

- *$S = S_1 \times S_2$ is the set of states;*
- *Σ_o is the set of observable events;*
- *$\delta : S \times \Sigma_o \to S$ is the transition function:*

$$\forall s_1 \in S_1, s_2 \in S_2, o \in \Sigma_o,$$
$$\delta((s_1, s_2), o) = (\delta_1(s_1, o), \delta_2(s_2, o));$$

- *$s_0 = (s_{01}, s_{02})$ is the initial state;*
- *$tag : S \to \{F\text{-}safe, F\text{-}sure, F\text{-}discriminable, F\text{-}undiscriminable, nonAdmissible\}$ is incrementally defined as follows,$\forall s = (s_1, s_2) \in S$:*

1. if $tag_1(s_1) = F\textit{-possible}$ and $tag_2(s_2) = \neg F\textit{-impossible}$ then $tag(s) = F\textit{-sure}$;

2. if $tag_1(s_1) = F\textit{-impossible}$ and $tag_2(s_2) = \neg F\textit{-impossible}$ then $tag(s) = nonAdmissible$;

3. if $tag_1(s_1) = F\textit{-impossible}$ and $tag_2(s_2) = \neg F\textit{-possible}$ then $tag(s) = F\textit{-safe}$;

4. if $tag_1(s_1) = F\textit{-possible}$ and $tag_2(s_2) = \neg F\textit{-possible}$, two cases hold:

(a) if there exists a sequence of transitions from s to a state s' such that $tag(s') = F\textit{-sure}$ or $tag(s') = F\textit{-safe}$, then $tag(s) = F\textit{-discriminable}$;

(b) else $tag(s) = F\textit{-undiscriminable}$.

From this definition can be directly derived the following results, for any state s of $\Delta(F)$:

Theorem 1 *1. $tag(s) = F\textit{-sure}$ $\equiv$ for any observable sequence σ such that $\delta^\star(s_0, \sigma) = s$, for any sequence of events w of the system G such that $Obs(w) = \sigma$, $F \in w$;*

2. $tag(s) = F\textit{-safe}$ $\equiv$ for any observable sequence σ such that $\delta^\star(s_0, \sigma) = s$, for any sequence of events w of the system G such that $Obs(w) = \sigma$, $F \notin w$;

3. $tag(s) = F\textit{-discriminable}$ $\equiv$ for any observable sequence σ such that $\delta^\star(s_0, \sigma) = s$, for any sequence of events w of the system G such that $Obs(w) = \sigma$, there exists at least one finite sequence of observable events $\sigma_{s\to s'}$ such that $\delta^\star(s, \sigma_{s\to s'}) = s'$ and $tag(s') \in \{F\textit{-sure}, F\textit{-safe}\}$;

4. $tag(s) = F\textit{-undiscriminable}$ $\equiv$ for any observable sequence σ such that $\delta^\star(s_0, \sigma) = s$, for any sequence of events w of the system G such that $Obs(w) = \sigma$, there is no finite sequence of observable events $\sigma_{s\to s'}$ such that $\delta^\star(s, \sigma_{s\to s'}) = s'$ and $tag(s') \in \{F\textit{-sure}, F\textit{-safe}\}$.

4.3 Active diagnoser

Let $F_1, \dots, F_n$ be the set of anticipated faults, let $\Delta(F_i) = (S_i, \Sigma_o, \delta_i, s_{0i}, tag_i)$ be the active diagnoser of fault F_i. The active diagnoser Δ of G is then defined as the union of specialised diagnosers as follows. For any sequence of observations σ, the specialised diagnoser $\Delta(F_i)$ reaches the state s_i with a tag $tag(s_i)$. The result of the active diagnoser Δ after observing σ is then: $s = s_1, \dots, s_n$ and $tag(s) = \{tag(s_1), \dots, tag(s_n)\}$. The active diagnoser then provides the following pieces of information:

1. *Current diagnosis:* for any fault F it provides the current status about the presence of F. For instance, if $\forall i \in \{1, \dots, n\}$, the status of F_i is safe, it means that no fault has occurred.
2. *Status of active diagnostic session:* it provides a general view about how useful the trigger of an active diagnostic session is. The optimal goal of such a session is to disambiguate between the presence and the absence of any fault F such that $tag(s)$ contains $F\textit{-discriminable}$ if possible (see next section for details). If there is a plan of actions that reaches these goals, its execution is necessarily represented as a sequence of transitions from state s to a state s' such that $tag(s')$ contains $F\textit{-sure}$ or $F\textit{-safe}$. Any execution from state s to the state s' that contains $nonAdmissible$ is not admissible by G.

5 Planning for diagnosis

5.1 Background about mission planning

The mission planning problem is described in [Chanthery *et al.*, 2005] or [Meuleau *et al.*, 2009]. An autonomous system must act autonomously for achieving a set of objectives in an uncertain environment. Each objective has an associated reward that depends on the operator interest. This problem is known as an over-subscription planning problem [Smith, 2004]. In this problem, it is infeasible to achieve all objectives. The mission planning system must choose a feasible subset of these that can be achieved within the time and resources limits and that maximizes expected return.

5.2 Planning for diagnosis problem formulation

The last section allows us to obtain the active diagnoser $\Delta = (S, \Sigma_o, \delta, s_0, tag)$. Suppose that Δ reached a state s^I where it exists at least one i in $\{1, \dots, n\}$ such that $tag(s^I) = (\dots, F_i\textit{-discriminable}, \dots)$. Then by Theorem 1: there exists at least one finite sequence of observable events σ such that $\delta^\star(s^I, \sigma) = s'$ and $tag_i(s'_i) \in \{F_i\textit{-sure}, F_i\textit{-safe}\}$. Let π_σ be the sequence of actions resulting from the projection of σ to Σ_a, then π_σ is a possible sequence of actions for deciding whether F_i has certainly occurred or not. The planning problem is to choose the best admissible sequence of actions to perform in this case in order to refine the diagnosis.

For the sake of readability, the formulation of the planning problem is decomposed into two steps in the same way that the active diagnosis. First, we formulate the problem for one fault F and then present the entire planning problem.

Planning problem for a single fault

Suppose that Δ reached a state s^I for which it exists **only one** F that is tagged $F\textit{-discriminable}$; the other faults F_j are tagged $F_j\textit{-sure}$ or $F_j\textit{-safe}$. Then the set of admissible sequences of actions for diagnosing F with certainty is contained in AP_F where:

$$AP_F = \{\pi_\sigma | \exists\sigma, \delta(s^I, \sigma) = s', \; s' \text{ is tagged } F\textit{-sure} \text{ or } F\textit{-safe}\}$$

By Theorem 1:

Proposition 1 *If it exists a sequence of actions that can be performed by the system and that refines the diagnosis, it is in AP_F.*

The objective here is to reformulate the problem into a classical planning problem. Thus, we decide to define the problem as a tuple $P = (s_i, S, A, T, Goal, C)$ where the following pieces of information are extracted from the active diagnoser:

- the initial state s_i is s^I;
- $A = \Sigma_a$;
- S the finite set of states of Δ;
- $T : S \times A \times S \to \{0, 1\}$ is the transition function:

- $T(s, a, s') = 1$ if s' can be reached when a is performed in s, i-e $\pi_{s \to s'} = a$;
- $T(s, a, s') = 0$ otherwise [1];

- The set of goal states $Goal = \{s' \in S$ such that s' is tagged $F\text{-}sure$ or $F\text{-}safe\}$;
- C a criterion to minimize.

The planning problem may be formulated as follows: finding a conditional plan of the type:

$action$;
if $observation_1$ **then** $action_1$
else if $observation_2$ **then** $action_2 \ldots$,

that will perform one of the sequences:
$< action; action_1, \ldots >$; $< action; action_2, \ldots >$, knowing that at least one of them leads the diagnoser from s_i to a state $s_p \in Goal$, minimizing a criterion C and respecting resources constraints. The conditional plan can be seen as a tree. Note that $action_1$, $action_2$, … could also be sequences of actions, followed by an observation.

Definition 6 *A plan is admissible from s_i iff resources constraints are respected and if it performs at least one sequence of actions $\{a_{i+1}, a_{i+2}, \ldots, a_p\}$ such that it exists $s_p \in Goal$ for which*

$$\forall k \in \{i+1, \ldots, p\}, T(s_{k-1}, a_k, s_k) = 1$$

General planning problem
Suppose that Δ reached a state s^I for which **there exists a subset** $D \subseteq \{1, \ldots, n\}$ such that for all i in D, s^I is tagged $F_i\text{-}discriminable$ and for all j in $\{1, \ldots, n\} \setminus D$, s^I is tagged $F_j\text{-}sure$ or $F_j\text{-}safe$. Then the set of sequences of actions for diagnosing all the F_i for which i is in D with certainty is contained in AP where:

$$AP = \{\pi_\sigma | \exists \sigma, \delta(s^I, \sigma) = s', \quad \forall i \in D, s' \text{ is tagged } F_i\text{-}sure \text{ or } F_i\text{-}safe\}$$

The set AP may be the empty set. Actually,

Proposition 2 *The set of sequence of actions that are admissible for diagnosing all the F_i for which i is in D is contained in the intersection of the AP_{F_i} sets.*

$$AP = \bigcap_{i \in D} AP_{F_i}.$$

The reformulation of the problem into a classical planning problem is a tuple $P = (s_i, S, A, T, Goal, C)$ where s_i, S, A, T and C are the same as the one of the planning problem for a single fault and $Goal$ is defined by

$$Goal = \{s' \in S \text{ such that } \forall i \in \{1, \ldots, n\}, \text{ if s is tagged } F_i\text{-}discriminable \text{ then s' is tagged } F_i\text{-}sure \text{ or } F_i\text{-}safe\}.$$

The planning problem remains the same.

[1] $T(s, a, s') = 0$ in particular if a is an action that cannot be performed in s for security or physical reason.

5.3 Planning algorithm

The problem of the search of the optimal tree can be formulated as a search in a AND/OR tree [Nilsson, 1998]. We use an iterative depth-first search that explores the graph as it was an AND-OR tree where OR nodes correspond to system states and AND nodes correspond to actions. We choose a depth-first search approach because once the first plan has been computed, the algorithm is any-time. So, even if all branches of the AND-OR tree are not explored for time reason, there exists an admissible plan. Finally, we come back to a widely studied problem, even in the diagnosis area that is the search in an AND-OR tree [Pattipati and Dontamsetty, 1992] excepted that the plan has to respect resources constraints. The problem is to determine the optimal solution tree in the AND-OR tree. This problem is NP-complete. The planning algorithm should use a heuristic search in order to choose the best action to perform in case of multiple choices, as in the AO* algorithm [Olive *et al.*, 2003] and prune the search tree. The heuristic function should be an easily computable estimation of the criterion from the current state to a goal state. The solution is a tree that we call the tree plan.

The on-line planning algorithm consists in these steps:

1. According to the active diagnoser, compute an AND-OR graph that is the entry of the planning problem;
2. Find the *best* tree with the heuristic iterative depth-first search algorithm. Send this tree to the controller that applies it;
3. If tree plan succeeds, i-e that the current state of Δ is in $Goal$, then *END* (Situation 1). The tree plan could fail for several reasons:
   - after an action, the active diagnoser is a state s where a fault is tagged $F\text{-}undiscriminable$. Then the tree plan fails and there is no possibility to refine the diagnosis, so *END* (Situation 2).
   - The tree plan is reduced to a limited number of actions and after the last action, the active diagnoser is in a state s where some faults are tagged $F\text{-}discriminable$. Then the tree plan fails but the process could iterate in (1) (Situation 3).

5.4 Criterion

In this paper, we propose to formalize a single aggregate objective function. The following criteria should be taken into account: Similarity between the results of the active diagnosis plan and the mission plan (a plan that performs a sequence of actions that finally leads the system to a state that is compatible with the mission achievement should be preferred. Similarly, a plan that performs a sequence of actions that is an obstacle to the mission achievement should be discarded); fault criticity; action cost.

Actions of A are separated into 3 types: *repair actions*: repairing a fault may involve one or more actions; *replan actions*: they represent computational actions; *other actions*: like move, observe, transmit information, etc. These actions are used for the achievement of the mission. This set of actions is denoted A_m.

Costs and rewards

The considered costs are the following:

- Each action $a \in A_m$ has a cost $C_a > 0$;
- Faults may be repairable or not [Cordier *et al.*, 2007]. For each fault F, this is modeled by a variable ϕ_F such that $\phi_F = 1$ if F is repairable; $\phi_F = 0$ otherwise. Each fault may be repaired with a cost $C_{repair_F} > 0$. For a fault F that is non repairable, $C_{repair_F} = \infty$.
- The cost of replanning from a goal state s' depends of the similarity between this active diagnosis goal and the states included in the mission plan defined in subsection 5.1. Because of the difficulty to determine the state s', it is supposed that the cost of the replanning action is constant, equal to $C_{replan} > 0$.

Let O be the set of possible objectives of the mission. Each objective $o \in O$ has an associated reward $R_o > 0$.

Criterion and constraints

The idea is to use a criterion that is similar to the one for mission planning. Let π_σ^m be the sequence of actions performed by the system for achieving its mission from a state s^I to a state s and O^m the subset of O that is really achieved if this sequence of actions is performed. We recall that the mission planning criterion from a state s^I to a state s is the following:

$$\begin{cases} \min\left(\sum_{a \in \pi_\sigma^m} C_a - \sum_{o \in O^m} R_o\right) \\ \\ \text{under the time and resources constraints} \end{cases}$$

In the same way, it is possible to define the planning for diagnosis criterion from a state s^I to a state s'. π_σ is the sequence of actions performed by the system for refining diagnosis from s^I to a state s'.

$$\begin{cases} \min(Costs - \mathit{Future}_R) \\ \\ \text{under the time and resources constraints} \end{cases}$$

where $Costs$ represents the costs of actions and Future_R represents an evaluation of the future rewards. If $s' \notin Goal$ then:

$$Costs = \sum_{a \in \pi_\sigma} C_a$$

If $s' \in Goal$ then let D_s be the subset of faults F such that s is tagged $F\text{-}sure$. Costs may be separated into 3 types:

- Costs due to actions in A_m that are in the sequence π_σ. This set of actions is denoted A_m^{pi};
- Costs due to reparation if it is possible;
- Costs due to replanning.

$$Costs = \sum_{a \in A_m^{pi}} C_a + \sum_{F \in D_s} \phi_F C_{repair_F} + C_{replan}$$

As it is not easy to evaluate the future rewards, we use a heuristic that gives the possible maximum reward. Let O^I be the set of objectives already achieved before s^I, let O_F be the set of objectives that may be achieved knowing that F occurred. Then

$$\mathit{Future}_R = \sum_{F \in D_s} (\phi_F \sum_{o \in O \setminus O^I} R_o + (1 - \phi_F) \sum_{o \in O_F \setminus O^I} R_o)$$

The use of future rewards allows to take into account the criticity of the fault.

6 Integrating Active diagnosis into an on-board architecture

To sum up the previous sections, three modules should cooperate at the decision level for autonomous systems. Two modules have been defined for refining the diagnosis:

- The *active diagnoser* monitors the system and detects situations where it is useful to trigger an active diagnosis session;
- The *planner for diagnosis* takes as input a planning problem P given by the active diagnoser and compute a tree plan for refining the diagnosis.

The *mission planner* takes as input the current state of the system and computes a mission plan that achieves a feasible subset of objectives within the time and resources limits and that maximizes expected return.

These three computational tasks have to be integrated in an on-board architecture. For dealing the conflicts that could appear, an execution control level has to be introduced.

6.1 The execution control level

The concept of execution control level was introduced by [Alami *et al.*, 1998] in the framework of autonomous mobile robots architectures. The role of the execution control level is to handle the conflicts between different functional modules and to maintain a logical description of their operating states. Several tools [Fleury *et al.*, 1997], [Barbier *et al.*, 2006] exist in the literature for designing the execution control level. These tools allow to design complex on-board architectures such as autonomous mobile robots or satellites, that require the integration of heterogeneous operational functions with various real-time constraints and algorithm complexities (mission planning, diagnosis, active diagnosis, guidance algorithm, trajectory computations, etc.); an homogeneous integration of these functions in the control architecture which requires coherent and predictable behaviors (starting, ending, error handling), and standard interfaces (control, parameterization, data flow).

In this article, we choose to design the execution control level with the ProCoSA tool because it allows to specify the links between modules and to manage their interactions.

6.2 The ProCoSA tool

In an architecture designed with ProCoSA, modules encapsulate functions that are dynamically started, interrupted or (re)parameterized upon asynchronous *requests* sent to the modules. Modules are so standardized servers that manage the execution of a set of functions or algorithms on a host machine. They can deal with several functions, synchronous or asynchronous, and execute several treatments in parallel. When a treatment ends, the module answers in an asynchronous way to the module that sent the request. This answer is an *event*. The data-processing components of the deliberative part of the vehicle such as a planning algorithm or active diagnosis are thus implemented as on-board modules. Off-line, ProCoSA allows to model the execution level as a Petri net by means of a convivial graphic user interface. It also generates an on-line controlling net in presence of constraints on the marking of a Petri net. In an architecture designed by ProCoSA, an automaton, called the Petri Player, manages on-line the update of the Petri nets marking.

The real-time level of this language will depend on the application; in this work, the architecture behaves like a soft real-time system. Indeed, there is no guarantee on time response mainly because ProCoSA internally uses a socket communication protocol and events treatments depend on the Petri Player state.

An original functionality of the ProCoSA Petri nets is the possibility to assign events and requests to transitions. Thanks to this functionality, the modules are integrated in the architecture in a coherent and homogeneous way: a transition is fired if the marking validates it and if an assigned event occurs. The crossing of this transition produces the assigned requests. These requests are either events toward other transitions or requests toward a module. Finally, modules can produce events.

6.3 Conflicts between planning for diagnosis and mission planning management

The solution proposed for dealing the conflicts between planning for diagnosis and mission planning is to stop the execution of the mission plan if the active diagnoser detects an ambiguous state that needs to be disambiguate. After that, the execution controller has to open a session of active diagnosis, apply the plan for diagnosis, repair if required, then close the active diagnosis session and launch a mission replanning. This solution is illustrated on Figure 2.

On the right of the figure, there are the three functional modules implied in the conflicts management: active diagnoser, planner for diagnosis and mission planner. The ProCoSA tool is used to specify the links between them and to manages their interactions thanks to the Petri net on the left of the figure. At the beginning, the system performs its mission. Then the active diagnoser detects a situation where an active diagnosis session is useful. It sends a message "ambiguous state": the execution controller stops the execution of the mission and sends a request of planning for diagnosis, associated with the problem P provided by the active diagnoser. The planner for diagnosis finds the best tree plan and sends it to the controller. The place "ACT FOR DIAGNOSIS" embodies the application of the tree plan. There are three possible situations described in Section 5.3:

- (Situation 1) The tree plan succeeds: the system is in a state where all faults are tagged $F\text{-}sure$ or $F\text{-}safe$. Then the controller sends a request to the mission planner to replan with the new mission context. The new plan may include repair actions.
- (Situation 2) The tree plan fails and there is no possibility to refine the diagnosis: the active diagnosis session has to be closed. The execution controller sends a request to the mission planner to replan with a degraded and uncertain mission context.
- (Situation 3) The tree plan fails but the active diagnoser detects a state where some faults are still tagged $F\text{-}discriminable$, the execution controller sends a new request of planning for diagnosis, associated with a new problem P provided by the active diagnoser.

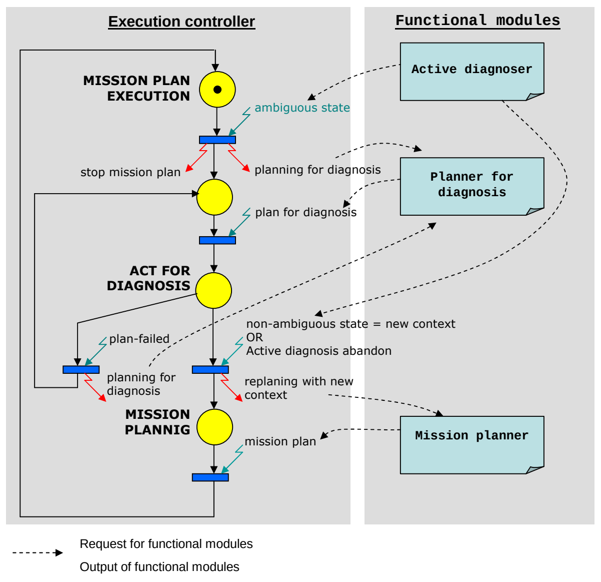

Figure 2: Conflicts management between mission planner and active diagnosis

If the mission planner is requested, it replans the mission and sends the best mission plan to the execution controller that controls its execution.

Several problems remains open :

- What is the criterion used by the controller for suspending the mission plan execution? Actually, the solution that consists in stopping the mission plan execution each time the active diagnoser detects an ambiguous state could imply that the mission is never performed. This criterion should help the execution controller to find a trade-off between refining the diagnosis and performing the mission.
- What is the criterion used by the controller for abandoning the active diagnosis session if (Situation 3) arrives more than once? If the plan for diagnosis fails again and again, a criterion should help the execution controller to chose between replanning for diagnosis or abandon the active diagnosis session. If the active diagnosis session fails, it is still possible to replan the mission under uncertainties.
- Are the repair actions a part of active diagnosis or a part of the actions chosen by the replanning? Because of our definition of *Goal*, the repair actions are rejected

in the mission replanning. The advantage is that even if a repairable fault is detected, the mission planner may choose a plan that not repair this fault because it is not worthwhile for the rest of the mission.

7 Conclusion and future works

This paper defines the concept of active diagnosis for discrete-event system without reducing the number of possible actions performed by the controller. This point is crucial for autonomous vehicles that realize a mission. The formal definition of an active diagnoser and the implementation of its generation in the framework of finite-state automata provide solid foundations for further works. This paper shows how the active diagnoser can be transformed into a planning problem and gives some key ideas for its resolution. It presents how the active diagnoser is integrated in an on-board architecture and how it is linked with the mission planner.

In the future, we will focus on how to embed an algorithm that provides the capability of an active diagnoser, for example using the techniques proposed in [Pencolé *et al.*, 2006]. Moreover, another idea is to interleave active diagnosis and mission planning in creating "artificial" objectives with reward. The mission planner will have to choose between performing actions for achieving real objectives or for refining the diagnosis regarding the underlying rewards.

References

[Alami *et al.*, 1998] R. Alami, R. Chatila, S. Fleury, M. Ghallab, and F. Ingrand. An architecture for autonomy. *Intern. Journal of Robotics Research*, 17:315–337, 1998.

[Barbier *et al.*, 2006] M. Barbier, J.-F. Gabard, D. Vizcaino, and O. Bonnet-Torrès. ProCoSA: a software package for autonomous system supervision. In *CAR'06 - First Workshop on Control Architectures of Robots*, 2006.

[Bayoudh *et al.*, 2008] M. Bayoudh, L. Travé-Massuyès, and X. Olive. Towards active diagnosis of hybrid systems. In *Proc of DX'08*, 2008.

[Chanthery and Pencolé, 2009] E. Chanthery and Y. Pencolé. Monitoring and active diagnosis for discrete-event systems. In *7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, 2009. To appear.

[Chanthery *et al.*, 2005] E. Chanthery, M. Barbier, and J-L. Farges. *Planning, Scheduling and Constraint Satisfaction: from Theory to Practice*, chapter Integration of Mission Planning and Flight Scheduling for Unmanned Aerial Vehicles. IOS Press, 2005.

[Cordier *et al.*, 2007] M-O Cordier, Y. Pencolé, L. Travé-Massuyès, and T. Vidal. Self-healability = diagnosability + repairability. In *proceedings of the 18th International Workshop on Principles of Diagnosis (DX'07)*, 2007.

[Fleury *et al.*, 1997] Sara Fleury, Matthieu Herrb, and Raja Chatila. Genom: A tool for the specification and the implementation of operating modules in a distributed robot architecture. In *Intern.Conf. on Intelligent Robots and Systems*, pages 842–848. IEEE, 1997.

[Hopcroft *et al.*, 2001] J.E. Hopcroft, R. Motwani, and J.D. Ullman. *Introduction to automata theory, languages, and computation (2nd ed)*. Addison-Wesley, 2001.

[Jiang and Kumar, 2004] S. Jiang and R. Kumar. Failure diagnosis of discrete-event systems with linear-time temporal logic specifications. *IEEE Transactions on Automatic Control*, 49(6):934– 945, 2004.

[Kuhn *et al.*, 2008] L. Kuhn, B. Price, J. de Kleer, M. B. Do, and R. Zhou. Pervasive diagnosis: The integration of diagnostic goals into production plans. In *AAAI 2008*, pages 1306–1312, 2008.

[McIlraith, 1995] S. McIlraith. Incorporating action into diagnostic problem solving (an abridged report). In *Working Notes of the 1995 AAAI Spring Symposium on Extending Theories of Action: Formal Theory and Practical Applications*, pages 139–144, 1995.

[Meuleau *et al.*, 2009] N. Meuleau, E. Benazera, R. I. Brafman, E. A. Hansen, and Mausam. A heuristic search approach to planning with continuous resources in stochastic domains. *Journal of Artificial Intelligence Research*, 34:p 27–59, 2009.

[Nilsson, 1998] N.J Nilsson. *Artificial intelligence, a new synthesis*. 1998.

[Olive *et al.*, 2003] X. Olive, L. Trave-Massuyes, and H. Poulard. Ao* variant methods for automatic generation of near-optimal diagnosis trees. In *Proc. of the International Workshop on Principles of Diagnosis (DX'03)*, 2003.

[Pattipati and Dontamsetty, 1992] K. R Pattipati and M. Dontamsetty. On a generalized test sequencing problem. *IEEE transactions on systems, man, and cybernetics*, 22(2):392–396, 1992.

[Pencolé *et al.*, 2006] Y. Pencolé, D. Kamenetsky, and A. Schumann. Towards low-cost diagnosis of component-based systems. In *6th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Process (SAFEPROCESS)*, 2006.

[Ramadge and Wonham, 1989] P.J.G. Ramadge and W.M. Wonham. The control of discrete event processes. In *IEEE Proc.: Special issue on Discrete Event Systems*, 1989.

[Sampath *et al.*, 1996] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Failure diagnosis using discrete event models. *IEEE Transactions on Control Systems Technology*, 4(2):105–124, March 1996.

[Sampath *et al.*, 1998] M. Sampath, S. Lafortune, and D. Teneketzis. Active diagnosis of discrete-event systems. *IEEE Transactions on Automatic Control*, 43, 1998.

[Smith, 2004] David E. Smith. Choosing objectives in oversubscription planning. In *14th International Conference on Automated Planning and Scheduling*, 2004.