Formal Design of Fault Detection and Identification Components using Temporal Epistemic Logic

Marco Bozzano, Alessandro Cimatti, Marco Gario, and Stefano Tonetta

Fondazione Bruno Kessler, Trento, Italy
{bozzano, cimatti, gario, tonettas}@fbk.eu

Abstract. Automated detection of faults and timely recovery are fundamental features for autonomous critical systems. Fault Detection and Identification (FDI) components are designed to detect faults on-board, by reading data from sensors and triggering predefined alarms.
The design of effective FDI components is an extremely hard problem, also due to the lack of a complete theoretical foundation, and of precise specification and validation techniques.
In this paper, we present the first formal framework for the design of FDI for discrete event systems. We propose a logical language for the specification of FDI requirements that accounts for a wide class of practical requirements, including novel aspects such as maximality and non-diagnosability. The language is equipped with a clear semantics based on temporal epistemic logic. We discuss how to validate the requirements and how to verify that a given FDI component satisfies them. Finally, we develop an algorithm for the synthesis of correct-by-construction FDI components, and report on the applicability of the framework on an industrial case-study coming from aerospace.

1 Introduction

The correct operation of complex critical systems (e.g., trains, satellites, cars) increasingly relies on the ability to detect when and which faults occur during operation. This function, called Fault Detection and Identification (FDI), provides information that is vital to drive the containment of faults and their recovery. This is especially true for fail-operational systems, where the occurrence of faults should not compromise the ability to carry on critical functions, as opposed to fail-safe systems, where faults are typically handled by going to a safe state. FDI is typically carried out by dedicated modules, called FDI components, running in parallel with the system. An FDI component processes sequences of observations, made available by predefined sensors, and is required to trigger a set of predefined alarms in a timely and accurate manner. The alarms are then used by recovery modules to autonomously guarantee the survival of the system.

Faults are often not directly observable, and their occurrence can only be inferred by observing the effects that they have on the observable parts of the system. Moreover, faults may have complex dynamics, and may interact with each other in complex ways. For these reasons, the design of FDI components

is a very challenging task, as witnessed by a recent Invitation To Tender issued by the European Space Agency [1]. The key reasons are the lack of a clear and complete theoretical foundation, supported by clear and effective specification and validation techniques. As a consequence, the design often results in very conservative assumptions, so that the overall system features suboptimal behaviors, and is not trusted during critical phases.

In this paper, we propose a formal foundation to support the design of FDI by introducing a *pattern based* language for the specification of FDI requirements. Intuitively, an FDI component is specified by stating which are the observable signals (the inputs of the FDI component), the desired alarms (in terms of the unobservable state), and defining the relation between the two. The language supports various forms of delay (exact, finite, bounded) between the occurrence of faults, and the raising of the corresponding alarm. The patterns are provided with an underlying formal semantics expressed in epistemic temporal logic [2], where the *knowledge* operator is used to express the certainty of a condition, based on the available observations. The formalization encodes properties such as *alarm correctness* (whenever an alarm is raised by the FDI component, then the associated condition did occur), and *alarm completeness* (if an alarm is not raised, then either the associated condition did not occur, or it would have been impossible to detect it, given the available observations). Moreover, we precisely characterize two aspects that are important for the specification of FDI requirements. The first one is the *diagnosability* of the plant, i.e., whether the sensors convey enough information to detect the required conditions. We explain how to deal with non-diagnosable plants by introducing the more fine grained concept of *trace diagnosability*, where diagnosability is localized to individual traces. The second is the *maximality* of the diagnoser, that is, the ability of the diagnoser to raise an alarm as soon as and whenever possible.

Within our framework, we cover the problems of (i) validation of a given specification, (ii) verification of a given diagnoser with respect to a given specification, and (iii) automated synthesis of a diagnoser from a given specification, using a synchronous and perfect-recall semantics for the epistemic operator. Moreover, we provide formal proofs for the correctness of the synthesis algorithm, we show that the specification language correctly captures the formal semantics and we clearly define the relation between diagnosability, maximality and correctness. The framework has been validated on an industrial setting, in a project funded by the European Space Agency [1]. The framework provides the conceptual foundation underlying a design toolset, which has been applied to the specification, verification and synthesis of an FDI component for a satellite.

It is important to remark the deep difference between the design of FDI components and diagnosis. A diagnosis system can benefit from powerful computing platform, it can provide partial diagnoses, and can possibly require further (post-mortem) inspections. This is typical of approaches that rely on logical reasoning engines (e.g., SAT solvers [3]). An FDI component, on the contrary, runs on-board (as part of the on-line control strategy) and is subject to restrictions such as timing and computation power. FDI design thus requires a deeper theory,

which accounts for the issues of delay in raising the alarms, trace diagnosability, and maximality. Furthermore, a consistency-based approach [3] is not applicable to the design of FDI: in order to formally verify the effectiveness of an FDI component as part of an overall fault-management strategy, a formal model of the FDI component (e.g., as an automaton) is required.

This paper is structured as follows. Section 2 provides some introductory background. Section 3 formalizes the notion of FDI. Section 4 presents the specification language. In Section 5 we discuss how to validate the requirements, and how to verify an FDI component with respect to the requirements. In Section 6 we present an algorithm for the synthesis of correct-by-construction FDI components. The results of evaluating our approach in an industrial setting are presented in Section 7. Section 8 compares our work with previous related works. Section 9 concludes the paper with a hint on future work.

2 Background

Plants and FDIs are represented as *transition systems*. A transition system is a tuple $S = \langle V, V_o, W, W_o, I, T\rangle$, where V is the set of state variables, $V_o \subseteq V$ is the set of observable state variables; W is the set of input variables, $W_o \subseteq W$ is the set of observable input variables; I is a formula over V defining the initial states, T is a formula over V, W, V' (with V' being the next version of the state variables) defining the transition relation.

A *state* s is an assignment to the state variables V. We denote with s' the corresponding assignment to V'. An *input* i is an assignment to the input variables W. The *observable part* $obs(s)$ of a state s is the projection of s on the subset V_o of observable state variables. The observable part $obs(i)$ of an input i is the projection of i on the subset W_o of observable input variables. Given an assignment a to a set of variables X and $X_1 \subseteq X$, we denote the projection of a over X_1 with $a_{|X_1}$. Thus, $obs(s) = s_{|Vo}$ and $obs(i) = i_{|Wo}$.

A *trace* of S is a sequence $\pi = s_0, i_1, s_1, i_2, s_2, \ldots$ of states and inputs such that s_0 satisfies I and, for each $k \geq 0$, $\langle s_k, i_{k+1}, s_{k+1}\rangle$ satisfies T. W.l.o.g. we consider infinite traces only. The observable part of π is $obs(\pi) = obs(s_0), obs(i_1), obs(s_1), obs(i_2), obs(s_2), \ldots$. Given a sequence $\pi = s_0, i_1, s_1, i_2, s_2, \ldots$ and an integer $k \geq 0$, we denote with σ^k the finite prefix $s_0, i_1, \ldots, s_k$ of π containing the first $k+1$ states. We denote with $\pi[k]$ the $k+1$-th state s_k. We say that s is *reachable* in S iff there exists a trace π of S such that $s = \pi[k]$ for some $k \geq 0$. We say that S is deterministic if i) there are no two initial states s_0 and s'_0 s.t. $obs(s_0) = obs(s'_0)$ ii) there are no two transitions $\langle s, i_1, s'_1\rangle$ and $\langle s, i_2, s'_2\rangle$ from a reachable state s s.t. $obs(i_1) = obs(i_2)$ and $obs(s'_1) = obs(s'_2)$.

Let $S^1 = \langle V^1, V^1_o, W^1, W^1_o, I^1, T^1\rangle$ and $S^2 = \langle V^2, V^2_o, W^2, W^2_o, I^2, T^2\rangle$ be two transition systems with $\emptyset = (V^1 \setminus V^1_o) \cap V^2 = V^1 \cap (V^2 \setminus V^2_o) = (W^1 \setminus W^1_o) \cap W^2 = W^1 \cap (W^2 \setminus W^2_o)$. We define the *synchronous product* $S^1 \times S^2$ as the transition system $\langle V^1 \cup V^2, V^1_o \cup V^2_o, W^1 \cup W^2, W^1_o \cup W^2_o, I^1 \wedge I^2, T^1 \wedge T^2\rangle$. Every state s of $S^1 \times S^2$ can be considered as the product $s^1 \times s^2$ such that $s^1 = s_{|V^1}$ is a state of S^1 and $s^2 = s_{|V^2}$ is a state of S^2.

We say that S^1 is *compatible* with S^2 iff i) for every initial state s^2 of S^2, there exists an initial state s^1 of S^1 such that $s^1_{|V^1_o \cap V^2_o} = s^2_{|V^1_o \cap V^2_o}$ and ii) for every reachable state $s^1 \times s^2$ of $S^1 \times S^2$, for every transition $\langle s^2, i^2, s'^2 \rangle$ of S^2, there exists a transition $\langle s^1, i^1, s'^1 \rangle$ such that $i^1_{|W^1_o \cap W^2_o} = i^2_{|W^1_o \cap W^2_o}$ and $s'^1_{|V^1_o \cap V^2_o} = s'^2_{|V^1_o \cap V^2_o}$.

3 Formal Framework

3.1 Diagnoser

A diagnoser is a machine D that synchronizes with observable traces of the plant P. D has a set $\mathcal{A}$ of Boolean alarm variables that are activated in response to the monitoring of P. We use *diagnoser* and *FDI component* interchangeably, and call *system* the composition of the plant and the FDI component.

Formally, given a set $\mathcal{A}$ of alarms and a plant transition system $P = \langle V^P, V^P_o, W^P, W^P_o, I^P, T^P \rangle$, a diagnoser is a deterministic transition system $D(\mathcal{A}, P) = \langle V^D, V^D_o, W^D, W^D_o, I^D, T^D \rangle$ such that: $V^P_o \subseteq V^D_o$, $W^P_o \subseteq W^D_o$, $\mathcal{A} \subseteq V^D_o$ and $D(A, P)$ is compatible with P; when clear from the context, we use D to indicate $D(\mathcal{A}, P)$. Given a trace π_P of P we denote with $D(\pi_P)$ the trace of D matching π_P. Only observable variables can be shared among the two systems and used to perform synchronization. This gives raise to the problem of partial observability: the diagnoser cannot perfectly track the evolution of the original system. This makes the diagnoser synthesis problem hard.

3.2 Detection, Identification, and Diagnosis Conditions

The first element for the specification of the FDI requirements is given by the conditions that must be monitored. Here, we distinguish between detection and identification, which are the two extreme cases of the diagnosis problem; the first deals with knowing whether a fault occurred in the system, while the second tries to identify the characteristics of the fault. Between these two cases there can be intermediate ones: we might want to restrict the detection to a particular subsystem, or identification among two similar faults might not be of interest.

For example, a data acquisition system composed of a sensor and a filter might have several possible faults: the sensor might fail in a single way (*sdie*) while the filter might fail in two ways ($fdie_{high}$ or $fdie_{low}$). The *detection* task is the problem of understanding when (at least) one of the two components has failed. The *identification* task tries to understand exactly which fault occurred. Similarly, e.g., if we can replace the filter whenever it fails, it might suffice to know that one of $fdie_{high}$ or $fdie_{low}$ occurred (this is sometimes called *isolation*).

FDI components are generally used to recognize faults. However, there is no reason to restrict our interest to faults. Recovery procedures might differ depending on the current state of the plant, therefore, it might be important to consider other unobservable information of the system.

We call the condition of the plant to be monitored *diagnosis condition*, denoted with β. We assume that for any point in time along a trace execution of

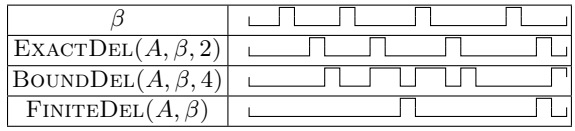


Fig. 1: Examples of alarm responses to the diagnosis condition β.

the plant (and therefore also of the system), β is either true or false based on what happened before that time point. Therefore, β can be an atomic condition (including faults), a sequence of atomic conditions, or Boolean combination thereof. If β is a fault, the fault must be identified; if β is a disjunction of faults, instead, it suffices to perform the detection, without identifying the exact fault.

3.3 Alarm Conditions

The second element of the specification of FDI requirements is the relation between a diagnosis condition and the raising of an alarm. This also leads to the definition of when the FDI is correct and complete with regard to a set of alarms.

An *alarm condition* is composed of two parts: the diagnosis condition and the delay. The delay relates the time between the occurrence of the diagnosis condition and the corresponding alarm; it might be the case that the occurrence of a fault can go undetected for a certain amount of time. It is important to specify clearly how long this interval can be at most. Interaction with industrial experts led us to identify three patterns of *alarm conditions*, which we denote with ExactDel(A, β, n), BoundDel(A, β, n), and FiniteDel(A, β):

1. ExactDel(A, β, n) specifies that whenever β is true, A must be triggered exactly n steps later and A can be triggered only if n steps earlier β was true; formally, for any trace π of the system, if β is true along π at the time point i, then A is true in $\pi[i+n]$ (Completeness); if A is true in $\pi[i]$, then β must be true in $\pi[i-n]$ (Correctness).
2. BoundDel(A, β, n) specifies that whenever β is true, A must be triggered within the next n steps and A can be triggered only if β was true within the previous n steps; formally, for any trace π of the system, if β is true along π at the time point i then A is true in $\pi[j]$, for some $i \leq j \leq i+n$ (Completeness); if A is true in $\pi[i]$, then β must be true in $\pi[j']$ for some $i-n \leq j' \leq i$ (Correctness).
3. FiniteDel(A, β) specifies that whenever β is true, A must be triggered in a later step and A can be triggered only if β was true in some previous step; formally, for any trace π of the system, if β is true along π at the time point i then A is true in $\pi[j]$ for some $j \geq i$ (Completeness); if A is true in $\pi[i]$, then β must be true along π in some time point between 0 and i (Correctness).

Figure 1 provides an example of admissible responses for the various alarms to the occurrences of the same diagnosis condition β; note how in the case of BoundDel$(A, \beta, 4)$ the alarm can be triggered at any point as long as it is within the next 4 time-steps. FiniteDel(A, β) is of particular theoretical interest since it captures the idea of diagnosis as defined in previous works [4].

An alarm condition is actually a property of the whole system since it relates a condition of the plant with an alarm of the diagnoser. Thus, when we say that a diagnoser D of P satisfies an alarm condition, we mean that the traces of the system $D \times P$ satisfy it.

Considering our previous example of the data acquisition system, we can define the following toy specification. $\beta_1 = (sdie \vee fdie_{high} \vee fdie_{low})$ indicates the fault detection condition, therefore we define the FINITEDEL(A_1, β_1) as the finite-delay fault detection. Another example could be identification of the sensor death ($\beta_2 = sdie$) within a bound: BOUNDDEL$(A_2, \beta_2, 5)$. Finally, we could be interested in knowing that some fault occurred in the filter with some precise delay information: EXACTDEL$(A_3, (fdie_{high} \vee fdie_{low}), 2)$.

3.4 Diagnosability

Given an alarm condition, we need to know whether it is possible to build a diagnoser for it. In fact, there is no reason in having a specification that cannot be realized. This property is called *diagnosability* and was introduced in [5].

In this section, we define the concept of diagnosability for the different types of alarm conditions. We proceed by first giving the definition of diagnosability in the traditional way (à la Sampath) in terms of observationally equivalent traces w.r.t. the diagnosis condition. Then, we prove that a plant P is diagnosable iff there exists a diagnoser that satisfies the specification. In the following, we will not provide definitions for finite-delay since they can be obtained by generalizing the ones for bounded-delay.

Definition 1 *Given a plant P and a diagnosis condition β, we say that* EXACTDEL(A, β, d) *is diagnosable in P iff for any pair of traces σ_1,σ_2 and for all $i \geq 0$, if $obs(\sigma_1^{i+d}) = obs(\sigma_2^{i+d})$ then $\sigma_1, i \models \beta$ iff $\sigma_2, i \models \beta$.*

Definition 2 *Given a plant P and a diagnosis condition β, we say that* BOUNDDEL(A, β, d) *is diagnosable in P iff for any pair of traces σ_1,σ_2 and for all $i \geq 0$ there exists a j, $i \leq j \leq i + d$, s.t. if $obs(\sigma_1^j) = obs(\sigma_2^j)$ and $\sigma_1, i \models \beta$ then $\sigma_2, k \models \beta$ for some k, $j - d \leq k \leq j$.*

An exact-delay alarm condition is not diagnosable in P iff there exists a pair of traces that violates the conditions of Definition 1: this would be a pair of traces σ_1 and σ_2 such that for some $i \geq 0$, $\sigma_1, i \models \beta$, $obs(\sigma_1^{i+d}) = obs(\sigma_2^{i+d})$, and $\sigma_2, i \not\models \beta$. We call such a pair a *critical pair*. Definition 2 is a generalization of Sampath's definition of diagnosability:

Theorem 1. *Let α be a propositional formula and β_α a condition that holds in a point of a trace if α holds in some point of its prefix, then α is d-delay diagnosable (as in [5]) in P iff* BOUNDDEL(A, β_α, d) *is diagnosable in P.*

The following theorem shows that if a component satisfies the diagnoser specification then the monitored plant must be diagnosable for that specification. In Section 6 on synthesis we will show also the converse, i.e., if the specification is diagnosable then a diagnoser exists.

Theorem 2. *Let D be a diagnoser for P. If D satisfies an alarm condition then the alarm condition is diagnosable in P.*

The above definition of diagnosability might be stronger than necessary, since diagnosability is defined as a global property of the plant. Imagine the situation in which there is a critical pair and after removing this critical pair from the possible executions of the system, our system becomes diagnosable. This suggests that the system was "almost" diagnosable, and an ideal diagnoser would be able to perform a correct diagnosis in all the cases except one (i.e., the one represented by the critical pair). To capture this idea, we redefine the problem of diagnosability from a global property expressed on the plant, to a local property (expressed on points of single traces).

Definition 3 *Given a plant P, a diagnosis condition β and a trace σ_1 such that for some $i \geq 0$ $\sigma_1, i \models \beta$, we say that* ExactDel$(A, \beta, d)$ *is* trace diagnosable *in $\langle \sigma_1, i \rangle$ iff for any trace σ_2 such that $obs(\sigma_1^{i+d}) = obs(\sigma_2^{i+d})$ then $\sigma_2, i \models \beta$.*

Definition 4 *Given a plant P, a diagnosis condition β, and a trace σ_1 such that for some $i \geq 0$ $\sigma_1, i \models \beta$, we say that* BoundDel$(A, \beta, d)$ *is* trace diagnosable *in $\langle \sigma_1, i \rangle$ iff there exists j, $i \leq j \leq i + d$ such that, for any trace σ_2 such that $obs(\sigma_1^j) = obs(\sigma_2^j)$ then $\sigma_2, k \models \beta$ for some $j - d \leq k \leq j$.*

A specification that is trace diagnosable in a plant along all points of all traces is diagnosable in the classical sense, and we say it is *system* diagnosable.

3.5 Maximality

As shown in Figure 1, bounded- and finite-delay alarms are correct if they are raised within the valid bound. However, there are several possible variations of the same alarm in which the alarm is active in different instants or for different periods. We address this problem by introducing the concept of *maximality*. Intuitively, a maximal diagnoser is required to raise the alarms as soon as possible and as long as possible (without violating the correctness condition).

Definition 5 *D is a maximal diagnoser for an alarm condition with alarm A in P iff for every trace π_P of P, $D(\pi_P)$ contains the maximum number of points i such that $D(\pi_P), i \models A$ in the sense that if $D(\pi_P), i \not\models A$ then there does not exist another correct diagnoser D' of P such that $D'(\pi_P), i \models A$.*

4 Formal Specification

In this section, we present the Alarm Specification Language with Epistemic operators (ASL_K). This language allows designers to define requirements on the FDI alarms including aspects such as delays, diagnosability and maximality.

Diagnosis conditions and alarm conditions are formalized using LTL with past operators [6] (from here on, simply LTL). The definitions of trace diagnosability and maximality, however, cannot be captured by using a formalization

based on LTL. Therefore, in order to capture these two concepts, we rely on temporal epistemic logic. The intuition is that this logic enables us to reason on set of observationally equivalent traces instead that on single traces (like in LTL). We assume the familiarity of the reader to LTL, but we provide a brief introduction to temporal epistemic logic, and then show how it can be used to verify diagnosability, define requirements for non-diagnosable cases and express the concept of maximality.

4.1 Temporal Epistemic Logic

Epistemic logic has been used to describe and reason about knowledge of agents and processes. There are several ways of extending epistemic logic with temporal operators. We use the logic KL_1 [2] and extended it with past operators.

A formula in KL_1 is defined as $\beta ::= p \mid \beta \wedge \beta \mid \neg\beta \mid O\beta \mid Y\beta \mid F\beta \mid X\beta \mid K\beta$. Note how this is an extension of LTL on the past, with the addition of the epistemic operator K. The intuitive semantics of $O\beta$ is that β was true in the past, while $Y\beta$ means that in the previous state β was true; the intuitive semantics of $K\beta$ is that the diagnoser *knows* that β holds in the current execution. The formal semantics of the epistemic operator K is given on indistinguishable traces:

$$\sigma_1, n \models K\beta \text{ iff } \forall \sigma_2, obs(\sigma_1^n) = obs(\sigma_2^n) \Rightarrow \sigma_2, n \models \beta.$$

Therefore, $K\beta$ holds at time n in a trace σ_1, if β holds in all traces that are observational equivalent to σ_1 at time n. This definition implicitly forces *perfect-recall* in the semantics of the epistemic operator, since we define the epistemic equivalence between traces and not between states. Moreover, the traces are compared with a synchronous semantics. Therefore, the semantics of our transition system is synchronous and with perfect-recall (compare [2]).

4.2 Diagnosis and Alarm Conditions as LTL Properties

Let $\mathcal{P}$ be a set of propositions representing either faults or elementary conditions for the diagnosis. The set $\mathcal{D}_\mathcal{P}$ of *diagnosis conditions* over $\mathcal{P}$ is any formula β built with the following rule: $\beta ::= p \mid \beta \wedge \beta \mid \neg\beta \mid O\beta \mid Y\beta$ with $p \in \mathcal{P}$. We use the abbreviations $Y^n\phi = YY^{n-1}\phi$ (with $Y^0\phi = \phi$), $O^{\leq n}\phi = \phi \vee Y\phi \vee \cdots \vee Y^n\phi$ and $F^{\leq n}\phi = \phi \vee X\phi \vee \cdots \vee X^n\phi$.

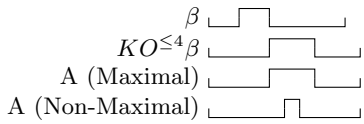
We define the *Alarm Specification Language* (ASL) in Figure 2, where we associate to each type of alarm condition an LTL formalization encoding the concepts of correctness and completeness. *Correctness*, the first conjunct, intuitively says that whenever the diagnoser raises an alarm, then the fault must have occurred. *Completeness*, the second conjunct, intuitively encodes that whenever the fault occurs, the alarm will be raised. In the following, for simplicity, we abuse notation and indicate with φ both the alarm condition and the associated LTL; for an alarm condition φ, we denote with A_φ the associated alarm variable A, and with $\tau(\varphi)$ the following formulae: $\tau(\varphi) = Y^n\beta$ for $\varphi = \textsc{ExactDel}(A, \beta, n)$; $\tau(\varphi) = O^{\leq n}\beta$ for $\varphi = \textsc{BoundDel}(A, \beta, n)$; $\tau(\varphi) = O\beta$ for $\varphi = \textsc{FiniteDel}(A, \beta)$.

Alarm Condition	LTL Formulation
$\textsc{ExactDel}(A, \beta, n)$	$G(A \rightarrow Y^n \beta) \wedge G(\beta \rightarrow X^n A)$
$\textsc{BoundDel}(A, \beta, n)$	$G(A \rightarrow O^{\leq n} \beta) \wedge G(\beta \rightarrow F^{\leq n} A)$
$\textsc{FiniteDel}(A, \beta)$	$G(A \rightarrow O\beta) \wedge G(\beta \rightarrow FA)$

Fig. 2: Alarm conditions as LTL (ASL)

Alarm Condition	Diagnosability condition
$\textsc{ExactDel}(A, \beta, n)$	$G(\beta \rightarrow X^n K Y^n \beta)$
$\textsc{BoundDel}(A, \beta, n)$	$G(\beta \rightarrow F^{\leq n} K O^{\leq n} \beta)$
$\textsc{FiniteDel}(A, \beta)$	$G(\beta \rightarrow FKO\beta)$

(a) Diagnosability Property.



(b) Example of Maximality.

Fig. 3: Diagnosability and Maximality.

4.3 Diagnosability as Epistemic Property

We can write the diagnosability tests for the different alarm conditions directly as epistemic properties that can be verified on single points of the traces (trace diagnosability) or on the entire plant (system diagnosability) (Figure 3.a). For example, the diagnosability test for $\textsc{ExactDel}(A, \beta, n)$ says that it is always the case that whenever β occurs, exactly n steps afterwards, the diagnoser *knows* that n steps before β occurred. Since K is defined on observationally equivalent traces, the only way to falsify the formula would be to have a trace in which β occurs, and another one (observationally equivalent at least for the next n steps) in which β did not occur; but this is in contradiction with the definition of diagnosability (Definition 1).

4.4 Maximality as Epistemic Property

The property of maximality says that the diagnoser will raise the alarm as soon as it is possible to know the diagnosis condition, and the alarm will stay up as long as possible. The property $K\tau(\varphi) \rightarrow A$ encodes this behavior:

Theorem 3. *D is maximal for φ in P iff $D \times P \models G(K\tau(\varphi) \rightarrow A_\varphi)$.*

Whenever the diagnoser knows that $\tau(\varphi)$ is satisfied, it will raise the alarm. For bounded- and finite-delay alarms, this guarantees also that the alarm will stay up if possible, since $K\tau(\varphi) \rightarrow XKY\tau(\varphi)$. An example of maximal and non-maximal alarm is given in Figure 3.b. Note that according to our definition, the set of maximal alarms is a subset of the non-maximal ones.

4.5 ASL_K Specifications

The formalization of ASL_K (Figure 4) is obtained by extending ASL (Figure 2) with the concepts of maximality and diagnosability, defined as epistemic properties. When *maximality* is required we add a third conjunct following Theorem 3.

	Template	$maximality = False$	$maximality = True$
$diag = System$	ExactDel	$G(A \rightarrow Y^n\beta) \wedge G(\beta \rightarrow X^nA)$	$G(A \rightarrow Y^n\beta) \wedge G(\beta \rightarrow X^nA) \wedge$ $G(KY^n\beta \rightarrow A)$
	BoundDel	$G(A \rightarrow O^{\leq n}\beta) \wedge G(\beta \rightarrow F^{\leq n}A)$	$G(A \rightarrow O^{\leq n}\beta) \wedge G(\beta \rightarrow F^{\leq n}A) \wedge$ $G(KO^{\leq n}\beta \rightarrow A)$
	FiniteDel	$G(A \rightarrow O\beta) \wedge G(\beta \rightarrow FA)$	$G(A \rightarrow O\beta) \wedge G(\beta \rightarrow FA) \wedge$ $G(KO\beta \rightarrow A)$
$diag = Trace$	ExactDel	$G(A \rightarrow Y^n\beta) \wedge$ $G((\beta \rightarrow X^nKY^n\beta) \rightarrow (\beta \rightarrow X^nA))$	$G(A \rightarrow Y^n\beta) \wedge$ $G((\beta \rightarrow X^nKY^n\beta) \rightarrow (\beta \rightarrow X^nA)) \wedge$ $G(KY^n\beta \rightarrow A)$
	BoundDel	$G(A \rightarrow O^{\leq n}\beta) \wedge$ $G((\beta \rightarrow F^{\leq n}KO^{\leq n}\beta) \rightarrow (\beta \rightarrow F^{\leq n}A))$	$G(A \rightarrow O^{\leq n}\beta) \wedge$ $G((\beta \rightarrow F^{\leq n}KO^{\leq n}\beta) \rightarrow (\beta \rightarrow F^{\leq n}A)) \wedge$ $G(KO^{\leq n}\beta \rightarrow A)$
	FiniteDel	$G(A \rightarrow O\beta) \wedge$ $G((\beta \rightarrow FKO\beta) \rightarrow (\beta \rightarrow FA))$	$G(A \rightarrow O\beta) \wedge$ $G((\beta \rightarrow FKO\beta) \rightarrow (\beta \rightarrow FA)) \wedge$ $G(KO\beta \rightarrow A)$

Fig. 4: ASL_K specification patterns.

When $diag = trace$ instead, we precondition the completeness to the trace diagnosability (as defined in Figure 3.a); this means that the diagnoser will raise an alarm whenever the diagnosis condition is satisfied and the diagnoser is able to know it. The formalizations presented in the table can be simplified, but are left as-is to simplify their comprehension. For example, in the case $diag = trace$, we do not need to verify the completeness due to the following result:

Theorem 4. *Given a diagnoser D for a plant P and a trace diagnosable alarm condition φ, if D is maximal for φ, then D is complete.*

A similar result holds for ExactDel in the non-maximal case, that becomes: $G(A \rightarrow Y^n\beta) \wedge G(KY^n\beta \rightarrow A)$. Finally, the implications for the completeness in the trace diagnosability case can be rewritten as, e.g., $G((\beta \wedge FKO\beta) \rightarrow (FA))$. Another interesting result is the following:

Theorem 5. *Given a diagnoser D for a plant P and a system diagnosable condition φ, if D is maximal for φ and φ is diagnosable in P then D is complete.*

An ASL_K specification is built by instantiating the patterns defined in Figure 4. For example, we would write $\text{ExactDel}_K(A, \beta, n, trace, True)$ for an exact-delay alarm A for β with delay n, that satisfies the trace diagnosability property and is maximal. An introductory example on the usage of ASL_K for the specification of a diagnoser is provided in [7].

5 Validation and Verification of ASL_K Specifications

Thanks to the formal characterization of ASL_K, it is possible to apply formal methods for the validation and verification of a set of FDI requirements. In *validation* we verify that the requirements capture the interesting behaviors and exclude the spurious ones, before proceeding with the design of the diagnoser. In *verification*, we check that a candidate diagnoser fulfills a set of requirements.

Validation In the following we focus on the validation of an alarm specification, but the same ideas can be applied to a set of diagnosis conditions. We consider a set of environmental assumptions E and a specification $\mathcal{A}_\mathcal{P}$. The environment assumption E may include both assumption on the plant's input and an abstraction of the plant. It can vary in a spectrum starting from trivially no assumption ($E = \top$), to some LTL properties, to a detailed model of the plant, going through several intermediate levels. The idea is that throughout the different phases of the development process, we have access to better versions of the plant model, and therefore the analysis can be refined. For example, it might be possible to provide some assumption on the maximum number of faults in the system, or on their dynamics, before a complete description of the subsystems is available.

Known techniques for requirements validation (e.g.,[8]) include checking their consistency, their compatibility with some possible scenarios, whether they entail some expected properties and if they are realizable, i.e., if there exists an implementation satisfying the requirements. In the following we instantiate these checks for the alarm specification.

In Section 6, we prove that we can always synthesize a diagnoser satisfying $\mathcal{A}_\mathcal{P}$, with the only assumption that if $\mathcal{A}_\mathcal{P}$ contains some system diagnosable alarm condition, then that condition is diagnosable in the plant. This means that any specification $\mathcal{A}_\mathcal{P}$ is consistent by construction (just consider a diagnosable plant). Moreover, the check for realizability reduces to checking that the plant is diagnosable for the system diagnosable conditions in $\mathcal{A}_\mathcal{P}$. The diagnosability check can be performed via epistemic model-checking (Section 4.3) or it can be reduced to an LTL model-checking problem using the twin-plant construction [9].

As check of possible scenarios, we would like that alarms should eventually be activated, but also that alarms are not always active. This means that for a given alarm condition $\varphi \in \mathcal{A}_\mathcal{P}$, we are interested in verifying that there is a trace $\pi \in E$ and a trace $\pi' \in E$ s.t. $\pi \models FA_\varphi$ and $\pi' \models F\neg A_\varphi$. This can be done by checking the unsatisfiability of $(E \wedge \varphi) \rightarrow G\neg A_\varphi$ and $(E \wedge \varphi) \rightarrow GA_\varphi$.

As check of entailed properties, it is interesting to understand whether there is some correlation between alarms in order to simplify the model, or to guarantee some redundancy requirement. To check whether $A_{\varphi'}$ is a more general alarm than A_φ (subsumption) we verify that $(E \wedge \varphi \wedge \varphi') \rightarrow G(A_\varphi \rightarrow A_{\varphi'})$ is a tautology. A trivial example of subsumption of alarms is given by the definition of maximality: any non-maximal alarm is subsumed by its corresponding maximal version. Finally, we can verify that two alarms are mutually exclusive by checking the validity of $(E \wedge \varphi \wedge \varphi') \rightarrow G\neg(A_\varphi \wedge A_{\varphi'})$. In general, the validation of alarm conditions requires reasoning in temporal epistemic logic, however, the validation of diagnosis condition only requires reasoning on LTL with past.

Verification The verification of a system w.r.t. a specification can be performed via model-checking techniques using the semantics of the alarm conditions:

Definition 6 *Let D be a diagnoser for alarms $\mathcal{A}$ and plant P. We say that D satisfies a set $\mathcal{A}_\mathcal{P}$ of ASL_K specifications iff for each φ in $\mathcal{A}_\mathcal{P}$ there exists an alarm $A_\varphi \in \mathcal{A}$ and $D \times P \models \varphi$.*

To perform this verification steps, we need in general a model checker for KL_1 with synchronous perfect recall such as MCK [10]. However, if the specification falls in the pure LTL fragment (ASL) we can verify it with an LTL model-checker such as NuSMV [11] thus benefiting from the efficiency of the tools in this area. Moreover, a diagnoser is required to be compatible with the plant. Therefore, we need to take care that the synchronous composition of the plant with the diagnoser does not reduce the behaviors of the plant. This would imply that there is a state and an observation that are possible for the plant, but not taken into account by the diagnoser. Compatibility can be checked with dedicated tools such as Ticc [12] based on game theory. However, here we require compatibility in all environments and therefore, compatibility can be checked by model checking by adding a sink state to the diagnoser, so that if we are in a state and we receive an observation that was not covered by the original diagnoser, we go to the sink state. Once we modified the diagnoser, we verify that $D \times P \models G\neg SinkState$.

6 Synthesis of a Diagnoser from an ASL_K Specification

In this section, we sketch an algorithm to synthesize a diagnoser that satisfies a given specification $\mathcal{A}_\mathcal{P}$. The algorithm considers the most expressive case of ASL_K (maximal/trace diagnosable), satisfying, therefore all other cases.

The idea of the algorithm is to generate an automaton that encodes the set of possible states in which the plant could be after each observations. The result is achieved by generating the power-set of the states of the plant, and defining a suitable transition relation among the elements of this set that only considers the observable information. We call the sets in the power-set *belief states*. Each belief state of the automaton can be annotated with the alarms that are satisfied in all the states of the belief state, obtaining the diagnoser.

Our algorithm resembles the construction by Sampath [5] and Schumann [13]. The main differences are that we consider LTL Past expression as diagnosis condition, and not only fault events as done in previous works. Moreover, instead of providing a set of possible diagnosis, we provide alarms: we need to be certain that the alarm condition is satisfied in all possible diagnosis in order to raise the alarm. This gives raise to a 3-valued alarm system, in which we *know* that the fault occurred, *know* that the fault did *not* occurred or are uncertain.

Given a plant $P = \langle V^P, V_o^P, W^P, W_o^P, I^P, T^P\rangle$, let S be the set of states of P. The *belief automaton* is defined as $\mathcal{B}(P) = \langle B, E, B_0, R\rangle$ where $B = 2^S$, $E = 2^{W_o^P \cup V_o^P}$ and $B_0 \subseteq B$ and $R : (B \times E) \to B$ are defined as follows.

We define $B_0 = \{b \mid \text{there exists } u \in 2^{V_o^P} \text{ s.t. for all } s \in b,\ s \models I^P \text{ and } obs(s) = u\}$: we assume that the diagnoser can be initialized by observing the plant, and each initial belief state must, therefore, be compatible with one of the possible initial observations on the plant. The transition function R is defined as follows $R(b, e) = \{s' \mid \exists s \in b \text{ s.t. } \langle s, i, s'\rangle \models T^P,\ obs(s') = e_{|V_o^P}, obs(i) = e_{|W_o^P}\}$: the belief state $b' = R(b, e)$ is a successor of b iff all the states in b' are compatible with the observations from a state in b.

The diagnoser is obtained by annotating each state of the belief automaton with the corresponding alarms. To do this we explore the belief automaton, and annotate with A_φ all the states b that satisfy the temporal property $\tau(\varphi)$: $b \models A_\varphi$ iff $\forall s \in b.s \models \tau(\varphi)$. It might occur that neither $K\tau(\varphi)$ nor $K\neg\tau(\varphi)$ hold in a state. In this case there is at least a state in the belief state in which $K\tau(\varphi)$ holds and one in which it does not hold. This pair of states represents uncertainty, and are caused by non-diagnosabile traces.

We define D_φ as the diagnoser for φ. For the propositional case $\tau(\varphi) = p$, $D_\varphi = \langle V^{D_\varphi}, V_o^{D_\varphi}, W^{D_\varphi}, W_o^{D_\varphi}, I^{D_\varphi}, T^{D_\varphi} \rangle$ is a symbolic representation of $\mathcal{B}(P)$ with $V_o^{D_\varphi} = V_o^P \cup \{A_\varphi\}$, $W_o^{D_\varphi} = W_o^P$ and such that every state b of D_φ represents a state in B (with abuse of notation we do not distinguish between the two) and, for all $v \in V_o^{D_\varphi}$, $v \in obs(b)$ iff for all $s \in b$, $v \in s$, and such that every observation e of D_φ represents an observation in E and $obs(e) = e_{|W_o^{D_\varphi}}$. The following holds:

Theorem 6 (Compatibility). *D_φ is compatible with P.*

Theorem 7 (Correctness, Completeness and Maximality). *D_φ is correct (i.e. $D_\varphi \times P \models G(A_\varphi \rightarrow \tau(\varphi))$), maximal (i.e. $D_\varphi \times P \models G(K(\tau(\varphi)) \rightarrow A_\varphi)$) and complete (under the assumption that if φ is system diagnosable, then φ is diagnosable in P).*

All other alarm conditions can be reduced to the propositional case. We build a new plant P' by adding a monitor variable $\overline{\tau}$ to P s.t., $P' = P \times (G(\tau(\varphi) \leftrightarrow \overline{\tau}))$, where we abuse notation to indicate the automaton that encodes the monitor variable. By rewriting the alarm condition as $\varphi' = \textsc{ExactDel}(A_\varphi, \overline{\tau}, 0)$, we obtain that $D \times P \models \varphi$ iff $D \times P' \models \varphi'$.

7 Industrial Experience

The framework described in this paper has been motivated by, and used in, the AUTOGEF project [1], funded by the European Space Agency. The main goal of the project was the definition of a set of requirements for an on-board Fault Detection, Identification and Recovery (FDIR) component and its synthesis. The problem was tackled by synthesizing the Fault Detection (FDI) and Fault Recovery (FR) components separately, with the idea that the FDI provides sufficient diagnosis information for the FR to act on.

The AUTOGEF framework was evaluated using scalable benchmark examples. Moreover, Thales Alenia Space evaluated AUTOGEF on a case study based on the EXOMARS Trace Gas Orbiter. This case-study is a significant exemplification of the framework described in this paper, since it covers all the phases of the FDIR development process. The system behavior (including faulty behavior) was modeled using a formal language and table- and pattern-based description of the mission phases/modes and observability characteristics of the system. The specification of FDIR requirements by means of patterns greatly simplified the accessibility of the tool to engineers that were not experts in formal methods. Specification of alarms was carried out in the case of finite delay, under the

assumption of trace diagnosability and maximality of the diagnoser. Moreover, different faults and alarms were associated with specific mission phase/mode and configurations of the system, which enabled generation of specific alarms (and recoveries) for each configuration. The specification was validated, by performing diagnosability analysis on the system model. The synthesis routines were run on a system composed of 11 components, with 10 faults in total, and generated an FDI component with 754 states. Finally, the correctness of the diagnoser was verified by using model-checking routines. Synthesis and verification capabilities have been implemented on top of the NuSMV model checker. We remark that the ability to define trace diagnosable alarms was crucial for the synthesis of the diagnoser, since most of the modeled faults were not system diagnosable.

Successful completion of the project, and positive evaluations from the industrial partner and ESA, suggest that a first step towards a formal model-based design process for FDIR was achieved.

8 Related Work

Previous works on formal FDI development have considered the specification and synthesis in isolation. Our approach differs with the state of the art because we provide a comprehensive view on the problem. Due to the lack of specification formalism for diagnosers, the problem of verifying their correctness, completeness and maximality was, to the best of our knowledge, unexplored.

Concerning specification and synthesis [14] is close to our work. The authors present a way to specify the diagnoser using LTL properties, and present a synthesis algorithm for this specification. However, problems such as maximality and trace diagnosability are not taken into account. Interesting in [14] is the handling of diagnosis condition with future operators.

Some approaches exist that define diagnosability as epistemic properties. Two notable examples are [15] and [16], where the latter extends the definition of diagnosability to a probabilistic setting. However, these works focus on finite-delay diagnosability only, and do not consider other types of delays and the problem of trace diagnosability.

Finally, we extend the results on diagnosability checking from [9] in order to provide an alternative way of checking diagnosability and redefine the concept of diagnosability at the trace level.

9 Conclusions and Future Work

This paper presents a formal framework for the design of FDI components, that covers many practically-relevant issues such as delays, non-diagnosability and maximality. The framework is based on a formal semantics provided by temporal epistemic logic. We covered the specification, validation, verification and synthesis steps of the FDI design, and evaluated the applicability of each step on a case-study from aerospace. To the best of our knowledge, this is the first work that provides a formal and unified view to all the phases of FDI design.

In the future, we plan to explore the following research directions. First, we will extend FDI to deal with asynchronous and infinite-state systems. In this work we addressed the development of FDI for finite state synchronous systems only. However, it would be of practical interest to consider infinite state systems and timed/hybrid behaviors. Another interesting line of research is the development of optimized ad-hoc techniques for reasoning on the fragment of temporal epistemic logic that we are using, both for verification and validation, and evaluating and improving the scalability of the synthesis algorithms. Finally, we will work on integrating the FDI component with the recovery procedures.

References

1. European Space Agency: ITT AO/1-6570/10/NL/LvH "Dependability Design Approach for Critical Flight Software". Technical report (2010)
2. Halpern, J.Y., Vardi, M.Y.: The complexity of reasoning about knowledge and time. lower bounds. Journal of Computer and System Sciences **38**(1) (1989) 195–237
3. Grastien, A., Anbulagan, A., Rintanen, J., Kelareva, E.: Diagnosis of discrete-event systems using satisfiability algorithms. In: AAAI - Vol. 1. (2007) 305–310
4. Rintanen, J., Grastien, A.: Diagnosability testing with satisfiability algorithms. In Veloso, M.M., ed.: IJCAI. (2007) 532–537
5. Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., Teneketzis, D.C.: Ieee transactions on control systems technology. Volume 4. (1996) 105–124
6. Lichtenstein, O., Pnueli, A., Zuck, L.: The glory of the past. In Parikh, R., ed.: Logics of Programs. Volume 193. Springer Berlin Heidelberg (1985) 196–218
7. Bozzano, M., Cimatti, A., Gario, M., Tonetta, S.: Formal Specification and Synthesis of FDI through an Example. In: Workshop on Principles of Diagnosis (DX'13). (2013) Available at URL https://es.fbk.eu/people/gario/dx2013.pdf.
8. Cimatti, A., Roveri, M., Susi, A., Tonetta, S.: Validation of requirements for hybrid systems: A formal approach. ACM Transactions on Software Engineering and Methodology **21**(4) (2012) 22
9. Cimatti, A., Pecheur, C., Cavada, R.: Formal Verification of Diagnosability via Symbolic Model Checking. In: IJCAI. (2003) 363–369
10. Gammie, P., Van Der Meyden, R.: Mck: Model checking the logic of knowledge. Computer Aided Verification (2004) 256–259
11. Cimatti, A., Clarke, E.M., Giunchiglia, E., Giunchiglia, F., Pistore, M., Roveri, M., Sebastiani, R., Tacchella, A.: NuSMV 2: An OpenSource Tool for Symbolic Model Checking. In: CAV. (2002) 359–364
12. Adler, B., de Alfaro, L., da Silva, L., Faella, M., Legay, A., Raman, V., Roy, P.: Ticc: A Tool for Interface Compatibility and Composition. In: CAV. (2006) 59–62
13. Schumann, A.: Diagnosis of discrete-event systems using binary decision diagrams. Workshop on Principles of Diagnosis (DX'04) (2004) 197–202
14. Jiang, S., Kumar, R.: Failure diagnosis of discrete event systems with linear-time temporal logic fault specifications. In: IEEE Transactions on Automatic Control. (2001) 128–133
15. Ezekiel, J., Lomuscio, A., Molnar, L., Veres, S.: Verifying Fault Tolerance and Self-Diagnosability of an Autonomous Underwater Vehicle. In: IJCAI. (2011) 1659–1664
16. Huang, X.: Diagnosability in concurrent probabilistic systems. In: Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems. (2013)