On the safety assessment of RPAS safety policy

Diego Couto, Kevin Delmas, Xavier Pucel

On the safety assessment of RPAS safety policy

Diego COUTO
ONERA
Toulouse, France
Email: diego.couto_reyes@onera.fr

Kevin Delmas
ONERA
Toulouse, France
Email: kevin.delmas@onera.fr

Xavier PUCEL
ONERA
Toulouse, France
Email: xavier.pucel@onera.fr

***Abstract*—Remotely Piloted Aircraft Systems (RPASs) should become shortly mainstream to meet the operational requirements of emerging applications such as autonomous transport or infrastructure monitoring. The integration of these systems in the airspace creates new issues, especially concerning safety. The variety of operational contexts where RPAS are involved defeats the safety assessment processes used for decades for large civil aircrafts. One of the major issues is the lack of standardisation of RPAS safety policies. We claim that including the safety policy during safety assessment is one of the prominent challenges for a safe integration of the RPAS in the airspace. Hence, we provide a formal modelling and analysis framework dedicated to the safety assessment of RPAS safety policies.**

I. Introduction

The Remotely Piloted Aircraft Systems (RPASs) should become shortly mainstream to meet the operational requirements of emerging applications such as autonomous transport [1] or infrastructure monitoring [2]. The integration of these systems in the airspace creates new issues, especially concerning safety as described in [3]. The variety of operational contexts where RPAS are involved defeats the safety assessment processes, like the ARP4754 [4], used for decades for large civil aircrafts.

One of the major issues for the safety assessment of RPAS is the lack of standardisation of their safety policies. We call a *safety policy* the set of procedures that are applied to select a control mode (nominal or degraded) for the Unmanned Aircraft Vehicle (UAV) depending on its health status. Designing a safety policy for an RPAS is significantly different than for a civil aircraft, mostly because the possible operational contexts are much more diverse. This means that new requirements such as modularity and distribution are applicable to RPAS safety policies, while common requirements such as dynamism, determinism and decision soundness are more challenging to meet.

We claim that integrating an analysis of the safety policy in the safety assessment is one of the prominent challenges for a safe integration of the RPAS in the airspace. In this paper, we provide a formal modelling framework dedicated to the safety assessment of RPAS safety policies. The contribution of this paper is four-fold: 1) provide a formal framework for modelling both the propagation of failures within the RPAS and the safety policy; 2) identify meaningful properties and requirements for evaluating a safety policy and ensuring the dependability of the RPAS; 3) describe an intelligible yet formal way to model the safety policy through a preference-based formalism; 4) use automated verification tools to perform the safety assessment.

The remainder of the paper is organised as follows: section II describes the major issues related to the modelling and analysis of a safety policy, and how these issues can be seen as an estimation problem; section III provides a formal description of the framework, illustrated on a simple yet comprehensive RPAS use-case; section IV positions the proposed approach with related work.

II. General overview of the approach

A. Reminder on safety processes

Complex systems are usually developed using a safety assessment process, that is a set of verification and validation activities performed throughout the system development process. For large civilian aircraft, the standard ARP4754 [4] describes a safety assessment process tightly linked to the development process [5]. Generally, a safety process can be seen as an instantiation of the following safety assessment pattern at various levels.

a) Hazard Analysis : Identify the failure conditions, in a given context, that may rise safety issues so-called *failure conditions* and allocate *safety objectives* to these conditions commensurate with the hazard's severity. These safety objectives enforce some bounds over safety indicators such as the minimal number of root failures, or the failure rate of a particular failure condition.

b) Safety Assessment: Assess the proposed architecture against the safety objectives. To perform the safety assessment, designers can rely on formalisms (for instance listed in the ARP4761 [6]) enabling the analyst to describe the contribution of the failures of the architecture's components to the failure conditions. As identified by [7], classical formalisms embrace an architecture-agnostic modelling. Consequently, adapting the safety models after an evolution of the system design may be cumbersome. Architecture-aware formalisms [8], [9] have been introduced to overcome the limitations of architecture-agnostic formalisms. Architecture-aware formalisms provide a way to define the dysfunctional behaviour of entities called *components* that are instantiated and connected to build the architecture of a *system*. Ultimately the interconnected components can be analysed by automatic solvers like [10], [11] to derive a safety assessment. This kind of approach is the so-called Model-Based Safety Assessment.

For general aviation, the advisory circular 23 [12] describes a standard high-level functional decomposition of aircraft systems. Based on this functional decomposition, it performs a high-level and generic Hazard Analysis providing standard failure conditions, and defines a generic safety policy. This

documents provides a framework where one can focus on the Safety Assessment and analyse the contributions of each system component to the standard failure conditions, provided the system meets the assumptions made in the document.

The high-level generic functional decomposition proposed in [3] could be a starting point for a similar document for RPAS. However, we found it impossible to extend this approach to a generic Hazard Analysis and a generic safety policy for RPAS. The main reason is the diversity of both use cases and UAV designs (quad-rotor, fixed wing, etc). The hazards can vary significantly according to the operational context (populated or desert area), and the ways to mitigate those hazards, i.e. the safety policy, depends heavily on the physical capacities of the UAV.

Instead of trying to reproduce a unique generic Hazard Analysis and safety policy as done in [12], we opt for an approach where Hazard Analysis and safety policy are tailored to each operation, and are specified early in the safety assessment. Our ambition is to provide a tool that facilitates this way of performing safety assessment.

B. What is a safety policy ?

The approach we propose for safety assessment is organised around the notion of *safety policy*. Informally, the safety policy provides a way to mitigate hazards by using appropriate behaviour, especially in the presence of faults.

Formally, we model a safety policy as a process that produces decisions about the *control mode* to apply using as input the *health status* of an RPAS. We first define these two concepts.

Definition 2.1 (Control Mode): We denote $\mathcal{M}$ the set of control modes of the UAV, where a control mode $m \in \mathcal{M}$ is a way in which orders from the pilot are translated into commands for the control mechanisms to act on the flight control surfaces. A control mode can be considered as a configuration of the UAV.

Example 2.1 (Control Mode): The Auto Pilot (AP) mode is a control mode that only receives a target position as input, and computes the trajectory to reach it as well as all the necessary manoeuvres required to follow this trajectory. Another control mode is the Emergency manual (EM), where the pilot sends low level commands (direct pitch, roll, yaw, thrust) and the UAV applies them without any type of filtering or assistance and sends a visual feedback (*e.g.*a First Person View camera). Some intermediate modes can be available, where the pilot provides some intermediate level inputs (speed, altitude, manoeuvre type), and the UAV translates them into low level commands.

Definition 2.2 (Health Status): Let $\mathcal{R}$ be the set of resources (hardware, software or functional artefacts) that are used in any control mode. Only a subset $\mathcal{R}_\mathcal{A} \subset \mathcal{R}$ of the resources can be monitored; the health status is composed of the availability and integrity of the monitored resources. It is classically estimated by analysing alarms obtained from RPAS monitoring.

Example 2.2 (Health Status): Let the Control and Command link (denoted C^2 link) be a monitored resource, $C^2 \in \mathcal{R}_\mathcal{A}$. The health status of the C^2 link is comprised of both the availability (*i.e* can the UAV and the pilot communicate) and the integrity (*i.e* is information transmitted correctly between the pilot and the UAV) of this resource.

Definition 2.3 (Safety policy): A *safety policy* is a procedure that specifies at each instant in time the control mode that should be enabled in the RPAS. It usually depends on the past and current health status as well as the current operational context of the UAV.

Safety policies can be distributed across several actors. In our approach, we focus on safety policies that are structured as illustrated in Figure 1, where the policy is distributed across two actors: the pilot and UAV. The procedure contains several steps: 1) the system uses monitors to detect abnormal conditions and raise alarms; 2) each actor receives alarms about a limited subset of monitorable resources, and estimates the health status according to these alarms; 3) each actor applies decision rules and determines a control mode to apply; 4) eventually one actor (the UAV in our case) is in charge of selecting the final control mode among the provided ones.

Note that the decision rules are not the same for all actors, partly because they do not have access to the same information, and partly because the rules must account for a potential loss of communication. Moreover we do not assume that the *pilot* is necessarily a human operator or a more complex system involving technical items. This distinction may be necessary to identify the safety objectives that can be considered for this actor. Nevertheless the main objective of this paper is to model the interaction between the actors and assess their impact on the safety, therefore the notion of actor's nature is out of the scope of this paper.

C. Modelling a safety policy: challenges

Modelling a safety policy and integrating its analysis in the safety assessment of an RPAS raises the following challenges.

a) Modularity: Across its lifetime, an RPAS is likely to be used in environments that present different hazards. Each time, the safety policy must be adapted to its environment. Similarly, various RPASs can be used for a given mission, but the safety policy must be tailored to each system. Finally, the tasks that are assigned to a RPAS may vary inside the same environment. A key point for the usability of RPAS is that the safety policy must be able to be decomposed into parts that are related to the UAV, to the environment and to the operations, so that these parts can be reused and/or adapted to new vectors, environments or operations.

b) Dynamism: The safety policy generally relies on the order in which the alarms occur, and is thus a dynamic process. So the first challenge is to capture its dynamics to assess its safety impacts on the RPAS.

c) Determinism: A safety policy is generally provided as a set of *rules*, that associate control mode to apply when a given condition over the health status is fulfilled (*i.e* if the communication is unavailable, return to the home point). Such rules are seldom *exclusive*, so for a given health status a set of rules may apply. Since a single control can be selected at each time step to control the UAV, some rules must be applied in priority. The challenge is to provide a formalism to prioritise

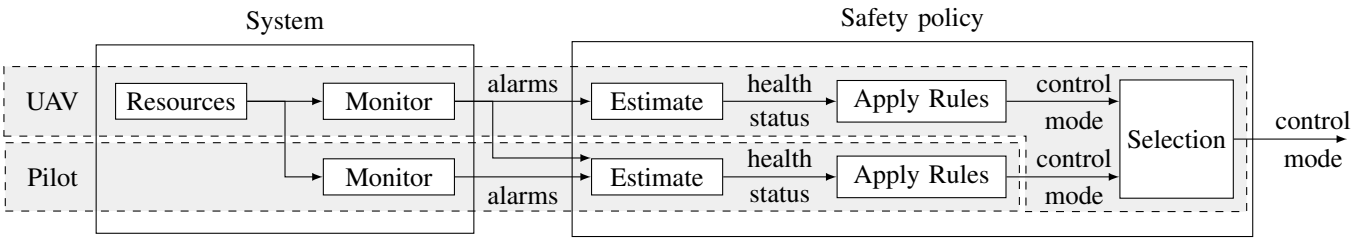

Fig. 1: Overview of a pilot/UAV distributed safety policy for a RPAS

these rules, in a way that is deterministic (the same inputs lead to the same outputs) and understandable by a pilot.

d) Distribution: A safety policy is usually distributed over several actors, typically the UAV and the pilot. The safety policy of the whole RPAS system (decomposed as the pilot and UAV local safety policies) must then contain a module in charge of the final decision when local safety policies disagree.

e) Decision: The final challenge for a safety policy, and not the least, is to make the right decisions. A safety policy uses as input the health status and the operational context to select the control mode to apply, but these inputs are obtained by on-board monitoring and/or pilot monitoring activities. However, because not all resources can be monitored, and because the monitoring can be itself fallible, most of the time these inputs are uncertain. This makes the decision challenge particularly difficult, and motivates the use of formal models and analyses to make sure the safety policy does not contain intricate bugs.

D. Modelling a safety policy as an estimation problem

Our approach addresses the decision challenge by decomposing it into two parts. The first part addresses the uncertainty about the health status and operational context by providing an estimate for them. This estimate is then used in the second part to select the control mode most appropriate to these estimated health status and operational context.

Some work like [13] maintain a set of possible health status sequences. Even if we assume an onboard computer could handle this large amount of information, we believe this approach is not relevant for RPAS. Indeed, presenting to the pilot a large set of possible failure scenarios will likely confuse him and increase the probability of human error.

In contrast we consider that a unique health status must be computed at each time step. When several health statuses can explain the observed alarms, only one of them will be chosen. This can be achieved using dedicated formalisms such as conditional preferences [14].

In particular, preference based estimation problem has been addressed in the literature by [15]. In this work, the estimation problem is formalised as a satisfiability problem over a set of constraints expressed with propositional logic and Past-Time Linear Temporal Logic (PTLTL). These constraints are split in two categories: 1) the hard constraints denoted Δ modelling the possible states and transitions of the system; 2) the soft constraints denoted Γ encoding the selection of a unique estimation among the system's states compliant with Δ and the observations as a conditional preference model.

In this paper we reuse the formalism from [15], and recall the main elements.

Definition 2.4 (Estimation Model): An estimation model is defined over a set of variables V partitioned into observable variables O and estimated variables E. All variables are either observed or estimated ($O \cap E = \emptyset$ and $O \cup E = V$).

The system behaviour is defined by a set Δ of hard constraints expressed over V using PTLTL. They express the temporal dynamics of the system and are always satisfied. The set of states that can be reached by the system is denoted S, and the initial state $s_0 \in S$.

The estimation strategy is defined by a sequence Γ of conditional preferences of the form $\big((v_0 \leftrightsquigarrow c_0), \ldots, (v_n \leftrightsquigarrow c_n)\big)$ where $v_i \in E$ is an estimated variable, and c_i is a formula expressed in PTLTL that represents the condition when we prefer to estimate v_i to true. For consistency reasons, c_i can only depend on past variables, present observable variables, and on present variables whose value is determined by a prior preference. Preferences are *soft constraints*, they are applied only when both values are possible for the variable, i.e. both values are consistent with the constraints of Δ.

In an estimation model, a system state is a Boolean assignment to variables of V, an observation is an assignment to O and an estimation to E. A sequence of states $(s_0, \ldots, s_k) \in S^*$ is a possible execution trace of the system if and only if it satisfies all the PTLTL formulas in Δ. Given a previous estimated state $\hat{s}$ and an observation o, the next estimated state is the assignment that extends o, and that satisfies preferences whenever possible (preferences are evaluated in sequential order). The estimation process is illustrated below.

Example 2.3 (Hard Constraints (Δ)): Let r_1 and r_2 be two UAV resources, respectively monitored by two alarms are *active low*: if their power supply fails, they will trigger. Let r_{pow} be a third resource that provides power to r_1, r_2 and their alarms. One can model the behaviour of this system with propositional (i.e. Boolean) variables h_1 (resp. h_2 and h_{pow}) representing that r_1 (resp. r_2 and r_{pow}) is healthy, and a_1 (resp. a_2) representing that the alarm for resource r_1 (resp. r_2) is triggered.

Formally, the model uses observable variables $O = \{a_1, a_2\}$ and estimated variables $E = \{h_1, h_2, h_{pow}\}$, and no more.

The behaviour "At every instant, the i$^{\text{th}}$ alarm triggered if and only if the power supply or monitored resource are not

ok" is represented by the following constraints:

$$\Delta = \left\{ \begin{array}{l} (\neg a_1) \Leftrightarrow h_{pow} \wedge h_1, \\ (\neg a_2) \Leftrightarrow h_{pow} \wedge h_2 \end{array} \right\}$$

The classical Boolean operators such as logical negation $\neg$, conjunction $\wedge$, disjunction $\vee$ and equivalence $\Leftrightarrow$ are available for modelling. In addition, past-time temporal operators such as yesterday Y, once O and since S can be used as well to express what dynamic behaviour is possible in the system.

When describing a state or observation, we denote $A = v_1 \cdots v_n \overline{v_{n+1}} \cdots \overline{v_{n+m}}$ the assignment where variables $v_1, \cdots, v_n$ are assigned to true, variables $v_{n+1}, \cdots, v_{n+m}$ are assigned to false, while variables not mentioned are unassigned.

Example 2.4 (Soft constraints (Γ)): Let us consider the initial state $s_0 = h_{pow} h_1 h_2 a_1 a_2$. If at the first time step, we receive the observation $o_1 = a_1 a_2$ then five possible states are consistent with Δ, s_0 being the previous state, and o_1 being the current observation. These states are $a_1 a_2 h_{pow} \overline{h_1 h_2}$, $a_1 a_2 \overline{h_{pow}} h_1 h_2$, $a_1 a_2 \overline{h_{pow}} h_1 \overline{h_2}$, $a_1 a_2 \overline{h_{pow} h_1} h_2$, and $a_1 a_2 \overline{h_{pow} h_1 h_2}$.

The principle of state estimation is to provide a decision procedure that selects one of these candidates, specified by a conditional preference model Γ. The preference language is expressive enough to implement the following reasoning: "If all the alarms are simultaneously triggered at the same time step, we prefer to explain them by a failure of the power supply", formally written as the following preference:

$$h_{pow} \leftsquigarrow \neg Y(a_1) \wedge \neg Y(a_2) \wedge a_1 \wedge a_2 \qquad (\gamma_1)$$

By adding two additional preferences about h_1 and h_2, we can build a complete estimation model. For example, the following model completes the reasoning with the optimistic principle that "If there is no alarm, or if the alarm is already explained by a fault in the power supply, we prefer to assume the resource is healthy":

$$h_1 \leftsquigarrow \neg a_1 \vee \neg h_{pow} \qquad (\gamma_2)$$
$$h_2 \leftsquigarrow \neg a_2 \vee \neg h_{pow} \qquad (\gamma_3)$$

Then the ordered sequence $\Gamma = (\gamma_1, \gamma_2, \gamma_3)$ defines an estimation model that selects state $a_1 a_2 \overline{h_{pow}} h_1 h_2$ for the previous state s_0 and current observation o_1. Note that preferences γ_2 and γ_3 can use the value of variable h_{pow} in their condition, as it is fixed first by γ_1. γ_1 however can only depend on past and observable variables.

This model can be used to generate an incremental estimator as described in [15]. In some instances, it is possible that an estimation model may be undefined for some of the system observation sequences as explained in [16]. In addition to this verification, their tool supports the verification of arbitrary properties for the pair system/estimator. In the section III, we describe how to model a safety policy with this approach, and illustrate how safety objectives can be expressed as properties for the pair system/estimator. We then show how automated verification can be used to perform safety assessment.

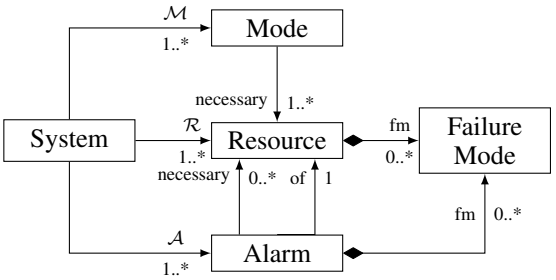

Fig. 2: Entity relation diagram of an RPAS safety policy.

III. Modeling framework

This section presents how we instantiate an estimation model to represent the safety policy of an RPAS. It is organised as follows. First we describe how each model part is implemented in terms of hard constraints and preferences. Second, we discuss how this model fits the requirements of modularity, dynamism, determinism, distribution and decision described in the previous section. Finally, we explain the analyses that can be run on the model that contribute to the safety assessment.

A. Safety policy model

The first step to model the safety policy consists in identifying the control modes, resources (with their failure modes) and alarms (with their failure modes) of the RPAS. These entities and their relations are captured by the diagram depicted in figure 2. More precisely, a RPAS (System) owns a set of control modes ($\mathcal{M}$), resources ($\mathcal{R}$) and alarms ($\mathcal{A}$).

Each control mode depends on a specific set of resources to be applicable, this set is described by the relation *necessary*. Each resource and alarm owns one or several failure modes given by the relation *fm*, modelling the considered ways resources and alarms fail. Each alarm provides information about a unique resource (*of* relation), however it depends on potentially several resources (*necessary* relation) to be triggered.

Example 3.1 (RPAS): The drone's mission is to inspect an infrastructure located in a pre-defined evolution zone nearby populated areas. The drone should not fly, land or crash outside the evolution zone. This RPAS is constituted of:

- 6 control modes named Autonomous (A), Semi-manual with Steering (P), Semi-manual with Guidance (G), Full Manual (M), Hovering (H) and Crash (C), i.e. $\mathcal{M} = \{A, P, G, M, H, C\}$.
- 8 resources named Steering (pi), Guidance (gu), Communication (rc), Propulsion (pr), Actuators (ac), Steering Law ($piLaw$), Guidance Law ($guLaw$), and Power Supply (pow), i.e. $\mathcal{R} = \{pi, gu, rc, pr, ac, piLaw, guLaw, pow\}$. Each resource $r \in \mathcal{R}$ has two failure modes, named Lost ($r.LS$), when the resource is completely lost, and Erroneous ($r.ES$) when the resource produces abnormal information or behaviour, i.e. $r.fm = \{r.LS, r.ES\}$.
- 5 alarms, respectively of the Steering ($alpi$), Guidance ($algu$), Communication ($alrc$), Propulsion

Mode	Resource							
	pi	gu	ac	pr	rc	pow	$piLaw$	$guLaw$
A	X	X	X	X	X	X	X	X
G		X	X	X	X	X		X
P	X		X	X	X	X	X	
M			X	X	X	X		
H	X	X	X	X		X	X	X
C								

Fig. 3: Resource dependencies of the control modes of the RPAS

($alpr$) and Actuator ($alac$) resources, i.e. $\mathcal{A} = \{alpi, algu, alrc, alpr, alac\}$. Each alarm $a \in \mathcal{A}$ has a failure false negative modes $a.FN$. All alarms depend on the Power Supply resource.

The resources required for each control mode are illustrated in figure 3.

To model the safety policy, one must provide:

1) the behavioural model of its system, here decomposed as the failure model and alarm model;
2) and the estimation and decision strategy of the policy, *i.e* the health status estimation strategy, and control mode selection strategy.

a) Failure model: This part of the model specifies assumptions over the failures modes of each resource: transient or permanent failures, exclusive or cumulative, *etc*. The assumptions about failure modes are expressed as hard constraints ($\Delta_{\mathcal{R}}$) that define the possible combinations of failures as well as the possible transitions between them. To do so, the occurrence of a failure mode f is modelled by a Boolean variable (also denoted f) true when the failure mode is observed on a given resource or alarm. Our model assumes that alarms and resources own a single nominal mode implicitly represented as the absence of failure modes. Therefore when a resource or alarm is in its nominal mode all its failure modes are set to false.

The analyst is free to use any pTLTL formula over failure mode variables to describe its failure model. Let us provide a set of recurrent patterns considered to model the RPAS.

Definition 3.1 (Permanent Failure): Once a resource r has entered a permanent failure mode $f \in r.fm$, then it cannot be repaired.

$$Y(f) \Rightarrow f$$

Definition 3.2 (Exclusive Failures): In a resource r with exclusive failure modes, at any time step t, only one failure mode is active:

$$\bigwedge_{\substack{f \in r.fm, \\ f' \in r.fm, \\ f \neq f'}} f \Rightarrow \overline{f}'$$

Definition 3.3 (Interleaved Failures): In a system with interleaved failures, at most one failure occurs at each time step. It is stated as:

$$\bigwedge_{\substack{r \in \mathcal{R}, r' \in \mathcal{R} \\ f \in r.fm, f' \in r'.fm \\ f \neq f'}} \big(Y(\overline{f}) \wedge f\big) \Rightarrow \big(Y(f') \Leftrightarrow f'\big)$$

Example 3.2 (Failure model): In our RPAS model, the failure modes for all resources and all alarms are exclusive, permanent, and interleaved. These constraints over the pi resource are:

Permanent: When pi was lost (resp. erroneous) then it remains lost (resp. erroneous)

$$(Y(pi.LS) \Rightarrow pi.LS) \wedge (Y(pi.ES) \Rightarrow pi.ES)$$

Exclusive: If pi is lost (resp. erroneous) then it cannot be erroneous (resp. lost)

$$(pi.LS \Rightarrow \overline{pi.ES}) \wedge (pi.ES \Rightarrow \overline{pi.LS})$$

b) Alarm model: This part of the model describes the nominal and dysfunctional behaviours of the alarms. The former specifies the conditions over the state of the monitored resource where the alarm should be triggered. The latter describes the impact of failures either in the alarm or in a resource that is necessary for the alarm *e.g.*the alarm no longer detects failures in the monitored resource. These rules are also encoded as hard constraints ($\Delta_{\mathcal{A}}$) that define the alarms behaviour in relation to the resource failures. Once again we consider that an alarm only provides a binary information *i.e* either the alarm is triggered or not. Therefore the Boolean variable a is true when the alarm a is triggered.

Once again, the analyst is free to use any pTLTL formula over failure mode and alarm variables to describe its alarm model. Let us provide a set of recurrent patterns considered to model the RPAS to model "active low" alarms with a "false negative" failure mode.

Definition 3.4 (Active low alarm with false negative): Let us consider an alarm a that owns a unique failure mode "false negative" denoted $a.fn$ that models a situation where the alarms always remains silent. Assuming the alarm a is "active low" then it is not triggered (denoted $\overline{a}$) if and only if the necessary and monitored resources are correct or it in failure mode fn, that is:

$$\overline{a} \Leftrightarrow \left(a.fn \vee \left(\bigwedge_{f \in a.of.fm} \overline{f} \wedge \bigwedge_{\substack{r \in a.necessary, \\ f' \in r.fm}} \overline{f'} \right) \right)$$

Example 3.3 (Alarm model): In our RPAS model, all alarms are active low with false negative. For $alpi$ the alarm model is:

$$\overline{alpi} \Leftrightarrow \big(alpi.fn \vee \big(\overline{pi.LS} \wedge \overline{pi.ES} \wedge \overline{pow.LS} \wedge \overline{pow.ES}\big)\big)$$

c) Health status estimation: The combination of hard constraints ($\Delta = \Delta_{\mathcal{R}} \wedge \Delta_{\mathcal{A}}$) describes how resources can fail, and how alarms are raised in consequence. However, only alarms are observable by the monitor, therefore the actual failure mode of the resources (i.e. the health status) must be deduced according to these signals and the knowledge of the

Alarm and Failure models. Recall that in our approach, we only keep one estimated health status at each time step, selected thanks to a preference model ($\Gamma_{\mathcal{R}}$). This preference model specifies which health status among those that are consistent with Δ to prefer, given the current alarms.

Here again, there are patterns that occur repeatedly in safety policy models. To illustrate them, let us describe the following typical estimation preferences that have been considered to model the safety policy of our RPAS.

Definition 3.5 (Preference of common causes): Let a resource r be necessary for several alarms $A_r = \{a \in \mathcal{A} | r \in a.necessary\}$, with $|A_r| \geq 2$. The common cause preference indicates that when multiple alarms are active, we prefer an estimation in which a unique failure explains all the alarms:

$$r.fm \leftsquigarrow \bigwedge_{a \in A_r} a$$

Definition 3.6 (Failure mode preference): It is often the case that, for a given alarm, several failure modes can explain it. It is then necessary to specify which one is the preferred explanation. For instance if an alarm a triggers off (was on, is now off), one may prefer to explain it with a false negative rather than the monitored resource being repaired:

$$a.fn \leftsquigarrow Y(a) \wedge \overline{a}$$

Note that preferences are soft equivalences. Once the alarm stays silent for two consecutive time steps, this preference will stop preferring the false negative explanation.

Definition 3.7 (Non monitored resource preference): When a resource r is not monitored and not necessary to any alarm then one must prefer to not consider the failure of this resource as the explanation of the actual state of the system. So let r be resource s.t. $\forall a \in \mathcal{A}, a.of \neq r \wedge r \notin a.necessary$, then for each $f \in r.fm$ we have:

$$f \leftsquigarrow \bot$$

Example 3.4: In our RPAS system, the health status estimation is based on the three previous preferences.

d) Control mode selection: The last part of the model describes the mechanism that selects a control mode. This mechanism may be expressed both with a set of hard constraints (denoted $\Delta_{\mathcal{M}}$) and preferences (denoted $\Gamma_{\mathcal{M}}$), since it is possible to select only one control mode at each time step.

The typical constraints that can apply to control modes include mode exclusivity (one mode selected at each time step) and applicability (a mode can only be selected when its necessary resources are estimated healthy) constraints.

Definition 3.8 (Mode applicability): A mode m can be selected (denoted by $m.select$) if its necessary resources are available, that is:

$$m.select \Rightarrow \bigwedge_{\substack{r \in m.necessary \\ f \in r.fm}} \overline{f}$$

Definition 3.9 (Exclusive modes): At each time step, exactly one mode must be selected:

$$\bigvee_{m \in \mathcal{M}} m.select, \qquad \bigwedge_{\substack{m \in \mathcal{M} \\ m' \in \mathcal{M} \\ m \neq m'}} m.select \Rightarrow \overline{m'.select}$$

In the RPAS use-case, the pilot is able to select a subset of modes so-called *manual modes* (denoted $\mathcal{M}_p = \{P, G, M\}$). Thus, the mode selection activity is distributed over the UAV and the pilot. To model this distribution, we consider for each pilot mode $m \in \mathcal{M}_p$ a Boolean variable $m.select_p$ meaning that the pilot selects the mode m. Since the modelling framework is based on constraint programming, the analyst can consider or not a set of assumptions over the pilot selection strategy. In the latter, the pilot selection variables are *free* variable *i.e* any possible assignments of these variables are considered in the subsequent analysis.

Since the pilot and the UAV may disagree on the mode selection, a strategy must be modelled. In the use-case, the priority is given to the pilot only if the selected mode is considered as safe by the UAV *i.e* the resources it rely on are available. This kind of selection strategy is formally provided by the following definition.

Definition 3.10 (Conditional pilot mode selection): Let $m \in \mathcal{M}_p$, then

$$\left(m.select_p \wedge \bigwedge_{r \in m.necessary} \overline{r.ES} \wedge \overline{r.LS} \right) \Rightarrow m.select$$

In the general case, several control modes may be applicable, and one must be chosen. A simple way to do so is to use an unconditional preference model.

Definition 3.11 (Order based policy): A simple way to select a mode is to consider a total, unconditional preference order over the control modes, for instance based on the automation level (more autonomous modes are preferred to less autonomous modes). Let $(m_1, \cdots, m_n)_{i \in 1, |\mathcal{M}|}$ be a sequence of exclusive control modes sorted by automation level, then the policy "Select m_1 if possible, otherwise select m_2 if possible, etc" is encoded as:

$$\Gamma_{\mathcal{M}} = \begin{pmatrix} m_1.select \leftsquigarrow \top \\ \vdots \\ m_n.select \leftsquigarrow \top \end{pmatrix}$$

It is important to note that, even though in example 3.11, the preferences do not explicitly depend on the health status, the preferences about failure mode variables are applied *before* the preferences about the control mode. As a consequence, the way failure modes are estimated can change which control modes are applicable, and thus weigh on the final choice of control mode.

Example 3.5: In our RPAS example, the control modes are exclusive, subject to the applicability constraint, and selected via an order based policy.

B. Evaluation of the modelling approach

The presented modelling approach gathers the constraints and preference to the activities identified in the figure 1. The failure model describes the dysfunctional behaviour of the system, and how failures affect the different functions of the system. It is tightly linked to the functional architecture. The alarm model identify the monitoring of the system and provides the assumption over failure detection capability of

these monitoring systems and there dependencies. Going back to our RPAS example, we detailed what are the resources (*e.g.*pi), the monitored ones (*e.g.*pi monitored by $alpi$) and the monitoring dependencies (*e.g.*$alpi$ depends on pow). The health status estimation can implement a very pessimistic strategy, or a very optimistic one, or a conditional one. It may depend on function-specific or alarm-specific variables, or may be very independent from the other model components. Finally, the control mode selection finalises the safety policy, and translates the health status into an actual control mode. Such a *modular* modelling enables the analyst to update only one part of the model with very limited impact on the other ones, which facilitates the reuse of model components from one mission to the other.

With the use of a temporal language, the *dynamic* aspects of the safety policy are easily represented and analysed.

As explained in [15], the estimation model selects a unique state in a *deterministic* way. This guarantees that the safety policy behaviour is reproducible for tests and analyses.

The disagreement solving between the pilot and the UAV is explicitly modelled in the framework, enabling the analyst to assess the safety impact of the selection strategy. As a means to represent a transition relation, conditional preferences are more concise than pure hard constraints. As such, our model is *expressive* enough to implement complex decision functions. At the same time, since the preferences are applied sequentially, it is quite straightforward to *explain* a given estimation, by stating the previous estimated state, and which preferences were applied or violated. The number of preferences stays small when compared to the potentially exponential number of explanations for an observation.

C. Safety objectives verification

We express safety objectives under questions of the form "Can a dynamic system reach a state that satisfies property X ?", where the dynamic system is the RPAS including its safety policy. These questions can be automatically verified using the safety policy model described in the previous section.

In an estimation problem, the observable variables receive exactly one value at each time step. Inversely, the estimated variables are associated to two values: their real value in the system, and their estimated value in the safety policy. When one models a safety policy as an estimation problem, whenever several failure modes are consistent with the observations, the safety policy chooses one failure mode among the candidates. This means that there are scenarios where the safety policy estimates a failure mode that is different from that of the system. In our model, this is expressed by the fact that the real value of the failure mode variables differs from the one that is estimated.

Example 3.6 (Fly-Away): A typical desired property for a RPAS would be to avoid the *Fly-Away* since it lead to a potential collision with on-ground or in-air obstacles. Such a feared event occurs when:

- the dependencies of the selected mode are not satisfied;
- and the propulsion is still available (otherwise it would result in a crash in the evolution zone);
- and the selected mode is not C.

One must then identify the situations where this event occurs, therefore the property would be "It exists an observation sequence where a fly-away occurs". Such informal property can be decomposed over each mode $m \in \mathcal{M}$ (except C) as the following formula:

$$\begin{array}{rcl} \phi_R & = & \bigvee_{r \in m.necessary} r.ES \vee r.LS \wedge \overline{pr.LS} \\ \phi_E & = & m.select \end{array}$$

A fly-away thus occurs if, for a given mode m, the above formula are satisfied.

Our analysis approach takes advantage of this situation to represent some safety objectives. Note that the evolution of the system's state depends only on the occurrence of failures. So one may not be interested to ensure that a safety policy fulfils the properties for an arbitrary large number of failures, since this policy may not likely to exist. A more relevant analysis would be to ensure that the safety policy is "reasonably safe" *i.e* ensures a set of properties up to a given number of failures. This analysis can be performed thanks to a bounded reachability analysis defined as follows.

Definition 3.12 (Bounded reachability analysis): Let Δ, Γ be an estimation model, ϕ_R and ϕ_E two sentences in PTLTL. The reachability analysis denoted $\text{REACHABLE}_{\Delta,\Gamma}(\phi_R,\phi_E,n)$ enumerates, up to the time step n , pairs (S_R, S_E) where:

- S_R and S_E have length n;
- at each time step, S_R satisfies Δ and S_E satisfies both Δ and Γ;
- at the last time step, S_R satisfies ϕ_R and S_E satisfies ϕ_E.

S_R represents the real system states, while S_E represents the estimated policy states.

Example 3.7 (Bounded reachability): Back to the example 2.3, one may want to identify the sequences of states where "The power supply failure is not detected after one time step". This analysis can be performed by using the bounded reachability by decomposing the property as "The power system failure" $Y(\overline{h_{pow}})$ and "Is not detected after one time step"h_{pow}. The sequences leading to this step, up to three time step would be obtained by calling $\text{REACHABLE}_{\Delta,\Gamma}(Y(\overline{h_{pow}}),h_{pow},3)$. Such a request would provide the an example of the table I where:

1) a failure of h_2 is (correctly) estimated at s_1;
2) according to Δ, the subsequent alarm a_1 can be interpreted either as a failure of h_1 or a failure of h_{pow} but according to the preference γ_1, the failure of h_1 is preferred over the failure of h_{pow} leading to a mis-estimation of the system state;
3) ultimately, the inability to estimate the failure of h_{pow} after one time-step is satisfied on s_3.

The reachability analysis has been performed on the RPAS to identify the situations for which the Fly-Away feared event (introduced in the example 3.6) occurs. For the sake of readability, the results of the reachability analysis provided in the table II has been processed to display the single new

Step	Observed variables	Estimated variables	
		System	Estimator
s_0	$\overline{a_1 a_2}$	$h_{pow} h_1 h_2$	
s_1	$\overline{a_1} a_2$	$h_{pow} h_1 \overline{h_2}$	
s_2	$a_1 a_2$	$\overline{h_{pow}} h_1 \overline{h_2}$	$h_{pow} \overline{h_1 h_2}$
s_3	$a_1 a_2$	$\overline{h_{pow}} h_1 \overline{h_2}$	$h_{pow} \overline{h_1 h_2}$

TABLE I: Reachability analysis result

Order	Failures		Comments
1	$piLaw.LS$ $piLaw.ES$		Undetectable steering control failure
	$guLaw.LS$ $guLaw.ES$		Undetectable guidance control failure
2	$a_{pi}.FN$ $a_{pi}.FN$	$pi.LS$ $pi.ES$	Steering sensors failure and monitoring false negative
	$a_{gu}.FN$ $a_{gu}.FN$	$gu.LS$ $gu.ES$	Guidance sensors failure and monitoring false negative
	$a_{rc}.FN$ $a_{rc}.FN$	$rc.LS$ $rc.ES$	Communication failure and monitoring false negative
	$a_{ac}.FN$ $a_{ac}.FN$	$ac.LS$ $ac.ES$	Actuator failure and monitoring false negative
	$a_{pr}.FN$	$pr.ES$	Erroneous propulsion and monitoring false negative

TABLE II: Safety assessment of the RPAS for the Fly-Away

failure occurring at each time step. Note that the uniqueness is enforced by the interleaving and permanent failure constraints.

The safety assessment identified two single contributors to the Fly-Away: the control laws. Since these resources are not monitored, a failure of one of these control laws is not detected by any alarm, therefore neither the pilot or the UAV initiates a mode change. The second set of scenarios involves a typical combination of alarm failure (false negative) and resource failure. Once again this situation boils down to the previous one *i.e* an undetectable failure. Note that the propulsion and the power supply loss do not contribute to the fly-away since it would result in a crash of the drone in the evolution zone.

The designer of the safety policy can use these information to enhance the policy for instance by adding monitoring points. Since the framework is modular, extra assumptions can be made by adapting the alarm model, without changing the remaining model.

IV. Related-work

1) Analysis of a safety policy: The integration of the safety policy in the safety assessment can be related to the formal verification of Fault Detection, Isolation and Recovery (FDIR) mechanisms. For instance the authors of [17] used several verification techniques to prove that a set of FDIR mechanisms meets some functional requirements. Another work [18] uses finite state machine to model the system and FDIR behaviours while failure propagation were modelled by Temporal Fault Propagation Graphs (TFPG). A dedicated model-checker is then used to assess some functional properties of the FDIR. Eventually [19] provides a comprehensive formal framework to either assess or even synthesise a safety monitoring. Nevertheless, the aforementioned papers do not consider the limited system's state observability, therefore do not consider the safety impact of this source of uncertainty. Indeed the particularity of our framework is to provide a preference-based modelling of the strategy (formalised in Γ) considered by the designer to select a given state when several system's states can explain the current observations.

In [20], the property of fault diagnosability is stated and analysed with respect to a set of conditions that can represent safety requirements. However the authors of [20] do not address one of the main aspects of autonomous systems like RPAS, identified by the authors of [21], that is the collaboration of human and technical elements to ensure the safety of the whole system. Especially considering the monitoring capability of each agent, their mitigation strategy and the arbitration in case of disagreement on the safety action to perform. In the presented framework, this distribution problematic is a core challenge addressed both by the modelling of the actor monitoring capabilities as hard constraints and their mitigation strategy as a preference model.

2) Diagnosis of discrete event systems: The problem of assessing the presence of faults in a system is known as Model-Based Diagnosis, and has been addressed in various ways in the literature, in particular for discrete event systems [22]. However, the question of which is the preferred diagnosis is usually reduced to a form of likelihood, independent from the operational context. [15] provides a way to specify a diagnosis selection strategy that accounts for the operational context, such as alarms and hazards, and the distribution of the safety policy.

3) Control of discrete event systems: The problem of selecting the control mode of the system could also be represented as a problem of control of a discrete-event system [23]. While this allows one to represent the safety effects of the dynamic control model selection, this requires to accurately represent the actions from the pilot in order to derive meaningful analyses. This may be done in future work. Controller synthesis techniques [24] could be applied to ease the description of safety policies and even improve them.

V. Conclusion

a) Summary: We presented in this paper a formal modelling framework dedicated to the safety assessment of RPAS safety policies. This framework enables the analyst to provide, in a modular way, the failure and alarm models formalising both the considered failures of the RPAS, the possible control modes and monitoring (and their own dependencies) used to handle these failures. A light formalisation of the distributed strategy considered to select a control mode (by the pilot and the UAV) and to select a mode in case of disagreement has been provided and illustrated on the use-case. Eventually we provided a way to automatically perform some qualitative safety assessments of the safety policy, especially to identify single point of failures.

b) Limitations and future works: The framework is currently built to deal with coarse grain fault models based on abstract failure modes (loss or erroneous). The analyst may need to express more precisely the impact of resource failures for instance in a quantitative way (*e.g.*propulsion blocked to a given value). To alleviate this expressiveness restriction, one

can consider using Satisfiability Modulo Theory (SMT) solvers to handle constraints and properties expressed over various kinds of objects *e.g.*reals values. Moreover the presented modelling of the pilot behaviour is simplistic and does not exploit the expressiveness of the framework. We are currently working on new modelling integrating the capability of the pilot to detect inconsistencies between downloaded data in absence of alarms, and conversely its ability to detect false positive. Concerning automatic assessment, the current readability analysis is based on iterative calls to a SAT solver to enumerate the scenarios without considering minimality constraints. Such an enumeration is a major bottleneck to provide a scalable analyser. Considering some minimality criteria (typically based on the inclusion) can drastically enhance the efficiency of the solver.

REFERENCES

[1] Airbus, February 2018. [Online]. Available: https://www.airbus.com/newsroom/press-releases/en/2018/02/vahana–the-self-piloted–evtol-aircraft-from-a–by-airbus–succ.html

[2] F. Flammini, C. Pragliola, and G. Smarra, "Railway infrastructure monitoring by drones," in *2016 International Conference on Electrical Systems for Aircraft, Railway, Ship Propulsion and Road Vehicles & International Transportation Electrification Conference (ESARS-ITEC)*. IEEE, 2016, pp. 1–6.

[3] K. J. Hayhurst, J. M. Maddalon, P. S. Miner, G. N. Szatkowski, M. L. Ulrey, M. P. DeWalt, and C. R. Spitzer, "Preliminary considerations for classifying hazards of unmanned aircraft systems," 2007.

[4] SAE, "Aerospace Recommended Practices 4754a - Development of Civil Aircraft and Systems," 2010.

[5] A. Legendre, A. Lanusse, and A. Rauzy, "Toward model synchronization between safety analysis and system architecture design in industrial contexts," in *Model-Based Safety and Assessment - 5th International Symposium, IMBSA 2017, Trento, Italy, September 11-13, 2017, Proceedings*, 2017, pp. 35–49. [Online]. Available: https://doi.org/10.1007/978-3-319-64119-5_3

[6] SAE, "Aerospace Recommended Practices 4761 - guidelines and methods for conducting the safety assessment process on civil airborne systems and equipment," 1996.

[7] T. Prosvirnova, "Altarica 3.0: a model-based approach for safety analyses," Ph.D. dissertation, Ecole Polytechnique, 2014.

[8] A. Arnold, G. Point, A. Griffault, and A. Rauzy, "The altarica formalism for describing concurrent systems," *Fundamanta Informaticae*, vol. 40, no. 2-3, pp. 109–124, 1999.

[9] Y. Papadopoulos and J. A. McDermid, "Hierarchically performed hazard origin and propagation studies," in *Computer Safety, Reliability and Security*. Springer, 1999, pp. 139–152.

[10] A. Rauzy, "Mathematical foundations of minimal cutsets," *Reliability, IEEE Transactions on*, vol. 50, no. 4, pp. 389–396, 2001.

[11] B. Bittner, M. Bozzano, R. Cavada, A. Cimatti, M. Gario, A. Griggio, C. Mattarei, A. Micheli, and G. Zampedri, "The xsap safety analysis platform," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2016, pp. 533–539.

[12] F. A. Administration, "Advisory circular ac 23.1309-1c: Equipment, systems, and installations in part 23 airplanes," Federal Aviation Administration, Tech. Rep., March 1999.

[13] Q. Gaudel, E. Chanthery, P. Ribot, and M. J. Daigle, *Diagnosis of Hybrid Systems Using Hybrid Particle Petri Nets: Theory and Application on a Planetary Rover*. Cham: Springer International Publishing, 2018, pp. 209–241.

[14] N. Wilson, "Computational techniques for a simple theory of conditional preferences," *Artificial Intelligence*, vol. 175, no. 7-8, pp. 1053–1091, 2011.

[15] C. Pralet, X. Pucel, and S. Roussel, "Diagnosis of intermittent faults with conditional preferences," in *Proceedings of the 27th International Workshop on Principles of Diagnosis (DX'16)*, 2016.

[16] X. Pucel and S. Roussel, "Intermittent Fault Diagnosis as Discrete Signal Estimation: Trackability analysis," in *DX 2017*, BRESCIA, Italy, Sep. 2016. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02003771

[17] E. Bensana, X. Pucel, and C. Seguin, "Improving fdir of spacecraft systems with advanced tools and concepts," *Proc. ERTS*, 2014.

[18] B. Bittner, M. Bozzano, A. Cimatti, R. De Ferluc, M. Gario, A. Guiotto, and Y. Yushtein, "An integrated process for fdir design in aerospace," in *International Symposium on Model-Based Safety and Assessmemt*. Springer, 2014, pp. 82–95.

[19] M. Machin, J. Guiochet, H. Waeselynck, J.-P. Blanquart, M. Roy, and L. Masson, "Smof: A safety monitoring framework for autonomous systems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 5, pp. 702–715, 2016.

[20] A. Paoli and S. Lafortune, "Safe diagnosability for fault-tolerant supervision of discrete-event systems," *Automatica*, vol. 41, no. 8, pp. 1335–1347, 2005.

[21] E. E. Alves, D. Bhatt, B. Hall, K. Driscoll, A. Murugesan, and J. Rushby, "Considerations in assuring safety of increasingly autonomous systems," 2018.

[22] J. Zaytoon and S. Lafortune, "Overview of fault diagnosis methods for discrete event systems," *Annual Reviews in Control*, vol. 37, no. 2, pp. 308–320, 2013.

[23] W. Wonham, K. Cai, and K. Rudie, "Supervisory control of discrete-event systems: A brief history–1980-2015," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 1791–1797, 2017.

[24] K. Chatterjee, L. Doyen, T. A. Henzinger, and J.-F. Raskin, "Algorithms for omega-regular games with imperfect information," in *International Workshop on Computer Science Logic*. Springer, 2006, pp. 287–302.