Model Based Safety Assessment of Concept of Operations for Drones

Evaluation de sécurité fondée sur les modèles de scénarios d’opérations de Drones

Pierre Bieber, Christel Seguin, Vincent Louis, Florian Many
ONERA DGA TA
Toulouse Balma
pierre.bieber@onera.fr vincent1.louis@intradef.gouv.fr
christel.seguin@onera.fr florian.many@intradef.gouv.fr

Résumé
Nous proposons d’utiliser l’évaluation de sécurité fondée sur les modèles afin d’assister l’analyse des risques des scénarios d’utilisation de drones. Cette analyse de risque permet d’allouer des exigences de sécurité aux acteurs et aux systèmes impliqués dans l’usage des drones. L’approche a été appliquée pour analyser plusieurs scénarios définis par le projet MIDCAS qui vise à gérer les risques de collision entre un drone et d’autres aéronefs dans l’espace aérien.

Summary
We propose to use model-based safety assessment in order to support the risk analysis of concept of operations for drones. This risk analysis helps to allocate safety requirements to all actors and systems involved in the drone operations. The approach was applied in order to analyse several scenarios defined by the MIDCAS project in order to manage collision risk between a drone and other aircraft in the airspace.

Context

1 CONOPS for Drones
A Concept of Operations (CONOPS) defines a set of usage scenarios that describes the roles of all actors involved in an operation. The European Aviation Safety Agency (EASA) has recently published a document about its regulation policy for drones [1]. In this document, risk assessment of CONOPS for drones is presented by EASA as mandatory in order to regulate drone usage in a safe and proportionate manner. Three categories of operations and their associated regulatory regime have been proposed in [1]:
- Open operations: risk-limited operations that do not require an authorisation by an authority,
- Specific operations: operation authorisations are issued by an authority with specific limitations adapted to the operation risks,
- Certified operations: the classical certification process (e.g. a certification process similar to the one applied to transport aircraft) for operations is used.

A challenge is to provide methods and tools to support efficiently the risk analysis of CONOPS in the “Specific operations” category. In this paper we propose an approach that assists the risk assessment of such operations based on the use of formal safety models of the operation scenarios. In the following section we first provide an illustrative case study from the MIDCAS (Mid-air Collision Avoidance System) project [2]. In particular we present an example of a usage scenario that should be analysed. In the “Method” section, we introduce the safety modelling framework. We first describe the basic components used to model scenarios and then we present the global model describing a MIDCAS scenario. In the “Result” section we describe the main results of this study. Finally the “Conclusion” section contains a description of related works and a list of possible further works.

2 Collision Avoidance Case-Study
The proposed approach has been applied to analyse several scenarios defined by the MIDCAS project to manage collision risk in the controlled airspace.

Figure 1. Separation, Loss of separation and Near mid-air collision

The previous figure shows graphically the main safety concerns related with these scenarios. The left picture shows a safe situation where the drone and another aircraft (called the intruder) are separated. Two aircraft are separated when their mutual horizontal distance and mutual vertical distance are greater than a given separation distance. The separation distances vary according to flight phases (take-off, approach, in cruise, ...). Typical distances for horizontal separation could range from 3 to 20 nautical miles, typical vertical separation distances range from 1000 to 2000 feet. The middle picture shows an unsafe situation called loss of separation where either vertical or horizontal separation distances are infringed. In that situation, aircraft are in the orange zone but they are not in the smaller red zone called near mid-air collision zone. The MIDCAS project considered that aircraft are in the near mid-air collision zone when their horizontal distance is smaller than 0.5 nautical miles or their vertical
distance is smaller than 500 feet. The right figure shows the unsafe situation where both aircraft are in the Near Mid-Air collision zone.

Loss of separation is considered to be less severe than near mid-air collision because there should be sufficient time for the Air Traffic Control to detect the situation and to propose to both aircraft a separation manoeuvre: for instance one aircraft decreases its altitude and the other aircraft keeps its original flight path. In the case of near mid-air collision, an embedded system such as the TCAS [3] for transport Aircraft or the pilot shall trigger a predefined anti-collision manoeuvre.

Actors involved in the MIDCAS CONOPS are:

- The Sense And Avoid System (SAAS) that is embedded in the drone. Activities performed by this actor are found in the Blue area of the following figure;
- Pilot in Command of the drone (PIC). Activities performed by this actor are found in the Yellow area;
- Air Traffic Control (ATC). Activities performed by this actor are found in the Green area;
- Pilot in command of the Intruder aircraft (IP). Activities performed by this actor are found in the Pink area.

Activities of PIC, ATC and IP represent human tasks performed by either the Pilot of the drone, the pilot of the intruder or Air Traffic Controllers. They also represent technical activities necessary to perform the human tasks such message exchange, aircraft position computation and visualisation, separation and anti-collision trajectory computation.

The CONOPS scenarios for a generic collision avoidance system are described by a set of diagrams. Each scenario details the sequence of operation steps of each actor in a given situation depending, for instance, of the type of equipment of the intruder aircraft. The following figure shows a collision avoidance scenario when the drone and the intruder are flying in airspace controlled by Air Traffic Controllers.

A basic activity called a task is graphically described by a blue rectangle. Two activities linked with an arrow means that when the first activity has completed its work the second activity is started. The first activity that starts the scenario is depicted by a green circle, while the last activity that ends the scenario is depicted by a red circle. A yellow diamond-shaped box is used to represent conditional activation. The box is labelled with a question and the outgoing arrow is labelled with an answer to the question.

![Collision Avoidance Scenario in Controlled Airspace](image-url)

According to this scenario, the role of the SAAS embedded system is first to provide information about the drone position and other aircraft to its pilot (SAAS _01), to assess whether there is a possible loss of separation (SAAS _02). If there is a loss of separation then the embedded system should report the conflict to the ATC (SAAS _03). The embedded system should assess whether the drone enters in a near collision zone (SAAS _04). If it is the case then it should trigger the anti-collision manoeuvre (SAAS _05).

The role of the pilot in command of the drone is first to observe the position of the drone and of possible intruders (PIC _01), to assess whether there is a possible loss of separation (PIC _02). If there is a loss of separation then the pilot should report the conflict to the ATC (PIC _03). When the ATC has issued separation manoeuvre instructions, the pilot should acknowledge reception of the ATC instructions (PIC _04) and command the drone to perform the separation manoeuvre (PIC _05).

The role of the ATC actor is to observe the position of aircraft in the controlled area (ATC _01), to receive information about loss of separation from the pilots (ATC _04), to prepare separation manoeuvres (ATC _02). If there is a loss of separation then the ATC should send its instructions to the drone and intruder pilots (ATC _03).

The role of the intruder pilot is to acknowledge reception of the ATC instructions (IP _01) and command the intruder aircraft to perform the separation manoeuvre (IP _02).
3 Library of AltaRica Components

Model based safety assessment approaches have been developed among other things to strengthen the links between system specifications and safety models and to automatically find out combinations of faults leading to undesired events for complex systems. AltaRica formal models have been successfully applied to assess safety of aircraft systems [4]. We propose to use the same approach in order to analyse operation scenarios such as the one that was presented in the previous section.

We developed a library of basic components that can be used to describe formally the propagation of faults within an operational scenario. In this section, we will focus on the model of the Task component. We wanted to model the causal dependency between connected tasks where the performance of one task influences the performance of following tasks. So we have defined an AltaRica node called Task with one input variable named perfo_in and one output variable named perfo_out to model respectively the performance of preceding tasks and the performance transmitted to following tasks once the current task is finished. The value of the output variable describes the performance of a task (its level of success):

- the value is equal to “ok” when the task is successfully completed,
- the value is equal to “lta” (e.g. less than adequate) when the task partially fails (for instance the task finished late or provided an imprecise result),
- the value is equal to “null” when the task totally fails (the task was not performed at all or it finished in an incorrect state).

The performance of a task depends not only on the performance of preceding task but also of the internal performance status of the task. When the internal status of the task is ok, the output performance is equal to the input performance; otherwise when the internal status is null or the input performance is null then the output performance is equal to null; otherwise the internal status is less than adequate and the output performance is less than adequate. In the AltaRica model, the state variable perfo_status represents the internal status of a task. We suppose that initially the internal status of a Task is ok.

The internal performance status can be partially degraded or totally lost after events partial_loss and total_loss.

```altrica
node Task
flow
  perfo_in : (ok, lta, null) : in ;
  perfo_out : (ok, lta, null) : out ;
state
  perfo_status : {ok, lta, null} ;
event
  partial_loss, total_loss ;
init
  perfo_status := ok ;
trans
  perfo_status = ok |- partial_loss -> perfo_status:= lta ;
  perfo_status = ok |- total_loss -> perfo_status:= null ;
assert
  perfo_out = case{
    perfo_status = ok : perfo_in ,
    (perfo_status = null) or (perfo_in = null) : null,
    else lta} ;
edon
```

Figure 3. AltaRica code of the Task component, and its graphical representation

The behavior of a Task is formally described in the AltaRica code presented in the previous figure (see references [4] or [5] for an introduction to the syntax and semantics of the AltaRica language). The declaration part of node Task introduces the input and output variables (perfo_in and perfo_out), the state variable (perfo_status) and the events (partial_loss and total_loss). The initialization part sets the state variable to its initial value. The transition part associates each failure event with a transition that can be triggered whenever the task can be achieved correctly in the current state (perfo_status = ok) then the state variable is changed according to the type of fault (lta for partial loss, and null for total loss). The assertion part gives an equation defining the value of the task output variable according to the value of the state variable and input variable in the current state.

Several colored icons are used to show graphically the current state of a task. When the internal status of the task is ok then the icon is green if the task output is ok, it is orange if the output is lta (this happens when the task provides a less than adequate result due to a fault in one of the preceding tasks) and red if the output is null. When the internal status of the task is equal to lta then the icon is orange with a black cross, and when the internal status of the task is null then the icon is red with a black cross.

The task model we have just presented does not model accurately the temporal aspects of a scenario because all tasks in a sequence are activated simultaneously. We have developed another library that models tasks that can be activated or deactivated according to their position in a scenario. We have decided not to use this more accurate model of tasks because the failure conditions of interest (loss of separation, near mid-air collision) do not require a precise temporal model. At this stage of the development of the MIDCAS system, we were not interested in knowing the exact chronology of task faults leading to these situations. Furthermore, analysis tools are less efficient with the accurate models of tasks because they include more events (partial_loss, total_loss events and activation, deactivation events) than the simpler model of tasks (partial_loss and total_loss events).
node TaskWithResource
flow
  perfo_in, resource : \{ok, lta, null\} : in;
  perfo_out : \{ok, lta, null\} : out;
state
  perfo_status : \{ok, lta, null\};
event
  partial_loss, total_loss ;
  init
    perfo_status := ok;
  trans
    perfo_status = ok \land partial_loss -> perfo_status := lta;
    perfo_status = ok \land total_loss -> perfo_status := null;
  assert
    perfo_out = case(
      (perfo_status = ok) \land (resource = ok) : perfo_in,
      (perfo_status = null) or (perfo_in = null)
    or (resource = null) : null,
    else lta;
  endon

Figure 4. AltaRica code of the Task component with resource and its graphical representation

Tasks were described at two levels of details. The task model we have just presented does not take into account the effect of equipment faults on the operation. The operation model based on this component is used to analyze the robustness of the CONOPS scenarios. At a lower level of abstraction, the task model includes details about equipment necessary to perform the operation. A new input called resource is added to the task component. We use the three possible values \{ok, lta and null\} to indicate whether the needed resource is working correctly or it is partially lost or it is totally lost. The assertions are modified in order to take into account the influence of the resource on the output of the task. When both the internal status of the task and the needed resource are ok, the output performance is equal to the input performance; otherwise when the internal status or the input performance or the needed resource is null then the output performance is equal to null; otherwise the output performance is equal to lta. The operation model using this task component has to be connected with a model of the resources. This operation model can be used to perform detailed safety analysis, such as common cause analyses that looks at the effects on several operations of shared equipment failures.

The library contains models for other components that are used to coordinate tasks. We will not describe in details these components because they are rather simple. We consider that a fault cannot originate from these components, these components do not include fault events and they do not contain a state variable. These components only contain flow variables, in the following we briefly explain how output values are computed for these components:

- start task: this component has one output that is always equal to ok;
- stop task: this component has one input that has to be connected with the preceding task and one output that is always equal to its input;
- merge: this component has two inputs connected with two preceding tasks, it has one output that is equal to the better performance value of the preceding tasks. We consider that ok is better than lta that is better than null. When one input is equal to lta and the other is equal to null then the output is equal to the best input performance e.g. lta. Similarly, if one input is equal to null or lta and the other input is equal to ok then the output will be equal to ok.
- conditional activation: this component has one input that has to be connected with the preceding task and one Boolean input that indicates whether the condition is true. The component has two outputs, if the condition is true the performance of the preceding task is forwarded on the first output and the value of the second output is null otherwise the performance of the preceding task is forwarded on the second output and the value of the first output is null.

The basic model in the following figure shows the various components of the AltaRica library that we have developed. The figure also illustrates the use of icons to quickly grasp the current state of a model. Component start is a start Task, its icon is a green circle meaning that this task is performed correctly. Components Task1a, Task1b, Task2a and Task2b are tasks. Task2a is performed correctly (green box icon), Task1a is partially lost (orange box with a black cross icon), Task2b is totally lost (red box with black cross icon), Task1b could be performed correctly but it performs less than adequately because it depends on Task1a that is performed less than adequately. The merge component produces lta output as it receives null input from Task2b and lta input from Task1b. CondActivation is a conditional activation component. As the Condition input is false the input from the merge component is forwarded towards the stop task called stop_no and null is forwarded to the stop task called stop_yes. Consequently stop_no provides a lta output whereas stop_yes provides a null output.

Figure 5. Basic Model
4 Operation Safety Model

For each CONOPS scenario such as the one described in figure 2 we build an operation safety model that uses components in the AltaRica library we have presented in the previous section. The operation safety model aims at having a graphical view that is as similar as possible to the graphical view of the CONOPS scenario. Due to limitations in the possible names of AltaRica component instances, we use short names such as `SAAS_02` instead of the longer names used in the scenario such as “SAAS_02 Assess UAV separation from other traffic”.

Each element (a component or a link between components) found in the CONOPS scenario has a counterpart in the operation model. Components are represented by either a basic task, a start task, a stop task or a conditional activation. The model contains two merge components that do not appear in the CONOPS scenario. Component `m_PIC` (resp. `m_ATC`) is used to formalize how the preceding tasks of `PIC_02` (resp. `ATC_02`) influence it. So the inputs of `m_PIC` are connected with the output of task `SAAS_03` and the output of task `PIC_01`. Similarly, the inputs of `m_ATC` are connected with the output of task `ATC_04` and the output of task `ATC_01`.

We also use several stop tasks (`SAAS_stop`, `PIC_stop`, `ATC_stop` and `IP_stop`) instead of only one stop task as in the original CONOPS scenario because we wanted to precisely illustrate how the sequence of tasks of each actor is finished.

![Operation Safety Model for Collision Avoidance Scenario](image)

Figure 5. Operation Safety Model for Collision Avoidance Scenario

Conditional activation components are named according to the condition being tested. Components `SAAS_sep`, `PIC_sep` and `ATC_sep` test whether a loss of separation occurs, component `SAAS_CA` tests whether a near mid-air collision occurs.

5 Operational Safety Assessment

We want to use the operation safety model in order to produce automatically safety assessments. Before performing the assessment we had to add new components.

Component `InitSit` gives the initial situation of the model to be assessed. In the following figure, the initial situation is a loss of separation. The `InitSit` component provides two Boolean output flows: one that sets the loss of separation condition used by `SAAS_sep`, `PIC_sep` and `ATC_sep` to true and the other flow sets the near-collision condition to false.

The model also includes observer components. These components are used as model “probes”. They enable to analyse the occurrence of some failure conditions. For instance, component `PIC_IP_Obs` is an observer that tests how both pilots (remote pilot in command of the UAV and pilot of the intruder aircraft) have finished their task. Its behavior is similar to a merge component. This observer can be used to analyse the situations where there is a total (resp. partial) failure to manage properly a loss of separation by the pilots in that case the output of the observer node is equal to `null` (resp. `lta`).

Results
The safety assessment aims at:

- identifying the effect of task faults on the overall operation, this can be done using an interactive simulator that runs faults and shows graphically the effect on the faults on the operation model components. For instance the previous figure shows a situation where tasks PIC_02 “Assess situation” and ATC_01 “Detects potential conflict situation with the intruder” are totally lost. This leads to the inability for the drone pilot and the intruder pilot to properly manage a loss of separation. The icons for tasks PIC_02 and ATC_01 are red with a black cross, this means that they are totally lost and they produce a null output that is merged in m_ATC and propagated by the following tasks down to the observer PIC_IP_Obs.

- computing the minimal combinations of faults leading to an undesired event, this was done using the Cecilia sequence generator. Table 1 gives the set of minimal combinations of faults leading to a null output of observer node PIC_IP_Obs.

- computing the probability of occurrence of failure condition, this can be computed using the set of minimal combinations of faults and the probability of occurrence of each fault appearing in this set.

<table>
<thead>
<tr>
<th>#</th>
<th>1st fault</th>
<th>2nd fault</th>
<th>3rd fault</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ATC_02.total_loss</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>ATC_03.total_loss</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>ATC_01.total_loss</td>
<td>ATC_04.total_loss</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>ATC_01.total_loss</td>
<td>PIC_02.total_loss</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>ATC_01.total_loss</td>
<td>PIC_03.total_loss</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>ATC_01.total_loss</td>
<td>SAAS_01.total_loss</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>IP_01.total_loss</td>
<td>PIC_04.total_loss</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>IP_01.total_loss</td>
<td>PIC_05.total_loss</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>IP_02.total_loss</td>
<td>PIC_04.total_loss</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>IP_02.total_loss</td>
<td>PIC_05.total_loss</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>ATC_01.total_loss</td>
<td>PIC_01.total_loss</td>
<td>SAAS_02.total_loss</td>
</tr>
<tr>
<td>12</td>
<td>ATC_01.total_loss</td>
<td>PIC_01.total_loss</td>
<td>SAAS_03.total_loss</td>
</tr>
</tbody>
</table>

Table 1. Minimal combination of faults leading the total loss of ability to manage a loss of separation

The two first steps of the safety assessment provide a qualitative feedback which is mandatory before doing the probability quantification. If an unacceptable fault combination leads to an unsafe situation, the CONOPS could have to be redesigned. If we analyze the minimal cut sets leading to a null output of observer node PIC_IP_Obs we find single faults (line #1 and #2 in Table 1). For instance, the loss of ATC_02 “Generates separation solution” leads to this situation. The reason is that ATC_02 is the only activity that computes a separation trajectory, consequently when this task is not performed properly both the drone pilot and the intruder pilot are unable to perform the separation. To improve this situation it could be envisioned that the pilot of the drone has a number of predefined separation strategies that could be applied safely when the ATC is not providing separation instructions. This would lead to adding a task PIC_06 “Commands Drone to comply with predefined separation instructions” whose output could be merged with that of task PIC_05 “Commands Drone to comply with (ATC) instructions”. In that case a single fault of the ATC would no longer lead to the null output of the observer.

Another combination of interest is # 6: the loss of ATC_01 “Detects potential conflict situation with an intruder” combined with the loss of SAAS_01 “Provides information for situation awareness”. This combination is not acceptable because it does not contain a fault of one of the drone pilot tasks, and the drone pilot should be involved in the management of separation. To correct this situation the drone pilot should be able to perform PIC_01 without receiving the situation information provided by drone. In that case we could change the CONOPS scenario by merging the output of SAAS_01 and PIC_01 and remove the direct link from
SAAS_01 to PIC_01. This would be simple to achieve when the pilot can visually observe the drone as the pilot observation activity would be independent from the drone position measurement activity but it would require specific observation equipment for the pilot when the drone is flying beyond line of sight. Currently the French regulation for small drones [6] requires the drone pilot to be able to visually observe the drone during all flight phases.

The generation of minimal combination of faults is also useful to allocate requirements to the components in the model. In the aeronautics domain, a Development Assurance Level ranging from A to E is allocated to components such as functions, software and hardware items. The DAL allocation process is based on an analysis of the minimal combination of faults, the analysis aims at computing the severity of the effect of a component fault caused by a development error. ONERA developed a tool [7] that automatically allocates a DAL to components respecting the allocation rules from ARP4754a [8]. This tool was applied to the minimal combination of faults in order to explore various possible DAL allocations for the CONOPS tasks.

The last step of the safety assessment is mandatory for assessing systems made only of physical components whose reliability is usually measurable. It is more questionable when dealing with human operators whose reliability is not measured in the aerospace domain. In this case, a quantitative assessment could be used in order to compare easily two CONOPS.

The safety assessment was also performed on a model that contains a preliminary description of the resources needed to perform the operation. The following picture shows a model where tasks are related with two communication resources:

- **comm_PIC_Drone**: used to exchange information between the drone and its pilot,
- **comm_ATC**: used to exchange information between the ATC and pilots.

The previous picture shows the effect of the total loss of resource **comm_PIC_Drone** this leads to the loss of a number of PIC and SAAS activities (SAAS_03, PIC_01, PIC_05) but as the intruder pilot does not use this communication resource and is still able to perform the separation manoeuvre then the output of the observer is ok. The loss of **comm_ATC** resource leads to a null output of the observer because the separation instructions cannot be sent to both pilots (task ATC_03 is lost). The following table gives the minimal combinations of both operation and resource faults leading to null output of PIC_stop component.

<table>
<thead>
<tr>
<th>#</th>
<th>1st fault</th>
<th>2nd fault</th>
<th>3rd fault</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ATC_02.total_loss</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>ATC_03.total_loss</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>PIC_04.total_loss</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>PIC_05.total_loss</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>comm_PIC_Drone.total_loss</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>comm_ATC.total_loss</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>ATC_01.total_loss</td>
<td>ATC_04.total_loss</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>ATC_01.total_loss</td>
<td>PIC_02.total_loss</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>ATC_01.total_loss</td>
<td>PIC_03.total_loss</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>ATC_01.total_loss</td>
<td>SAAS_01.total_loss</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>ATC_01.total_loss</td>
<td>PIC_01.total_loss</td>
<td>SAAS_02.total_loss</td>
</tr>
<tr>
<td>12</td>
<td>ATC_01.total_loss</td>
<td>PIC_01.total_loss</td>
<td>SAAS_03.total_loss</td>
</tr>
</tbody>
</table>

**Table 2. Minimal combination of faults leading to PIC loss of separation**
6 Comparison with MIDCAS safety assessment

The results were compared with results described in MIDCAS document “Scenario Assessment” [9]. This document analyses the CONOPS scenarios using an event tree that can be described as a table (see Table 3). The left part of the table (columns 1 to 6) describes a combination of MIDCAS function faults and assumptions about the environment of the MIDCAS (drone, intruder aircraft, ATC), the right part (column 7) provides the effect of this combination (Loss of separation, mid-air collision, no safety effect).

The MIDCAS functions considered are:
- EF1. Provide Situational Awareness to the drone pilot, this function is represented by activities (SAAS_01, SAAS_02, SAAS_03, PIC_01, PIC_02, PIC_03) in the CONOPS scenario;
- EF2. Provide Separation Manoeuvre to the drone, this function is represented by activities (PIC_04, PIC_05);
- EF3. Provide Collision Avoidance, this function is represented by activities (SAAS_04 and SAAS_05);

The assumptions about the MIDCAS environment:
- Drone-PIC communication link status: either ok or null. This is represented by resource comm_PIC_Drone in the Operation safety model;
- Is the intruder aircraft able to perform an anti-collision manoeuvre ?: yes, no or undefined. This is represented by activities (IP_01, IP_02) in the CONOPS scenario;
- ATC separation instructions: either ok or null. This is represented by activities (ATC_01,ATC_02, ATC_03, ATC_04) in the CONOPS scenario.

To compare our results with the MIDCAS results we have generated the set of minimal combination of faults for the event tree. There are 3 single faults leading to the loss of separation (see Table 4).

Table 3: Table equivalent to the event tree - Extract

<table>
<thead>
<tr>
<th>EF1 (Provide Situational Awareness)</th>
<th>Drone PIC Comm.</th>
<th>EF2 (Provide Separation Manoeuvre)</th>
<th>ATC</th>
<th>Intruder ?</th>
<th>EF3 (Provide Collision Avoidance Manoeuvre)</th>
<th>End effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>null</td>
<td>ok</td>
<td>ok</td>
<td>ok</td>
<td>undefined</td>
<td>ok</td>
<td>separation loss</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ok</td>
<td></td>
<td>null</td>
<td>separation loss</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>yes</td>
<td>ok</td>
<td>separation loss</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>no</td>
<td>ok</td>
<td>separation loss</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>no</td>
<td>null</td>
<td>collision</td>
</tr>
</tbody>
</table>

Table 4. Minimal combination of faults for loss of separation generated from the event tree

Table 4 has to be compared with Table 2 (and not table 1) because the event tree focus on the loss of separation due to the drone pilot, this is similar to null output of PIC_stop component in the operation model. The difference between Table 2 and Table 4 can mainly be explained by the difference in the granularity of faults considered in the event tree and the operation model. In the MIDCAS analysis one fault represents several faults in the Operation Model so one line in Table 4 can represent several lines in Table 2. For instance, the fault of the ATC (line #1 in Table 4) represents lines #1, #2, #6 and #7 in Table 4 because ATC_fault represents ATC_0.total_loss, ATC_02.total_loss, ATC_03.total_loss, ATC_04.total_loss and comm_ATC.total_loss. Similarly the fault of EF2 “Provide Situational Awareness” (line #2 in Table 3) can be related with the loss of PIC_04 or PIC_05 that appear in lines #3 and #4 of Table 2.

Combinations #8 to #12 in Table 2 have no direct counterpart in the event tree analysis. The reason is again the difference in fault granularity. For instance, as ATC_01.total_loss is related, in MIDCAS, with ATC_fault and PIC_02.total_loss is related with EF2.fault then combination #8 in Table 2 (ATC_01.total_loss, PIC_02.total_loss) could be represented by {ATC.fault, EF2.fault} in Table 4 but this combination is not minimal because it contains {ATC.fault} that is a minimal combination leading to loss of separation. Consequently Table 4 does not contain {ATC.fault, EF2.fault}. Combinations #9 to #12 have no counterpart in Table 4 for the same reason.

7 Related Work

Currently, the risk assessment of aviation CONOPS is based on safety models like the “bow-tie” model [10] or adaptation of event-trees [11]. Such models first clarify the conditions of occurrence of a fault and its effects. Secondly they identify barriers that reduce either the occurrence frequency of a fault or the severity of the effect of this fault. Methods like event trees tend to lead to build manually complex combinations of heterogeneous conditions such as failure occurrence, human error or specific operational conditions (e.g. hypothesis about the air space category the drone is flying in). It is hard to argument the completeness and consistency of such models with respect to a CONOPS or with respect to a system specification.

In [12], the authors propose to analyse use-case scenarios in order to derive safety requirements. This method guides the safety analyst when building a manual risk assessment but the safety assessment is not performed automatically. The automation of safety assessment is likely to be needed when changes in the assumptions of the CONOPS occur frequently or a lot of new
operation scenarios are created. In [13] an approach based on the analysis of UML scenarios described as message sequence charts is proposed. The part of the risk analysis that associates possible deviations with each message exchange is supported by a tool but the analysis of the final effect of the deviation on the operation has to be performed by the analyst.

A more automated approach is proposed in [14] where a tool analyses scenario models described using the BPM notation and it can propose improvement of the scenario based the analysis results. This tool focus on quantitative safety requirements and does not seem to be able to deal with qualitative requirements related with the analysis of minimal combinations of faults.

8 Lessons Learnt and Perspectives

The proposed approach has been applied to analyse several scenarios defined to manage collision risk. Building the operational safety model with AltaRica helped to clarify several aspects of the initial CONOPS scenarios. The main clarifications that were needed during the safety modelling phase are:

- More rigorous description of dynamic aspects of the CONOPS: clarification of dependencies between actor activities and priorities between operation steps allocated to the same actor.
- Better explanations of hypothesis on the operation environment: clarification of what operation steps are triggered when a fault is detected or when an external condition becomes true.
- Better explanation of required aircraft functions in order to complete successfully the operation: we have only analyzed basic requirements in terms of communication resources, similar study should be performed for other types of resources such as computers, displays or electric power supplies.
- More detailed decomposition of some operation steps: communication actions were often grouped together with the action computing the data to be transmitted. For a better assessment of the effect of communication equipment faults, communication and computation steps were separated.

In the future we intend to re-use the developed modelling framework in order to analyse other types of CONOPS. We first want to model CONOPS for ground robots operating in an airport. There are interesting safety analysis to be performed on these systems because they should combine two antagonistic safety principles: stop in order to avoid colliding with a smaller object and perform a rather quick anti-collision manoeuvre to avoid being hit by an aircraft. We could also model new CONOPS such as the one recently published [15] by ICAO that describes the Global Aircraft Tracking potential solutions.

References