Addressing safety assessment of autonomous robot operation and design with Model Based Safety Assessment

Jean-Loup Farges, Claire Saurel, Christel Seguin, Frédéric Deschamp, Alexandra Favre-Bonté, Alain Ruaudel, Augustin Desfosses, Marc Laval

Addressing safety assessment of autonomous robot operation and design with Model Based Safety Assessment

Jean-Loup Farges,
Claire Saurel,
Christel Seguin
ONERA
2 av E. Belin, Toulouse
{firstname.name}@onera.fr

Frédéric Deschamp,
Alexandra Favre-Bonté,
Alain Ruaudel
LGM
1 rue E. Arin, Toulouse
{firstname.name}@lgm.fr

Augustin Desfosses,
Marc Laval
STERELA
5 imp. Pedenau, Pins-Justaret
{firstname.name}@sterela.fr

1 Introduction

Robots were initially designed to substitute human beings operating in manufactures for repetitive and fastidious tasks or in uninhabited planets for exploration. Those robots are typically automated robots which are operated in segregated or highly controlled environments. However, robots are increasingly expected to be used for tasks that are dangerous or difficult for human beings. Nowadays mobile robots are more and more autonomous and present in less and less segregated environments. Autonomous are designed for doing complex operations, but also for being able to cope with unexpected changes of their working environment, without producing any damage.

This context induces specific safety analysis issues. First, the safety principles of a new concept of robotic operation and the possibility to control the robot environment are highly dependent of the application domain. Since, by definition, there is no experiment report about new concepts, it is not easy to define technical safety requirements for their design. So, before going any further, it is necessary to specify and asses the safety of the concept of operation at an abstract level. Second, the robots remain more often under the supervision of human supervisors and the safety assessment shall be holistic in order to integrate role sharing between such semi-autonomous robots and their supervisors. Third, the ability of the robot to adapt its action often relies on complex components that may be safety critical. Thus an accurate allocation of safety objectives shall also be performed at hardware equipment level.

The methodology proposed here addresses those issues combining classical safety analyses of the robot at functional and hardware level with a less classical safety analysis of the concept of operation of the robot considering the share of roles. This methodology leads to three interconnected safety analysis corresponding respectively to three system design steps: 1) robot operation analysis in its environment, 2) functional system architecture analysis, 3) physical architecture analysis (hardware level). Thus there are three complementary abstraction levels analysis of the system. The different levels are not explicitly linked together. However, the approach consistency is achieved by traceability between analyses.

Fault Tree Analysis is one widely accepted mean to assess whether a system meets its safety requirements. A fault tree is a Boolean function which models how a top level hazard is caused by combination of low level faults. It can be built either manually or generated automatically from a system model, using the Model Based Safety Assessment (MBSA) approach. MBSA presents two advantages with respect to manual approaches in the context of application to autonomous robots:

- Complexity of systems and reconfiguration features can be explicitly modelled using not only dysfunctional dependencies but also functional dependencies between components.
- An improvement of the robot often corresponds to local changes in the system impacting only few components of the model but impacting globally the fault tree. For more and more autonomous robots, improvements are frequent.

For this reason the methodology proposed here is implemented through a MBSA approach, more specifically using the AltaRica language and associated tools. The application of the methodology to autonomous mobile robotics is a complex task and raises questions about the way to get self-consistent whole models which deal with operations as well as with functional and physical architectures, and which are easy to validate, extend or to update, in order to be used for a safe design. These questions are illustrated here through a case study corresponding to an industrial application.

The next section gives background information on MBSA, AltaRica language and its application approach for autonomous robot. Then another section presents a case study for applying MBSA to an actual autonomous robot designed for runway lights inspection. The three following sections provide details of the application of MBSA at three steps of robot design: external functional analysis, internal functional analysis and detailed hardware design. The case study is used in those sections to exemplify the proposed MBSA approach. A last section presents conclusions and possible extensions of this work.

2 Model Based Safety Assessment for robot operation

2.1 MBSA and AltaRica

MBSA is an approach to formally perform safety assessment for complex reconfigurable autonomous systems. “Formally” means here that it provides language and tools to describe concepts (operations, functional architecture, physical architecture...) without any ambiguity, to automatically simulate propagations of faults, and to compute minimal configurations of faulty components which lead to undesired events. MBSA then helps to validate the modelled concept and to derive safety requirements for the next design steps.

Amongst all the languages available in the literature to perform MBSA, we have chosen to use AltaRica, a formal modelling language to describe both functional and dysfunctional behaviour of a system [1]. Moreover, the language is carried out by several tools. We use the tool Cecilia™ OCAS from Dassault Aviation to edit graphically the model then to analyse it by several means: simulation, automatic generation of minimal cuts (i.e. the shortest scenarios leading to the failure condition) or sequences (i.e. ordered cuts). Successful modelling and safety analysis with AltaRica is described for simple operational models [1] and for systems models described only at a chosen abstraction level [3].

An AltaRica model is a network of interconnected components so called “nodes”. A node is atomic or composed of interconnected sub-nodes. Each node has a finite number of:

– *flow variables*. They are the inputs and the outputs of the node used to link the node and its environment (other nodes).
– *state variables*. These internal variables memorize current or previous functioning mode (for example, failure mode). In our models, these variables (flow and state) belong to finite domains of values (Boolean or enumerated)
– *events*. They label changes of the value of state variables. They model the occurrences of fault, human action or a reaction to a change of one input value.

The node dynamic is defined by:

– *init*. This is used to assign initial value to state variables
– *transitions*. They describe how the state variables are modified. They have the following format: “G(s,v) |- E ->s_” where G(s,v) is a Boolean condition on state variables s and input variables v, E is the event and s_

is the effect of the transition on state variables. If the condition G is true, then the event E can be triggered and state variables are modified as described in s_.

– *assertions*. These equations describe how output variables are computed from inputs and state variables.

These concepts are illustrated by the following example, used to model the completion of robot actions at the operational level.

```
node funIO
//declaration of variables and events
  flow
input: bool: in;
output: bool: out;
  state
ok : bool;
  event
fail;
//dynamic of the states
  init
ok:=true;
  trans
ok |- fail -> ok:= false;
//function performed in each states
  assert
output = case { ok : input,
                 else false} ;
edon
```

This component has one *input* and one *output* variables both ranging over the Boolean domain {true, false}, one Boolean state variable and one event *fail*. At the initial instant, the node is in state "*ok:=true*". The event *fail* describes a failure which leads the node into the state "*ok:=false*". "*fail*" can be triggered only if the node is in state "*ok*". The assertion means that the output value is equal to the input one if the node is in state "*ok:=true*". In other cases, the output value is "*false*".

2.2 Strategy of application of MBSA for robot operation

We propose three steps of application of MBSA to refine progressively the safety assessment of a robot operation.

The first step deals with the analysis of *the robot operation* in its environment. The model of the concept of operation highlights the hazards that shall be mitigated in all phases of operation and the coarse grained activities undertaken to detect and mitigate the risks: supervisor tasks, external robot functions and communication between robot and supervisor to achieve situation awareness, action planning and execution. The model integrates adverse environmental conditions technical failure modes and human error in order to assess whether the role sharing is robust enough. Moreover, this model clarifies the priority between potential conflicting decisions. Such conflicts occur when the remote supervisor and the on board automatism have different views of the situation or when two hazards need simultaneously to be mitigated by incompatible actions (e.g. stop versus exit). At the end, this step of analysis provides an allocation of safety objectives for the supervisor and the technical system that maintain the operation at an acceptable level of risk.

The second step deals with the analysis of the *functional robot architecture*. The model aims at clarifying how the external robot functions are implemented by internals functions, ranging from robot navigation, guidance and piloting to fault detection and reconfiguration logic. It integrates functional failure modes and safety objectives allocated to these lower level functions in order to assess whether the internal functional architecture meets the safety objectives allocated to the high level functions. Moreover, this specification shall implement the same priorities between recovery mechanisms than the ones that have been specified at the operational level. New requirements may be derived from this design step.

The third step deals with the analysis of the physical architecture of robot (hardware level). This model describes how the low level functions are implemented on hardware devices (processors, energy chain, actuators, ….) . It integrates physical failure modes and this more detailed specification shall be compliant with all the requirements previously defined.

3 Case study

In order to show what kinds of models and safety analysis can be produced at each level with AltaRica MBSA framework, a case study is proposed. It includes a robot which has to check lights on runways in an airport

Runways are delimitated by rows of lights which help pilots to identify the landing runway and safely land. In order to be in compliance with an ICAO requirement (annex 14), it is mandatory to regularly inspect these lights: a light is considered as deficient if its light intensity is lower than half its value required by ICAO annex 14, and in this case it has to be replaced. This tedious work used to be manually carried out at night by two supervisors driving a vehicle equipped with a Photometric Airfield Calibration. The aim of the industrial supervised 4MOB–Aero system is to do this job. The 4MOB–Aero system is composed of a mobile robot, specially equipped to perform light measurements; this robot is supervised and controlled by an supervisor who communicates with the Air Traffic Controllers (ATC) of the airport.

In order to get authorization to exploit such a system, it is necessary to prove that its design and operational procedures prevent the occurrence of some undesired events, such as robot damage, collision with an aircraft or human beings on the runaway or degradation of the inspected infrastructure in case of an unexpected change in the environment or a system failure.

3.1 MOB-Aero functional requirements

During the mission preparation, the robot is informed about the inspected zone, led by the supervisor on the runway and put at a starting position close to the first inspected light.

4MOB may be operated beyond visual line of sight of its supervisor but it shall be continuously monitored through datalink.

After a start telecommand from the supervisor, the robot is expected to perform the light inspection as follows:

- follow a track of lights while controlling each of them,
- after the last light of a row, move to the first light of the second row to control, and so on, until all the lights of the specified zone have been controlled.

Then it comes back to its starting place and waits.

For safety of aircrafts or other devices, the light control mission is only performed on closed runways. The airport is supervised by ATC who will request to park the robot in a safe place if the runway needs to be open in emergency. This request will be managed by the robot supervisor, the resulting requirement is:

R1: If the supervisor asks the robot to exit from the runway, the robot shall stop its mission and exit the runway.

In case of detection of collision risk on the runway or robot failure, the robot has to immediately stop into a safety mode. One resulting requirement is:

R2: In case of detection of collision risk, the 4MOB shall perform an emergency stop.

3.2 MOB-Aero safety requirements

We identified the following undesired events (UE) with a safety impact on human being or goods during the 4MOB-Aero operation, or which concern mission unavailability:

UE1: collision of the robot with intruders (aircraft, obstacle…) on the runway

UE2: degradation of the runway (contamination with robot debris, light degradation)

UE3: abortion of the inspection mission before it has been finished (availability problem)

UE4: injuries of robot supervisor

UE5: provision of erroneous light measure….

In the following we focus on UE1 which can be caused by combinations of technical failures, human error or adverse external conditions. The study will be carried on more specifically the two following failure conditions (FC):

FC1: the robot operation cannot end with a runaway exit whereas it is requested (violation of R1 which may lead to UE1)

FC2: the robot operation cannot end with an emergency stop in presence of an obstacle on the runway (violation of R2 which may lead to UE1 or UE2)

The discussions with 4MOB-Aero experts have led us to classify the severities of the undesired event UE1 and failure conditions FC1 and FC2 as "CAT" (for: catastrophic), inspired by the safety classification used in the aircraft domain (CAT, HAZ, MAJ, MIN, No Safety Effect). The class definition and the associated safety objectives are specified by aeronautic regulation such as AC 25.1309-1A for large civil aircraft. Such regulation is not available for a robot operated in an airport. We proposed the following safety objectives for CAT FC1 and FC2:

- The operational procedures and robot architectures shall be designed so that no single failure or human error leads to FC1 or FC2
- The occurrence of FC1 or FC2 shall be rare:
  1) Reliability level of hardware components of the system shall be compatible with this objective in order to cover correctly random failure occurrence;
  2) Design Assurance Level of any technical equipment of the system shall be compatible with this objective in order to cover correctly design error occurrence.
  3) Training of the human supervisor shall be compatible with this objective in order to prevent correctly human error occurrence.

4 Analysis of robot operation

This section illustrates how we achieve a model and its analysis to support the safety analysis of the robot operation. We focus on the following sub-phase: the robot moves from a starting point to a lamp to be controlled, on a closed runway.

4.1 Model overview

The model performed for the safety analysis of the operation contains the following parts.

The sub-model of the *operation* itself is structured as follows. At the highest level, the operation is made of successive phases: mission preparation, light inspection, measures analysis after the mission completion and robot maintenance. The "light inspection" phase is presented in figure 1 in appendix. It is decomposed following a classical robotic paradigm in OperationMonitoring (i.e. knowledge acquisition), OperationControl (i.e. decision making), and RobotAction (i.e. move performance). The last refinement level shows how these high level activities are jointly performed by actions or communications of the three actors of the scenario: the robot, its supervisor, and the airport ATC. Figures 2, 3 in appendix show respectively the details of the operation monitoring, operation control and robot moves.

The model contains also *observations of environmental events* (OpenRunway, RobotKO, …) which can affect safety if there are no mitigated. In figure 1, these conditions are on the top left of the figure, they correspond to outputs of the RealWorld component and are the inputs of OperationMonitoring.

The *undesired events and failure conditions* are modeled by specific components called observers (at the right of the scheme in figure 1). For instance, the observer CollisionWithObstacle of FC2 is true whenever both DebrisOnRunway is true while EmergencyStop doesn't work (false). The IntempestiveNormalMissionStop component models an observer for an availability problem rather than a safety failure: the robot cannot succeed in doing its light inspection mission, but there is no damage. All other observers deal with failures with potential safety effect during the light inspection mission.

4.2 Failure propagation logic

Operational failures may be generated by adverse external conditions (like runway open), by human errors or failure of technical components. In order to deal homogenously with failure propagation, we introduce failure modes per kind of actions whatever it is performed by the robot or by the operators. So, only one general failure mode has been defined for actions with effect on the physical world: failure performance – which covers both loss and erroneous (cf component funIO of section 2). Two failure modes have been defined for cognitive actions (knowledge acquisition and decision making) which are managing pieces of knowledge or decision that are true or false (i.e. Boolean statements): stuck-on true and stuck on-false. For instance, a wrong alarm is an alarm observation stuck on true, whereas an alarm loss is the observation stuck on false.

It is worth noting also that the model propagates not only failure modes (i.e. the possible deviations from the normal activity) but also normal activity outcomes that are useful to control or observe the failure propagation, such as an alarm. Safety functions were integrated in the operational model in order to make the system behaviour safe whatever imagined undesired event occurs. These monitoring and failure management functions are distributed between all actors. For instance, during the operation monitoring, ATC is in charge of managing the runway status (open or closed) and communicating this status to the robot supervisor who builds its own knowledge about the runway status (see first line of figure 2). During the operation control, the supervisor merges all pieces of knowledge to make decision about robot moves and it sends the decision to the robot (see figure 3). Sensors and on board logic can also be used by the robot to monitor the situation and decide the following physical actions.

The definition of the safety policies was challenging, due to the need to manage conflicting safety functions: stop the robot to avoid collisions with ground obstacles and exit the runway on demand of the air-traffic controller, for situations where the robot would have to perform them simultaneously (for instance, runway becoming open while a detection of collision risk). The priority has been given to the most urgent action: emergency stop. The priority management has been modelled using Boolean gates to merge knowledge and decisions (cf figure 3). This approach can be reused for other applications.

Under all these modelling hypotheses, observers introduced at the top level of the model respectively reflect that:

- the robot is hit by an intruder aircraft while it could not leave the open runway to go to a safe place (FC1)
- the robot hits an obstacle on the runway because it could not perform an emergency stop (FC2).

4.3 Safety Analysis

We first validated the model with operation designer using interactive graphical simulation of Cecila OCAS. It consists for safety engineers in triggering events of the RealWorld model or some component failures in the whole model. Then the tool computes the next stable situation and new failures can be injected. Graphical conventions tied to the values of failure modes of AltaRica components show which parts of the operation are more or less degraded: green means that the modelled action performs correctly or that a described condition is true; red means that the action fails or that a described condition is false. Examples of simulation runs are presented in the appendix. They were more specifically used to validate the priorities between EmergencyStop et EmergencyExit requests (cf figure 5 in the appendix).

Cecilia OCAS also provides a sequence generator which computes minimal combinations of component failures leading to undesired events: in other words, making observers of the AltaRica model true. 3 sequences leading to FC2 (collision with an obstacle) were computed (cf Figure 7). They combine the presence of an obstacle with three single failures of the following robot actions can lead to FC2:

- OperationControl.RobotDecideEmergencyStop
- OperationMonitoring.RobotAnalyseObstacle
- RobotAction.StopMove

So the operator procedure does not help to secure the operation, the derived recommendation is to add independent redundant functions in the robot architecture to make the operation safer. This will be checked on the functional architecture of the robot. If it is not possible, the procedure shall be changed. Moreover, a high design assurance level and high reliability objective are requested for the robot functions and hardware that will implement this action.

Similarly, 28 sequences leading to FC1 (collision with an aircraft) were compute. They contain single failures of actions of ATC (2), supervisor (13) and robot (13) actions. Moreover, they

highlight that an inadvertent emergency stop will prevent the emergency exit.

4.4 Lessons learnt

The novelty of this experiment is the complexity mainly introduced by the distributed management of conflicting safety goals. The proposed modelling structure supports modular development of sub-models (for the operation monitoring, decision making and acting), that were simply integrated by connecting all inputs and outputs of the different sub-models.
We have got both a more readable and complete overall model for both safety and designer engineers, while the computation load of the sequence generator remains tractable. In case of need of debugging the model, its modular and refinement-based structure makes corrections and their impact on the rest of the model more easily localisable.

5 Analysis of robot functional architecture

This section illustrates how we achieve a model and its analysis to support the safety analysis of the robot functional architecture. We focus on the architecture of on-board functions that are specified to control and command the robot moves. The failure conditions analysed at this level are the failure modes of the robot actions that lead to FC1 and FC2

5.1 Model Overview

The model performed for the safety analysis of the robot functional architecture contains as previously the studied system view, the hypothesis about the system environment and the failure conditions (cf figure 8 in appendix)..
The environment contains the obstacle (present or not on the runway) and the operator who can control the modes of robot moves: transit to a line of lights, following the line, runway exit or stop move.
The studied system view is the onboard robot functional architecture. It depicts low level functions that will implement the high level actions identified during the operation specification: monitoring of obstacles and system health, situation assessment and selection of piloting modes, move stop, normal move, runway exit, communication with the operator station. For instance, the vision based navigation is used when following a line of lights whereas a GPS based navigation is used as primary mean in other cases.
The definitions of the failure conditions are refined according to this design choice. For instance, the collision with an obstacle may results not only from the presence of an intruder on the runway but also from error in the robot trajectories.

5.2 Failure propagation logic

Failure modes of the monitoring functions are the same than previously (stuck at true / false). The failure modes of the control function are loss or erroneous behaviour.
The main difficulty of this model is to manage the various functional modes. The mode evolutions have been modelled by transitions between functional states of the mode manager function. Then switches select the outputs of the functional lanes according to the active mode.

5.3 Safety analysis

As previously, simulation and sequence generations have been performed to validate the model and analyse the failure conditions. We assume that the collision with an obstacle is classified CAT and the loss of emergency exit is classified HAZ.
The collision with an intruder is due to the intruder presence and 2 singles failures: loss of obstacle detection or loss of emergency stop loss. We assume that the probability to have an intruder is circa 10^{-2} per hour of operation; thus the probability of loss of detection or emergency stop shall be less than 10^{-4} per hour of operation probability of to reach a safety target of 10^{-6} impacts per operation hour. It is homogeneous with requirement of development assurance level D of aeronautic standards.
The collision due to error of trajectories is caused by double failures. 12 scenarios combine 1 error in the navigation or pilot with the non detection of the error; 12 scenarios combine a detected error with the loss of the emergency stop. For these scenarios, the probability of presence of light on ground is 1. So the probability target for the 12 navigation and piloting sub-function shall be around 10^{-3}.

5.4 Lessons learnt

The robot function model is a pivot model between the operation and the hardware models. The operation models helped to identify all the high level needed functions and their priority.
The robot function model refines this hypothesis and it eases to debug the mode management and the reconfiguration logic. We are investigating how to use it to simplify the hardware models and analysis.

6 Analysis of robot hardware architecture

This section illustrates how we achieve a model and its analysis to support the safety analysis of the robot hardware architecture. We focus on the architecture of on-board hardware that is used to implement the control and command functions.

6.1 Model Overview

The hardware model describes the hardware architecture used to implement the robot's functions, components failure modes and their impact on the functions allocated.
The aim of the experiment is to perform a complete hazard analysis and then implement a safety monitor on the robot. The results obtained help to the justification of level Dal D requirements.
Contrary to operational and functional level, hardware level is based on a lower level of architecture. Different level of the analysis can be considered as black box such at an upper level with all control components: PC card, Command card and Power card. Following robot architecture, each card can be decomposed at a lower level by block functions such as "connectors, relay, converters, and microprocessor" (See figure 9). The hardware model is in line with the current safety analysis practices.

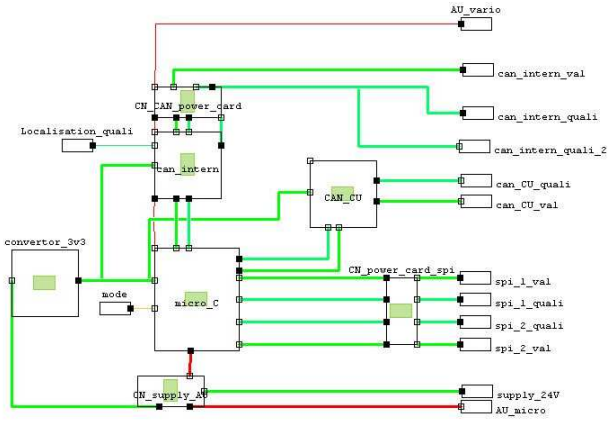

Figure 9.: Model at command card level

6.2 Failure propagation logic

Each block integrates failures modes defined for each component type such as loss or erroneous. Consequently a Failure Mode Effects Analysis has been performed indirectly for all components in order to have an exhaustive model approach
Data flow types are defined by two types: Quality which is defined by the following state as {ok, erroneous, none} and the Value {True or false}.The quality of output and inputs are distinguished from value state in order to have an exhaustive representation of all degraded mode.

At operational and functional level, MBSA modelling allows propagation of functional failure modes and nominal values related to the robot's functions. In order to have continuity with the other model level, the model takes in account several robot modes define in previous analysis. This choice has been done in order to consider each undesired events which could happen in robot operational phases.

Components are related with function and state, with the difficulty of each component can be non-used during a state. Moreover, components can have more than one function. Then all components and blacks blocks are linked with undesired events.

6.3 Safety analysis

A preliminary risks analysis allows identifying 7 undesired events applicable at this analysis stage.

Validation of the model is a very important phase because of the important number of components, functions and modes of the model. The validation has been performed in two steps, using respectively simulation and sequence generation. The first step aims at validating the nominal mode of the robot in case of cascading failure and reconfiguration essentially. The second step ensures the validity of failure effects and propagations. As the robot control is a highly integrated system, basic physical components may achieve different functions leading model validation complex.

The model allows having two kinds of results. The first one is to find all scenarios leading to an undesired event (such as impacting some lights on runway, or erroneous trajectory). This analysis is based on a sequence generator taking into consideration positions of events. The analysis of this information allows identifying the weak points, at robot design point of view and procedure point of view.
The other one is the probability of occurrence of undesired event, in order to compare with probability targets. The hardware model is in line with the current safety analysis practices: it supports the minimal cut set computation of the undesired events as well as their probability of occurrence. In order to have failure rate results, a reliability analysis of the robot was performed and failure rate of components were added in the model. So that results can be exported with cut set calculation. Because of the dynamic approach of the modelling only cut set can be obtained and be used for failure rate calculation of undesired event. One of the problems we have been dealing with at this level were the dynamic nodes. The solution finds is to use both methods: functional blocks and organic components in the model.

6.4 Conclusion

These results such as critical results failure rates, allows identifying critical point of the architecture and implementation of a safety monitor. Finally, the safety analysis contributes to define a monitoring function with a high level of independency from the main control function. These results help to check if the robots meet DAL D requirements, the level of certification for authority's assessments.

7 Conclusion

AltaRica code may be complex to unfamiliar readers. Coaching not only on the language but also on its methodology of use is needed for an efficient use of the approach. Moreover, the use of graphical simulation greatly helped the model co-validation by safety and design experts.

As several models had been first designed in parallel by different teams in the project, the question of clarifying the chosen perimeter and hypothesis for modelling was important in order to get consistent safety analysis results, from the operational level to the hardware architecture level. This suggested some updates of the initial models, in order to get a complete and consistent model of the system we wanted to analyse the safety, and a readable model for all people who need to learn from safety analysis for their design work.
One model containing all kinds of sub-operations is too difficult to read, debug or update. Models mixing both operational and functional concepts or detailed functional and hardware concepts have similar drawbacks. So, the principle of a level-structured and more modular model was successfully demonstrated:

- The traceability between levels is quite good.
- The MBSA approach has been applied to the operation of an industrial system that is more complex than systems previously addressed using the same approach.
- Violations of safety requirements by the robot were detected and corrected.

On the modelling point of view, we still have to deal with some methodological questions in depth in order to make the connections and the traceability more readable between all the abstraction model levels. Despite the remaining questions, the stakeholders have nevertheless a good confidence level that the three levels methodology is a good methodology that can be generalized to many other contexts. Practically, future works will include increased dissemination of the method in the industrial community and actions towards authorities in order to check that the methodology in general and its results for the case study are acceptable for authorizing operations.

8 Acknowledgements

The results presented in this paper were obtained during an experiment that was performed in the context of the CPSLabs European Union (EU) Horizon 2020 project, grant agreement number 644400, which helps to transfer academic methods and tools to industrial designers. The authors thank the other members of the CPSLabs project and the EU for giving them the opportunity to use AltaRica to experiment a formal MBSA approach in a robotic industrial application. The authors also thank Dassault Aviation for providing them temporary CECILIA OCAS licences.

9 References

1. A. Arnold, A. Griffault, G. Point, A. Rauzy, A "The AltaRica Formalism for Describing Concurrent Systems". In Fundamenta Informaticae, vol. 40, n°2-3, pp. 109--124, IOS Press (2000)
2. P. Bieber, C. Seguin, V. Louis, F. Many "Model Based Safety Assessment of Concept of Operations for Drones", Lambda Mu 20, Sait-Malo 2016
3. Pierre Bieber and Christel Seguin, chapter 3. "Safety Analysis of Embedded Systems with the AltaRica Approach", in Industrial Use of Formal Methods: Formal Verification, Jean-Louis Boulanger Editor, Wiley, 2013
4. "Guidelines for development of civil aircraft and systems" EUROCAE ED-79A / SAE ARP 4754A, 2010

10 Appendix

10.1 Top level view of the operation model

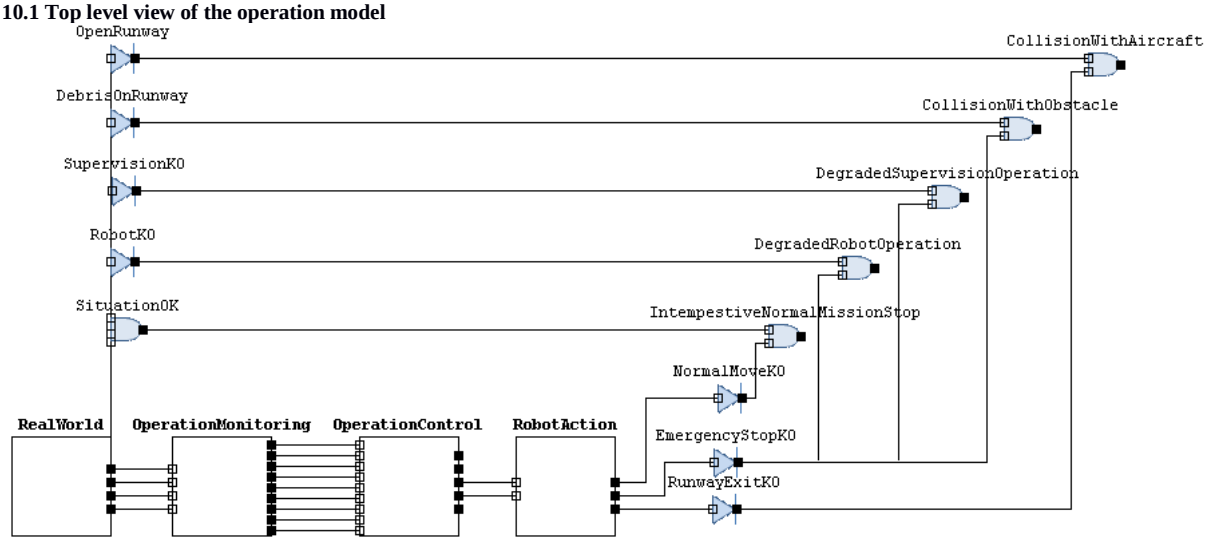

Figure 1. Global operational model

10.2 OperationMonitoring model

This model describes knowledge communication and acquisition activities between actors.

White boxes on the left of the figure correspond to environmental events (cf Figure 1) and are respective ouputs of The RealWold component of the model. Other ones describe knowledge status of actors.

Other icons describe knowledge analysis, send or receive actions of actors.

The supervisor has a central knowledge of the situation and of the status of all actors. Only the supervisor status is known both by ATC and the robot

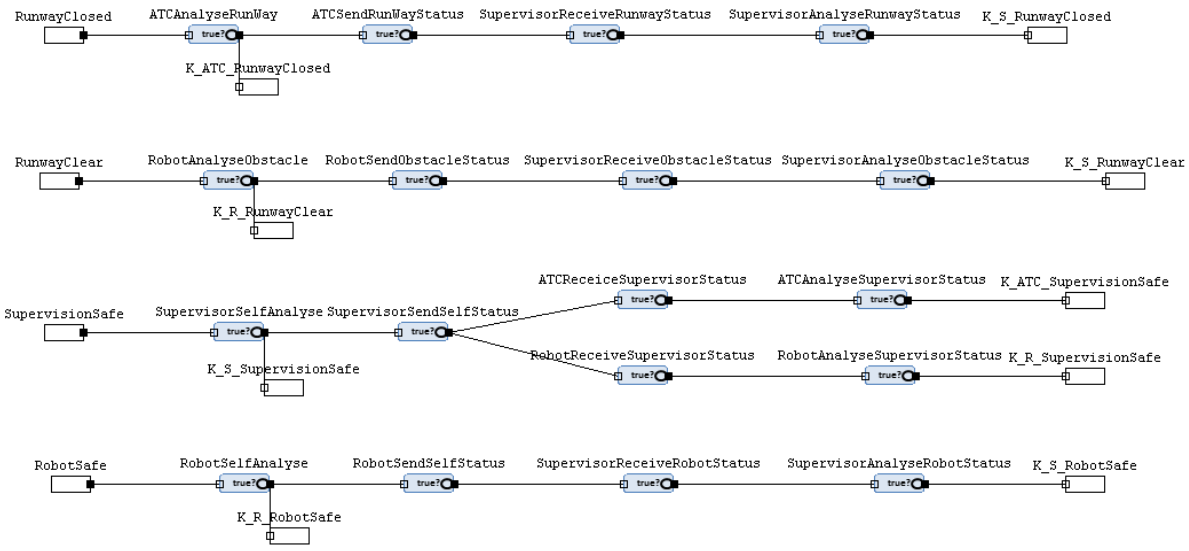

Figure 2. Operation monitoring model

10.3 ControlMonitoring model

A priority mechanism is here modelled between two safety functions. Whenever the supervisor or the robot assesses that an Emergency stop is needed, Emergency stop is decided for the robot. If the supervisor is informed that the runway cannot be closed, it decides to order a runway exit to the robot only if an Emergency stop has not been decided.

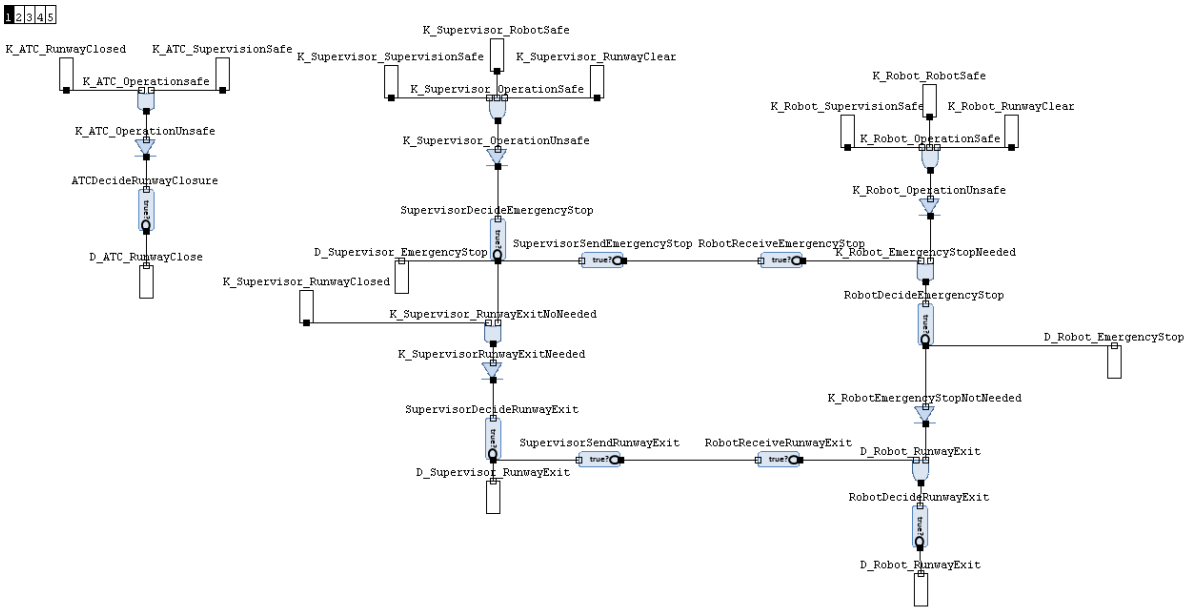

Figure 3. Control operational model

10.4 Operation simulation

For instance, first suppose there are debris on the runway while the runway keeps closed, the robot and the supervisor being OK: RunwayClear is set to false in the RealdWorld component (DebrisOnRunway gets true, that produces a green colored link to the observer CollisionWithObstacle). The simulation shows that the robot performs an EmergencyStop (green), so that the observer CollisionWithObstacle is colored in red. This means that the collision with debris is avoided. Since the runway is supposed to be closed, there cannot be any CollisionWithAircraft, so this observer is red. See Figure 4. Presence of debris on the runway.

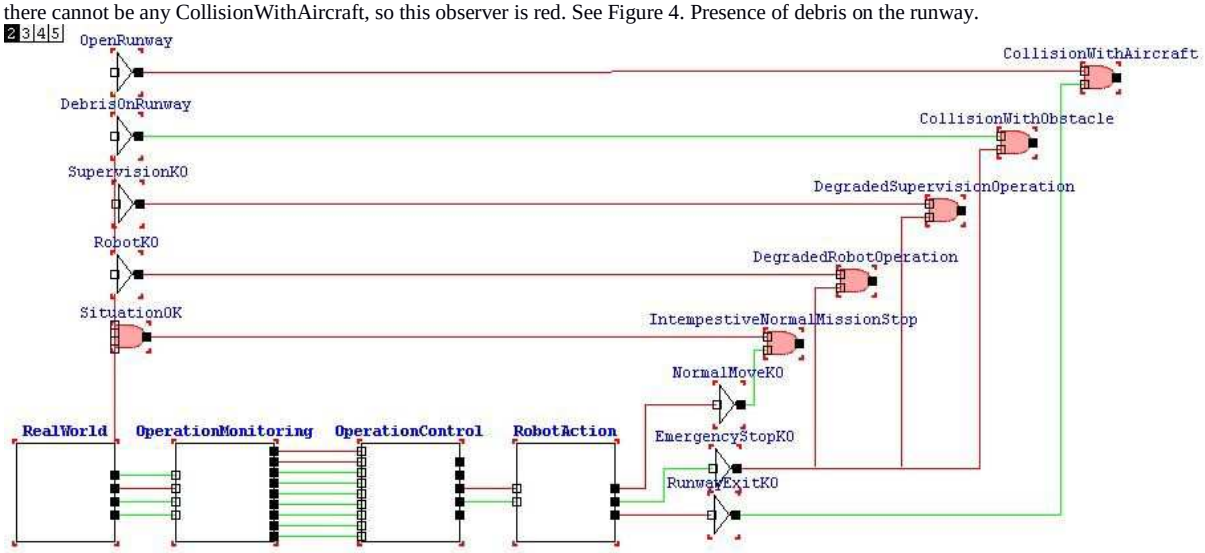

Figure 4. Presence of debris on the runway

Let then suppose, with one more simulation step, that the runway becomes open (the robot can meet aircrafts landing on it), by setting RunwayClosed to false in the RealWorld component. OpenRunway becomes true (green), EmergencyStop is correctly performed by the robot (green output), while the RunwayExit task fails (red output). The simulation shows that though the robot is not crashed by debris (the CollisionWithObstacle observer is colored in red), it cannot avoid landing aircrafts (the CollisionWithAircraft observer is colored in green). This is compliant with the priority mechanism modelled to keep with situations where both an EmergencyStop and a RunwayExit are required. See Figure 5. Presence of debris on an open runway".

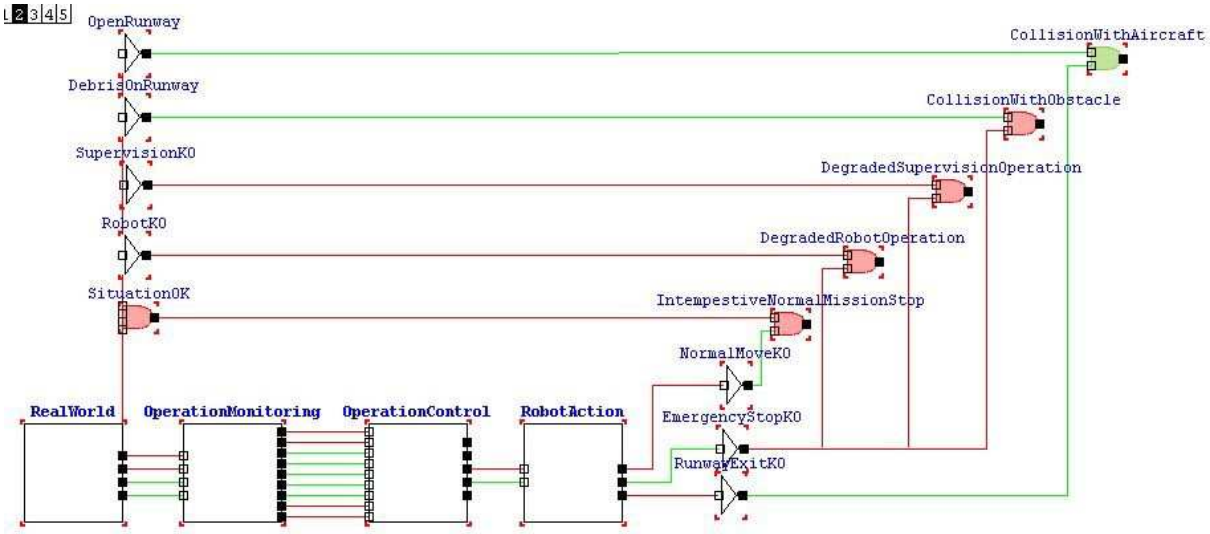

Figure 5. Presence of debris on an open runway

You can also put a failure mode in a action, so see how it is propagated trough the model. Let suppose moreover that the action RobotAnalyseObstacle fails in the component OperationMonitoring (depicted with a crossed icon), and "believes" that the RunWay is always clear (green color). The simulation shows that the EmergencyStop cannot be performed, so that the robot will get in a CollisionWithObstacle (green), while CollisionWithAircraft cannot occur. See Figure 6. Presence of undetected debris on an open runway

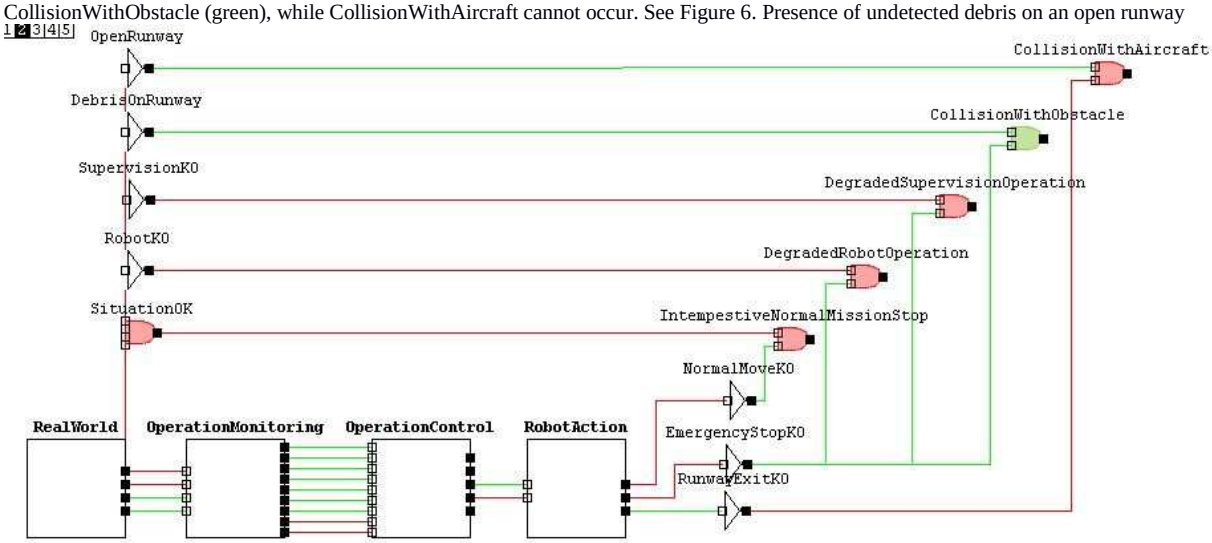

Figure 6. Presence of undetected debris on an open runway

10.4 Sequences of the operation model leading to the collision with an obstacle on the runway

```
/*
orders(MCS('CollisionWithObstacle.O.true')) =
orders  product-number
2       3
total   3
end
*/
products(MCS('CollisionWithObstacle.O.true')) =
{'OperationControl.RobotDecideEmergencyStop.stuck_on_false', 'RealWorld.RunWayClear.is_false'}
{'OperationMonitoring.RobotAnalyseObstacle.stuck_on_true', 'RealWorld.RunWayClear.is_false'}
{'RobotAction.StopMove.fail', 'RealWorld.RunWayClear.is_false'}
end
```

Figure 7. Sequences computation for FC2 (Collision with an obstacle on the runway)

10.5 Top level view of the model of the functional architecture

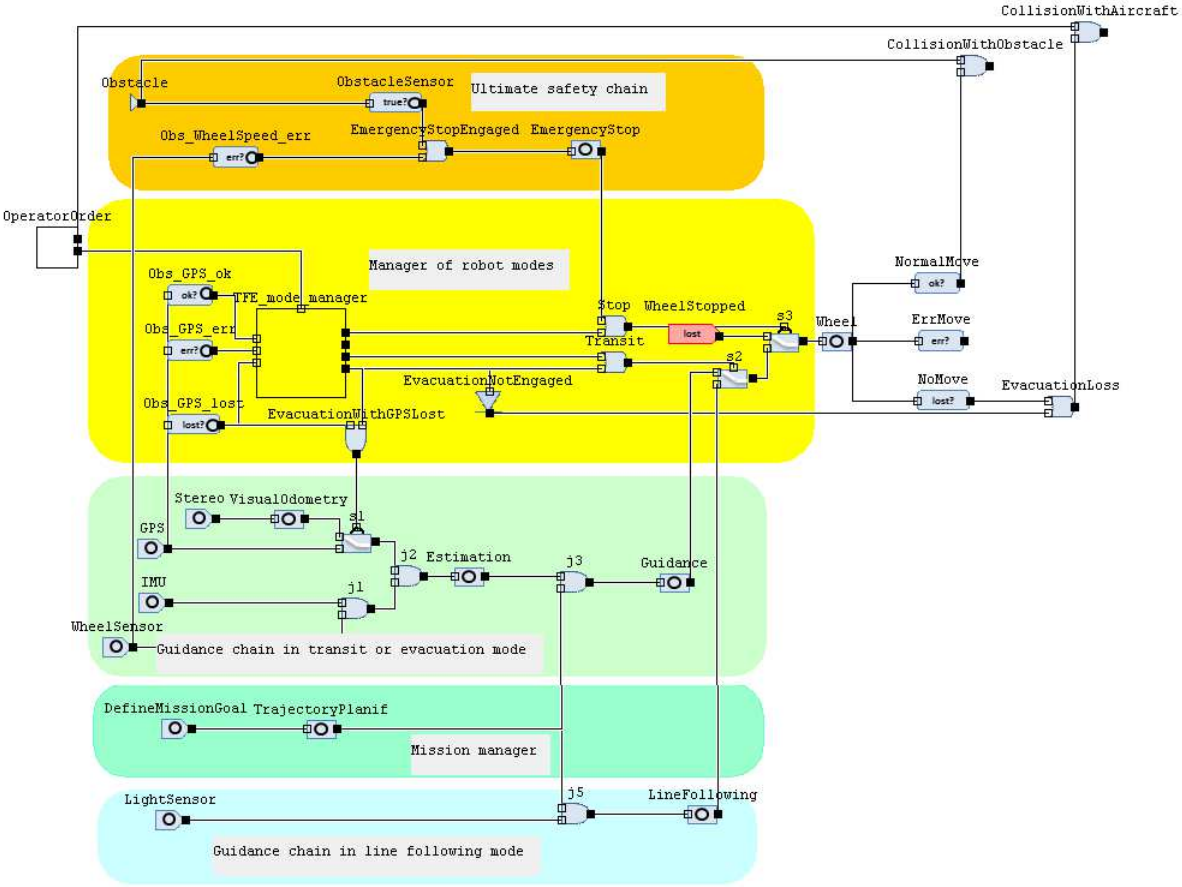

Figure 8. Functional architecture of the robot control