Safety Analyzes of Mechatronics Systems: a Case Study

Benjamin Aupetit, Michel Batteux, Antoine Rauzy, Jean-Marc Roussel

Safety Analyzes of Mechatronics Systems: a Case Study

B. Aupetit [*,**] **M. Batteux** [*] **A. Rauzy** [***] **J.-M. Roussel** [****]

[*] *IRT SystemX, 8, avenue de la Vauve, 91120 Palaiseau, France, (e-mail: {benjamin.aupetit, michel.batteux}@irt-systemx.fr)*
[**] *LGI, CentraleSupélec, Grande Voie des Vignes, 92290 Châtenay-Malabry, France*
[***] *IPK/NTNU, S. P. Andersens veg 5, Valgrinda*3.306, 7491 Trondheim, Norway, (e-mail: antoine.rauzy@ntnu.no)*
[****] *LURPA, ENS Paris-Saclay, 61 avenue du Président Wilson, 94235 Cachan, France, (e-mail: jean-marc.roussel@ens-cachan.fr)*

Abstract: In this article, we present a safety analysis of the case study "Landing Gear" proposed recently by Bonniol and Wiels. This case study mixes both physical (hardware) elements and control (software) elements and is representative of a large class of mechatronics systems.
For this analysis, we used AltaRica 3.0 as modeling language and stochastic simulation as analysis tool.
This experience sketches a methodology to assess the effects of hazards, failures and uncertainties in mechatronics systems.

Keywords: Case study "Landing Gear", Model Based Safety Assessment, Stochastic Simulation, AltaRica 3.0

1. INTRODUCTION

Boniol and Wiels proposed recently a case study, a landing gear system, as a benchmark for techniques and tools dedicated to the verification of behavioral properties of systems (Boniol et al., 2014). This use case is very interesting because it involves both hardware and software elements in an intricate way. For this reason, it is representative of a large class of mechatronics systems.The cited book present different modeling approaches of the software part of the system.

In this article, we revisit this case study from the point of view of the safety analyst. The main objective of safety analyzes is to assess the likelihood that something goes wrong. Safety regulations and standards such as (ARP4754A, 2010) and (ARP4761A, 2004) prescribe to perform probabilistic safety assessments for every critical system, including of course avionic systems such as landing gears. As of today, these analyses focus almost exclusively on hardware. Conventionnal methods such as Fault Trees or Event Trees (see e.g. Rausand and Høyland (2004)) are widely used and well mastered. They are however clearly not expressive enough to address mechatronics systems. For this class of systems, at least some abstraction of the software/control part has to be embedded into the model.

AltaRica 3.0 (see e.g. Prosvirnova et al. (2013)) is a high-level modeling language dedicated to probabilistic safety assessments. AltaRica 3.0 is a prototype-oriented modeling language (see e.g. Noble et al. (1999)). Its semantic is described in term of Guarded Transition Systems (Rauzy, 2008). As we shall show here, it has the expressive power to describe not only hardware parts, but also to describe the control part of mechanical systems.

Calculations of probabilistic safety indicators are computationally hard (namely #P-hard (Valiant, 1979)). Therefore practical means (algorithms and heuristics) have to be found to push back the combinatorial explosion. Monte-Carlo simulation is one of the tools at hand. It is widely used in risk assessment (see e.g. Zio (2013) for a recent monograph). Of course, it is not the universal panacea: no exact result can be obtained, only confidence ranges. But it proves to be a useful tradeoff in many situations where the problems at stake are too complex to be solved exactly.

The main contribution of this article is to show the capacities of AltaRica 3.0 and stochastic simulation to perform a meaningful probabilistic safety assessment of mechatronics systems. Moreover it shows that this can be done at a reasonable engineering cost.

The remainder of this article is organized as follows. Section 2 provides the reader with some background about this work. Section 3 presents briefly the case study and the AltaRica 3.0 model we designed. Section 4 gives more details about the AltaRica 3.0 description of the control part of the system. Section 5 reports experimental results. Finally, section 6 concludes the article.

2. BACKGROUND

Safety analyzes of physical systems have two characteristics: failures are stochastic (not preventable and random

⋆ This research work has been carried out in the framework of the Technological Research Institute SystemX, and therefore granted with public funds within the scope of the French Program "Investissements d'Avenir".

occurrences), and the state-space of models can be huge. Therefore, those analyzes must be probabilistic (see e.g. Henley and Kumamoto (1981)). Moreover, mechatronics systems are composed of a physical and a control part. Nevertheless classical safety analysis focus on the physical part. Control part assessment is part of the software assessment domain. But dysfunctional behavior of such systems is the result of the combination of their software and their hardware. Therefore, as already discussed (e.g. Piriou (2015)), a safety analysis of such systems must take into account both parts.

The preferred mathematical representation for those analysis are stochastic timed automata. They are subject of several work in progress. Stochastic Discreet Event Systems (Zimmermann, 2007) or SAML (Güdemann et al., 2012) are based on such mathematical basis. Several tools are available to assess those kinds of models (e.g. PRISM (Kwiatkowska et al., 2011), COSMOS (Ballarini et al., 2015), UPPAAL (Bengtsson et al., 1996)). But those tools are for software assessment, not safety analyses.

The AltaRica project started at the end of the 90's. AltaRica 3.0 is the third version of the language. It focuses on computation efficiency, and usability (reuse and readability of models) and is intended for industrial use. It is designed for safety analyses, but it may be expressive enough for the control part to be modeled along the physical part.

3. CASE STUDY

In this section, we give a partial description of the "landing gear" case study, as well as some elements about the AltaRica 3.0 model we designed. The reader interested by a full description of the case study should refer to Boniol and Wiels presentation in Boniol et al. (2014).

3.1 Architecture of the system

The landing gear system is made of three parts: a mechanical part, a controller and a pilot interface, see Fig. 1. Each part is further decomposed. In total, the system involves about 72 interacting components.

The structure of the AltaRica 3.0 model reflects the physical architecture of the system. AltaRica 3.0 provides structuring constructs stemmed from both object-oriented and prototype-oriented paradigms (see e.g. Batteux et al. (2015)). Prototypes are used to represent components with a unique occurrence in the system. Classes and instances of classes are used to describe components with several occurrences or that can be re-used from a model to another one via libraries of modeling components. As shown in Fig. 2, the pilot interface and the analogical switch are modeled using a prototype, while every electro-valve are instances of the ElectroValve class.

3.2 Description of the behavior

The functional behavior of the system is conveniently described by means of communicating timed automata. In order to perform our safety analysis, we had to make several hypotheses about failure modes of components and

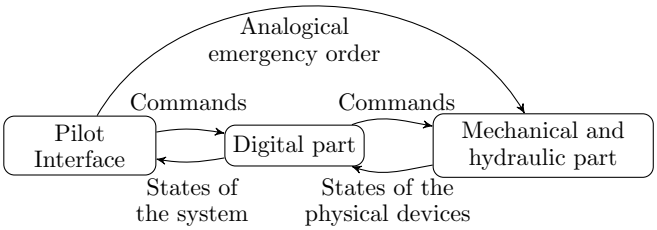

Fig. 1. Architecture of the Landing Gear System

- LandingSystem
  - PilotInterface
    - Handle
    - Lights
  - Control
    - ComputingModule1: *ComputingModule*
    - ComputingModule2: *ComputingModule*
  - Mechanical
    - AnalogicalSwitch
    - Hydraulic
      - ElectroValveGeneral: *ElectroValve*
      - ElectroValveCloseDoors: *ElectroValve*
      - ElectroValveOpenDoors: *ElectroValve*
      - ElectroValveExtendGears: *ElectroValve*
      - ElectroValveRetractGears: *ElectroValve*
    - FrontSet: *Landingset*
    - LeftSet: *Landingset*
    - RightSet: *Landingset*

Fig. 2. AltaRica 3.0 model partial structure (italic are class)

about probabilities of these failure modes, which are not available in Boniol et al. (2014).

First, we assumed that failures are permanent. Second, we split components into two categories: components failing on demand (with a given probability), and components failing stochastically (with a constant failure rate). In both cases, the values we took (see Table 1) are realistic but not real.

Table 1. Failure modes of components

Component	Failure type	Rate/Probability
Analogical Switch	on-demand	10^{-6}
Electro-Valve	on-demand	10^{-4}
Cylinder	constant rate	$10^{-5}h^{-1}$
Sensor	constant rate	$10^{-4}h^{-1}$
Computing Module	constant rate	$10^{-6}h^{-1}$

The semantic of an AltaRica 3.0 model is defined in terms of Guarded Transition Systems (GTS) (see e.g. Prosvirnova (2014)). GTS are automata whose states are described by variables. Variables may be Boolean, integer, floating point numbers, or members of finite sets of symbolic constants. Transitions between states are composed by a guard, an instruction, and an event with a stochastic or deterministic delay. Flow variables are used to propagate data between components. Their values are computed after each firing of a transition, using assertions.

As shown in the remainder of this paper, functional behaviors and various failures modes with their effects can be described using AltaRica 3.0 .

3.3 Pilot interface part

The pilot interface is made of a Up/Down handle. This handle is used to launch the retracting and the extending landing gear sequences. The handle behavior is similar to a memory: it remains in the state it has been set. It may therefore be modeled as an AltaRica 3.0 state variable.

A set of lights reports the current positions of doors and gears as well as the current health of equipments. The behavior is to only display a received data, no information is stored: they may be represented by means of flow variables.

3.4 Control part

The control part is made of two identical computing modules executing in parallel the same software. This software is in charge of controlling gears and doors, detecting anomalies and informing the pilot. In Boniol et al. (2014), the software is not specified, because this definition may be part of the study. There is a partial specification of the required behavior, with operating sequences and timing constraints: the definition used in this modeling is described section 4.

Each computing modules receive 54 discrete (or Boolean) input values and emits 8 boolean electrical outputs (3 towards the pilot interface and 5 towards the mechanical part). For the AltaRica 3.0 model, the two computing modules are part of the Control block (Fig. 3). This block distributes inputs to each computing modules, and aggregates outputs from them.

Inputs Control part inputs are directly linked to the computing modules inputs. Inputs are the Boolean values of the sensors: position of the handle of the pilot interface, and sensors of the mechanical part. Each sensor is triplicated: for each position, 3 inputs have to be taken into account. To ease the model, those 3 inputs are grouped in a record.

Outputs Outputs of the computing modules are electrical data to the pilot interfaces (three lights), and electrical orders to the hydraulic electro-valves. They may be contradictory: each one of the two computing modules may give a different output. The outputs of the control part are the composition of the outputs of the computing modules, as an electrical OR: if one of them is true, then the order is true.

3.5 Mechanical part

The mechanical part contains the landing sets and the hydraulic circuit.

Landing sets The mechanical part is made of three identical landing sets (front, left and right). Each landing set is made of a door and a landing gear. A cylinder opens and closes the door. Two latching boxes lock it in closed

```
block Control
  ComputingModule Module1;
  ComputingModule Module2;
  /* Orders to PilotInterface */
  Boolean vfOutLockedDown( reset = true );
  Boolean vfOutManeuvering ( reset = false );
  Boolean vfOutPhysicalFailure ( reset = false );
  /* Orders to hydraulic */
  Boolean vfOutEVGeneral (reset = false);
  Boolean vfOutEVCloseDoors (reset = false);
  Boolean vfOutEVOpenDoors (reset = false);
  Boolean vfOutEVRetractGears (reset = false);
  Boolean vfOutEVExtendGears (reset = false);
  /* From Left Gear sensors */
  ThreeValues vfInLeftDoorOpen (reset = false);
  [...]
  assertion
    vfOutLockedDown := Module1.vfOutLockedDown
                    or Module2.vfOutLockedDown;
  [...]
end
```

Fig. 3. Partial AltaRica 3.0 model of the control part

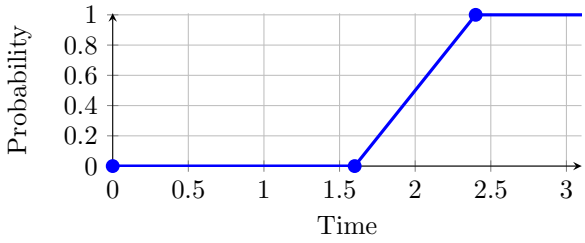

Fig. 4. Cumulative Probability Distribution for the front gear down to high movement duration

position. A cylinder extends and retracts the landing gear. Two latching boxes lock it in high and down position.

The extension sequence is composed of several step: unlocking and opening the door, unlocking, extending, and locking the gear, and closing and locking the door. The retraction sequence is similar. Each step is defined with a duration, which depends on the landing set (front, left or right). As this is a physical process, the duration is defined with a 20 percent variation.

To model this behavior, these cylinders are described in the AltaRica 3.0 model as automata, with user-defined delays for transitions. User-defined delays are Cumulative Probability Distribution described with a set of points (Batteux and Rauzy, 2013). With this mechanism, the delay will be stochastically determined, in the range of 20 percent around the mean value (Fig. 4).

Hydraulic Circuit Electro-valves set the pressure in various portion of the hydraulic circuit. There is one general electro-valve, and 4 others: one for each movement (extension/retraction of the landing gears, and opening/closing of the doors). Each movement of the three landing sets depends on the same electro-valve. They are activated by electrical orders coming from the control part. An analogical switch prevent any electrical order to the general electro-valve if there was no new order from the pilot. A set of triplicated discrete end-position sensors inform the control part about the state of each cylinder.

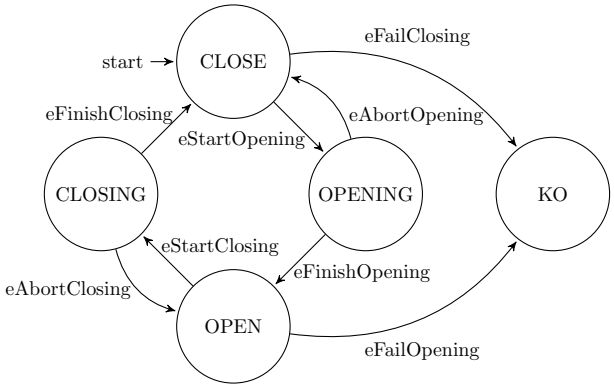

Fig. 5. Electro-valve automata

A time is needed to open and close the electro-valve. Each movement has a probability to fail (as defined in 3.2). Fig. 5 show the automata representing the behavior of the electro-valve. The corresponding AltaRica 3.0 model is showed Fig. 6. This class is then instantiated in the AltaRica 3.0 model for each one of the 5 electro-valves.

4. MODELING THE CONTROL

The system presented in Boniol et al. (2014) specify constraints the software running onto the two computing modules must meet.

As this is a mechatronic system, its behavior is the result of both its mechanical part and its control part. Therefore a model of the control part is needed in order to be able to assess every requirement.

To perform our study, we designed a model of the software embedded onto the two computing modules. This model is of course an abstraction of the software that would actually run. Nevertheless, it respects fully the specifications. The model of the software is divided in two parts: control and monitoring.

4.1 Control

The control part controls the actuators (electro-valves) according to the values given by sensors and to the commands of the pilot. It is represented by the guarded transition system graphically described Fig. 7. For the sake of clarity, transitions corresponding to a counter-order (the handle going from DOWN to UP, or from UP to DOWN, during an ongoing sequence) are not totally described and are dotted.

4.2 Monitoring

The monitoring part reports to the pilot about the current state of the system. It uses the three lights of the Pilot Interface in order to provide these data. The red light must be on in case of anomaly. Sensors anomalies can be easily detected: each sensor is triplicated. If, at some point, one sensor has a different value than the two others, it is definitively discarded. If the two remaining sensors have different values, then there is an anomaly. Some orders have to be executed within a time interval. For example, the hydraulic circuit must be pressurized 2

```
domain ElectroValveState { CLOSE, OPEN,
                           CLOSING, OPENING }
class ElectroValve
  Boolean vsKO ( init = false );
  ElectroValveState vsState (init = CLOSE);
  //Input, Output and Command
  Boolean vfIn, vfOut, vfE (reset = false);
  //Time to Open and close
  parameter Real pOpening = 1.0;
  parameter Real pClosing = 3.6;
  event eAbortOpening (delay = Dirac(0));
  event eAbortClosing (delay = Dirac(0));
  event eStartClosing (delay = Dirac(0),
                       expectation = 0.9999 );
  event eStartOpening (delay = Dirac(0),
                       expectation = 0.9999 );
  event eFailClosing (delay = Dirac(0),
                      expectation = 1.0e-4 );
  event eFailOpening (delay = Dirac(0),
                      expectation = 1.0e-4 );
  event eFinishClosing (delay = Dirac(pClosing));
  event eFinishOpening(delay = Dirac(pOpening));
  transition
    eFailClosing: not vsKO
              and vsState == OPEN
              and vfE == false
              -> vsKO := true;
    eFailOpening: not vsKO
              and vsState == CLOSE
              and vfE
              -> vsKO := true;

    eStartClosing: not vsKO
              and vsState == OPEN
              and vfE == false
              -> vsState := CLOSING;
    eFinishClosing: not vsKO
              and vsState == CLOSING
              -> vsState := CLOSE;
    eStartOpening: not vsKO
              and vsState == CLOSE
              and vfE
              -> vsState := OPENING;
    eFinishOpening: not vsKO
              and vsState == OPENING
              -> vsState := OPEN;
    eAbortOpening: not vsKO
              and vsState == OPENING
              and vfE == false
              -> vsState := CLOSE;
    eAbortClosing: not vsKO
              and vsState == CLOSING
              and vfE
              -> vsState := OPEN;
  assertion
    if vsState != CLOSE then vfOut :=: vfIn;
end
```

Fig. 6. AltaRica 3.0 modeling of the electro-valves class

seconds after the general electro-valve opening order,and not be pressurized 10 seconds after the closing order. If this time interval is not respected, then there is an anomaly. Anomaly detection have been incorporated into the model (Fig. 8).

The green light (gears are locked down) and the orange light (system is in motion) are commanded by the control part. Any detected anomaly must be reported to the pilot using the red light. These commands can easily be

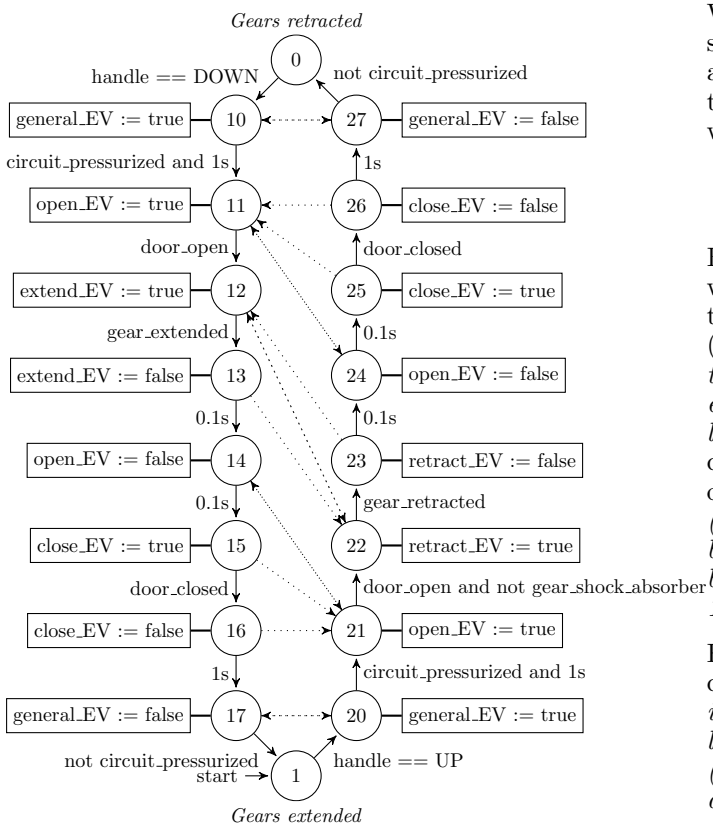

Fig. 7. Sequence control automata

```
Boolean vsHydraulicMustBePressurized
                (init = false);
event eHydraulicMustBePressurized
                (delay = Dirac(0));
event eHydraulicMustNotBePressurized
                (delay = Dirac(0));
event eHydraulicTooLongToBePressurized
                (delay = Dirac(2));
event eHydraulicTooLongToNotBePressurized
                (delay = Dirac(10));
transition
 eHydraulicMustBePressurized:
     vfOutEVGeneral and
     not vsHydraulicMustBePressurized
  -> vsHydraulicMustBePressurized := true;
 eHydraulicMustNotBePressurized:
     not vfOutEVGeneral and
     vsHydraulicMustBePressurized
  -> vsHydraulicMustBePressurized := false;
 eHydraulicTooLongToBePressurized:
     vsHydraulicMustBePressurized and
     not vfInPressure.vfOut and
     not vsAnomaly
  -> vsAnomaly := true;
 eHydraulicTooLongToNotBePressurized:
     not vsHydraulicMustBePressurized and
     vfInPressure.vfOut and
     not vsAnomaly
  -> vsAnomaly := true;
```

Fig. 8. AltaRica 3.0 monitoring of the hydraulic pressure

described in the AltaRica 3.0 model by using flow variables and assertions on the software state.

We assumed that each computing module is watching the state of the other one. If the latter does not answer, an anomaly is reported to the pilot by the red light. We added this classical (in the aeronautic domain) feature because we assumed that the computing modules can also fail.

5. ANALYSIS

Boniol and Wiels article provides also a set of requirements which are of two different types. First, reachability properties which involve only the current state of the system e.g. (R_{31}) *When the command line is working (normal mode), the stimulation of the gears outgoing or the retraction electro-valves can only happen when the three doors are locked down.* Second, timeout properties that require a certain action to be achieved within a limited amount of time, e.g. (R_{11}) *When the command line is working (normal mode), if the landing gear command handle has been pushed DOWN and stays DOWN, then the gears will be locked down and the doors will be seen closed less than 15 seconds after the handle has been pushed.*

For this study, we added the following requirement which corresponds to a catastrophic event: (R_a) *When the pilot interface indicates that the aircraft is ready to land (green light on) without any problem (red light off) or movement (orange light off), then every gear is locked down and every door is locked.*

Reachability properties, like (R_{31}), can be expressed easily as a Boolean expression over variables encoding the state of the system. Therefore, they can be encoded by means of AltaRica 3.0 observer. For the requirement to be checked, it should never be found false.

Timeout properties are described by means of a guarded transition system describing the behavior of a timeout.

5.1 Experiments

To complete the model, we added a representation of the pilot. The virtual pilot commands the handle of the Pilot Interface to perform cycles of take-off and landing, every two hours (except in case an anomaly is detected). It allows to simulate the model for several cycles, and to stop when there is an anomaly detected.

The complete AltaRica 3.0 model is made of 129 components (instances of classes and blocks), 997 variables (state and flows), and 504 transitions. The flattened model is made of about 14000 lines. Finally, the size of the state space (disregarding timing issues) is about 2.6×10^{105}.

2×10^8 histories of 10 hours have been generated by stochastic simulation. The stochastic simulator ran on 4 parallel processes to divide execution times. The computation takes about 17 hours on a 4 cores Intel Xeon E312xx (Sandy Bridge) at 2.6GHz with 8 GB RAM.

5.2 Experimental results

All requirements defined in Boniol et al. (2014) have been taken into account. Results are showed in Table 5.2. Three of them are not satisfied. Each other is satisfied: meaning that it has never been found false for all 2×10^8 histories.

It does not mean that it will always be true. In fact, the probability for an history to found such a requirement as false is included in the confidence interval at 95%: $[0; 1.5 \times 10^{-9}]$ (according to Hanley and Lippman-Hand (1983)). One can notice that in the aeronautic domain, 10^{-9} is the bound to reach for critical aspects. It shows that stochastic simulation can be used to obtain a useful estimation of a probabilistic indicator, without the cost of a complete state-space exploration (as in conventional model-checking).

Table 2. Results on requirements

Req.	Type	Checked	Req.	Type	Checked
(R_a)	Observer	100%	(R_{61})	TimeOut	100%
(R_{11})	TimeOut	99.9999945%	(R_{62})	TimeOut	100%
(R_{12})	TimeOut	99.9999940%	(R_{63})	TimeOut	100%
(R_{21})	Observer	100%	(R_{64})	TimeOut	100%
(R_{22})	Observer	100%	(R_{71})	TimeOut	100%
(R_{31})	Observer	100%	(R_{72})	TimeOut	100%
(R_{32})	Observer	**0%**	(R_{73})	TimeOut	100%
(R_{41})	Observer	100%	(R_{74})	TimeOut	100%
(R_{42})	Observer	100%	(R_{81})	TimeOut	100%
(R_{51})	Observer	100%	(R_{82})	TimeOut	100%

5.3 Reliability

The results of this simulation indicate that 0.78% of the generated histories have a problematic failure (a physical component has failed, or 2 sensors from the same group have failed). Although this reliability indicator is not acceptable (according to aeronautic standards), each ones of those problematic failures have been reported to the pilot as an anomaly (because (R_a) was never found false).

6. CONCLUSION

In this article, we reported the results of a probabilistic safety analysis we made on the case study "Landing Gear" proposed recently by Boniol and Wiels with AltaRica 3.0 and Monte-Carlo simulation. This study was especially interesting because we had to take into account both software behavior, random failures of hardware, and detection of those failures by the software.

Of course, Monte-Carlo simulation is by no means sufficient to prove formally properties of the software. But we showed how we were able to check a significant set of properties of the system (hardware + software). Furthermore, stochastic simulation performances are relatively independent to the state-space size, unlike conventional model-checking. This experience outlines a methodology to assess the safety of mechatronics systems in presence of hazards, failures and uncertainties.

REFERENCES

ARP4754A (2010). *Guidelines for Development of Civil Aircraft and Systems*. Society of Automotive Engineers, Warrendale, Pennsylvania, USA.

ARP4761A (2004). *Guidelines and methods for conducting the safety assessment process on civil airborne systems and equipment*. Society of Automotive Engineers, Warrendale, Pennsylvania, USA.

Ballarini, P., Barbot, B., Duflot, M., Haddad, S., and Pekergin, N. (2015). HASL: A new approach for performance evaluation and model checking from concepts to experimentation. *Performance Evaluation*, 90, 53–77.

Batteux, M. and Rauzy, A. (2013). Stochastic simulation of altarica 3.0 models. In *Proceedings of the European Safety and Reliability Conference, ESREL 2013*. CRC Press, Amsterdam (The Netherlands).

Batteux, M., Prosvirnova, T., and Rauzy, A. (2015). *System Structure Modeling Language (S2ML)*. AltaRica Association. Archive hal-01234903, version 1.

Bengtsson, J., Larsen, K., Larsson, F., Pettersson, P., and Yi, W. (1996). *UPPAAL — a tool suite for automatic verification of real-time systems*, 232–243. Springer Berlin Heidelberg. doi:10.1007/BFb0020949.

Boniol, F., Wiels, V., Ameur, Y.A., and Schewe, K.D. (eds.) (2014). *ABZ 2014: The Landing Gear Case Study*, volume 433 of *Communications in Computer and Information Science*. Cham, Switzerland.

Güdemann, M., Lipaczewski, M., Struck, S., and Ortmeier, F. (2012). Unifying probabilistic and traditional formal model-based analysis. In *Proceedings of 8. Dagstuhl-Workshop on Model-Based Development of Embedded Systems (MBEES)*.

Hanley, J. and Lippman-Hand, A. (1983). If nothing goes wrong, is everything all right?: Interpreting zero numerators. *JAMA*, 249(13), 1743–1745.

Henley, E. and Kumamoto, H. (1981). *Reliability engineering and risk assessment*. Prentice-Hall.

Kwiatkowska, M., Norman, G., and Parker, D. (2011). PRISM 4.0: Verification of probabilistic real-time systems. In G. Gopalakrishnan and S. Qadeer (eds.), *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, volume 6806 of *LNCS*, 585–591.

Noble, J., Taivalsaari, A., and Moore, I. (1999). *Prototype-Based Programming: Concepts, Languages and Applications*. Springer-Verlag, Berlin and Heidelberg, Germany.

Piriou, P.Y. (2015). *Contribution à l'analyse de sûreté de fonctionnement basée sur les modèles des systèmes dynamiques, réparables et reconfigurables*. Ph.D. thesis, ENS Cachan.

Prosvirnova, T. (2014). *AltaRica 3.0: a Model-Based Approach for Safety Analyses*. Thèse de doctorat, École Polytechnique, Palaiseau, France.

Prosvirnova, T., Batteux, M., Brameret, P.A., Cherfi, A., Friedlhuber, T., Roussel, J.M., and Rauzy, A. (2013). The altarica 3.0 project for model-based safety assessment. In *Proceedings of 4th IFAC Workshop on Dependable Control of Discrete Systems, DCDS'2013*, 127–132. International Federation of Automatic Control, York, Great Britain.

Rausand, M. and Høyland, A. (2004). *System Reliability Theory: Models, Statistical Methods, and Applications, 2nd Edition*. Wiley-Blackwell, Hoboken, New Jersey.

Rauzy, A. (2008). Guarded transition systems: a new states/events formalism for reliability studies. *Journal of Risk and Reliability*, 222(4), 495–505.

Valiant, L.G. (1979). The complexity of enumeration and reliability problems. *SIAM Journal of Computing*, 8(3).

Zimmermann, A. (2007). *Stochastic Discrete Event Systems: Modeling, Evaluation, Applications*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Zio, E. (2013). *The Monte Carlo Simulation Method for System Reliability and Risk Analysis*. Springer Series in Reliability Engineering. Springer London.