Advanced Multi-System Simulation Capabilities with AltaRica

C. Kehren; ONERA; Toulouse, France
C. Seguin; ONERA; Toulouse, France
P. Bieber; ONERA; Toulouse, France
C. Castel; ONERA; Toulouse, France
C. Bougnol; AIRBUS; Toulouse, France
J.-P. Heckmann; AIRBUS; Toulouse, France
S. Metge; AIRBUS; Toulouse, France

Keywords: safety, modeling, requirements, simulation, model checking

Abstract

Recently, AIRBUS and ONERA were involved in the ESACS (Enhanced Safety Assessment for Complex Systems) European project. The aim of this project was to investigate new safety assessment techniques based on the use of formal design languages and associated tools. Two case-studies based on AIRBUS aircraft were used to validate the approach. Both a complete hydraulic system and an electric system were modelled. We also built a model depicting the two systems and their interconnections and performed a safety analysis focusing on failure propagation.

In this paper, we report how the combination of these two medium sized models was assessed and analysed with the AltaRica language. With respect to analysis, we explain how we used Cecilia OCAS, developed by Dassault Aviation, a French aircraft manufacturer. Simulation was first performed interactively with graphical views of the system that help to understand precisely how failures propagate inside a system as well as between systems. Then we used a model checker that performs symbolically an exhaustive simulation of the system. As a main result, we found out that these tools and the underlying safety approach were very efficient to assess whether qualitative safety requirements are fulfilled by a system design or not.

Introduction

During the last three years, AIRBUS and ONERA were involved in the ESACS (Enhanced Safety Assessment for Complex Systems) European project. This project aimed at developing safety assessment techniques based on the use of formal specification languages and associated tools. So called formal models are traditionally used to specify the expected normal behaviors of software based system. ESACS partners investigated first how to generalize such models to deal with faulty behaviors of various kinds of systems. Then they proposed new tools or new uses of existing tools to check whether the generalized formal models met qualitative safety requirements. These tools provide not only interactive simulation capabilities but also take advantage of formal language features to support advanced capabilities such as model-checking or fault tree generation. The approach was validated on some existing aircraft systems. In this paper we present how AIRBUS France and ONERA tackled modeling and assessment of two aircraft systems using the AltaRica language and a subset of the associated tools.

ESACS approach (ref. 1) raised two main issues. The first issue is to get formal system models meaningful for safety analysis? ESACS partners are interested in failure propagations in complex dynamic systems. So they consider formal notations for reactive systems, used to support system design such as Statechart (models are automata), Scade (models are equations between synchronous data flows) or dedicated to safety such as AltaRica (models mixing automata and equation concepts). To cope with failure propagations, system models can either be produced by system designers and then extended with failure modes specified by safety engineers, or can directly be produced by safety specialists using libraries. It is worth noting that formal models of failure propagations should have the correct granularity level to ease model exploitation. On one hand, advanced simulation capabilities have good performances when the analyzed model does not go into detailed arithmetic computations. On the other hand, a correct granularity is reached when the scenarios, leading to a failure condition, extracted by the tools are similar to what safety analysts would have envisioned if they had to design a fault tree. In order to get the appropriate granularity at first shot, we chose to define libraries of AltaRica components that focus on failure mode propagation and abstract details of nominal behaviors.

The second issue is related to the choice of the adequate techniques for assessing qualitative safety requirement of complex dynamic systems. Interactive simulation facilities enable to perform a preliminary bottom up analysis since failures can be injected and their effects computed not only locally but at system or even aircraft level. This will be detailed later on. Top down analyses are guided by qualitative requirements such as "no single failure leads to the system loss". We propose to use model-checkers to assess such kind of requirements. They perform "exhaustive" simulation to check whether a requirement is always met. Moreover, they can distinguish subtle temporal situations such as a transient loss of a function (during a recovery phase for instance) from a permanent one. Two case-studies based on AIRBUS aircraft were used to validate the approach. Both a complete hydraulic system and an electric system were modeled and assessed. Then, another system depicting the two systems and their interconnections was built and analyzed.

The paper has the following structure. First section describes the studied aircraft systems and focuses on their safety requirements and architecture. Section 2 introduces the AltaRica language through examples. We explain the modeling philosophy used to build the hydraulic and electric libraries at a satisfying granularity level. Section 3 deals with the benefit of advanced simulation capabilities to assess qualitative safety requirements on dynamic models. We show how the models were analyzed using interactive simulation facilities of Cecilia OCAS and SMV (Symbolic Model Verifier) model-checker. The following section is dedicated to problems related with the combination of systems regarding modeling and safety assessment point of view. We illustrate our proposals with the combination of the hydraulic and electric systems. Last section presents a conclusion of our work and the lessons learnt so far.

Case-studies Presentation

Both systems studied in this paper produce and provide the aircraft systems with power. The role of the hydraulic system is to supply hydraulic power to devices which ensure aircraft control in flight like the flaps, slats, or spoilers as well as devices which are used on ground like the braking system. The role of the electrical system is to deliver AC/DC power to all loads of the aircraft such as displays, motors, or computers. As the loss of devices powered by these systems could lead to the loss of aircraft control, both systems share the same main safety requirement: total loss of (hydraulic or electrical) power is considered to be catastrophic. The probability of occurrence of this failure condition should be smaller than 10^{-9} per flight hour and no single failure should lead to this failure condition.

Hydraulic Generation and Distribution System: The hydraulic system is mainly composed of three independent sub-systems which generate and transmit the hydraulic power to the consumers. Three kinds of pumps were used in the model of an A320-like hydraulic system. The first one is the Electric Motor Pump (EMP) which is powered by the electric system, the second one is the Engine Driven Pump (EDP) that is powered by one of the two aircraft engines and the last one is the RAT pump that is powered by the Ram Air Turbine. The hydraulic system also contains other types of components such as tanks, valves and gauges.

To meet its main safety requirement, the system is constituted of three channels: Green, Blue and Yellow. The Blue channel is made of one electric pump EMPb, one RAT pump and two distribution lines: prioritary (Pdistb) and non-prioritary (NPdistb). When priority valve PVb is closed consumers connected to Npdistb do not receive hydraulic power. The Green system is made of one pump driven by engine 1 EDPg and two distribution lines Pdistg and NPdistg. The Yellow system is made of one pump driven by engine 2 EDPy, one electric pump EMPy and two distribution lines Pdisty and NPdisty. Moreover a reversible Power Transfer Unit (PTU) transmits pressure between green and yellow channels as soon as the differential pressure between both channels exceeds a given threshold.

These components are controlled by crew actions and reconfiguration logics. The RAT is automatically activated in flight when both engines are lost. The EMPb is automatically activated when the aircraft is in flight or on ground when one engine is running. EMPy is activated by the pilot on ground. We assumed that EDPy, EDPg were activated whenever the corresponding engine was started.

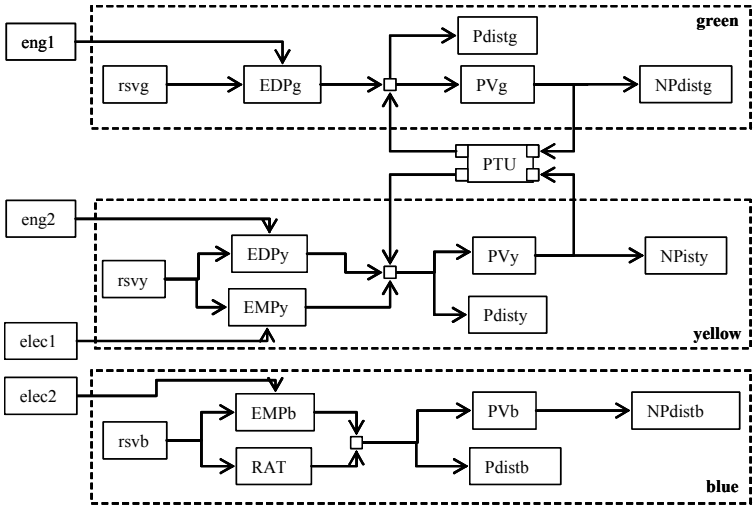

Figure 1 − Hydraulic System Architecture

Electrical Generation and Distribution System: The electrical system includes generators, bus bars, contactors, circuit breakers, Transformers/Rectifiers Units (TRU) and junctions. To meet its main safety requirements the electrical system of an A320-like aircraft is organized in two sub-systems: nominal electrical system and emergency system. The nominal system is composed of 2 main generators GEN1 powered by engine 1, GEN 2 powered by engine 2 and an auxiliary power unit APU. The emergency system is composed of an emergency generator CSM_G powered by the Ram Air Turbine, automatically deployed in case of main generators loss. Electricity is supplied to the electrical loads through four distribution bus bars in the normal system: ACside1, DCside1, ACside2, DCside2 and two essential bus bars in the emergency system: ACess and DCess. The AC to DC conversion is performed by transformers TR1, TR2 and TRess. The system contains several circuit breakers to limit the propagation of short circuits.

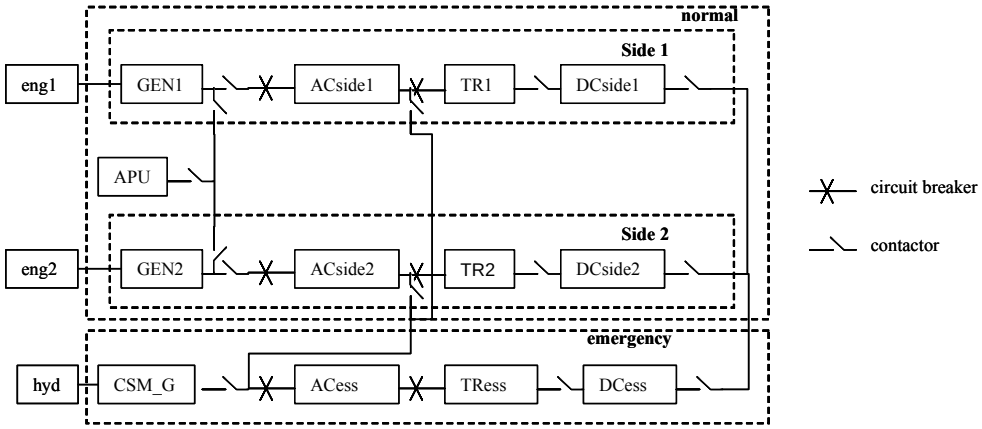

Figure 2 − Electrical System Architecture

Contactors are controlled in order to implement various reconfigurations. For instance, all nominal generators (GEN1, GEN2 or APU) can be used to provide electricity to any distribution bus bar in the normal system or in the emergency system when one or two generators are not available. Other reconfiguration rules apply to the loss of transformers TR1 and TR2. When all nominal generators are lost, the emergency generator only provides power to essential bus bars.

System Modeling in AltaRica

The AltaRica Language: AltaRica (ref. 2) is a formal language developed at LaBRI (Laboratoire Bordelais de Recherche en Informatique) for modeling both functional and dysfunctional behaviours of systems. Thanks to the

language well defined semantics and syntax, safety assessments of AltaRica models can be analysed by numerous reliability or validation tools. Moreover, its capacity to realise compositional and hierarchical models is a great advantage when complex systems must be modelled. The development of AltaRica models is supported by Cecilia OCAS workshop (ref. 3) of Dassault Aviation that provides graphical edition and simulation facilities and integrates fault tree generators. We will now briefly describe this language.
Each system component is modelled by a "node". A node is a mode automaton (ref. 4) defined by three well identified parts. First part is the declaration of the different kinds of node parameters: state, flow and event. States are internal variables which memorize current functioning modes (failure modes or normal ones). Flows are node inputs or outputs. Possible types of states and flows are integer interval, enumeration and boolean. Events are phenomena, which trigger transitions from an internal state to another. They can model pilot actions or the occurrence of failures, or reactions to input conditions (the key word "no_event" is used in this case). This particular event plays a significant role when modeling the impact of cascading failures in the system.

The second part describes the automaton transitions. A transition is a tuple `g |- evt -> e` where `g` is the guard of the transition, `evt` is an event name and `e` is the effect of the transition. The guard is a boolean formula over state or flow variables. It defines the configuration in which the transition is fireable if the event `evt` occurs. The effect `e` is a list of assignations of value to state variables. So the transition part describes how functioning or failure states can evolve.

The third part is a set of assertions. Assertions are atomic equalities or more structured equations using `if-then-else` or `case` construction. They establish relations between the states and the flows of the component and so, describe how component outputs are determined by component inputs and current functioning mode.

These concepts are illustrated by the following example. The component `block` has one `input`, one `output` flow ranging over the domain `{no, low, ok, max}`, one boolean internal state `ok` and one `failure` event. The transition means "if the system is `ok` and if `failure` occurs then the system is no more `ok`". The assertion means "if the system is `ok` then the `output` is equal to the `input` else the `output` value is `no`". We used here the `case` structure but we could use similarly an `if then else` structure.

```
node block
 state
  ok : boolean;
 flow
  input : {no, low, ok, max} : in;
  output :{no, low, ok, max} : out;
 event
  failure;
 trans
  ok |- failure -> ok:= false;
 assert
  output = case {ok : input,
  else : no};
edon
```

In a system model, instances of such nodes are interconnected by assertions which plug input-output flows. Hierarchy of nodes can be used to build complex components and structure the system model.

Case-Studies Modeling: The main step prior to model a system is to collect information on it (e.g. architecture, failure modes). We particularly paid attention to Airbus Functional Hazard Assessment document performed on aircraft functions that describes the failure conditions, effects and severity levels (i.e. catastrophic, hazardous, major or minor) and to the System Safety Assessment which demonstrates that safety objectives are met. In this section we describe how to model a system using these documents as inputs and use the example of the hydraulic pipe as an illustration.

Failure Modes: For the electrical and hydraulic systems, failure modes that could cause the loss of energy supply (voltage or hydraulic power) were modeled. We considered that all components could fail to generate, transmit or deliver energy. We also supposed that short-circuits could occur in bus bars, whereas leaks could occur in pipes.

Finally, blocked positions for valves and contactors were also considered. Table 1, hereunder, sums up the failure modes considered in our component libraries.

Part of Electrical System Library		**Part of Hydraulic System Library**	
Component	***Failure Modes***	**Component**	***Failure Modes***
Wire	*none*	pipe	*leakage*
Contactor	*failed, stucked*	reservoir	*failed, leakage*
circuit breaker	*failed, stucked*	pump	*failed, SC, overheat*
transformer	*failed, SC*	valve	*failed, stucked*
consumer	*failed, SC*	consumer	*failed, leakage*
bus bar	*failed*	PTU	*failed, stucked*
generator	*failed, SC*		

Table 1 − Failure Modes

Failure Propagation: We have to know what kind of information are exchanged between the components and how failures will be propagated through pipes or wires. This information is really specific to each system. If we consider a hydraulic circuit pipe we cannot model a leakage only by considering the absence or the presence of fluid in the pipe. Indeed, the real consequence of a leakage is a sudden pressure decrease for all the components located downwards the faulty component and, at last, a lack of fluid in the circuit. As a result, a pipe must transmit the couple fluid/pressure in order to take into account and to correctly propagate the leakage information throughout the model. Moreover, as all the components (i.e. downwards but also upwards) have to be informed of such a failure, the fluid/pressure signal has to be bidirectional. Of course we have exactly the same worries concerning the electrical system with the short-circuit propagation and we decided to use the same modeling approach.

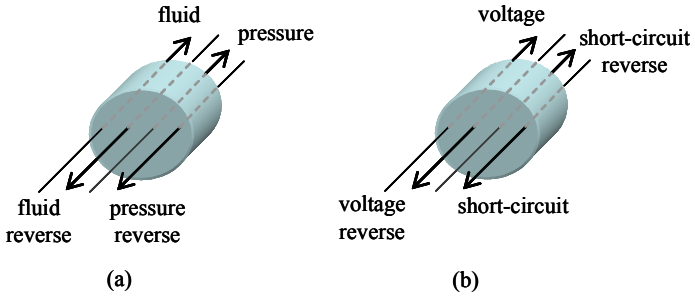

Figure 3 − Pipe (a) and Wire (b) Sections

In the component models, failure propagation will be modelled by assertions that constrain the values of flow variables. The following example shows in details the failure propagation in the pipe model.

Example:

```
node pipe
 flow
  output_pressure : {max,ok,low,no} : out;
  output_fluid : {yes,low,no} : out;
  output_pressure_reverse_info : {max,ok,low,no} : in;
  output_fluid_reverse_info : {yes,low,no} : in;
  input_pressure : {max,ok,low,no} : in;
  input_fluid : {yes,low,no} : in;
  input_pressure_reverse_info : {max,ok,low,no} : out;
  input_fluid_reverse_info : {yes,low,no} : out;
 state
  state_ : {ok,leakage};
 event
  leak;
 trans
  (state_ = ok) |- leak -> state_ := leakage;
 assert
```

```
 output_fluid = (case {
 (state_ = ok) : input_fluid,
 (state_ = leakage) and ((input_fluid = yes) or (input_fluid = low)) : low,
 else no}),
 input_fluid_reverse_info = (case {
 (state_ = ok) : output_fluid_reverse_info,
 (state_ = leakage) and ((input_fluid = yes) or (input_fluid = low)) : low,
 else no}),
 output_pressure = (case {
 (state_ = ok) and not((input_fluid = no)) : input_pressure,
 else no}),
 input_pressure_reverse_info = (case {
 (state_ = ok) and not((input_fluid = no)) : output_pressure_reverse_info,
 else max});
init
 state_ := ok;
edon
```

When a pipe is not leaking, output pressures and output fluid levels are equal to input ones. When a leak occur, the fluid level decreases from `yes` to `low` until the reservoir is empty (the input fluid level is `no`). Moreover, while the pipe is not empty (fluid different from `no`), the leak increases the upwards pressure and decreases the downwards one.

In Cecilia OCAS workshop, each node is associated to an icon and belongs to a library. Once the component library created, the system is easily and quickly modelled. Components are dragged and dropped from the library to the system architecture sheet and then linked graphically. The whole hydraulic system model is made of about 15 component classes and the electrical system model uses about 20 classes of components.

Safety Assessment Techniques

Formal Safety Requirements: As stated in the case-study presentation section, the main safety requirement for the systems under study is: "*Total loss of hydraulic (or Electrical) power is classified catastrophic*". We also considered two related requirements: "*Loss of two hydraulic channels (or two bus bars) is classified major*", "*Loss of one hydraulic channel (or one bus bar) is classified minor*". We associate with this set of safety requirements six qualitative requirements of the form "*if up to N individual failures occur then the loss of N+1 power channels of system S shall not occur*" with N = 0,1,2 and S = Electrical or Hydraulic.

To model these qualitative requirements we first have to model the loss of N+1 power channels. Let N = 2 and S = Hydraulic, so we consider the total loss of hydraulic power. A first approach consists in using propositional formula 3_Hyd_Loss that would be true whenever the value of flow output_pressure of the distribution lines of the three hydraulic channels is equal to no**.** But this formula fails to adequately describe the failure condition. It could hold in evolutions of the system during a small period of time and then it would no longer hold as the hydraulic power is recovered due to appropriate activation of a backup such as the RAT for instance. The correct description of the failure condition should model the fact that hydraulic power is definitively lost. Hence we use Linear Temporal Logic (ref. 5) operators to model a failure condition. The following temporal formula models the permanent loss of hydraulic power:

```
Permanent_3_Hyd_Loss: F G 3_Hyd_Loss
```

where F is the eventually (or Finally) operator, G is the always (or Globally) operator. Formula Permanent_3_Hyd_Loss can be read "*eventually Hydraulic power is totally lost in all future time steps*". So the general form of qualitative requirements we check is:

```
No_N+1_S_Loss: G upto_N_failures -> ~ F G N+1_S_Loss
```

with N = 0,1,2 and S = Electrical or Hydraulic and `upto_N_failures` is a property that holds in all states of a system such that up to N individual failures have occurred.

Graphical Interactive Simulation: A Safety Engineer can check the effect of failure occurrences on the system architecture using Cecilia OCAS graphical interactive simulator. The system architecture is depicted by a set of

interconnected boxes that represent nodes of the AltaRica model. Icons are associated with a node state. For instance, a green box is displayed if a distribution line delivers power and a red box is displayed otherwise. These icons help to rapidly assess the component current state.

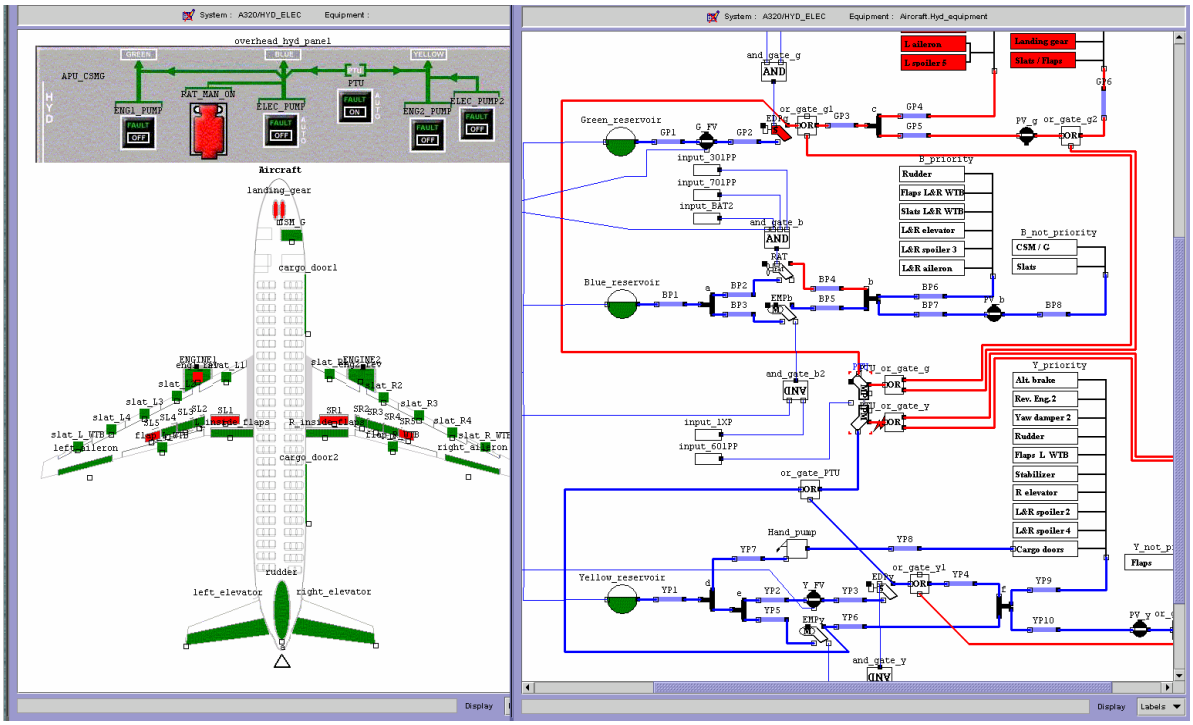

Figure 4 − Cecilia OCAS Graphical Simulator

To observe more complex situation such as the loss of several channels, special nodes called "observers" are added into the model. An observer internal state only depends on the value of other components outputs.
First, the simulator computes the initial state. Then, when the safety engineer selects a node the simulator proposes the set of events that can be performed at this step. This is the set of events with a guard that is true in the current state. The safety engineer chooses an event and the resulting state is computed by the simulator. As failures are events in the AltaRica model, the safety engineer can inject several failure events into the model in order to observe whether a failure condition is reached (such as loss of one or several power channels).

Figure 4 shows the graphical user interface of Cecilia OCAS. The Hydraulic system is displayed in the right window. All basic icons represent a component (tank, pump, distribution line …) of this system. The left window displays a set of observers that show whether aircraft devices powered by the hydraulic system are available or not. At the top of this window, we designed a control panel similar to the aircraft panel with button components that are used to activate or inactivate components in the hydraulic system.

Model-checking: A model-checker as Cadence Labs SMV (ref. 6) performs symbolically an exhaustive simulation of a finite-state model. The model-checker can test whether the qualitative requirements stated as temporal logic formulae are valid in any state of the model. Whenever a formula is not valid, the model-checker produces a counter-example that gives a sequence of states that lead to a violation of the safety requirement.

We developed tools to translate a model written in Altarica into a finite-state SMV model. Thus, we were able to check that both system models enforced their qualitative safety requirements. All requirements were verified in less than ten seconds although the truth value of some formulae depended in each state on as much as 100 boolean variables.

The model checker was very useful to debug a preliminary version of the electrical system model where the control of contactors was not properly defined. We extracted from the counter-example generated by the model-checker a scenario of events and then simulated it with OCAS Altarica simulator. We found several scenarios with one or two failure events and several (six or seven) no-event transitions that lead to a counter-example we would have never found by ourselves when using the interactive simulator to explore the electrical system behavior.

Multi-System

Any system model has to include some details about systems in interface. The A320-like hydraulic system model includes a description of the engine and electrical systems in order to know whether EDP and EMP pumps or the PTU are powered. Similarly, the electrical system includes a description of the engine and hydraulic systems in order to know whether normal and emergency generators are powered. In the models described in the previous sections, systems in interface were described at a very high level of abstraction. Hence, some failure propagations across system boundaries could not be observed. In this section, we explain how to plug the hydraulic and electrical system models in order to obtain a multi-system model that is useful to gain a better insight of inter-system failure propagation.

Modeling: To be able to plug two system models together they must include links with their environment. This means that components interacting with other systems should have interface flows.

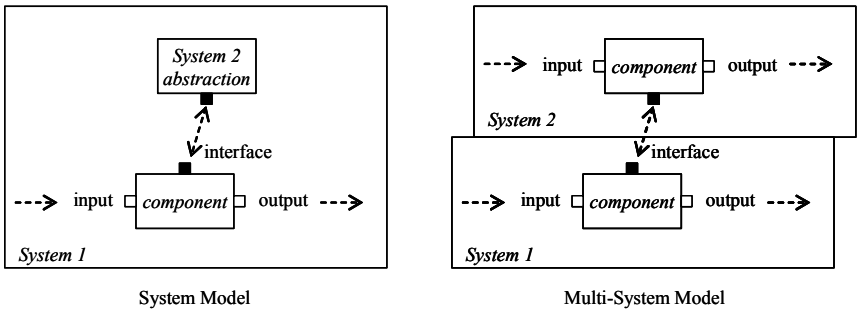

Figure 5 − Link Between Systems

As shown in figure 5, in a single system model, interface flows are linked to an abstract model of the system in interface. When developing a multi-system model this abstract model is replaced by a more detailed model. For instance, a hydraulic pump has two interface flows depicted in figure 6: one is shared with the controller from which it gets an activation order, another is shared with the electrical system for power supply.

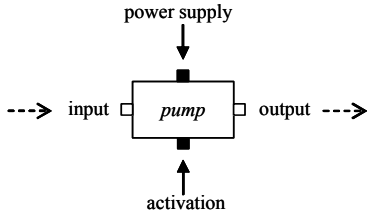

Figure 6 − Pump Interfaces

Example: The power supply interface flow is composed of an output variable named `elec_interface_SC` which transmits short circuit information to the electrical system and an input named `elec_interface_voltage` which transmits voltage to the pump. This input value in the hydraulic system is directly dependent on the architecture of the electrical system.

```
flow
 elec_interface_SC : bool : out;
elec_interface_voltage : bool : in
assert
 elec_interface_SC = (case {(status = SC): true, else false});
```

This "power supply" interface models short circuit propagation from the hydraulic system to the electrical system. Whenever the pump is in a short circuit state (the value of variable status is equal to SC) then interface flow `elec_interface_SC` is set to true so that the short circuit can be propagated over the electrical system. During the analysis of the hydraulic system, the propagation of a short circuit originating at a pump could not be properly assessed. The abstract model of electrical system we used did not take into account short-circuits. Having replaced this abstraction by the electrical system detailed model, the propagation of a short circuit originating in the hydraulic system can be traced even beyond the limits of this system.

Analysis: The analysis methodology described in the previous section was applied to this case-study: interactive analysis using the simulation capabilities of Cecilia OCAS, model-checking of requirements, and simulation of counter-examples found by the model-checker. With respect to analysis, the main interest of a multi-system model is that increased level of detail helps to better assess if safety requirements are met. In the following we give two examples where we improved the safety assessment.

To check the safety requirements of the standalone hydraulic system, we did not take into account situations where one engine was shut down by the pilot. In the abstract model of the electrical system, if one engine is shut down and the other fails, then electrical power is lost and, on ground, all hydraulic pumps are lost. So in this model, if combined with a particular pilot action, a single failure could lead to the total loss of hydraulic power. In the multi-system model, the electrical model includes the APU that can generate electricity on ground. Therefore we were able to show that shutting down one engine on ground is a safe operation.

Similarly, in the standalone electrical system, we assumed that the CSMG generator was always available to recover the loss of all the nominal generators (i.e. GEN1, GEN2 and APU). In the abstract model of the hydraulic system, the RAT is always activated, so the CSMG is always powered. This was a too restrictive assumption. In the multi-system model, the hydraulic model includes the correct activation rule for the RAT (i.e. it is activated in flight when engines or normal electrical power are totally lost). Using this detailed model, we confirmed that the safety requirements of the electrical system were enforced.

Finally, we did not notice any performance issue on our two medium-sized models combination neither concerning the overall simulation, nor concerning model-checking.

Future Work and Conclusion

Our experiment shows that safety system modeling and analysis are possible and fruitful using a formal approach provided that models have the right level of detail. We have to observe that our models should not confine to failure propagation related with the functional analysis. They should also include failure propagations that could be related to system-level risks as specification errors, assumptions, synergistic considerations through-out the life-cycle, energy effects, …Previous works such as references 7 and 8 present modeling approaches that, as ours, abstract nominal physical details and focus on failure propagation. However, the author main goal was to generate fault trees, so their models focus on system architecture and do not enable temporal analysis of highly dynamic reconfiguration mechanisms such as contactor activation rules in the electrical model. It is worth noting that a similar abstract approach is followed by Dassault Aviation that uses Cecilia OCAS to generate fault trees from AltaRica models.

During the ESACS project, other partners experimented a different approach (ref. 1). They started from models only describing the nominal behavior of a system. Then this model was extended with failure mode specification. For several case-studies, complementary work had to be performed in order to get a tractable model for the analysis tools. For instance, some parameter values were frozen and the number of time steps was reduced.

In this paper, we showed that we could state formally interesting qualitative and temporal safety requirements of aircraft systems and perform assessment analysis with interactive simulation and model-checking tools without performance problems. Moreover, the ability to produce and to simulate a counter-example is very useful. However, we think that a limitation of the model-checking tools we used lies in the inability to produce the set of all counter-examples with a given number of failure events. We will investigate this topic in a follow-up project called ISAAC (Improvement of Safety Analysis of Aircraft Complex systems) that started at the beginning of 2004. We plan to

study how to mechanically generate bounded length failure sequences and to generate dynamic fault trees (refs. 9, 10).

Acknowledgement

The work described in this paper has been developed within the ESACS Project, a European sponsored project, G4RD-CT-2000-00361.

References

1. M. Bozzano, A. Villafiorita et al. ESACS: an integrated methodology for design and safety analysis of complex systems. ESREL 2003 European Safety and Reliability Conference, 2003.

2. A. Arnold, A. Griffault, G. Point, A. Rauzy. The AltaRica formalism for describing concurrent systems. Fundamenta Informaticae n°40, p109-124, 2000.

3. GFI Consulting. Cecilia Workshop. http://www.gfic-oasys.com/french/cecilia/o_presentation.htm.

4. A. Rauzy. Mode automata and their compilation into fault trees. Reliability Engineering and System Safety, 2002.

5. Z. Manna, A. Pnuelli. The temporal logic of reactive and concurrent systems. Springer − Verlag, New-York, 1992, ISBN 0-387-97664-7.

6. K.L. MacMILLAN. Symbolic Model Checking. Kluwer Academic Publishers, 1993, ISBN 0-7923-9380-5.

7. Y. Papadopoulos, M. Maruhn. Model-based automated synthesis of fault trees from Matlab-Simulink models. DSN2001, Int. Conf. on Dependable Systems and Networks (former FTCS), Gothenburg, Sweden, pages 77-82, ISBN 0-7695-1101-5, July 2001.

8. Fenelon, P., McDermid, J.A., Nicholson, M. & Pumfrey, D.J. Towards Integrated Safety Analysis and Design, ACM Computing Reviews, Vol. 2, No. 1, p. 21-32, 1994.

9. R. Manian, J.B. Dugan, D. Coppit, K.J. Sullivan. Combining various techniques for dynamic fault tree analysis of computer systems. Proceedings of the International Symposium on High Assurance System Engineering, 1998 http://www.computer.org/proceedings/hase/9221/92210021abs.htm.

10. M. Bouissou, J.L. Bon. A new formalism that combines advantages of fault-trees and Markov models: Boolean logic Driven Markov Processes. Reliability Engineering and System Safety, Volume 82, Issue 2, Pages 149-163, November 2003.

Biography

P. Bieber, C. Castel, C. Kehren, C. Seguin, ONERA, 2 avenue Edouard-Belin, BP4025, 31055 Toulouse cedex 4, France, telephone − +33 5 62 25 26 42, facsimile − +33 5 62 25 25 93, email − {name}@onera.fr.
Pierre Bieber, Charles Castel, Christophe Kehren and Christel Seguin work at ONERA-CERT (French national aeronautics and space research center located in Toulouse, France) in the Modeling and Information Processing Department. Their recent work is about using formal methods to help aircraft system safety assessments.

C. Bougnol, J.-P. Heckmann, S. Metge, AIRBUS, 316 route de Bayonne, 31300 Toulouse, France, telephone − +33 5 61 93 40 38, facsimile − +33 5 61 93 31 55, email − {name}@airbus.com.
Christian Bougnol, Jean-Pierre Heckmann and Sylvain Metge work at AIRBUS France in the department in charge of the safety application policy for AIRBUS aircrafts.