EXPERIMENTS IN MODEL BASED SAFETY ANALYSIS: FLIGHT CONTROLS

Romain Bernard[1], Jean-Jacques Aubert[1], Pierre Bieber[2], Christophe Merlini[1], Sylvain Metge[1]

(1) Airbus, (2) Onera

Abstract: Since the ESACS and ISAAC projects, Airbus and Onera have been investigating failure propagation models and more specifically AltaRica model-based safety analysis. This paper presents results and lessons learnt from an industrial system architecture modeling experiment: rudder control system of the Airbus A340-500/600 aircraft. After introducing failure propagation model construction and analysis, the paper focuses on modeling the reconfigurations, the command/monitoring architecture and finally the latent failures. The main advantage of this approach is the improved readability of safety analysis results that facilitates a quick understanding of the system behaviour. This improves the communication between the safety and design communities. *Copyright © 2002 IFAC*

Keywords: Aerospace, Fault Tolerant System, Formal Methods, Flight Control, Safety.

1. INTRODUCTION

The aeronautical industry is constantly improving its design, definition and development processes. All manufacturers are now building a virtual aircraft before any component has actually been physically produced. Models support development phases and take benefit of available computation capabilities to provide relevant data or a better view of the aircraft. Whereas many domains (e.g. structure design, software development) now integrate a model-based approach, safety analyses are still based on Dependence Diagrams (DD), a.k.a. Reliability Block Diagrams (RBD) or Fault Trees (FT). Several weak points have been identified in the use of DD and FT:

- These representations may be difficult to read for anyone who is not a safety specialist, impacting the communication between system designers and safety engineers, especially when system designers have to validate the completeness of the safety analysis.
- The increasing complexity of aircraft architecture and technologies impact the analysis load: it becomes more and more costly to perform an "exhaustive" analysis and to provide a compact representation.
- Both formalisms are only static: dynamic aspects, such chronology of failure modes necessary to understand a scenario containing hidden failure modes of back-up components, are not presented.
- A DD/FT has to be manually written for each identified Failure Condition (FC, i.e. feared event); in the event of architecture update, each DD/FT has to be checked and updated if needed.

In order to improve safety assessment techniques, Airbus has been involved in ESACS (Enhanced Safety Assessment for Complex Systems), see (Åkerlund, et al. 2003) and ISAAC (Improvement of Safety Activities on Aeronautical Complex systems), see (Åkerlund, et al. 2006) European projects. Those projects aimed at developing safety assessment techniques based on the use of formal specification languages and associated tools. Among the formal languages and tools investigated during both projects, the AltaRica language, see (Arnold, et al., 2000), and the Cecilia^{TM} OCAS tool developed by Dassault were selected. The first lessons learnt based on ESACS experiments indicated that this new approach could overcome the weak points previously mentioned, but these experiments were based on a limited interaction with the system designers. So, it was decided to perform during ISAAC an experiment that would directly involve system designers in order to assess whether the approach and tools could be integrated into the current Airbus Safety Process.

The second section of this paper describes the selected case study: the Rudder Control System. Subsequently, the principles of model-based safety analysis using AltaRica are presented in the third section of the document. The fourth section details the main steps in the application of the modelling and analysis approach to the Rudder Control System. Finally a comparison of the AltaRica approach with other model-based safety analysis approaches will precede the conclusion, focused on the industrial applicability of the approach.

2. RUDDER CONTROL SYSTEM

The first experiments in model based safety analysis performed by Airbus and Onera were introduced during the ESACS project. The case studies were the A320 electrical and hydraulic generation systems, see (Bieber, et al, 2004). In order to enrich this panel and to increase the set of aircraft systems particularities, Flight Controls were selected for modelling. This example allowed us to explore problems related to the modelling of computer based system and control loop architecture.

Flight control systems command mobile surfaces (ailerons, rudder…) in order to rotate the aircraft around the 3 axes: pitch, roll and yaw. Since the introduction of fly-by-wire systems, pilot inputs are interpreted by the flight controls computers that move the surfaces as necessary to achieve the desired flight path modification. Of the numerous flight control systems, the rudder control system has been selected as the support of this modelling activity.

Each mobile surface may be identified as composed of three parts:

- Resources: providing power to control and actuation systems. Resource systems such as electrical and hydraulic generation are complex systems analysed independently. The analysis of consumer systems, such as flight control systems, limits its consideration to a resource being either available or lost.
- Actuation system: acting on the connected surface. A servo-control is an hydraulic actuator.
- Control system: transmitting pilot or autopilot orders to actuators. There are several types of flight control computers: Primary Computers (PRIM), Secondary Computer (SEC) that provide backup to the PRIM computers, and the Backup Control Module (BCM) that is used as an ultimate backup.

The design of mobile surfaces and their control system architecture depend on the structure efforts and technological means available to actuate them. The actuation system architecture and the behaviour expected are input data for the control system design.

The A340-500/600 rudder actuation system is composed of 3 actuation lines acting simultaneously in nominal situation. Each line is composed of a PRIM controlling one servo-control (S/C). When no PRIM is able to contribute to the actuation (failure of the PRIM or of its associated S/C), the SEC is then activated. Finally, the ultimate back-up is enabled when no computer, either PRIM or SEC, is available.

Flight control systems are built to very stringent dependability requirements both in terms of safety and availability, see (Lacaze, et al., 2004). The most important qualitative safety requirement is that a catastrophic consequence must not be due to a single failure, a control surface jam or a pilot control jam. In order to avoid unintended behaviour, the actuator and the computer outputs have to be monitored.

In order to deal with the safety issue, computers are built according to a COM/MON (command and monitoring, cf. Fig.1) principle: two channels (a command channel and a monitoring channel), active or waiting simultaneously, receive feedbacks from several sensors and compute commands to be transmitted to actuators. The MON channel monitors for two kinds of unintended behaviour:

1. intrinsic computer failure: MON channel compares its computed commands with those received from COM channel.
2. controlled component failure: MON channel monitors feedback from several sensors on the actuator and on the actuated mobile surface.

When the difference between the two commands exceeds a predetermined threshold (i.e. one of the two channels is faulty) or if a feedback is out of range, both channels have to be switched off.

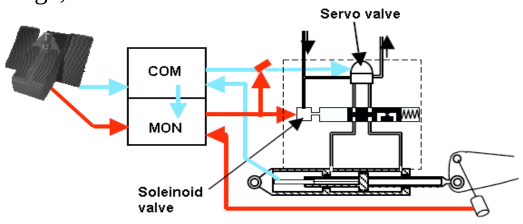

Fig. 1. Command Monitoring architecture

3. INTRODUCTION TO FAILURE PROPAGATION MODELLING AND ANALYSIS

A failure propagation model, see (McDermid and Pumfrey, 1994; Heinher et al. 2001), describes what failure modes can originate in a system component and how these failure modes are propagated from one component to others. Using AltaRica we associate a formal behaviour with each component. Data considered are discrete: e.g. correct command, lost hydraulic power, instead of the actual data values.

3.1. AltaRica language

AltaRica is a formal language developed by the LaBRI (Laboratoire Bordelais de Recherche en Informatique), jointly with French industrial partners (especially Dassault Aviation and Elf Aquitaine…) in order to model critical systems for safety purpose.
Each component, called a node, is composed of the declaration of variables and events, and the definition of transitions and assertions. These concepts are illustrated by the following example. Component SEC sends a correct order to an actuator when it is powered, activated and it receives a consistent feedback, else it does not send anything. If the SEC is in a correct state, a failure may occur and put the

SEC in a lost state. This behaviour is illustrated by the continuous-line part of figure 2.

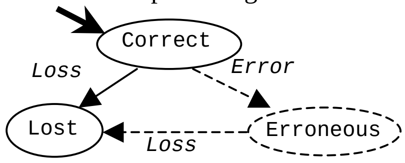

Fig. 2. SEC automaton

Given the domain `{correct, lost}` called `StatusType`, the AltaRica model of the SEC is presented in the following tables.

The state variable table declares that `Status` is an internal variable of type `StatusType` that is initially equal to `correct`. The value of a state variable can only be modified by transitions.

State	Type	Initial Value
`Status`	`StatusType`	`Correct`

Tab. 1. State variable of SEC

The flow table declares input/output/local variable(s), typed, not initialised. They are used to model data exchanged with interfaced components.

Flow	Type	Direction
`Resource`	`Boolean`	`In`
`Activation`	`Boolean`	`In`
`Feedback`	`StatusType`	`In`
`Order`	`StatusType`	`Out`

Tab. 2. Flow variables of SEC

The event table declares events and tells whether they are triggered internally by the system or externally by its environment. In the latter case, an occurrence probability law may be associated with the event. In the following all internal events will be named `update`.

The transition table contains the set of transitions describing events. A transition is composed of a guard (Boolean expression based on state variables and input/local flow variables, defining in which conditions the transition may be triggered), the name of the event labelling the transition, and the set of state variable new affectations. In the following table, a transition is associated with event `Loss` that may only be triggered when component SEC is not lost (i.e. `Status` is different from `lost`). The new value of `Status` is `lost`.

Event	Guard	New affectations
		`Status`
`Loss`	`Status != Lost`	`Lost`

Tab. 3. Event `Loss` of SEC

The assertion table contains the set of output/local variables computation formulae. Computation formulae of an output variable may be based on state, input and local variables. Local variables are mainly used to factorise redundant expressions. The following table defines how the `Order` output is computed. A nominal `Order` is sent (`Order=correct`) whenever `Resource` and `Activation` are true and `Status` and `Feedback` are `correct` otherwise no order is sent (`Order=lost`).

Assertion	Case	Value
`Order`	`Resource and Activation and Status = correct and Feedback = correct`	`correct`
	`Else`	`lost`

Tab. 4. SEC Output computation

An AltaRica model, composed of several components linked together, may be considered as a representation of an automaton of all possible behaviour scenarios. Each node of the automaton is a configuration, i.e. a function that associates a value with each state and flow variable. The initial configuration is first defined by initial value of components state variables. Then, assertions are computed and values are associated with outputs. These values are propagated to connected components hence values can be associated with inputs. In the automaton, two configurations are connected by an arrow labelled with an event name whenever, in the first configuration, the guard of the transition associated with that event is satisfied and, the second configuration is equal to the first configuration with state variables modified according to the assignments of the corresponding transition.

A scenario of the model is a path in the automaton that starts from the initial configuration and moves from configuration to configuration by selecting an arrow labelled with an event. If, from a current configuration, it exists an arrow labelled with an `update` internal event, then external event cannot be selected from this configuration. This means that `update` internal events are given priority with respect to external ones.

3.2. Model analysis means

Industrial tools provide services to perform the analysis of the model. Automatic and interactive simulations facilitate the exploration of scenarios of events potentially leading to a FC.

Probably the most useful tool is the interactive graphical simulation. The tool describes graphically the system model and its current configuration. The colour of links between components indicates the current value associated with the linked variables. The icon of a component depicts the component current status. For instance, green colour means true or correct, red means false or lost, purple means erroneous or runaway and grey means not active or stand-by mode. It is also possible to inspect component variable values by clicking on them. In the current configuration, all external events that can be triggered are presented to the user. The user selects one event, then all internal events that can be triggered are automatically triggered and the new configuration is computed so the user can look at the consequences of each failure event.

Using the simulation, it is possible to validate with the system designer that the composition of nodes, separately developed, behaves consistently with the system actual behaviour. Once the AltaRica model has been validated, the simulation allows a quick understanding of the system and its reconfigurations in case of failure occurrence.

Automatic simulation can be used either to directly generate Minimal Size Sequences (MSS) of events leading to a FC or to generate a FT from which Minimal Cut Sets (MCS) can be computed with appropriate tools. The algorithm is described in (Rauzy, 2002). MSS keep track of the chronology of events whereas in a cut set it is usually not possible to determine what failure event occurred first.

With current AltaRica tools, the user has to specify an Output variable O and a value V, and MSS/MCS or FT leading to situations where O=V are generated. If a FC concerns several components or several outputs, a new component has to be created: it is called an Observer. It is a virtual component because it does not exist in the actual system architecture.

The case study reported in this paper being focused on qualitative analyses, tools and results presented in this paper do not mention quantitative aspects. Nevertheless, as failure rates may be defined while modelling, several AltaRica tools may either compute the probability of a FC based on the probability of each MCS or propose a stochastic simulation, see (Boiteau, et al. 2006).

4. MODELLING EXPERIMENTS

4.1. Incremental model construction

To model the Rudder Control System we first developed a library of AltaRica components. For the A340/500-600 case-study, the model is composed of the following component classes: `PRIM` and `SEC` for computers, `ServoCtl` for actuation, `ResourcePower` for electrical and hydraulic circuits, `BCM` and `BPS` for the ultimate back-up and `Rudder`. The `Rudder` encapsulates both the sensors providing feedbacks and the observer part, above-mentioned, for the FC "total loss of control of the rudder".

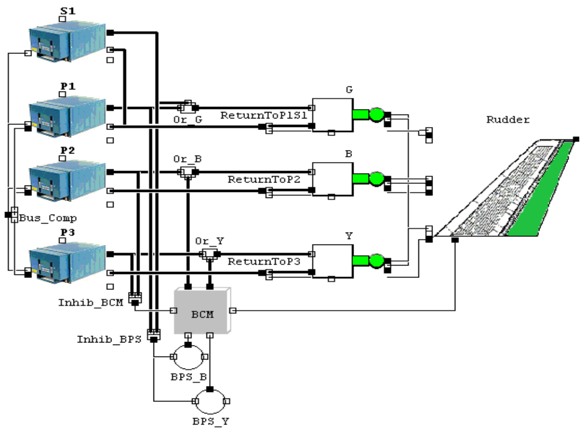

Fig. 3: A340/500-600 Rudder Control System

The Rudder Control model (see Fig. 3) is built by creating instances of the classes of components (e.g. `P1`, `P2` and `P3` are instances of `PRIM`) stored in the library and linking these instances.

As a progressive methodology, the first step focused on the loss of components or group of components, and then the effect of erroneous behaviour. This aims at validating separately mechanisms for the reconfiguration in case of partial loss and mechanisms for detecting erroneous behaviours. Two versions of the library of components were developed. In the first library, most of the components contain failure mode `loss` leading to the total loss of the component. In the second library, components also contain failure mode `error`. Some components as the rudder, being partly an observer, do not contain any failure mode.

It is rather easy to enrich AltaRica nodes in order to deal with a new failure mode. First, in order to add the erroneous case to the internal state of each component and to the failure propagation, the `FailureType` domain previously defined is enriched with a new value: `erroneous`. Then failure mode `error` is declared and a new row is added in the transition table to describe the dotted line in figure 2 that relates the correct and erroneous states.

Event	Guard	New affectations
		`Status`
`Error`	`Status = Correct`	`Erroneous`

Tab. 5. AltaRica code of failure mode `Error`

Finally, the assertion table has to be extended to take into account situations when state variables or inputs are erroneous. For instance, in table 6, we consider that the `Order` output of `SEC` is erroneous if either its `status` or the rudder `feedback` is erroneous.

Assertion	Case	Value
`Order`	`Resource and Activation and Status = correct and Feedback = correct`	`correct`
	`Resource and Activation and Status != lost and (Status = erroneous or Feedback = erroneous)`	`erroneous`
	`Else`	`lost`

Tab. 6. Enriched assertion for `Order` output

Two models were built that are graphically similar to figure 3. The first model is based on the basic library of components and is used to assess Rudder Control loss. The second model is based on the enriched library and is used to analyse Rudder Control erroneous behaviour.

4.2. Modelling reconfigurations

Flight controls systems are highly redundant, thus the rudder control system has several backups that are not always active as, e.g. SEC computer and BCM.

An activation order is modelled with a flow variable (e.g. a Boolean variable called `Activation`) sent from a component, defining the activation rule, to the back-up component to activate. Then, to model the activation into the back-up component, input variable `Activation` is used into guards and assertions to differentiate the activation or the stand-by status of the component, see table 4.

SEC computer S1 activation rule is defined in component `Bus_Comp`, see figure 3, that models inter-computer communication. Table 7 gives the assertion that defines S1 activation. Similarly, components Inhib_BCM and Inhib_BPS contain the rules for BCM and BPS activation.

Assertion	Case	Value
S1_Activation	P1_status=lost and P2_status=lost and P3_status=lost	true
	Else	false

Tab. 7. S1 activation rule

4.3. Modelling Hidden/Active failure modes

Failure modes impacting secondary components are divided into two categories: failures that remain undetected until component is required, called "hidden failures", and failures occurring when the component is active or detected before activation, called "active failures".

This notion can be illustrated by enriching the SEC model described in section 3.1, event Loss described in table 3 would be replaced by a hidden failure and an active failure described hereunder in table 8. The guards of these transitions include details about the activation of the component.

Event	Guard	New affectations
		Status
Hidden_Loss	not Activation and Status != Lost	lost
Active_Loss	Status != Lost	lost

Tab. 8. SEC Hidden/Active failures transitions

Given the system architecture presented in figure 3, and considering a theoretical failure mode leading to the loss of all primary computer, called "`PRIM: Loss of all PRIMs`", and failure modes presented in table 8 for the SEC. The cuts leading to the activation of the ultimate backup would be:

```
'PRIM: Loss of all PRIMs'&'SEC: Hidden Loss'
'PRIM: Loss of all PRIMs'&'SEC: Active Loss'
```

Whereas the sequences produced for the same analysis would be:

```
'SEC: Hidden Loss';'PRIM: Loss of all PRIMs'
'SEC: Active Loss';'PRIM: Loss of all PRIMs'
'PRIM: Loss of all PRIMs';'SEC: Active Loss'
```

The sequences provide the basis for a more precise analysis because irrelevant scenarios are removed such as, for instance, hidden loss of the SEC happens after it is activated.

4.4. Modelling COM/MON principles

In order to model the COM/MON architecture of a computer, a monitoring component (MON) receives data from a command component (COM) and several feedbacks from sensors, and then compares several data together. To detect any inconsistency, the MON component has to be available, meaning in a correct state, energized and sure that the hydraulic circuit powering the S/C is intact. The selected inconsistency detection reaction is: when an error is detected, the computer does not contribute anymore to the actuation. The detection is there an `update` event that switches off the component (`Status=lost`).

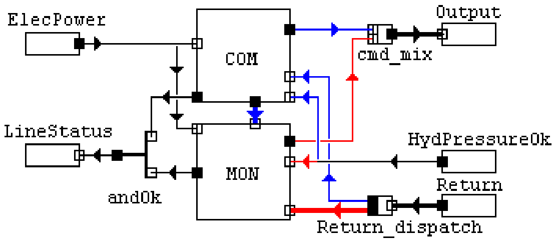

Fig. 4. Command Monitoring model

To detect an intrinsic computer failure, the MON computes locally an order according to its status and several feedbacks, and compares it to the order received from the COM component. Table 9 illustrates the code of this monitoring rule.

Event	Guard	New affectations
		Status
Update	Available and InputFromCom != Mon_Order	lost

Tab .9 Monitoring rule code

4.5. Avoiding permanent loops

The rudder actuation system is a looped system: e.g. S/C feedback returns from sensors depend on the S/C internal status and the order received, but the computation of the order depends also on the feedbacks. A loop may lead the simulator to run into an unlimited/permanent loop, so it is necessary to avoid it. The solution applied has been to add "delays" into components:

- Inputs contributing to the loop are stored into state variables via an `update` event.
- Assertions include state variables, instead of sensors feedbacks inputs, in order to consider the delay into the output computation.

Illustration: given a component with an internal state variable `InStatus` and an input variable `Input`, both defined as `status_type` typed, an instantaneous event `Update` (see table 10) will affect the `Input` value to the `InStatus` variable.

Event	Guard	New affectations
		InStatus
Update	InStatus != Input	Input

Tab. 10. Delay modelling into input acquiring

5. CONCLUSION

Other approaches for model-based safety analysis were recently experimented by partners from ESACS project and in the U.S., see (Heimdahl et al., 2005). Their approach differs from the AltaRica one with respect to how the failure propagation model is obtained. In their approach, a system model developed by System Designers is extended with failure modes by Safety Specialists in order to perform safety analysis. The main advantage of this approach is that the extended system model requires limited model validation by system designers as they have developed the original system model. The main drawback is that system models often rely on constructs that currently go beyond the current scope of model-checkers (as real arithmetics) so the original model needs to be simplified. So this approach is certainly the way to go in order to perform true model-based safety assessment in the future. But in the mean time, the AltaRica approach can be used as an interim approach, especially for the architecture validation in early conception phases.
The goal of the A340 Rudder case study was to evaluate the accuracy of results generated from the model. To perform the comparison, components failure modes were mapped to FC analysis failure modes and computation FCs. MCS generated from the model were consistent with MCS from the analysis manually performed, currently based on the engineering judgement and his system knowledge. This gave confidence into this approach and lead Airbus to launch internal R&D projects investigating viability of the industrialization of an AltaRica model-based safety analysis.

More generally, regarding to the language and industrial tools, several advantages have been identified to this approach:

- Graphic layer allows building models close to description diagram, for a quick identification of system architecture.
- Interactive simulation completed with dynamic graphic (icons and link colours updated after each event triggered) ease the understanding of the system behaviour: any user may trigger an event and view the consequences into the system (loss of components, reconfigurations…).
- Implementation of MCS/MSS automatic generation algorithms ensures an exhaustive exploration and eases the analysis update in case of architecture modification.
- Since failure modes of a component are relevant for several failure conditions, a model will allow performing several analyses.

Following ESACS/ISAAC projects and the Airbus A340-500/600 rudder case study, several research areas are currently investigated:

- Model sharing: Safety analyses and Operational Reliability analyses being focused on FM, a model may be common to both analyses, each one defining its own observer.
- Particular Risks Analysis (PRA) & Common Cause Analysis (CCA): failure propagation models add functional aspects into geometrical models, i.e. complete input data for PRA/CCA.
- Multi-system analysis: compositional property of AltaRica allows linking several mono-system into a multi-system; this should help to assess aircraft level properties.

REFERENCES

Åkerlund, O., P. Bieber, E. Böde, C. Bougnol, M. Bozzano, M. Bretschneider, C. Castel, A. Cavallo, M. Cifaldi, A. Cimatti, A. Griffault, C. Kehren, B. Lawrence, A. Lüdtke, S. Metge, C. Papadopoulos, R. Passarello, T. Peikenkamp, P. Persson, C. Seguin, L. Trotta, L. Valacca, A. Villafiorita, G. Zacco (2003). ESACS: an integrated methodology for design and safety analysis of complex systems. In: *Proceedings of ESREL*. Balkema publisher.

Åkerlund, O., P. Bieber, E. Böde, M. Bozzano, M. Bretschneider, C. Castel, A. Cavallo, M. Cifaldi, J. Gauthier, A. Griffault, O. Lisagor, A. Lüdtke, S. Metge, C. Papadopoulos, T. Peikenkamp, L. Sagaspe, C. Seguin, H. Trivedi, L. Valacca (2006). ISAAC, a framework for integrated safety analysis of functional, geometrical and human aspects. In: *Proceedings of ERTS*.

Arnold, A., A. Griffault, G. Point, A. Rauzy (2000). The AltaRica formalism for describing concurrent systems. In: *Fundamenta Informaticae* p109-124.

Bieber, P., C. Bougnol, C. Castel, J.P. Heckmann, C. Kehren, S. Metge, C. Seguin (2004). Safety assessment with AltaRica: lessons learnt based on two aircraft system studies. In: *Proceedings of World Computer Congress*, IFIP

Boiteau, M., Y.Dutuit, A.Rauzy, J.P.Signoret, (2006). The AltaRica Data-Flow language in use : Assessment of production availability of a multistates system. *Reliability Engineering and System Safety*, 91:747-755, 2006.

M.P..Heimdahl, A.Joshi, M.Whalen, (2005) Model-Based Safety Analysis: Final Report, NASA Technical report.

G.Heiner, J.A.McDermid, Y.Papadopoulos, R.Sasse, (2001)Analysis and synthesis of the behaviour of complex programmable electronic systems in conditions of failure, In *Reliability Engineering and System Safety*

Lacaze, I., J. Souyris, P. Traverse (2004). Airbus fly-by-wire: a total approach to dependability. In: *Proceedings of World Computer Congress*, IFIP

McDermid, J., D. Pumfrey (1994). A Development of Hazard Analysis to aid Software Design. In: *Proceedings of COMPASS.*

Rauzy, A., (2002). Modes automata and their compilation into fault trees. *Reliability Engineering and System Safety*, 78:1-12, 2002.