SAFETY ASSESSMENT WITH ALTARICA

Lessons learnt based on two aircraft system studies

Pierre Bieber[2], Christian Bougnol[1], Charles Castel[2], Jean-Pierre Heckmann Christophe Kehren[2], Sylvain Metge[1] and Christel Seguin[2]

[1]AIRBUS, 316 route de Bayonne 3100 Toulouse cedex 03 France; [2]ONERA, 2 avenue E. Belin 31055 Toulouse cedex 04 France

Abstract: AIRBUS and ONERA used the AltaRica formal language and associated tools to perform safety assessments. Lessons learnt during the study of an electrical and hydraulic system are presented.

Key words: dependability, aircraft, formal methods

AIRBUS and ONERA were recently 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. We used the AltaRica (Arnold et al. 2000) formal language that is supported by Cecilia OCAS workshop developed by Dassault Aviation. Two case-studies based on AIRBUS aircraft electrical and hydraulic systems were used to validate the approach (Kehren et al. 2004b). In this paper we present lessons we learnt during ESACS. Lessons are sorted in three categories: **Advantages** are situations where the use of AltaRica was clearly positive, **Difficulties** are situations where the use of AltaRica was not directly positive but we found out how to circumvent the difficulties and the remaining situations are considered to be **Limitations**.

1. SYSTEM AND REQUIREMENT MODELLING

1.1 System Modelling

The first step of ESACS approach (Bozzano et al. 2003) is to obtain a formal model that is suitable to perform safety assessment of the system under study. We followed the modeling approach defined in (Fenelon et al. 1994) that abstracts nominal physical details of components and focuses on failure propagation. We defined libraries of (electrical, hydraulic, computer, ...) component models and used them to build the safety model.

Advantages: Each system component is modelled by an Altarica node that can be regarded as a mode automaton (see Rauzy 2002). In Cecilia OCAS workshop, each node is associated to an icon and belongs to a library. Once the component library created, the system is modelled easily and quickly. 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. Furthermore, hierarchy of nodes can be used to build complex components and structure the system model. We were able to combine in a common model both models of the electrical and hydraulic systems in order to assess whether interface safety requirements were met.

Difficulties: It can be difficult to adequately model physical system failure propagation. If we consider a hydraulic circuit pipe, a leakage cannot be modelled only by considering the absence or the presence of fluid in the pipe. Indeed, the real consequence of a leakage is a quick pressure decrease for all the components located downwards the leaking component and, at last, an absence of fluid in the circuit. As a result, a pipe must transmit the pair (fluid, pressure) in order to propagate correctly 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.

Limitations: As explained above, components are linked with bidirectional flows. The actual topology of the electrical and hydraulic systems includes several loops that lead to potentially circular definitions in the formal models. Altarica and other data flow languages reject models that include syntactical circular definitions. This is rather pessimistic, because our models were rejected although all the loops include valves or contactors such that there is no cycle in the failure propagation. We applied the usual solution that consists in adding a time delay in the loops. One consequence is that failure propagation is not instantanneous, it needs several time steps to reach a correct state.

1.2 Formal Safety Requirements

The second step in the ESACS methodology is to formalize the safety requirements. We limited our study to qualitative safety requirements of the form *"if up to N individual failures occur then the loss of N+1 power channels shall not occur"* with N = 0,1,2. To observe situations such as the loss of several channels, special AltaRica nodes called observers are added to the model.

Difficulties: AltaRica observers do not make the difference between and instantaneous loss of power that allows power to be recovered later due to a reconfiguration and a permanent loss of power that does not allow any recovery. We used Linear Temporal Logic operators to model the several temporal flavors of the loss of channels. By now, Safety Engineers are not familiar with temporal logic operators and they might need some training in order to be able to formalize safety requirements. To limit this difficulty, we proposed to define a library of useful safety requirement formulae. When we studied the Electrical system we reused the safety requirement formulae developed for the hydraulic system study.

2. SAFETY ASSESSMENT TECHNIQUES

2.1 Graphical Interactive Simulation

A Safety Engineer can check the effect of failure occurrences on the system architecture using Cecilia OCAS graphical interactive simulator. 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 in the model a number of failure events in order to observe whether a failure condition is reached (such as loss of one or several power channels).

Advantages: Several icons can be associated with a component of the model depending on its state. For instance, a green box is displayed if the observer receives power and a red box is displayed otherwise. These icons help to rapidly assess if a failure condition occurs.

2.2 Fault Tree generation

Fault tree analysis is a well established technique among safety engineers. Cecilia OCAS includes a fault tree generator. This tool efficiently produces a Boolean formula (a fault tree) that describes all the sequences of failure events of the Altarica model that lead to a given observer state.

Advantages: Thanks to fault tree analysis packages the Safety Engineer can compute minimal cut sets of a failure condition and investigate what is the minimal number of failure events that lead to it. If failure occurrence rates are associated with the failure events of the AltaRica model, quantitative analysis (probabilistic computations) can be performed as well.

Limitations: The current algorithm has strong limitations on the form of the AltaRica model that can be taken into account: the order of occurrence of events in the model should not make any difference on the state of the system. So we cannot apply this tool to our models because, for a given sequence of failure events, some combinations with time delay events (that we had to add in order to avoid circular definitions) are not equivalent due to incorrect failure propagations. To overcome this limitation, Cecilia OCAS also includes a sequence generator tool that explores the state space of the model in order to find bounded length sequences that lead to a given failure conditions.

2.3 Model-checking

A model-checker as Cadence Labs SMV (McMillan 1993) 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 describes 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.

Advantages: 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 sequence 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) time delay transitions that would lead to a counter-example that we never found by ourselves when we used the interactive simulator to explore the electrical system behavior.

Limitations: The model-checking tools we used was unable to produce the set of all counter-examples with a given number of failure events.

3. CONCLUDING REMARKS: SAFETY ARCHITECTURE PATTERNS

During ESACS we found out that libraries of components and safety requirement formulae were very useful at the modeling stage. We proposed to apply a similar approach at the validation stage by using a library of safety architecture patterns. A safety architecture pattern (Kehren et al. 2004a) describes a typical safety architecture as a triplication or a primary/backup. These pieces of architecture contain intrinsic safety properties and can be used to demonstrate the fulfillment of aircraft system safety requirements. We applied this approach to check the safety requirements of the electrical system. We compared the results with the classical approach. Nevertheless this new approach does not have the purpose to compete with the classical one but to bring more methods in some aircraft system development phases, especially the preliminary ones, for supporting the engineering judgement.

The certification process of the civil aircraft is supported by guidance and recommended practices provided in ARP4754 "Certification Considerations for Highly-Integrated or Complex Aircraft Systems". Amongst different guidelines, one of them is associated to the assignment of the Development Assurance Level (DAL) to items of an aircraft system architecture. The DAL assignment takes under consideration the safety repercussion of item failure scenarios (failure conditions) and different architecture features, such as redundancy, monitoring, or partitioning to eliminate or contain the degree to which an item contributes to a specific failure condition. Depending on these considerations, a DAL level is assigned to an item from E to A, reflecting the necessary quality effort during the aircraft/system/item development. The guidance is summarized under a table format that sorts the DAL assignment by different architecture features.

In particular, it is in a big interest to formalize the DAL assignment concern by using the safety pattern concept. By this way, model checking techniques can confirm automatically the full independence of item failures within an aircraft system architecture and bring at early aircraft design steps a model to guide the DAL assignment.

ACKNOWLEDGEMENT

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

REFERENCES

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

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

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

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

C. Kehren et alter, Architecture patterns for safe design, in proceedings of the first AAAF Conference on Complex and Safe System Engineering, 2004.

C. Kehren et alter, Advanced Multi-System Simulation Capabilities with AltaRica, in proceedings of the International System Safety Conference, 2004.

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