A practicable safety modeling methodology for aircraft systems using Altarica

Su Duo, Shaojun Li

China Aero-poly technology Establishment, AVIC, Jingshun Rd7, Chaoyang District, Beijing 100191, China

Abstract

With the increasing system scale and complexity, safety analysis based on formal models has been widely used in the development of aircraft products. However, when many departments or suppliers participate in the joint development of safety models, lots of problems arise. For example, models are weakly matching; models are incorrect, incomplete or with low reusability and difficult to maintain. To solve these problems, a practical safety modeling methodology based on Altarica, which contains three phases like information collection, model construction and model V&V, is proposed to establish a more structured, systematic and efficiency way in this paper. Detailed processes and relevant constraints are declared for each phase. Meanwhile, to improve the model management, it’s also discussed how to use the shared database to enhance the model reusability and simplify the model modification. At last, a hydraulic system is taken as an example to show how to the safety modeling methodology in practical.

1. Introduction

With the increasing system scale and complexity, safety analysis based on formal models is developed with more advanced model description capacity and automated analysis process, and has been highly accepted by safety-critical industries in different areas [1], such as aviation, railway transport and nuclear power, etc. However, with the wide use of formal models in the development of aircraft products, lots of problems on model construction, integration and management arise [2][3][4], especially when multiple...
departments or suppliers participate within the process together. These problems could be summarized as follows.

First of all, because of the lack of standard interface definition among different aircraft system models, models couldn’t be assembled in the process of the model integration, which leads to continuous iteration and sometimes even model reconstruction. Model reconstruction is a disruptive problem which means previous work wasted. Second, for the lack of unified modeling process and good modeling constraints, the correctness, consistency and completeness issues of Altarica models have emerged. For example, without consideration of the architecture and function characteristics during different phases and missions, many problems such as incomplete configuration, inconsistency between models and actual function process as well as incorrect failure propagation could be introduced. Meanwhile, the lack of modeling constraints leads to the bad model readability and disordered hierarchies, which highly increase the risk of introducing human errors. Finally, in the process of modeling complex systems, as the design updates and refines, the difficulty and workload to modify models are greatly increased. Therefore, it’s quite necessary to improve the ability of model maintenance and reuse.

Well-defined modeling process and modeling constraints can guarantee the correctness and completeness of the models and ensure that models match with others. References [4] [5] introduce the modeling process of the electronic system, hydraulic system and transmission system, but not fully define the information that should be collected before modeling (such as system configuration in different phases), as well as the necessary constraints (such as the naming rules). A Shared model database can be used to increase reusability. Reference [6] proposes constructing the safety architecture model library to improve the efficiency of modeling, but it doesn’t specify the contents of the shared database and how to reduce the difficulty of modifying models with the help of shared database so as to improve the model management ability.

In order to solve problems above, a whole safety modeling process, which contains three phases like information collection, model construction and model verification, is proposed in this paper first. The sub-processes of three phases are also specified as well as the modeling rules and constraints. Then, researches on shared-database construction are carried on to improve the model reusability and simplify model modification process. Finally, a case study on the hydraulic system is performed to show how to implement the modeling process.

2. The Altarica language

Altarica is a formal language developed by the computer science laboratory of Bordeaux jointly with industries partners [7], and has been widely used in the aviation areas. The language is carried out by the tool Simúía™ EADS APSYS (for example) which provides a graphical interface to design models and allow analyzing them by different ways such as simulation, automatic generation of minimal cuts (i.e. shortest scenarios leading to the failure condition) or sequences (i.e. ordered cuts).

Altarica language is hierarchical and compositional. Each component is described by a model automaton [8]. The basic unit to model a system component is called a “node” and is composed with three different parts: 1) the declaration of variables and events; 2) the definition of transitions; 3) the definition of assertions.

Each component has a finite number of flow variables and state variables. Flow variables are the inputs and the outputs of the node: they are the links between the node and its environment. State variables are internal variables which are able to memorize current or previous functioning mode (for example, failure mode). In our models, these variables (flow and state) are either Boolean or enumerated. Then, each node owns also events which modify the value of state variables. These events are phenomenon such as a failure, a human action or a reaction to a change of one input value.
The transitions describe how the state variables are modified. They are written such as “G(s,v) |- E - >s_” where G(s,v) is a Boolean condition on state s and input variables v, E is the event and s_ is the effect of the transition on state variables. If the condition G is true, then event E can be triggered and state variables are modified as described in s_. The assertions describe how output variables are constrained by the input and state variables.

Failures are propagated via nodes by their inputs and outputs. Failures can also be propagated by synchronizations which simulate the failure of several components at the same time. Hierarchy of nodes can be used to model complex components and build the model of the global system. Once the global model is obtained, the AltaRica model allows analyzing failure condition. Different tools can calculate, for example, minimal cut sets or the occurrence rate of a failure condition.

3. The modeling process

Safety models are usually used to take safety assessment of identified risks or hazards which exist for reasons of endogenous and exogenous causes. Therefore, before describing the modelling process, we have to stress that identification of risks in a structured and systematic way is the basis of normalized, systematic and structured safety modelling process [9]. After finishing identification of risks or hazards, the modelling process could start.

The modeling process contains three phase: information collection, model construction and model verification and validation. Each phase owns different sub-processes and relevant rules and constraints.

3.1. Information collection

3.1.1. Sub-process of information collection

Complete information collection is quite necessary before constructing a model. Incomplete information would lead to an incorrect model, which means more efforts to modify the model later. In order to clearly specify the system design, we define the information collection process as shown in Fig.1.

a) Specify the system architecture, external entities and external interfaces

The system architecture, external entities and external interfaces should be specified first. The architecture is the basis for the model. External entities contain origin producers of model input, target consumers of system output, and other entities representing technological exchanges or measures with external environments. External entities could be used to specify model inputs and outputs.

There exist three kinds of model inputs: (1) fluids like energy or supplied flows; (2) command and control flows issued by the operator or pilot; (3) configuration transmitted manually or automatically to the system and referring to the state of the architecture in different flight phases and missions.

According to the hierarchical level of the modelling system, the model may be later used to be integrated into a much higher level model. Meanwhile, the modelling system may have a quite high hierarchical level itself, which means it needs to assemble sub-system models (or supplier models) for this system. Therefore, in order to successfully assemble supplier models later, it’s necessary to specify the interfaces among different sub-models. The interface definition is declared in 3.1.2.

b) Build the function tree to specify the functions and services to model

Build the function tree, and specify the safety-relevant functions and services to model according to the aircraft FHA and system FHA results. The system architecture is hierarchical. In order to be
consistent with the hierarchical architecture, the system fiction should be hierarchical also, which could be reflected by the function tree.

![Diagram](image)

Fig. 1 Information collection process

c) Specify system breakdown structure

The function chains and relevant blocks/entities (blocks/entities refer to the subjects that output relevant functions which could be system, sub-system, components with different breakdown levels) could be determined after specifying the functions to model. Some entities could be regrouped to reach a proper level of precision. In order to build a hierarchical and readable model, the break down structure should be then specified. The division method of system breakdown structure is declared in 3.1.2. The regroupment and the structure should be validated by designers to ensure correctness.

d) Analysis the functions and services to be modeled

After Process c), internal functional analysis (such as building function flow diagrams) has to be prepared to identify all functional chains contributing to the functions. List the elementary functions and relevant entities contributing to the main functions, and the following information should be collected:

- blocks / entities involved in the transmission of the elementary functions
- for every blocks, elementary input and output functions connected
- for every output of an entity, input elementary functions needed
- relevant states of the inputs and outputs (failed or normal, etc.)
- possible specific dependency polynomial concerning an output state
- the physical states of the equipment-level entities (only the bottom-level entities must have physical states corresponding the failure modes themselves)
3.1.2. Rules and constraints

This part declares the rules and constraints that should be focused on.

a) Interface definition

The integration of different supplier models into a higher level model would be failed if the supplier models own different interface types. Therefore, data transmitted between models shall have the same type and type family name, which should be determined as soon as possible, even in the aircraft-level design phase. The process principles could be summarized as follow:

- Step 1: aircraft level function architecture is defined (definition of systems involved). The integrator can define all the interfaces in the document that gathers all the links {emitter / receiver / data transmitted}
- Step 2: Emitter / Receiver / Integrator agree on type / family name and store it in another document.

b) Model decomposition structure

The decomposition structure could be useful to improve the readability and reduce human errors. A practical decomposition levels are shown in Fig.2.

![Fig. 2 Model decomposition level](image)

In the first level, the system and external entities could be defined. In the second level, the system is decomposed into subsystems and the function networks formed by sub-systems are defined. Meanwhile, all output functions of sub-systems could be integrated into the output block. In the third level, the subsystems, output blocks and external blocks of Level 2 could be decomposed to declare the functional entities composing the subsystems, the function groups contained by the output block of Level 2, etc. The graphical decomposition structure pattern is shown in Fig.3.
3.2. **Model construction**

3.2.1. **Sub-process of model construction**

After collection of information, it's time to start model construction and build a hierarchical model like Fig.3. The model could be constructed from the top level to the bottom. The precise process is as follows.

a) **Build the top level model**

   Top level model contains the system block, the external input and output blocks and the connection links among them.

b) **Build the second level model**

   The second model contains the sub-system block, the output block and the connection links among them.

c) **Build the third level model**

   The third level model contains not only the elementary block, the elementary input and output and the connection links among them, but also the output function blocks decomposed by the output block of Level 2. There are two ways to connect the elementary blocks as shown in 3.2.2.

d) **Edit the elementary block**

   Edit the elementary block and specify its reliability data, input and outputs, and the polynomial logics.

e) **Create missions and phases**
Create missions and phases to declare the architecture characteristics during different flight phases and missions.

3.2.2. Rules and constraints

a) Naming rules

When the system model is made up of different sub-system models developed by different teams, naming rules must be unified. Each component / equipment / type / operator name should respect a dedicated nomenclature. For example, each component / equipment / type / operator might have the following kind of reference:

E24CAL-0001-001-001-00-1 with digits definition as
- 1: Project code (A320 : A320_AIRCRAFT)
- 2-3: ATA chapter / reference of the sub level system in the product breakdown
- 4: Item code (C: Component, E: Equipment, T: Type, O: Operator)
- 5-6: Partner code
- 7-10: Sequence number
- 11-13: Second sequence number
- 14-15: Special code
- 16: version

b) Polynomial connection of elementary blocks

There are two ways to connect the elementary blocks with the main functions and services to be provided by the system as shown in Fig.4.

- Connect according to the topological structuration of the functional network. Describe all elementary functional flows exchanged by the different entities, and identify those logical and functional chains constituting at the end the different contributions to the main service provided.
- Connect according to system composition. Determine the elementary entities contributing to the main function, and connect these entities with functions issued from the supporting entities directly to the main entity.

![Fig. 4](image-url)

Fig. 4 (a) Modelling according to the topological structuration; (b) Modelling according to the composition
3.3. Model verification and validation

3.3.1. Sub-process of model V&V

Model V&V is to ensure the correctness and completeness of models.

a) **Theoretical V&V**

V&V Checklists, FMEA/FTA/Reliability diagrams could be used to support V&V process. FMEA/FTA/Reliability diagrams could be used to check if the results are consistent with the previously known causes of failure conditions. V&V checklists should contain as many requirements as possible to guarantee the correctness and completeness as talked detailed in 3.3.2.

b) **Practical V&V**

Practical V&V is valid only if a physical model or real sample of the model can be used. Practical V&V works through producing real failures on the real system, and check if there is a coherency between the real effects produced and those predicted by the FMECA generated from the model.

3.3.2. Rules and constraints

a) **Theoretical V&V should contain contents like applicability, input requirements and means of compliance, model assumptions/limitations.**

Applicability could be verified through check the system baseline of configuration managements, referenced documents and system designs. These input requirements could be failure conditions probability, high level requirements, in service lessons learnt and interfaced system requirements. As to assumptions/limitations, one should check the correctness of model perimeter, abstraction level, behavior, etc.

b) **Practical V&V is valid only if a physical model or real sample of the model can be used.**

4. Model management

For a complex system, the difficulty to maintain its model is huge. Especially along with the continuous updates of system design, the workload to refine models is hard to predict and there is a high possibility of introducing new errors in the modification process. Use of shared database to manage model is put forward to enhance management ability and decrease the difficulty of model modification.

4.1. The structure of shared database

A shared database could improve the model reusability and the efficiency of modelling. The shard database could store common information and integrated into the modelling tools. The database could contain the following sub-libraries.

a) **Safety architecture pattern library**

Safety architecture pattern describe the pieces of architectures or micro-architectures widely used in the safety-critical systems [8], such as redundancy design. The properties of these micro-architectures are stored in the library. For example, for a double-redundant and cold-standby system, properties like a node within two blocks, internal and external inputs and outputs of the node, and transition relations between
two blocks, are defined in the library. When one has to build such a system, the work is only to call this node in library and assign these properties.

b) **Standard component library**

The failure modes of common electronic equipment, hydraulic equipment and standard mechanical parts are relatively fixed. Altarica nodes can be established for these components and stored in the *Standard Component Library*.

c) **Failure mode library**

Failure mode library is used to store common failure modes. The failure mode could be directly called in the process of describing Altarica nodes. The common failure modes include: leak, binding, error-input, no-input, short-circuit, open-circuit and so on. Failure modes could be classified in accordance with the equipment type.

d) **Rules and limitation library**

Some rules and constraints about the models could be recorded in the rules and limitation library, such as unified naming rules and that equipment of Level 3 must have its own physical failure state. In the model V&V phase, these rules could be called to check the model automatically.

4.2. **Database call relationships**

In the process of shared database design, the call relationships of different libraries should be considered to further enhance the model reusability and improve the modelling efficiency. For example, when calling the standard component library, one can call failure mode library at the same time to add failure modes for the component node. When calling the redundancy architecture in the safety architecture pattern library, one can call the standard component library to instantiate the blocks in the architecture pattern. Fig.5 describes the call relationships.

![Fig. 5 Database call relationships](image-url)

4.3. **Model update**

The shared database could be used to simplify the model update. Before constructing the model, a special shared database should be established for the project. And then, the model could be established
based on the special database. When the parameters of equipment changes, one only need to update the relevant information in the database, and then the whole model could follow the update.

5. Case study

The modelling process is applied to a hydraulic system of a helicopter with the detailed results described as follows.

5.1. Information collection

a) Specify the system architecture, external entities and external interfaces

The hydraulic system contains two main hydraulic subsystems (named A and B) and a cold backup C. Subsystem A and C provide pressure and flow for the left cavity of the rotor booster. B is designed for the right. When A failed, C starts working. Subsystem A and B are powered by the engine. However, C relies on an electric machine. The system architecture, external entities, and interfaces are shown in Fig.6. Since C doesn’t work at first, its initial state is spare (configuration information).

Fig. 6 Hydraulic systems Architecture

b) Build the function tree to specify the functions and services to model

The function tree is translated into Table1. According to the FHA results [9], A, B and C failed to provide hydraulic pressure and flow are a catastrophic event. That’s, all functions in Tables 2 have to be modelled.

Table 1 Function tree of the hydraulic system

| Function of Level 1 | Functions of Level 2 | Functions of Level 3 |
|---------------------|----------------------|----------------------|
| providing pressure and flow for the rotor booster | providing pressure and flow for the left cavity of the rotor booster | providing pressure and flow from A |
|                      | providing pressure and flow for the right cavity of the rotor booster | providing pressure and flow from B |
|                      | providing pressure and flow from C |

 c) Specify system breakdown structure
According to the decomposition method in Chapter 3, the hydraulic system is decomposed as shown in Table 2.

Table 2 the breakdown level of the hydraulic system

| Level 1 | Level 2 | Level 3 |
|---------|---------|---------|
| Hydraulic system | Oil tank | ... |
| Sub-system A | Pump | Providing pressure and flow for the left cavity of the rotor booster |
| Assembled valves | ... | Providing pressure and flow for the right cavity of the rotor booster |
| Pipe | ... | |
| Sub-system B | | |
| Sub-system C | | |

Providing pressure and flow for the rotor booster

External power source

| Engine | Engine |
|--------|--------|
| Electric machine | Electric machine |

| Rotor booster | Left cavity of the rotor booster | Left cavity of the rotor booster |
|---------------|---------------------------------|---------------------------------|
|               | Right cavity of the rotor booster | Right cavity of the rotor booster |

*d) Analysis the functions and services to be modeled*

Taking providing pressure and flow for the left cavity of the rotor booster from A as an example, the function flow diagram is built in Fig.7. Through function and failure analysis, one can specify elementary blocks, their inputs and outputs and physical failure states as well as the output polynomials.

Fig. 7 Function flow of Sub-system A
Table 3 Function analysis and failure analysis results

| Elementary block | Physical state itself | Input function (and state) | Output function (and state) | Output polynomial |
|------------------|-----------------------|----------------------------|-----------------------------|------------------|
| Tank             | Leak                  |                            | Providing oil (normal, failed) | And             |
| Pump             | Stuck                 | Providing oil (normal, failed) | Providing hydraulic pressure and flow (normal, no-pressure, low-pressure, high-temperature, etc.) | The polynomial should be described for each abnormal output function state. |
|                  | Cracked               |                            |                             |                  |
| Assembled valves | Leak                  | Providing hydraulic pressure and flow (no-pressure, low-pressure, high-temperature, etc.) | Distributing pressure and flow (normal, no-pressure, low-pressure, high-temperature, etc.) | The polynomial should be described for each abnormal output function state. |
| Pipe             | Broken                | Distributing pressure and flow (normal, no-pressure, low-pressure, high-temperature, etc.) | Transmitting pressure and flow (normal, no-pressure, low-pressure, high-temperature, etc.) | The polynomial should be described for each abnormal output function state. |

5.2. Model construction

Construct the model from the top level to the third level. In this case, the Simfia toolsets provided by EADS APSYS were adopted. The partial model is shown in Fig.8.

5.3. Model V&V

In this case, “Sub-system B can't provide pressure and flow” was taken as a top event to generate a fault tree as shown in Fig.9. Through validation the correctness of this fault tree by relevant system designers, the correctness of this model is verified correct partially.
6. Conclusion

A practical safety modelling methodology for aircraft systems is proposed in this paper. The information necessary to build a complete and correct model, the model construction process and model V&V methods are specified as well as the rules and constraints. The methodology could normalize the safety modelling process and enhance the model readability, correctness and completeness. In order to improve model management and enhance the reusability and modification capacity, the management based on a shared database is also discussed in this paper. In the following research, the modelling differences among different aircraft systems would be studied according to their own characteristics. And more attention should be paid on how to systematically create a more structured sharing database and make use of it more efficiently.

Acknowledgements

This paper is finished with the help of my dear workmates. They share lots of experiences on how to build a dependable model and avoid unnecessary man-made faults. Their advices inspired me, and promoted the accomplishments of this paper.

References

[1] A. Joshi, M. Whalen, M. Heimdahl. Model-based safety analysis final report, NASA contractor report, NASA/CR-2006-213953, 2006.
[2] Bieber, P., Bougnol, C., Castel, C., Heckmann, J.-P., Kehren, C., Metge, S., Seguin, C. 2004. Safety Assessment with AltaRica - Lessons learnt based on two aircraft system studies. In 18th IFIP World Computer Congress, Topical Day on New Methods for Avionics Certification, August 26th, 2004, Toulouse (France), LNCS series (copyright Springer-Verlag): 505-510
[3] Humbert, S., Seguin, C., Castel, C., Bosc, J-M. 2008a. Deriving Safety Software Requirements from an AltaRica System Model. In: SAFECOMP08, Michael D. Harrison and Mark-Alexander Sujan.
[4] R. Adeline, J. Cardoso, P. Darfeuil, S. Humbert, and C. Seguin. Toward a methodology for the AltaRica modeling of multi-physical systems. ESREL 2010, Rhodes, Greece, 2010.
[5] C. Kehren et al., Advanced Multi-System Simulation Capabilities with AltaRica, in proceedings of the International System Safety Conference, 2004.
[6] C. Kehren et al., Architecture patterns for safe design, in proceedings of the first AAAF Conference on Complex and Safe System Engineering, 2004.
[7] G. Point and A. Rauzy. Altarica - constraint automata as a description language. European Journal on Automation, 1999. Special issue on the Modelling of Reactive Systems.
[8] A. Rauzy. Mode Automata and their compilation into fault trees, Reliability Engineering and System Safety, 78:1-12, 2002.
[9] D Kritzinger. Aircraft system safety: Military and civil aeronautical applications. Cambridge: Woodhead, 2006, p 43-56/