A Practicable MBSA Modeling Process Using Altarica

Shaojun Li* and Duo Su

Airworthiness and Safety Technology Research Center
China Aero-ploy Technology Establishment, AVIC
Beijing, China
jinerli@126.com

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, it's quite difficult to build a complete, accurate and consistent safety model, especially for dynamic complex systems. 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 are declared for each phase. At last, a hydraulic system is taken as an example to show how to apply the safety modeling methodology in practical.

Keywords: Safety, model based safety analysis, formal modeling, modeling process, Altairca.

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 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, modeling a safety model needs large amounts of information, such as interfaces, system architectures, function flows and failure data, etc. Insufficient information collection could not only increase modeling difficulties but also delay safety assessment progress. For example, 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). No detailed researches provide reasonable, ordered and limited steps to gather sufficient information for a complete model. Second, with lack of well-defined procedures, many man-made errors are introduced in the

* Corresponding author.

F. Ortmeier and A. Rauzy (Eds.): IMBSA 2014, LNCS 8822, pp. 1–13, 2014.

disordered modeling process. Finally, after modeling, model verification and validation may be ignored to check the consistency between the model and the real system.

To solve the problems above, a practicable safety modeling process using Altarica is proposed in this paper. First, the whole framework of safety modeling is divided into three phases as information collection, model building and model V&V. And then, the sub-processes contained in each phase are depicted as well as their rationality. Finally, one of safety assessment cases which this modeling process is applied to is chosen to prove the validity of this process.

2 Fundamental Principles

2.1 Static Safety Models

Static safety models describe the propagation of the effects of failure modes, also we could say that they model on failure logic. The failure logic modeling (FLM) approach emerged in the 1990s. FLM comes from traditional safety analysis method such as FTA and FMEA， but it overcomes the problems of great difficulty to modify , reuse and application to large systems. A component's failure logic describes how deviations of component inputs (input failure modes) combine with each other and with internal abnormal phenomena (internal failures) to cause particular deviations of behavior of component outputs (output failure modes) as shown in Fig.1.The system's FL are composed from the FL of individual components. Altarica could be used to model failure logics as well as FPTN[6], FPTC[7], Hip-Hops[8],etc.

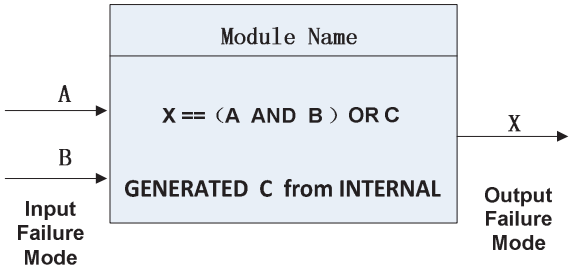

Fig. 1. Failure logic of a component

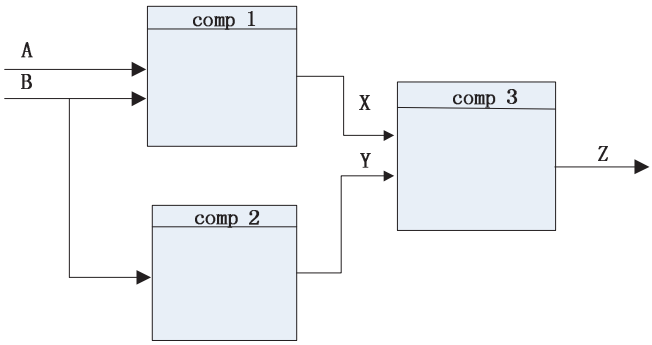

Fig. 2. FPTN notion of a system

According to connectives in the design models, a failure logic model of the system can be composed from the characterizations of individual components by connecting output failure modes of the component to input failure modes of other components, as shown in Fig. 2. Boxes can be seen a set of fault trees. Causes of output failure modes (such as Z in Fig. 2) can be deduced through the graphs and fault trees could be built in that way.

2.2 Dynamic Safety Models Using Altarica

Altarica is a formal language that has been widely used in the aviation areas[9]. Altarica could not only be used to model static failure logic, but also build a dynamic safety mode automaton. The characters of Altarica are introduced as follows.

AltaRica language is hierarchical and compositional. Each component is described by a mode automaton [10]. 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.

2.3 Simifa

Many tools have been developed to support building and analyzing Altarica models. In this paper , we adopted tool Simfia™ EADS APSYS 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).

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 modeling process, we have to stress that identification of risks in a structured and systematic way is the basis of normalized, systematic and structured safety

modeling process. After finishing identification of risks or hazards, the modeling process could start.

The modeling process contains three phase: information collection, model construction and model verification and validation. Each phase owns different sub-processes as shown in Figure 3.

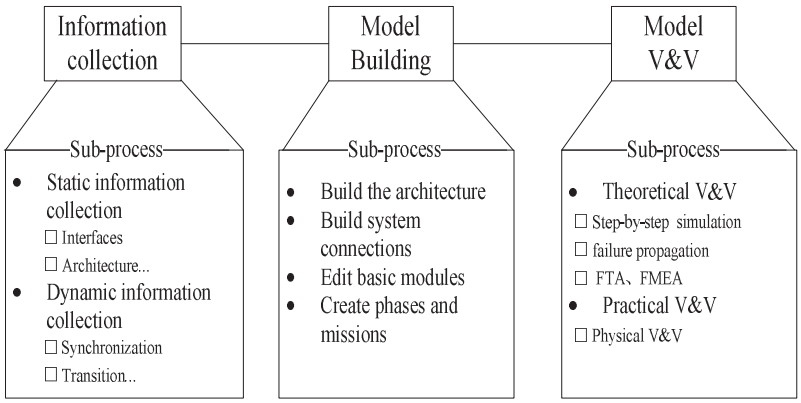

Fig. 3. Modeling phases and relevant processes

3.1 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. The information used to build static and dynamic modes is concluded as follows.

3.1.1 Static Information Collection

The static information collection process is shown in Fig.4.

1. 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 modeling system, the model may be later used to be integrated into a much higher level model. Meanwhile, the modeling 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.

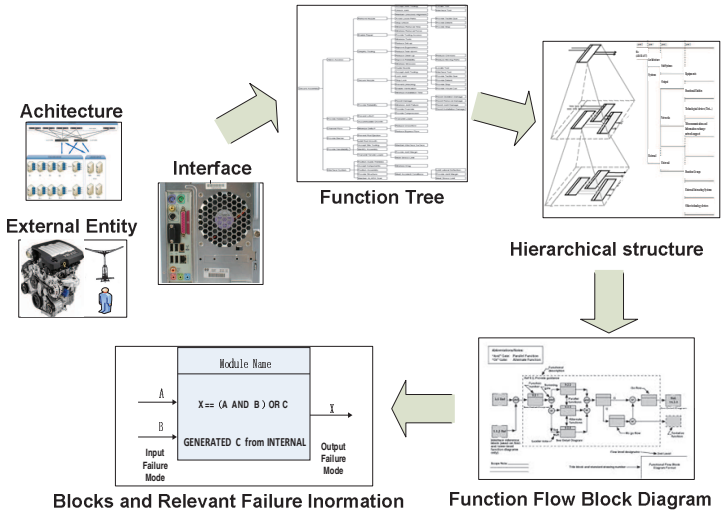

Fig. 4. Information collection process

2. 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.

3. 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 regroupment and the structure should be validated by designers to ensure correctness.

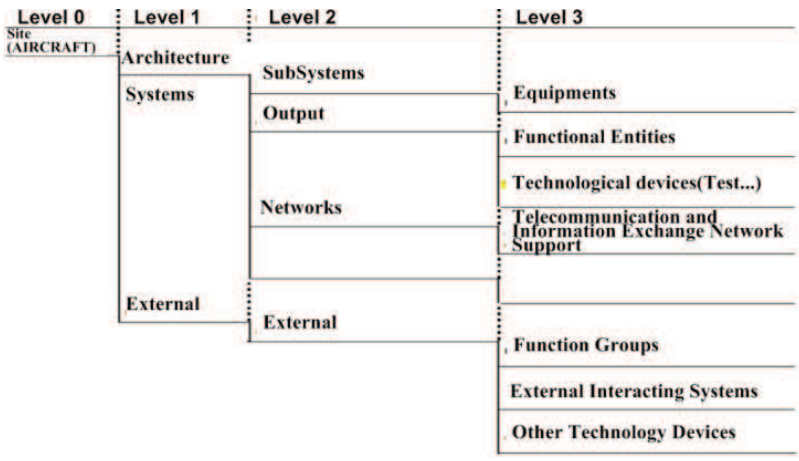

Fig. 5. Model decomposition level

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 Figure 6.

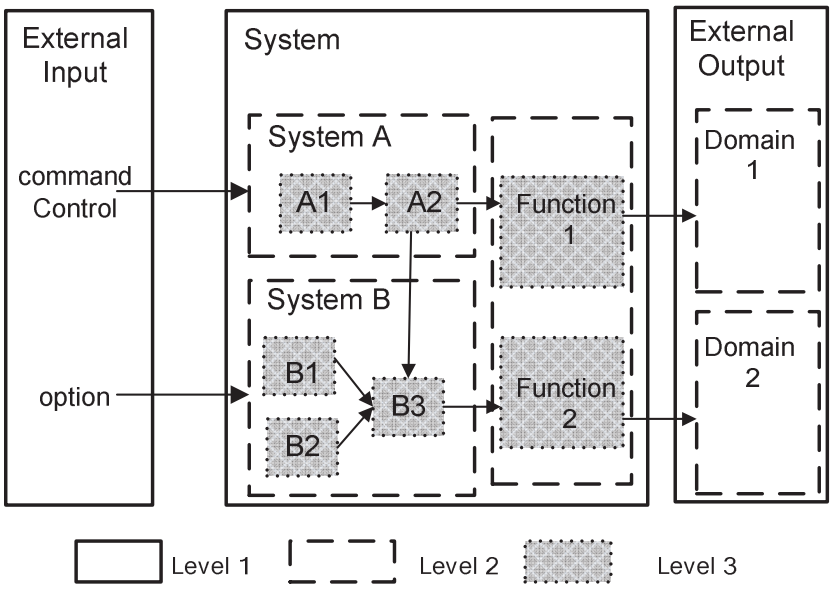

Fig. 6. Structured Model Pattern

4. Analysis the functions and services to be modeled

After Process 3, 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:

(a) blocks / entities involved in the transmission of the elementary functions
(b) for every blocks, elementary input and output functions connected
(c) for every output of an entity, input elementary functions needed
(d) relevant states of the inputs and outputs (failed or normal, etc.)
(e) possible specific dependency polynomial concerning an output state
(f) 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 Dynamic Information Collection

Dynamic information is used to build Altarica mode automata. The static information above is also helpful. However, the static information should be processed. Since the dynamic model is based on the theory of mode automata, a state diagram is quite useful to gather the necessary dynamic information.

A state diagram could state the dynamic information as follows:

- inputs and outputs of the object
- states that the object owns
- the polynomials of outputs, inputs and self-states
- the way that one state transformed to another
- the synchronized transition events happened to a system
- the initial state

State explosion is a problem that maybe introduced by the dynamic models. In order to reduce the states and simply the models, the states of certain object should be determined by its function failure modes instead of hardware failure modes. Hardware failure modes are usually much more than function failure modes, and it is difficult to determine its output when hardware failure mode occurs.

3.2 Model Construction

After collection of information, it's time to start model construction and build a hierarchical model like Fig.6. The precise process is as follows.

1. Build the architecture

According to the architecture and system breakdown structure, build a hierarchical architectural model. Determine the constituent blocks of different level models. It starts from the system level, and then the sub-system level until the bottom item level.

2. Build the connection among different levels and different blocks

According to the function flow charts, the connection among different blocks of different levels could be built. There is no need to clearly define the characters of connection, which could be detailed in the following steps. However, the pre-defined connection could avoid missing input information when editing the blocks in the next step. It would be much more reasonable to connect the blocks from the top level to the bottom level.

There are two ways to connect the blocks with the main functions and services to be provided by the system. For dynamic models， only is the first is allowed since dynamic models represent actual operation situations.

- 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.

3. Edit the bottom blocks

For static models, there are three steps to edit the elementary blocks or bottom blocks. First, define the input and output of the blocks. Then, define the states of the each block. At last, define the logic relations among outputs, inputs and states.

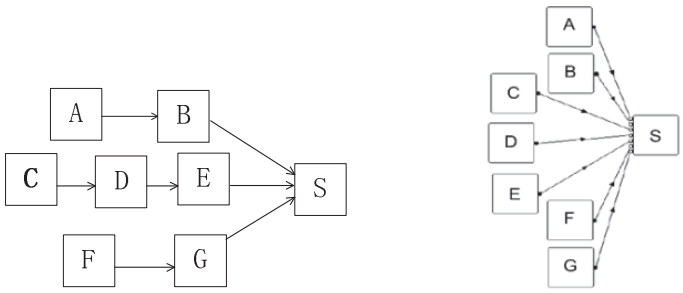

Fig. 7. (a) Modeling according to the topological structuration; (b) Modeling according to the composition

For dynamic models, more work needs to do except for the above three steps. The forth step is define the state transitions. Some transitions may happen at the same time. After editing the elementary blocks, the synchronous transition should be defined in the block of the corresponding higher level.

4. Create missions and phases

Create missions and phases to declare the initial state. When safety analysts try to build static models, this step is necessary if the system has different configurations during different phases. For dynamic model, it is always necessary to define the initial state.

3.3 Model Verification and Validation

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

1. 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.

Step-by-step simulation is supported by lots of analysis toolsets today. For static models, it is possible to choose a failure state, and observe how it influences other items. Similarly, after setting up a trigger event (or transition), it is possible to observe how the system operates. In that way, we could verify if the model is built correct.

2. 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.

4 Case Study

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

4.1 Information Collection

1. 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.8. Since C doesn't work at first, its initial state is spare (configuration information).

2. Build the function tree to specify the functions and services to model

The function tree is translated into Table1. According to the FHA results, 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 modeled.

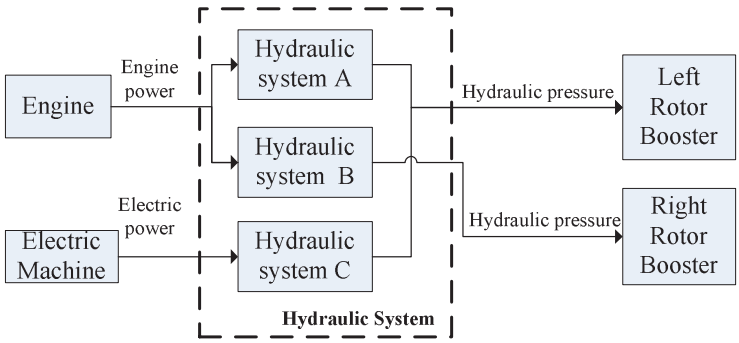

Fig. 8. Hydraulic systems Architecture

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 from C
	providing pressure and flow for the right cavity of the rotor booster	providing pressure and flow from B

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

4. 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.9. Through function and failure analysis, one can specify elementary blocks, their inputs and outputs, physical failure states and output polynomials.

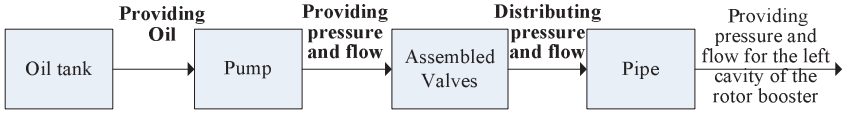

Fig. 9. Function flow of Sub-system A

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 Cracked	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.
…	…	…	…	...

5. State diagram analysis

Take the pump of sub-system B as an example to show how to determine the dynamic information. In order to take a complete description of state charts of the pump, we need to specify its inputs and outputs, states, asserts, transitions, synchronism, and initial states one by one. As talked in 3.1.2, to reduce the number of states, it is sensible to simply the states. Therefore, for the pump, its dynamic information could be assumed as follows:

- two states of inputs: normal(with oil providing by the tank), and failed (without oil providing)
- two states of outputs: normal (providing oil with certain pressure and flow), and failed (disabled to provide oil with certain pressure of flow)
- two self-states: normal and failed
- asserts: when the pump works normal and its input is normal, then its output is normal; If its input is failed or the pump is failed, its output is failed
- Transition: when the pump failed, its state transformed from normal to fail.
- Synchronism: when the pump's state converts to failed, the state of pump of sub-system C turns to working from waiting.
- initial state: normal

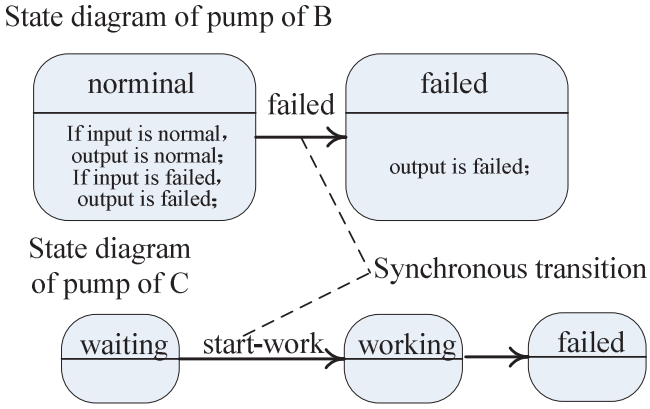

Fig. 10. State diagram of pump of B

4.2 Model Construction

Construct the model according to the procedures talked in 3.1. In this case, the Simfia toolsets provided by EADS APSYS were adopted. The partial model is shown in Fig.11.

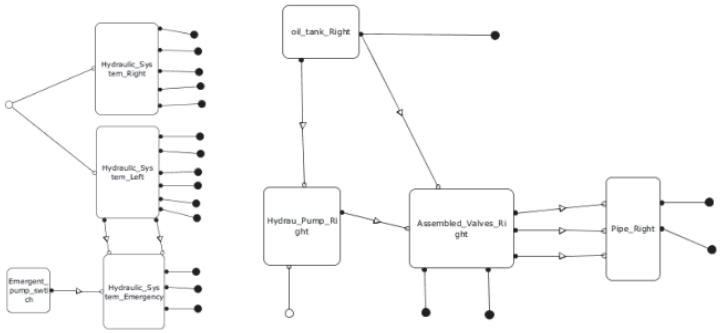

Fig. 11. (a) System level model; (b) Sub-system level model

4.3 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.12. Through validation the correctness of this fault tree by relevant system designers, the correctness of this model is verified correct partially.

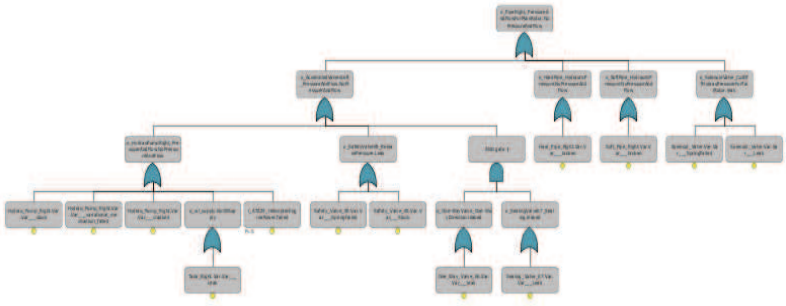

Fig. 12. Fault tree of "System B can't provide pressure and flow"

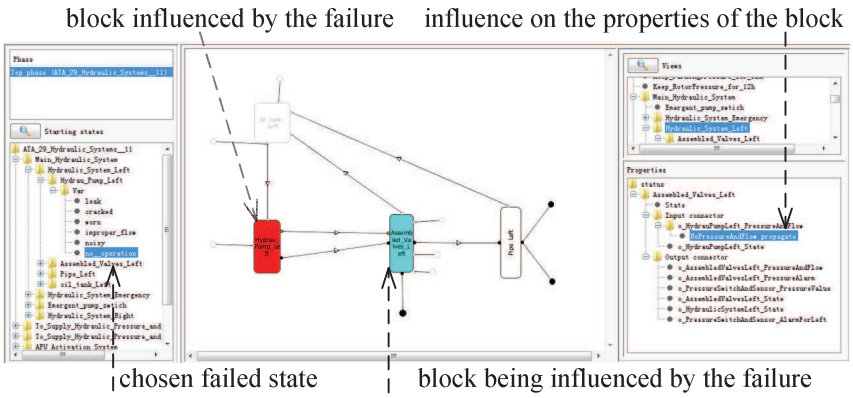

Fig. 13. Step by step simulation

Step-by-step simulation could help to verify the correctness of models. Choose the failure state of one block, and check its influence on other blocks step by step.

In Figure 13, the "No-operation" state of pump of A is chosen, the red block (the left block in the Fig.12 which is displayed in red color in the toolset) stands for the block that has been influenced by the failure, and the blue block (the right block in the Fig.12 which is displayed in blue color in the toolset) stands for the block being influenced right now. The detailed influence is shown on the value of the properties on the lower right corner of the figure.

5 Conclusion

A practicable safety modeling process using Altarica is proposed in this paper. The methodology could normalize the safety modeling process and enhance the model readability, correctness and completeness, and avoid unnecessary modeling errors.

References

1. Joshi, A., Whalen, M., Heimdahl, M.: 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.: 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. IFIP-AICT, vol. 155, pp. 505–510. Springer, Heidelberg (2004)
3. Humbert, S., Seguin, C., Castel, C., Bosc, J.-M.: Deriving Safety Software Requirements from an AltaRica System Model. In: Harrison, M.D., Sujan, M.-A. (eds.) SAFECOMP 2008. LNCS, vol. 5219, pp. 320–331. Springer, Heidelberg (2008)
4. Adeline, R., Cardoso, J., Darfeuil, P., Humbert, S., Seguin, C.: Toward a methodology for the AltaRica modeling of multi-physical systems. In: ESREL 2010, Rhodes, Greece (2010)
5. Kehren, C., et al.: Advanced Multi-System Simulation Capabilities with AltaRica. In: Proceedings of the International System Safety Conference (2004)
6. Fenelon, P., McDermid, J.A.: An Integrated Toolset for Software Safety Analysis. Journal of Systems and Software (1993)
7. Paige, R., et al.: FPTC: Automated Safety Analysis for Domain-Specific Languages. Models in Software Engineering, 229–242 (2009)
8. Papadopoulos, Y., Walker, M.: Engineering failure analysis and design optimisation with HiP-HOPS. Engineering Failure Analysis, 590–608 (2011)
9. Point, G., Rauzy, A.: Altarica - constraint automata as a description language. European Journal on Automation (1999)
10. Rauzy, A.: Mode Automata and their compilation into fault trees. Reliability Engineering and System Safety 78, 1–12 (2002)