Aircraft operational reliability - A Model-based approach and case studies

Kossi Tiassou

THESE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par *Institut National des Sciences Appliquées de Toulouse*

Discipline ou spécialité : *Systèmes Informatiques*

Présentée et soutenue par *Kossi Blewoussi TIASSOU*
Le *6 février 2013*

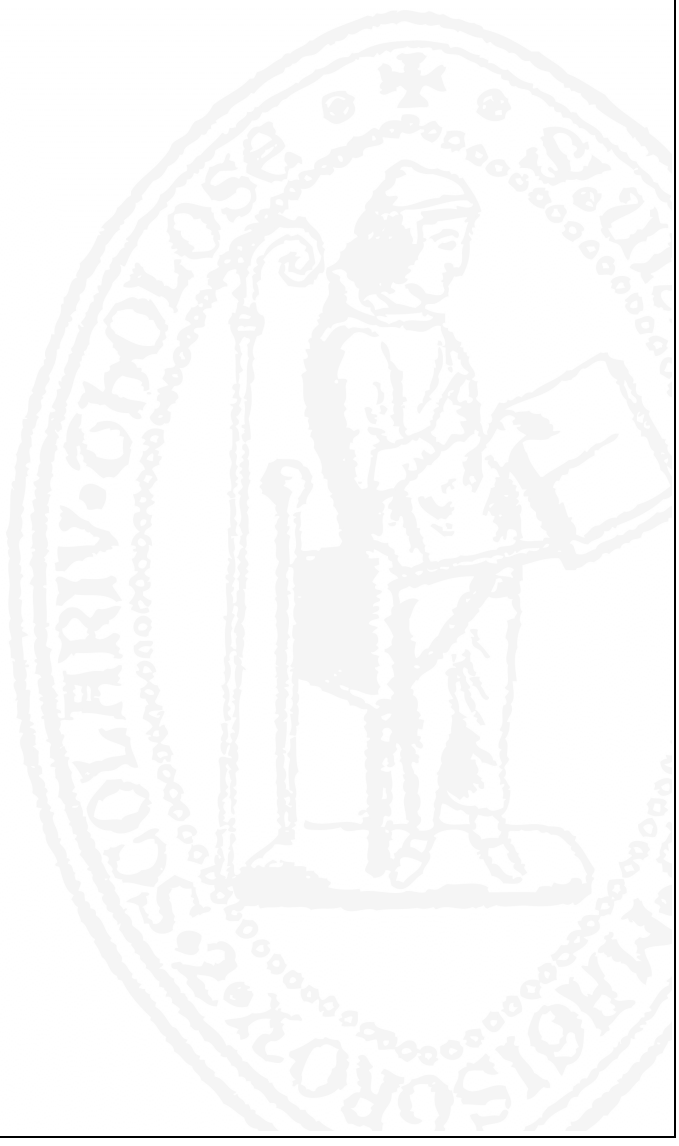
Titre : *Fiabilité opérationnelle des avions - Approche basée sur les modèles et cas d'étude*

Aircraft operational reliability — A Model-based approach and case studies

---

JURY
Karama KANOUN : Directeur de thèse
Christel SEGUIN : CoDirecteur de thèse
Felicita DI GIANDOMENICO : Rapporteur
Jean-Marc BOSC : Rapporteur
Mohamed KAÂNICHE : Examinateur
Kamel BARKAOUI : Examinateur
Chris PAPADOPOULOS : Invité

Ecole doctorale : *SYSTEMES*
Unité de recherche : *LAAS-CNRS*
Directeur(s) de Thèse : *Karama KANOUN et Christel SEGUIN*
Rapporteurs : *Felicita DI GIANDOMENICO et Jean-Marc BOSC*



Acknowledgements

I would like to express my gratitude to all those who have made this dissertation journey possible and have ensured its successful achievement.

The dissertation work has been carried out at the *Laboratoire d'Analyse et d'Architecture des Systèmes* of the French National Center for Scientific Research (LAAS-CNRS), within the Dependable Computing and Fault Tolerance research group (TSF). I would like to thank Raja Chatila and Jean Arlat, respectively former and current directors of LAAS, and Karama Kanoun the head of the TSF research group for having offered me the opportunity to carry out my PhD thesis work at LAAS.

The work has been carried out in the context of an Airbus project @Most (Airbus Maintenance Operations Solutions and Technologies), especially the DIANA (Decision Impact ANAlysis) package which involves collaborators from Airbus, ONERA (*Office National d'Etudes et de Recherches Aéronautiques*) and ISAE (*Institut Supérieur de l'Aéronautique et de l'Espace*). I would like to express my gratitude to all of them, especially Chris Papadopoulos, for the time devoted. He had extensively contributed to the work with fruitful suggestions and comments.

My sincere thanks to Felicita Di Giandomenico and Jean-Marc Bosc, the thesis reviewers (*rapporteurs*), for their interest in the subject and the time devoted. I also thank Kamel Barkaoui for having accepted to be part of the thesis committee.

My utmost gratitude goes to my dissertation advisors, Karama Kanoun, Christel Seguin and Mohamed Kaâniche, who have accepted to undertake this adventure with me. They have been patient and indulgent during all the thesis work. I have learnt a lot from their advices and guidance.

I would like to thank all the members of the Dependable Computing and Fault Tolerance research group who somehow supported me. I am grateful to Yves Crouzet for his help, especially during the thesis defense preparation, and Sonia De Sousa for her effort in carrying out the administrative tasks related to the defense organization. I am also thankful to all the fellow PhD students with whom I have mostly interacted. Amongst my fellow colleagues, the effort made by my former officemate Ossama Hamouda in welcoming me and giving me first guidance at the beginning of the thesis is acknowledged and appreciated.

I express my gratitude to all my friends who have encouraged me and have believed in the achievement of this project.

My family has been far from me but has kept this eagerness in asking about how the work was going. I was encouraged by many persons, from both the close and the extended family members. You have been of great support. Thank you all of your kindness and support.

Abstract

Dependability assessment, by system manufacturer, during aircraft design, based on stochastic modeling, is of common practice, but model based operational dependability assessment online, during missions' achievement, is seldom done. Usually, the stochastic assessment addresses aircraft safety.

This thesis addresses aircraft operational dependability modeling to support mission and maintenance planning, as well as the achievement of the missions. We develop a modeling approach, based on a meta-model that is used as a basis: i) to structure the information needed to assess aircraft operational reliability and ii) to build a stochastic model that can be updated dynamically. The update concerns the current state of the aircraft system, a mission profile and the maintenance facilities available at the flight stop locations involved in the mission. The aim is to enable operational reliability assessment online. Two case studies, based on aircraft subsystems, are considered for illustration. We present examples of evaluation results that show the valuable role of operational dependability assessment during aircraft mission.

Contents

Figures

Tables

Acronyms

@MOST	Airbus Maintenance Operations Solutions and Technologies
ACARS	Aircraft Communication Addressing and Reporting System
ARP	Aerospace Recommended Practice
ATA	Air Transport Association
BITE	Built In Test Equipments
DIANA	Decision Impact ANAlysis
EASA	European Aviation Safety Agency
ECAM	Electronic Centralized Aircraft Monitoring
EMF	Eclipse Modeling Framework
FAR	Federal Aviation Regulations
FCOM	Flight Crew Operating Manual
FDE	Flight Deck Effects
FMECA	Failure Modes, Effects and Criticality Analysis
FTA	Fault Tree Analysis –
LRU	Line Replaceable Unit
MCC	Maintenance Control Centre
MEL	Minimum Equipment List
MMRR	Minimum Mission Reliability Requirement
MOF	Meta-Object Facility
MR	Mission Reliability
MSG	Maintenance Steering Group
MTTF	Mean Time To Failure
PFR	Post Flight Report
PMS	Phased Mission System
QRH	Quick reference handbook
SAE	Society of Automotive Engineers
SAN	Stochastic Activity Network
SR	System Reliability

Introduction

Air transportation has become a common means of travel for a growing number of people. With this increasing interest in air transportation and the competitive market aircraft operators have to deal with, it is essential for aircraft manufacturers and airlines to develop means that can support aircraft operation for an optimal exploitation. New approaches need to be developed to improve operational capabilities so as to achieve and maintain higher levels of service delivery, minimize disruptions and avoid economical losses due to inoperability and customer dissatisfaction.

The @MOST (Airbus Maintenance Operations Solutions & Technologies) project has been set up by Airbus to examine new approaches to aircraft operation and maintenance, that can enhance operability and minimize costs. The principal target of the project is to develop predictive means in order to achieve operation efficiently and accomplish maintenance activities just in the right time.

In the context of @MOST, the DIANA (Decision Impact ANAlysis) project aimed at developing a model-based dependability assessment framework that can be used to analyze and ensure operational dependability, and by this way ensure success and efficiency in aircraft operations with regard to disruptions related to failures. For this purpose, the current approach that focuses on the dependability analysis during the aircraft design phase cannot be considered sufficient. The design phase analysis is most of the time concentrated on safety assessment or, when addressing operational reliability, the analysis focuses on the relative perceived reliability of one technology over another. In addition, the dependability assessment framework is intended to be used while the aircraft is in service so as to consider the current operational information for an adapted dependability assessment. To the best of our knowledge, model-based dependability assessment, in real time, during operation, has not yet received the focus it deserves in the dependability assessment community. Yet, the constant demand for efficiency and the evolving nature of today's systems require the use of accurate data in operation for up-to-date dependability assessment so as to support choices.

The research summarized in this dissertation has been carried out in the context of the DIANA project and concerns the development of a model-based dependability framework for assessing aircraft operational dependability, focusing especially on the model construction and the use of the model during missions' achievement. The objective of the dissertation is the establishment of the dependability model that can enable operational dependability assessment while an aircraft is in service, so as to improve the likelihood of success in achieving its missions. Aircraft systems dependability models have already been considered in the literature, especially for safety analyses during the design phase, but the scope of the assessment in DIANA requires a dedicated study to tackle the aspects that are not straightforward. The substantial motivations for carrying out the dissertation work are presented in the beginning of chapter I. These are principally the need to take into account all operational relevant constraints, instead of the special events related to safety, and the need to consider current operational information for the assessment.

To identify the role that dependability assessment can play during aircraft operation, and understand its importance in aircraft maintenance planning, it is necessary to have an overview of the different ways that aircraft operations can be carried out, and take a look into the underlying principle of the maintenance practice in the field. The construction of the model-based assessment framework, especially the operational and maintenance aspect requires, in addition to dependability analysis techniques, an insight into aircraft operations and maintenance together with how they are organized. The global cultural background for carrying out the work is given in the rest of chapter I. We present dependability concepts and analysis methods, give an overview of aircraft operation and describe the maintenance policies that support the current aircraft maintenance. We also present a review of works aiming at improving aircraft operability, from modeling studies contributing to safety reinforcement, through design and maintenance planning support studies for operational dependability enhancement, to post operational disruptions management. The objective is to clarify the place that model-based dependability assessment will take in the process of improving aircraft operability and avoiding operational disruptions.

The scope of the model-based dependability assessment framework, including the role of model-based dependability assessment in the process to achieve continuous operability, is presented in the end of chapter I. That is, as the aircraft may have to achieve different kinds of missions with different requirements, the assessment framework will be providing evaluations of the probability to succeed missions or the risk of encountering an adverse situation, considering the current operational information. The evaluation concerns the adaptation of the mission profile and maintenance planning to the current operational state of the aircraft. Missions and maintenance planning can be adjusted based on the evaluation. The assessment can be done after major events during operation.

The second chapter aims at presenting the developed modeling approach. The challenges related to the construction of the model are addressed. The model has to enable the evaluation of relevant reliability measures. Besides the complexity of aircraft systems, the consideration of the current information in operation so as to have an adapted model must also be tackled. Furthermore, as different types of aircraft do exist, and in order to harmonize the construction of the model, it is worth establishing a common basis for the model construction. We consider updating the model in operation in order to cope with the need to take into account the current operational information, and we consider the best practices in aerospace system dependability modeling to design the model. We establish the common basis for the model construction through meta-modeling. The meta-model is presented based on a detailed specification of the model content, which should help in a concrete construction of the model.

The modeling approach is established considering stochastic state-based technique so as to have a good expressiveness. Thus, any formalism supporting stochastic state space technique can be used to build the model. The model is, in particular, intended to be developed using the AltaRica language that has been used by Airbus to develop models for safety analyses. However, due to the fact that AltaRica and its current supporting tools were mostly developed for qualitative analysis, we have also selected Stochastic Activity Network (SAN) formalism to experiment the quantitative analysis aspects of the model. The project DIANA intends to develop a tool that can support the processing of the model in AltaRica in order to obtain quantitative results. The SAN formalism is supported by an

academic tool and is well known for quantitative dependability and performability analyses. We present a comparative analysis of the two formalisms in chapter III. Methods to transform models between the two formalisms are also examined.

The formalisms are used to develop case studies based on aircraft subsystems. We present the case studies developed using SAN in chapter IV. The implementation of the modeling approach using AltaRica is given in annex.

We firstly consider the rudder control subsystem that consists, inter alia, of flight control computers and servo-controls. Based on the related requirements, we analyze different scenarios of failures, of failure distribution changes due to wear out mechanisms, and of mission changes. Then we consider the electrical power supply subsystem also to analyze its impact on the successful achievement of the missions.

The assessment results show that events, like failures that do not prevent the accomplishment of the mission, can have significant impact on the likelihood of success of the mission.

Finally, we conclude by reminding the problem addressed, and our principal achievements in dealing with it. Possible directions for the future development of the research study carried out are also presented.

I Context and Background

This chapter presents the general background of the work that is reported in this dissertation. The dissertation addresses model-based dependability assessment in the context of aircraft operation. The global objective is to improve aircraft operation by enhancing the service delivery. Dependability concepts and analysis techniques are reviewed together with aircraft operation achievement in order to get an appropriate basic background of the subject and clarify the area that model-based dependability assessment will address. The dependability assessment targets failures and maintenance issues that may result in disruptions. The fundamental principles of aircraft maintenance are also overviewed.

This chapter is organized as follows: The chapter opens up by stating the objective of the work, after which, an overview of dependability concepts and analysis methods is given. This is followed by a description of aircraft operations planning and achievement. An overview of the main considerations in aircraft maintenance and its accomplishment is provided. Subsequently, a literature review of the work related to the subject is presented. Finally, a summary giving the thesis orientation is presented.

I.1 Motivation and Objective

As for any critical system, special attention is paid to an aircraft's dependability issues during its design phase. Safety has always been the major concern and significant works resulting in certification requirements and standards (FAR 25.1309 / EASA CS 25.1309, SAE ARP4754, ARP4761 for examples) have been carried out to analyze and ensure its attainment. Qualitative as well as quantitative techniques have been developed to assess and tackle the major issues that may arise. Aircraft systems are, for example, designed such that no single failure can lead to a catastrophic event. Another consideration based on a quantitative analysis viewpoint, is that the risk of occurrence of such an event must not exceed 10^{-9} per flight hour. These measures lead to several rules and recommendations that must be applied while the aircraft is in service. Programs of regular maintenance activities are also established to maintain an aircraft's functional state. The problem now is the continuous operability of the aircraft while respecting these rules. Indeed, operational issues may arise if for example, due to a failure, the aircraft current mission does not comply with the aircraft functional state. The maintenance team must be able to promptly cope with the problem. Furthermore, the operational dependability analyses at the design phase are based on general assumptions, considering the whole operational life of the aircraft, for average behavior analyses, which may not cover accurately all the various specific situations that may be encountered. An assessment during operations considering the current operational conditions could provide more appropriate analyses.

The objective of the dissertation is to use dependability modeling to analyze and assess in service operations in order to improve the service availability. This consists in developing models that can be used to forecast the behavior of the aircraft considering its operational state, the profile of its mission, if one is assigned, and maintenance activities. The model is intended to be integrated in a tool that can be used whenever needed during the aircraft

operation, to support continuous monitoring of the aircraft operability. The aim is to provide, in every context, as much as possible information on the current state and reliability trend of the aircraft, in order to anticipate adverse situations and increase the likelihood of operation success. The tool is intended to be used by all the actors involved in the aircraft operation. Pilots and flights planning crew need to define and prepare the aircraft missions using the indication on its future behavior. The maintenance team needs to plan and be prepared for maintenance activities using an estimation of the components reliability and the probable time the maintenance will be required.

The model is expected to integrate runtime information when it is solicited to provide an up-to-date indication. Indeed, the indication can be useful only if the model is representative of the current situation in service. Therefore, the different changes and choices that may take place during operations must be taken into account. The different actors involved in the aircraft operation must be given a means that allows them to provide the necessary information (data about the planned operations and maintenance activities).

The work is not specifically dedicated to safety, but rather considers operational requirements that cover safety issues as major property while ensuring a continuous achievement of missions. The fundamental idea is to enable operational dependability assessment in service, based on a model that allows for an adaptation to in-operation situations. However, the model construction can be based on the safety modeling studies performed during the aircraft construction process, and the modeling approach must also offer the possibility to be used for the traditional safety analyses. The latter use case will not be during operation; it will concern the aircraft system manufacturer who will be the builder of the model, using the proposed modeling approach and considering the system specificity.

Before the description giving more details on aircraft operation and maintenance in section I.4, an overview of dependability concepts is presented firstly in section I.2, putting more focus on dependability evaluation in section I.3.

I.2 Dependability Concepts

Dependability is defined in [Avizienis et al. 2000; Avizienis et al. 2004][1] as the ability to deliver service that can justifiably be trusted. A service delivered by a system is its behavior as perceived by its user(s); a user is another system that interacts with the former. The behavior of a system consists in the sequence of states that the system exhibits in order to do what it is intended for. Correct service is delivered when the service implements the system function, i.e., what it is intended to do.
Dependability is a generic concept that is led by three groups of fundamental concepts: its attributes, the threats to its attainment and the means to reach the desired dependability goals.

[1] All the definitions given in this section are from these references.

I.2.1 Dependability attributes

The dependability attributes represent different aspects of the service delivery. They are used to express and analyze the quality of the service delivered or expected from the system. Based on the needs of the user(s), several kinds of attributes can be found, but they are almost compositions or specializations of the following basic ones:

- *Reliability*, which characterizes the continuity of correct service;
- *Availability*, which characterizes the readiness for correct service;
- *Safety*, which characterizes the absence of catastrophic consequences on the user(s) and the environment;
- *Integrity*, which characterizes the absence of improper system alterations;
- *Maintainability*, which characterizes the ability to undergo modifications and repairs.

An additional fundamental attribute is considered while addressing security: confidentiality, which is defined as the absence of unauthorized disclosure of information. *Security* is defined as the concurrent existence of i) availability for authorized actions only, ii) confidentiality, and iii) integrity where 'improper' means 'unauthorized'.

Due to the imperfections generally inherent to all systems, the achievement of these attributes must be interpreted in a relative sense, not in an absolute, deterministic sense. The requirements for the attributes must be specified in terms of acceptable levels, and some of them may not be required for a given system.

I.2.2 Threats to dependability

The threats to dependability are faults, errors and failures. They are the circumstances at the origin of an incorrect service delivery. Their effects deteriorate the level of satisfaction of the dependability attributes.

- A *failure* is an event that occurs when the delivered service deviates from correct service; it is a transition from correct service to incorrect service delivery.
- An *error* is the part of the system state that may cause a subsequent service failure; a failure occurs when an error reaches and alters the sequence of the system external states in which the service consists.
- A *fault* is the adjudged or hypothesized cause of an error.

A system may not always fail in the same way. The ways a system can fail are its failure modes, and they are usually ranked according to their severities. The period during which incorrect service is delivered is called an outage.

Based on the notion of failure, an alternate definition of dependability, which provides a criterion for deciding if the service is dependable, is given as the ability to avoid service failures that are more frequent and more severe than is acceptable.

I.2.3 The means for dependability

To contain the threats and improve dependability, actions may be undertaken from the early design phase of the system, to its use phase. At the current stage of dealing with dependability issues, the means to attain the various attributes can be grouped into four major categories:

- *Fault prevention*, which deals with how to prevent the occurrence or introduction of faults;
- *Fault tolerance*, which deals with how to deliver correct service in the presence of faults;
- *Fault removal*, which deals with how to reduce the number and severity of faults;
- *Fault forecasting*, which deals with how to estimate the present number, the future incidence, and the likely consequences of faults.

Fault prevention is more related to general engineering processes and is handled by quality control techniques employed during design and development of systems. Fault tolerance is carried out via the implementation of error detection and system recovery mechanisms. Redundancies and diversity are part of the techniques used for fault tolerance. Fault removal can be carried out both during the development phase, and during the use phase of a system. Fault removal during the development phase consists of verification, diagnosis and correction. Fault removal during the use phase of a system consists in a corrective or a preventive maintenance. Fault forecasting is conducted by carrying out an evaluation of the system behavior with respect to fault occurrence or activation.

The implementations of the techniques offered by the means are themselves subject to imperfections. Therefore, their combined utilization is strongly recommended in order to enhance dependability. Moreover, they are not mutually exclusive at all; they are complementary. Fault prevention and fault tolerance are aimed at providing the ability to deliver a service that can be trusted. However, their underlying techniques can also be sources of errors and the system internal faults may still produce errors besides. Fault removal and fault forecasting are aimed at reaching confidence in the ability to deliver a trustable service, by justifying that the system is truly dependable.

This dissertation concerns fault forecasting during the use phase, in the context of aircraft operation. It is focused on developing an evaluation framework that can support choices and fault removal activities during the aircraft use phase.

I.3 Dependability Evaluation

Dependability evaluation can be conducted using either a qualitative analysis or a probabilistic analysis.

I.3.1 Qualitative evaluation

Qualitative or ordinal evaluation aims to identify, classify, and rank the failure modes, or the combinations of events (component failures or environmental conditions) that could

lead to system failures. Qualitative dependability evaluation can be conducted based on two majors categories of approaches:

- Approaches driven by an analysis from the causes to the effects, which are aimed at analyzing the consequences of an event, usually a component failure, on the whole system.
- Approaches that are based on a backward analysis, which aim to characterize the possible causes of a given situation.

The most popular representatives of the two groups are respectively failure modes, effects (and criticality) analysis – FME(C)A [Wei 1991; ECSS 2001], and fault tree analysis – FTA [Lee et al. 1985; Vesely and Roberts 1987; Ericson 1999]. They are presented in the next subsections.

I.3.1.1 Failure modes, effects and criticality analysis – FMECA

FMECA is an inductive approach whose principle is to analyze, for each component, the consequences of its possible errors so as to identify systematically all the failure modes of this component as well as their consequence for the system. The approach can be applied during the system design, development, and even its use phase. It is convenient to apply FMECA as early as possible to detect design weaknesses and reorient choices according to dependability requirements. The lack of details about the actual system may, however, be a handicap for its efficiency at an early design phase. It can be applied as an accompanying process from the design to the system use phase. During the early design phases, the FMECA can be used to verify the feasibility in regard to the expressed system requirements, and when more details are provided thanks to advances in the design and development, it can be used to verify and maintain the compliance with the requirements. During the use phase, it can be used as a guide to collecting field data for assessing analysis accuracy, and for developing maintenance troubleshooting procedures [Bowles 1998]. The FMECA of complex systems is usually performed based on the system functional structure followed by an analysis at the component level when the information needed becomes available.

In general, the application of an FMECA consists in listing in a table, based on the functional or structural description of the system, the various failure modes of each component and their characterizations. Each failure mode is characterized by [Laprie et al. 1995]:

- its possible causes;
- its effect, which can be local, i.e., only the component behavior is affected, or propagated up to the system level;
- the detection means;
- the corrective actions, especially when dealing with a catastrophic failure mode;
- its criticality.

The criticality of the failure mode is a categorization of the failure mode based on the severity, the frequency of occurrence, and sometimes, the possibility of detecting earlier symptoms. In some cases, the criticality of the failure mode is not taken into account. In

that case, the approach is referred to as FMEA. Failure modes with identical effects can be combined and summarized in a Failure Mode and Effects Summary - FMES.

Iterating the application of the method based on a refinement of the decomposition may help identify the failure modes that are not straightforward. By characterizing the system failure modes, the FMECA table represents a valuable documentation and a basis for the system validation and for the system support during its use phase. However, it is worth noting that the approach has some limitations. For a complex system, it is practically impossible to reach the objective of covering all the failure modes. Also, the approach is not designed to address combinations of failures, since each failure mode is addressed separately. Actually, given the number of failure modes that may be identified, considering their combinations raises the problem of combinatorial explosion. Deductive approaches like fault tree analysis intrinsically copes with combination of failures.

I.3.1.2 Fault tree analysis

Fault tree analysis (FTA) is a deductive approach, which consists in describing the combinations of events that may lead to an undesirable event, such as a catastrophic failure. As for FMECA, a fault tree analysis can be applied at any phase, from the design to the use phase of the system. The specificity of fault tree analysis is that it is not a systematic analysis of all the possible failures; FTA targets events of particular importance and only failures related to the targeted event are examined.

An FTA is based on a graphical representation of the events using logical connectors or gates. Many logical connectors can be found in the literature but the fundamental ones are the AND and OR gates. The resulting diagram, called fault tree, consists in successive levels of events; the top-level event, i.e., the tree root, is the undesirable event. The analysis starts with the undesirable event, and then iteratively determines (deduces) the causes using a systematic backward-stepping process, until reaching events considered elementary.

The primary benefit of constructing a fault tree is that it helps in gaining significant insights of the causes of the top event. The fault tree can also be processed to derive further refined information. The principal qualitative exploitation is based on the computation of minimal cut sets. A cut set is a collection of elementary events that can lead to the undesirable event at the tree root, and it is minimal when it doesn't contain any other cut set. The analysis of these minimal cut sets allows highlighting the critical events related to the occurrence of the undesirable event. A particular attention is paid to minimal cut sets containing a single event and precautions are taken regarding their realization. It is worth noting that minimal cut sets corresponding to intermediate events in the fault tree can also be computed.

The activity of constructing a fault tree represents a qualitative analysis activity, but the obtained fault tree represents also a basis that can be used for quantitative evaluation. Besides, qualitative and quantitative evaluations are not mutually exclusive. Model-based approaches (section I.3.2.3), taking into account the dynamics of the system, can be used for both qualitative and quantitative evaluation.

I.3.2 Quantitative evaluation

Quantitative or probabilistic evaluation aims to evaluate in terms of probabilities the extent to which some of the dependability attributes are satisfied; those attributes are then viewed as measures.

I.3.2.1 Quantitative measures

The alternation of correct-incorrect service delivery is quantified to define reliability, availability, and maintainability as measures of dependability. Several other measures (like the mean time to failure - MTTF, mean up time - MUT, etc) characterizing the behavior of the system, with regards to occurrences of failures and the system recovery, can be considered. Two main categories of measures can be distinguished [Laprie et al. 1995]:

- Measures that characterize the sojourn time in the state where the correct service is being delivered: e.g., reliability and MTTF, which measure the delivery of correct service before failure.
- Measures that consider the correct service delivery with respect to its alternation with incorrect service delivery; they correspond to the various forms used to measure availability.

Generally, for fault tolerant systems, several modes of service delivery can be distinguished. These modes represent different levels of service delivery and may range from full capacity to emergency service. From a dependability evaluation viewpoint, two main extreme cases can be identified:

- Several modes of correct service completion and a single mode of incorrect service.
- A single mode of correct service delivery and several modes of incorrect service.

The safety measure corresponds to a special case of the measures that consider this aspect of the service delivery, when all the failure events that may affect the system are not catastrophic. Safety is actually reliability with respect to catastrophic failures. The state of correct service and the states of incorrect service due to non-catastrophic failures are grouped into a safe state.

In general, the measures related to these kinds of systems, which take into account the impact of the service degradation, are called performability measures [Meyer 1992].

An overview of the techniques for the evaluation of the measures is presented in the following.

I.3.2.2 Quantitative dependability evaluation techniques

Quantitative evaluation is carried out based on two main approaches: measurement-based evaluation and model-based evaluation. The two approaches are nevertheless complementary since the model needs input data that may be obtained by measurement.

Measurement-based evaluation consists in testing the system or observing its behavior during its use phase, in order to collect data characterizing the targeted measure. Measurement-based evaluation is attractive as it provides the most accurate information. However, it is costly and it may take a long time to have an effective result in case of faults that are seldom activated. Besides, it is only applicable to systems that are already developed. For systems that are not yet built, but not limited to them, model-based evaluation can be applied using parameters from a similar system or parameters defined in the system specification.

Model-based evaluation consists in using an abstract representation to analyze the system behavior. It has the advantage of being usable all over the lifecycle of the system. During the design phase, model-based evaluation can be conducted to help make appropriate choices concerning dependability requirements. Solutions that best characterize the dependability of the system can be selected among various candidate alternatives, based on the results of their evaluation. Model-based evaluation is also useful in investigating further the solution chosen. Sensitivity analyses can be carried out with respect to some of the design parameters. Model-based evaluation is still a good solution to analyze a system that is already in use in order to improve its dependability.

The approach developed in this dissertation concerns model-based evaluation using up-to-date data collected during the use of the system. The main characteristics of model-based evaluation are presented in the following sections.

I.3.2.3 Model-based evaluation

Model-based dependability evaluation generally consists of three basics steps: i) the definition or choice of the measure(s) to evaluate, ii) the model construction, and iii) the model processing. The choice of the measure to evaluate depends on the requirements of the system. The measures should reflect the goals of the system service delivery. The model construction consists in describing the behavior of the system based on its architecture and its elementary processes. The model processing corresponds to the computation of the dependability measure(s). Depending on the experience gained in the use of the system under consideration, an additional step, which consists in validating the model, is taken into account.

The following subsections give more details about the model construction and the model processing.

I.3.2.3.1 Model construction

To build a dependability model, one has to determine carefully the facets to be represented as features in the model. A trade-off is to be made between the tractability of the model and the full representation of all the aspects of the system. The elements to include depend on the measure to evaluate, the data available (the model parameters) and the modeling method. Two main groups of dependability modeling methods can be distinguished: combinatorial methods and state-space methods.

Combinatorial methods include the use of fault trees [Ericson 1999], reliability block diagrams [Bennetts 1982] and similar methods that generally capture a static view of the

system with respect to an objective or a feared situation, which accounts for the measure to evaluate. The resulting models are concise and easy to understand. Fault trees are the most commonly used. The diagram resulting from an FTA represents a model for a quantitative evaluation of the undesirable event. Combinatorial models have efficient solution methods but they are highly limited in dealing with complex systems, especially systems with complex stochastic dependencies.

State-space methods are the most appropriate when dealing with complex systems and where it is necessary to use combined measures when performing an evaluation. They can deal with almost all the aspects of systems dynamics, including stochastic interaction between the system components. Historically, state-space methods have been explored in the context of mathematical models that specify probabilistic assumptions about time durations and transition behavior [Nicol et al. 2004]. Therefore, the resulting models are usually supported by strong mathematical theories. Markov processes are the most widely used, especially Markov processes with discrete state space, usually referred to as Markov chains. Time-homogeneous Markov chains are the most predominant. For measures that are based on a continuous time scale, the choice of a continuous-time Markov chain (CTMC) as the underlying process for the model construction, assumes that the waiting time until the occurrence of an event is exponentially distributed.

In practice, due to the complexity of the systems generally manipulated, it is uncommon to directly build the model as a Markov process or any other random process that may be used. The model is thus developed using high-level graphical or textual description formalisms. The model construction consists then in describing the system using the formalism features.

The various stochastic extensions of Petri nets are the most popular among the graphical description formalisms, encountered in the literature. Stochastic Petri Nets (SPNs) [Molloy 1982] are extensions of the initial Petri Nets with timed transitions for which the firing time distributions are assumed to be exponential. Generalized Stochastic Petri Nets (GSPNs) [Ajmone Marsan et al. 1984; Marsan et al. 1995] are SPNs extensions that allow the transitions of the underlying Petri nets to belong to two different classes: immediate or instantaneous transitions (represented by thin bars) and timed transitions (represented either by white rectangular boxes or thick bars) GSPNs have been widely used to model the dependability of component-based systems (e.g., [Kanoun and Borrel 1996; Fota et al. 1997; Fota et al. 1999; Betous-Almeida and Kanoun 2004]). Deterministic and Stochastic Petri Nets (DSPNs) [Marsan and Chiola 1987] have been introduced as an extension of GSPNs, to allow the modeling of events having deterministic occurrence times. A transition can be specified to be immediate, exponential, or deterministic. The Stochastic activity network (SAN) formalism [Meyer et al. 1985; Sanders and Meyer 2001] represents another extension with more flexible firing rules based on the introduction of gates, and the use of new terminologies. The SAN formalism supports the use of any kind of distribution function in the specification of the transitions, which are called activities.

Textual formalisms are based on the paradigms of programming languages and come out with a list of keywords and syntax rules. One of the motivations that lead their development is that systems designers and developers are used to programming languages and thus, can easily master the use of these formalisms in a context of integrating dependability analyses to the design and development process. The AltaRica language

[Arnold et al. 1999; LABRI 2010] represents an example of these languages. AltaRica is developed to allow the construction of "user friendly" models that are very closed to the system architecture. The Figaro language [Bouissou et al. 1991; Bouissou 1993], developed at EDF (Electricité de France), represents another example. Figaro is an object-oriented language, which allows modeling specialists to develop modeling components that can be easily used by system designers for dependability analysis. Many other textual formalisms such as the PRISM language [Kwiatkowska et al. 2009] and AADL (see e.g., [Rugina et al. 2008; Rugina et al. 2011]) can also be found. Despite the fact that they are based on textual descriptions, it is possible in some cases (for example AltaRica and AADL) to use graphical representations while developing the model.

In the context of this dissertation, the AltaRica and SAN formalisms have been considered. The former has been used to model a number of aeronautic systems for safety analysis. The SAN formalism is used to model a wide variety of systems (e.g., [Hamouda et al. 2009]) as it allows for quantitative analyses and is supported by an existing tool that offers many model processing means.

I.3.2.3.2 Model processing

The computation of the dependability measures can be done either by solving the model to obtain an "exact" solution, or by estimating the measure using statistical methods based on simulation. The computation of an "exact" solution depends on the content of the model, especially the elementary processes used to parameterize the model. Model solvers usually provide "exact" solutions only for models that contain, exclusively, exponentially and deterministically distributed events. State space models containing non-exponential distributed events can be, however, transformed to exponentially distributed events model using the method of stages [Betous-Almeida and Kanoun 1997]. The method transforms a non-Markovian process into a Markovian one. Statistical estimation based on simulation can be used on any arbitrary model.

Since dependability models are in practice constructed using high-level formalisms, many tools have been developed to support the construction, and they usually integrate one or several model processing engines. A wide range of tools has been proposed for Petri-nets and their extensions (SURF-2 [Béoumes et al. 1993], TimeNet [German et al. 1995; Zimmermann 2010], GreatSPN [Chiola et al. 1995] for instances). There are also tools such as the Möbius tool [Daly et al. 2000], which are aimed at supporting several formalisms. The Möbius tool supports the SAN formalism and provides analytical model solvers as well as statistical estimation based on simulation. The tool Cecilia Ocas [Bieber et al. 2004], provided by Dassault Aviation, is one of the well known supporting tools for the AltaRica formalism. It does not provide intrinsically quantitative evaluation means, but it is designed to allow the integration of other modules including quantitative evaluation modules. The DIANA project includes the development of a quantitative evaluation tool EPOCH [Teichteil-Königsbuch et al. 2011] capable of processing AltaRica models.

This dissertation is aimed at addressing the construction and processing of dependability models considering the aircraft systems, their operation and maintenance. The next section presents the global context of aircraft operation and maintenance.

I.4 Aircraft Systems, Operation and Maintenance

Due to the criticality of its activities, aviation represents the transportation mode that is managed with the most important care. Aircraft systems are designed in such a way to avoid severe failures. Flights are carefully prepared and achieved considering procedures that are aimed at ensuring their successful completion. A maintenance program is required for aircraft operation, and the program has to be approved by the regulatory authority.

I.4.1 Aircraft systems

An aircraft is composed of several subsystems, which deliver its high-level functions. The Air Transport Association (ATA) specification, which provides a common referencing standard for aircraft documentation, devotes chapter 21 up to chapter 49 to their classification. These are, for example, the air conditioning and pressurization system (chapter 21), the electrical power supply system (chapter 24), the flight controls system (chapter 27), the hydraulic power supply system (chapter 29)…

These subsystems are designed considering dependability requirements that have been identified earlier at the beginning of the development process. Generally, their detailed designs include several redundancies (resulting from dependability requirements) and complex reconfiguration scenarios that are aimed at surviving failures and skipping the failed components in the loop of the function delivery. The systems are also designed with functional interfaces that allow for interaction with the other subsystems.

For the purpose of repair times optimization, the subsystems are made of Line Replaceable Units (LRUs), which interact so as to provide the functions that are required in the achievement of a flight. Examples of LRU are flight control computers, hydraulic and electrical power generators, ... An LRU can be involved in several functions. For instance, the flight control computer that is the subject of the study in [Meyer et al. 1980] is involved in eight functions.

The ability of an LRU to deliver the service required in the accomplishment of a function depends on its current state, which can take several forms. For example, a flight control computer may be simply operational, erroneous, or totally lost; alternatively one may consider that a flight control computer is either operational or not. Due to the interactions between the LRUs, the ability to deliver the service also depends on the state of the interacting LRUs or subsystems.

While in service, the LRUs are subject to errors and failure events that lead to changes in their state. When an LRU is no longer operational, it may be maintained or replaced during a stop before undertaking any other flight, or at a convenient time, after some flights, depending on its criticality.

The services the aircraft systems have to deploy correspond to those required by the missions. An aircraft mission consists of a series of flights, and each of them may have its specific requirements. Furthermore, a flight consists of different phases with different requirements in terms of functions to deploy. For example, the landing gear deployment is not needed during cruise, while it is absolutely required during the landing phase.

I.4.2 Aircraft operation

Aircraft operations involve the planning and the achievement of the planned missions that consist of flights. It is worth noting that the word flight may be misunderstood in the aviation industry, due to different declinations of its meaning. Generally, a flight is a trip through air, from a location to another one and which is assigned an identification number. A flight may include one or more intermediate stops, dividing it into several legs (also called segments). A flight that does not involve any intermediate stop is called a *non-stop flight*. Technically, a flight leg corresponds to a flight cycle, i.e., a takeoff and landing.

The following gives an overview of the operations planning and achievement.

I.4.2.1 Operation planning

The planning involves activities from the airline high-level organization down to specific airport station crews. There is primarily strategic planning, over a given period, which defines the service the airline will offer to passengers [Clarke and Naryadi 1995]. Based on the desired schedule of services established by the commercial department, the set of flights to be presented to the customers is defined. In particular the origin and the destination of the flights, and a global timetable for their achievement are established. This schedule of flights to be proposed to customers is used as a basis for crew scheduling and aircraft scheduling. The crews scheduling and aircraft scheduling are analogous [Grandeau 1995]. They basically consist in a resource allocation problem. Sequences of flights are assigned to crew members and specific aircraft within the airline's fleet.

The particularity in the aircraft scheduling is that one has to take into account maintenance bases capable of servicing the aircraft type. Usually, the aircraft are assigned to fly repeating patterns of flight legs called rotations, which start and end at a maintenance base. Aircraft have periodic maintenance and inspection requirements after various numbers of flight hours. The rotations must account for all maintenance and inspection requirements.

The aircraft rotations are initially scheduled based on the aircraft types, before assigning a physical aircraft corresponding to the type. The process of assigning a physical aircraft to the flights is also called tail number assignment due to the fact that all aircraft have an identifying number on their tail, and airlines generally refer to specific aircraft using their tail numbers.

After the planning of the global activities of the airline, the ultimate step is the execution of these activities. The scheduled flights are executed on a daily basis, and they may be readjusted, if necessary, to cope with irregular events. Prior to its achievement, each flight requires planning tasks and careful preparation to ensure its safe and successful achievement.

I.4.2.2 Flight achievement process

To ensure the correct achievement of the flights, great care is taken concerning the operational state of the aircraft. The accomplishment of the flights is subject to a dispatch process that involves the consideration of various information to make a decision, in

compliance with approved procedures. Figure I.1 summarizes the process to dispatch an aircraft as it is today [Papadopoulos and Bernard 2008].

Whilst operating an aircraft or during maintenance, the anomalies that have occurred are reported based on different means. The ECAM (Electronic Centralised Aircraft Monitoring) monitors the aircraft systems and provides the cockpit crew with warnings, which are classified from level 1, the highest, down to level 4. These notifications are reported together with other Flight Deck Effects (FDE) and crew observations in the aircraft logbook. The Built In Test Equipments (BITE) also deliver warnings that are recorded in Post Flight Reports (PFR). In some cases, a part of these messages is transmitted in real time to the Maintenance Control Centre (MCC) through ACARS (Aircraft Communication Addressing and Reporting System). During their activities, the maintenance crew also report the anomalies observed.

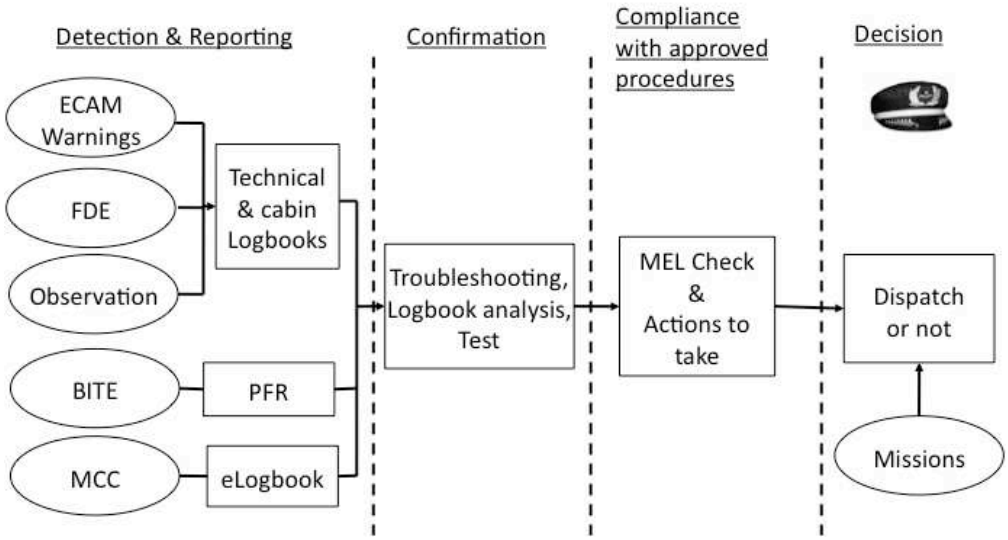


Figure I.1 Dispatch process

The warning and messages collected are checked in order to have insights into the problems and to determine the actual sources among the potential candidates incriminated by the warnings. This is done by means of troubleshooting using the troubleshooting manual, logbook analysis to try to replicate what is described, Post Flight Report Analysis to complete information received, and test that is part of the troubleshooting process to eliminate candidates. This investigation is conducted by the maintenance crew and may begin even before the aircraft is landed.

Once a component has been identified as inoperative, it is necessary to determine if it is critical for the flight. The Minimum Equipment List (MEL) is referred to, where either components are clearly marked as Go, or No-Go components. Or alternatively they may be Go-If components with a list of conditions to be validated to allow the aircraft to pursue its mission. If certain conditions are invalidated the component becomes a No-Go component. The components that are not mentioned in the MEL are considered No-Go components. Figure I.2 summarizes the different scenarios related to a component in the MEL.

- The Go components can be left failed.

- No-Go components obligatorily require corrective actions.
- The Go-If components may be left in a failed state or may be required to be removed or disconnected, but they require operational limitations (Go-If-o) or maintenance (Go-If-m) procedures to be applied, e.g. if anti-ice protection is unavailable then an aircraft may be dispatched as long as it is operated outside of an environment that has a risk of ice build-up. The system may be disabled by pulling and placarding a circuit breaker. In some cases, it is necessary to carry out a change to an operating procedure, e.g., flying below a certain altitude, or flying with the gear extended.

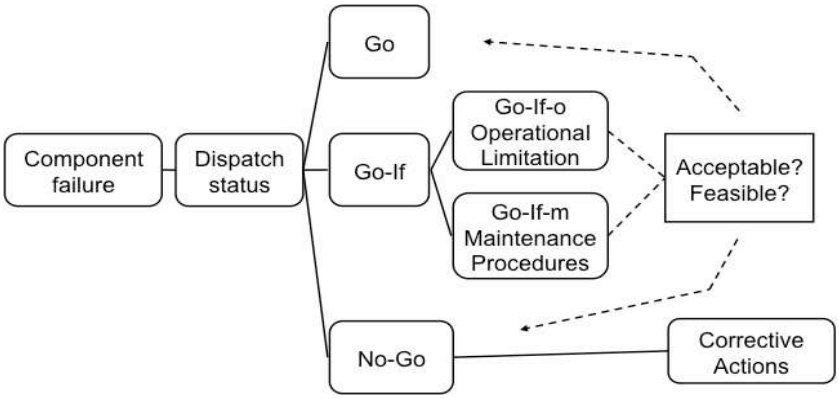


Figure I.2 Components status in MEL

The pilot, as part of his preparation activities, has to get an insight into the potential anomalies and their causes, if any. Once he i) has understood what the current state of the aircraft is, ii) has understood the imposed limitations in the way he can operate the aircraft, and iii) has applied them to the intended mission considering the prevailing weather conditions, he can then determine whether he is capable of flying the aircraft or if some actions are required to be taken in addition to what is described in the MEL. The airline helps the pilot by collecting the information that is pertinent for the pilot to take a decision on whether the aircraft can be used to achieve the flight or not. The decision must consider the whole planned mission, especially the flights to achieve after the current one, and more generally other missions linked to the mission in question.

The decision to dispatch the aircraft is very important at each flight since it not only deals with the ability to achieve the flight, but also the beneficial use for other missions. Nevertheless, it is still a technical preparation. To finally perform the intended flight, the flight handling crew must cope with the other activities ensuring the full readiness for dispatch. Activities before actual departure include passengers boarding, baggage and cargo processing, fueling, and other ...

When all ground activities are completed, the pilot requests the clearance to taxi to departure runway and initiates the take-off. During the take-off roll, the flight crew monitors the aircraft centerline tracking, engine parameters and conditions both inside and outside of the aircraft [Midkiff et al. 2004]. The take-off may be aborted (called rejected take-off) if a critical problem occurs. The abortion on runway may cause multiple tire failures due to heavy braking, and may result in a significant period of runway unavailability since the runway may need to be decontaminated.

During the flight, the flight crew must be constantly prepared for the possibility of contingencies requiring diversion of the aircraft to an en-route alternate airport. The causes of a diversion include medical emergencies, aircraft equipment problems, terrorist activities in-flight, unacceptable holding times, and fuel shortage.

Abortion during take-off and diversion in-flight represent operational interruptions that occur during the actual execution of the flight. A flight may be also subject to an operational interruption before its actual execution.

I.4.2.3 Operational interruptions

During the achievement of the planned flights, there may be several issues causing disruptions in the planned operations execution. The effect of the disruptions may range from simple readjustment noticeable only at the airline side, to deviations in the schedule affecting the passenger's travel. The latter directly affects the service, and ultimately the airline revenue. Nevertheless, whatever the disruption that may arise, one should be able to contain the situation in order to avoid the potential cascading effect. From a flight achievement viewpoint, the major undesirable situations that may be encountered, called operational interruptions, are delays, cancellations, in-flight turn-backs and diversions.

Among the factors at the origin of an operational interruption, several factors external to the aircraft in charge of the flight can be identified. Unfavorable weather condition may lead to the closure of an entire airport, resulting in the cancellation of all the flights departing and arriving at this airport. The flights heading to this airport at the closure time must be diverted. Some less severe weather conditions may only lead to flight delays. The flight crew, the ground handling services as well as the air traffic control services can also be causes of delays and cancellations. A disruption in the accomplishment of the airport service can easily affect the full achievement of the flight.

The factors related to the operational state of the aircraft to which the flight is assigned are failure events and maintainability problems. Any equipment failure may cause an interruption during the achievement of the aircraft planned rotations. A critical failure during a flight can cause a turn-back or a diversion to another airport. In this case, the aircraft should be routed to the nearest maintenance station that can accomplish the repair. However, thanks to the multiple redundancies in the aircraft system, the flight can be fully completed in most cases of failure. The interruption will then concern the next flight. The dispatch decision process should cope with the issue before the subsequent flight. If maintenance is required, the ability to promptly cope with the problem depends on the resources available at the considered airport. Otherwise the flight can be delayed or even cancelled if no spare aircraft is available.

The occurrence of an operational interruption can be very severe, especially the cascading effect. One or several aircraft may have to change from their current planned missions to a different mission. The replacement of an aircraft by another one may inject deviations in the missions of the latter, which may also affect other aircraft missions. Diversions to an airport where the airline does not have a base will leave the airline with aircraft, passengers, and crewmembers all at a wrong location with fewer options for recovery.

Airlines usually collect statistics on operational interruption occurrences. They evaluate the dispatch reliability, which corresponds to percentage of flights that are not cancelled and which are achieved without delay; and the operational reliability, which considers in addition to delays and cancellations, in-flight turn-backs and diversions.

To contain disruptions, airlines reserve margins in the service schedules and when a disruption happens, they try to re-plan the missions in such a way to resume the initial schedule as quickly as possible. For the disruptions resulting from failures, efficient maintenance policies can considerably reduce the problem.

I.4.3 Maintenance

In the early days of aviation, it was not mandatory to elaborate a maintenance program for aircraft operation. With the increasing development of the field and the notable complexity of aircraft systems, a maintenance program has been required for every aircraft in service. The maintenance program specifies the maintenance policies to be applied to the various components of the aircraft system.

I.4.3.1 Maintenance policies

A maintenance policy defines which type of maintenance must be performed on a system or a system component. Three main categories of maintenance policies can be distinguished: failure-based maintenance, time-based maintenance and condition-based maintenance. Condition-based maintenance policy can be itself split into inspection-based maintenance and examination-based maintenance [Kumar 2000].

I.4.3.1.1 Failure-based maintenance policy

Failure-based maintenance policy is an approach where the system or the system component considered is operated until failure. It consists in carrying out corrective maintenance after failures, and there is no planned intervention until the occurrence of the failure. Failure-based maintenance is usually applied to components that are non-safety critical and non operational relevant for the global system. It is also applicable to redundant and fault tolerant systems. The advantage of a failure-based maintenance policy is essentially the full use of the operating life of the component. The drawback is that the repair activities are always achieved as an unscheduled maintenance. When it is applied to an inappropriate component or system, the consequence of a failure can be disastrous.

I.4.3.1.2 Time-based maintenance policy

For components that are safety critical or whose failure may lead to significant economical consequences, it is important to prevent the failure or reduce the likelihood by carrying out regular maintenance actions. Time-based maintenance policy corresponds to the approach where preventive maintenance activities are carried out at a predetermined frequency to restore, overhaul, or replace the component. The frequency may be based on the operating times or units such as number of flight hours, number of takeoffs or landings … etc. The advantages of a time-based maintenance policy are the improvement of safety, the efficiency of the maintenance activities performed, as they are prepared in advance, and the reduction of unforeseen service disruptions. The disadvantages are essentially the

waste of operating life, the service downtime caused by the preventive maintenance activities and their cost. As the maintenance activities are carried out on a predetermined basis, it may not be globally efficient since the actual operating conditions may not always comply with the planning assumptions. Furthermore, the time-based maintenance is essentially aimed at reducing the faults accumulated with the time; the failure pattern of the component may not be time dependent.

I.4.3.1.3 Condition-based maintenance policy

Condition-based maintenance is the policy that is aimed at overcoming the drawbacks of the previous maintenance policies. The principle is to base the achievement of the maintenance activities on the actual state or performance trend of the component in service. It consists in detecting or monitoring changes in the component or system condition, which are likely to indicate an incipient failure, and achieving the maintenance when it is cost effective. It is suited to components for which cost effective techniques, capable of assessing the component condition and detecting the failure before it happens, exist. According to the techniques used, the condition-based maintenance policy can be classified as inspection-based maintenance or examination-based maintenance.

Inspection-based maintenance: It is an approach based on periodic inspections to determine whether the condition of the component is satisfactory or denoting a change that needs to be contained. The frequency of the inspections is determined before the component is put in service.

Examination-based maintenance: It is based on the use of indicators that provide information about the condition and a prediction of the future behavior of the component or the system. The indication is in form of numerical results whose analysis determines the next action to take. It is a dynamic approach.

Condition-based maintenance is designed to be a cost effective policy that enables a fuller use of the operating life of components whilst ensuring a good level of safety and reliability. However, it cannot be applied to every component. Its effectiveness depends on the accuracy of the monitoring technique. The maintenance program of complex systems usually combines all these kinds of approaches. The following presents considerations that lead an aircraft maintenance program.

I.4.3.2 Aircraft maintenance program

The objectives of an aircraft maintenance program are stated by the Air Transport Association of America (ATA) as follows [Kinnison 2004]:

- To ensure the realization of the inherent safety and reliability levels of the equipment;
- To restore safety and reliability to their inherent levels when deterioration has occurred;
- To obtain the information necessary for design improvement of those items whose inherent reliability proves inadequate;

- To accomplish these objectives at a minimum total cost, including maintenance costs and the costs of resulting failures.

These objectives do not take into account the operator's needs when the aircraft is in service. They are rather aimed at maintaining the safety and reliability level defined by the aircraft manufacturer. The maintenance program may have to be adjusted to fit airline operations in the field. The consideration of customer satisfaction may drive maintenance decisions.

The maintenance program is established based on the maintenance-planning document (MPD), which is provided by the aircraft manufacturer. The MPD is made of an initial maintenance program and other information such as the location and the numbering of the aircraft components. The initial maintenance program specifies the required maintenance tasks and the recommended frequencies.

Initial maintenance program development approaches

There have been two basic approaches to the development of an aircraft maintenance program: the process-oriented approach and the task-oriented approach. These two approaches correspond to different evolutions of procedures issued by the Maintenance Steering Group (MSG), a group of airlines and manufacturer representatives, as MSG-1, MSG-2 and MSG-3. MSG-1 and MSG-2 correspond to the process-oriented approach, and MSG-3 corresponds to the task-oriented approach.

The process-oriented approach focuses on individual items to determine which kind of maintenance must be applied. It uses three primary maintenance processes:

- Hard-time: a time-based maintenance, which requires that the item be periodically overhauled or removed. It concerns items that have predictable life limit.
- On-Condition: an inspection-based maintenance that requires that the item be periodically inspected or checked against some appropriate physical standard to determine whether it can be kept in service. The purpose is to remove the unit from service before failure during normal operation occurs [FAA 1978]. It concerns the items that have detectable wear out.
- Condition-Monitoring: a maintenance process for items that have neither Hard-Time nor On-Condition maintenance as primary maintenance process. No predetermined preventive maintenance task is associated to the item. The operator has to develop an appropriate condition-monitoring program to handle the item failure. The condition-monitoring program usually consists in tracking the item failure by means of data collection and analysis to help in predicting and avoiding the failure. The concerned items are usually operated to failure.

The process-oriented approach helped improve considerably aircraft maintenance by introducing different approaches to maintenance based on the component specificity, instead of applying systematically the traditional time-based approach to all the system components. However, the air transportation association of America published in 1980 [Chenevier 2001] other procedures, the MSG-3 procedures which consists in the task-oriented approach.

The task-oriented approach still uses the underlying philosophy of the previous maintenance processes (i.e., periodic maintenance, condition check and monitoring), but with a different approach. It consists in establishing the scheduled maintenance based on the tasks to achieve, instead of categorizing the items with a given type of maintenance. An item may be subject to a combination of tasks corresponding to different maintenance processes. The MSG-3 methodology is used to identify the suitable scheduled tasks to prevent the failures. The methodology is essentially a reliability-centered maintenance methodology, which is a consequence driven approach. It uses logical diagram for determining the suitable maintenance tasks. The analysis is made of two levels. The first level is intended to determine the failure effect category. The failure is categorized based on its impact on safety, operations and economy. The second level is aimed at determining the suitable maintenance task necessary for the prevention of the failure. The resulting tasks are accomplished based on predefined intervals.

Scheduled maintenance intervals

There are various types of maintenance intervals. Airlines can define their own named intervals, but they must either maintain the integrity of the initial maintenance requirements or receive the approval of the regulation authority. The standard intervals are [Kinnison 2004]:

- Transit checks: They are performed after landing and before the next take-off of the day. They consist of oil level check and a general visual inspection (walk around). If any problem is found, the resolving action will be however an unscheduled maintenance.
- 48-hour/daily checks: As named, they are done once every 48 hours or day and concern more detailed tasks than the transit checks.
- Hourly checks: They concern components that have maintenance tasks assigned based on the number of hours they have been operating. They are applied to items such as engines, flight control systems.
- Operating cycle limit checks: They are applied to items whose usage depends on the number of cycles performed. For example tires, brakes and landing gears, which are used only during take-off and landing.
- Letter checks (A, B, C, D): These correspond to the traditional grouping of aircraft maintenance. Essentially used before MSG-3 revision 2 [ATA MSG 1993], they were based on the simple belief that each part on an aircraft requires periodic overhaul. Thus, for modern aircraft, maintenance checks are based on the number of operating hours and cycles. The industry, however, generally still refers to maintenance intervals as ‘A’, ‘B’ and so on. The tasks carried out increase from A to D.

The program associated to these tasks represents the scheduled maintenance activities. They are intended to detect the failing components and repair or remove them. Some checks, like transit-checks, do not lead to immediate maintenance of the components found inoperative. They are either handled during higher-level maintenance tasks if they are deferrable (MEL conditions), or treated during an unscheduled maintenance.

Activities accomplishment

In the working viewpoint, maintenance activities are divided into on-aircraft maintenance and off-aircraft maintenance [Kinnison 2004]. As implied by the denomination, the difference between them is whether the tasks are performed at/on the aircraft or somewhere else; the faulty component may be removed from the aircraft and sent to the appropriate shop for repair. On-aircraft maintenance is itself divided into two categories:

- Line maintenance: it is performed at gate and includes everything from transit and daily/48-hours checks to A-checks. The aircraft is kept in service. It is worth noting that many specific denominations, which can be qualified of line maintenance "sub types" exist. These are base, outstation, turnaround and overnight maintenance.
- Hangar maintenance: it concerns major maintenance activities or modifications of the aircraft. The type of activities addressed are scheduled tasks above A-check (C, D); modification of aircraft or systems by service bulletin, airworthiness directive, or engineering order; fleet campaigns; special inspections required by airline or operational conditions; painting of aircraft; and aircraft interior modifications.

This dissertation considers only line maintenance activities, since the objective is to deal with operational issues during missions' achievement.

I.5 Aircraft Operational Dependability Assessment and Improvement Related Work

The dissertation concerns aircraft operational dependability assessment. The assessment is intended to be used for missions and maintenance planning adjustment. The objective is to improve the aircraft ability to achieve its missions without significant disruptions, especially disruption due to failure. Several studies have been carried out in the context, involving safety issues, reliability prediction principally during design phase, maintenance planning and optimization, and operational disruption management. The following gives an overview of the related work.

I.5.1 Aircraft safety assessment usable during operation

As safety represents a fundamental issue that must be covered before considering operational issues, several studies focus on safety analysis to address aircraft operation. The safety assessment is mandatory at design phase for certification purpose, and recommended processes and methods have been given (for instance SAE ARP 4754 and ARP 4761) for its achievement. The studies presented in this section represent the complementary studies which address operational issues in addition, or whose underlying principles can also be used to support operations achievement.

Usually, the analysis is aimed at verifying the fulfillment of safety and operational requirements. It is the case in [Ramesh et al. 2008], where mathematical models and solutions to determining the average probability of failure during a flight are presented. The aim is to provide means for demonstrating that the probability bound required for the system is respected. Redundant components with constant failure rate as well as aging

components with Weibull failure distribution are analyzed considering the whole operational life of the aircraft and the components maintenance mechanisms. The maintenance of a component is considered instantaneous. In [Prescott and Andrews 2005], the safety analysis is aimed at ensuring that the deferment rules of maintenance activities meet standards requirements.

[Tomaszek and Ważny 2009] analyze scenarios of aircraft operation safety using a simple model consisting of three possible aircraft states: operational state, repair state and complete loss of airworthiness state. The authors determine the probability of being in each of these states at a given time, considering different scenarios of reaching the state of complete loss of airworthiness. The analysis is rather appropriate to an elementary component.

Due to the particular complexity of aircraft systems, it is necessary to consider the components interaction and analyze the failure propagation. An approach to the evaluation of the cascading effect of aircraft system failures is presented in [Biswaws and Shrimali 2001]. The approach is composed of two steps. The first step consists in defining a relational matrix that represents the interdependencies between the aircraft subsystems. The second step develops, for each subsystem, the effects of its failure modes on the dependent subsystems identified in the first step. The authors claim that the outcome is very useful for flight crew in preparing their emergency procedures. The approach, however, is purely aimed at analyzing safety.

[Meyer et al. 1980] propose a model to assess the performability of a flight computer considering different levels of service accomplishment based on four attributes: safety, no change in flight plan, no operational penalties, and no economic penalties. The model considers a single flight profile and distinguishes different phases, which require different functions provided by the computer. The model is structured in three levels: the mission level which represents the different degrees of service accomplishment, the aircraft level which represents the functions deployment and the operating environment, and computer level that represents the computer internal behavior according to the different flight phases. The study does not deal with maintenance issues before or after the flight.

In [Sachon and Paté-Cornell 2000], delays and safety in airline maintenance are addressed. A probabilistic risk analysis model is developed in order to quantify the effect of airlines maintenance policies on the occurrences of delays and in-fight safety. The model developed is an influence diagram consisting of three tiers: a tier composed of decision variables that represent the qualification of the maintenance personnel, the time to start maintenance activities and the maximum number of deferrals; a tier consisting in a ground model that represents maintenance resources availability and the ability to maintain; and a tier representing the scenario of a crucial event occurrence.

I.5.2 Aircraft operational dependability assessment at design phase

The works aimed at enhancing aircraft operational dependability are mostly carried out during aircraft design and development phase for reliability prediction and improvement. Analytical approaches as well as simulation models can be found in the literature.

The design enhancement is the purpose of [Bineid and Fielding 2006], which aimed at developing a dispatch reliability design methodology. From an allocated dispatch reliability at the aircraft level, the methodology derives the corresponding dispatch reliability at component level, which is to be compared with a predicted dispatch reliability based on the components failure and maintenance rates. The components of the aircraft system are assumed to be in series configuration, and their failures and maintenance rates are constant.

In [Hugues et al. 2002], the dispatch reliability is predicted using Markov processes. The approach is based on estimating the dispatch reliability as the sum of dispatch reliabilities corresponding to the aircraft subsystems. Dispatch events are also the subject of [Saintis et al. 2009]. The paper presents a modeling approach based on the fault tree of the targeted aircraft system, together with a computing algorithm to estimate the bounds of the dependability measure. It considers a series of flight cycles and provides a means of evaluating the occurrence probability of one of three events at each cycle: "No Go dispatch", "Accepted Degraded Mode" which corresponds to the case where a "Go-If" occurs and the airline accepts to perform the corresponding tasks, "Refused Degraded Mode" which is a "Go If" that is not accepted by the airline. The probability that more than one component failure occurs during a flight is considered negligible.

[Balaban et al. 2000] propose a simulation model that takes into account the fact that more than one failure can occur during a flight. The model was aimed at estimating the mission capable rates of new configurations proposals for the enhancement of the USA air force C-5 aircraft, so as to determine the best configuration. The mission capable rate is estimated considering three primary levels of readiness: fully mission capable, partially mission capable, and not mission capable. The paper does not detail the content of the model.

The Ultra Reliable Aircraft Model (URAM) presented in [Jones et al. 2002] appears to be a general discrete events simulation framework for aircraft design enhancement with regard to reliability. The model covers i) the system architecture representation using state-space-based techniques, ii) the system functions based on a mapping with the system components states, and iii) maintenance by considering a grouping of the components so as to represent maintenance of the functions. The model is used as an interactive tool for the investigation of different failure scenarios. The output produced is the probability of achieving a desired Maintenance Free Operating Period (MFOP) length. The MFOP modeling is also addressed in [Chew et al. 2008], together with Phased-Mission System (PMS) modeling. The model developed was solved by simulation.

Generally, aerospace systems dependability modeling is addressed as a PMS modeling problem. The consideration of the different phases of a PMS is led by the fact that during different time intervals, identified as phases, the system is in different configurations and different functions are required according to the phase. The PMS modeling has been tackled considering different characteristics of the mission profiles and the system. The durations and the sequencing of the phases represent one of the characteristics that are generally considered. For the systems whose mission profile can be defined in advance, the durations and the sequencing of the phases are considered known in advance and thus, deterministic durations are used in the description of the phases. This approach is considered for example in [Meyer et al. 1980] to analyze the single flight mission. Other

aspects are whether the trajectory of the mission profile is static or dynamic, and whether during each phase, the system has time-homogeneous Markov process property or not. Any combination of these aspects can be covered by approaches based on Markov regenerative processes, with the possibility to have analytical solution under some given conditions [Mura and Bondavalli 2001].

Concerning the structuring of the model of a PMS, the distinction of different levels of abstraction and the use of sub models represent a flexible approach for decoupling the mission profile characterization from the system description, and thereby facilitate the consideration of changes in the mission profile. The example considered in [Mura and Bondavalli 2001] is structured using two sub models representing the mission profile and the system, whereas [Meyer et al. 1980] distinguish three levels of abstraction.

The works aimed at design phase analyses, are useful in ensuring the best inherent operational dependability to the system. While in service, the inherent dependability has to be maintained. The following presents some studies aimed at improving maintenance planning and achievement.

I.5.3 Aircraft maintenance

As presented in subsection I.4.3.2, the current aircraft maintenance programs are elaborated using reliability centered maintenance process, which is a consequence driven approach. However, one cannot ensure its total efficiency, and the resulting program application needs to be supported.

In [Ahmadi and Soderholm 2008], the objective is to support the reliability centered maintenance process that is applied while developing an aircraft maintenance program. The operational consequences of aircraft system failures are analyzed using event tree analysis. The paper discusses and categorizes the possible consequences of failures taking into account the flight phase during which they have occurred. Given a failure, the analysis determines the corresponding categories and computes the annual cost the failure may generate.

In service maintenance planning and achievement also represent an important aspect that has been dealt with in the literature. The issue that is mostly addressed concerns the consideration of maintenance in the problem of assigning aircraft to missions (see e.g., [Clarke et al. 1996; Moudani and Mora-Camino 2000]). Given a flight schedule, [Sriram and Haghani 2003] propose an approach to determining the best aircraft assignment and line maintenance planning that optimize costs. The approach considers only scheduled maintenance of types A and B, and doesn't take into account any unexpected maintenance constraint, such as failures during the achievement of the flights, which require an accomplishment or deferment of maintenance activity.

A decision support approach to maintenance planning and deferment is presented in [Papakostas et al. 2010]. That is, thanks to redundancy, the aircraft can continue operating in degraded mode with some equipment inoperative; however, the risk of occurrence of an operational interruption is increased. A method is proposed to decide on the maintenance deferment at a given stop, taking into account optimization criteria: cost, remaining useful

life and operational risks. It is worth noting that the work is not aimed at assessing the operational dependability. It uses the dependability measure as input.

After planning the maintenance activities, their effective accomplishment can also be a concern. [Gupta et al. 2003] present a stochastic simulation model for the efficient planning of the maintenance technicians' jobs. The model is principally focused on the labor utilization and the workload management.

Researches on inherent reliability enhancement and on efficiency of maintenance are aimed at avoiding failures that may result in operational disruptions. Other studies have been targeting the efficient actions to alleviate the disruption when it occurs.

I.5.4 Aircraft operation disruption management

The problem of recovering from operational disruption has been the subject of several studies [Le et al. 2011; Filar et al. 2001] belonging to the operational research community. The approaches proposed are aimed at resuming normal operation as soon as possible so as to reduce the economical losses.

The first paper [Teodorović and Guberinić 1984], commonly found in the literature, addresses the problem as resulting from the unavailability of one or more aircraft. The paper is aimed at reducing delays, using a network model. [Yan and Yang 1996] present a modeling framework based on network flow techniques, which is aimed at managing the disruption resulting from a single aircraft failure. The paper proposes four strategic models to handle the disruption, considering (1) only flights cancellations; (2) cancellations and ferry of spare aircraft; (3) cancellations and delays; and (4) cancellations, ferry of spare aircraft and delay. The models corresponding to (1) and (2) can be solved efficiently; (3) and (4) are NP-hard. The approach doesn't consider maintenance constraints.

Due to the complexity of the problem, the use of heuristics has been considered. [Argüello et al. 1997] describe an approach based on the heuristic GRASP (Greedy Randomized Adaptive Search Procedure) to reschedule the aircraft routings when one or several aircraft cannot be operated. The heuristic is based on randomized neighborhood search. The objective is to resume the original schedule within one day, while minimizing the cost of the reassignment. The approach was experimented and validated using data from Continental Airlines.

The occurrence of an operational disruption may not necessarily be due to an aircraft failure. Other factors such as crew unavailability, air traffic control and weather problems, may cause disruptions. Therefore, the disruption management has been a prevalent issue for airlines. [Clarke 1998] presents an overview of the practice in Airline Operations Control Centers (AOCC) for dealing with irregular airline operations together with a literature review of the research related to disruptions management. [Kohl et al. 2007] present the numerous aspects of the disruption management and an overview of the current related work. The paper is aimed at reporting on the experience gained in the DESCARTES (DEcision Support for integrated Crew and AiRcrafT recovery) research project, involving British Airways.

An extensive survey of model formulations and solution approaches to the disruption management problem can be found in [Clausen et al. 2010]. The survey distinguishes researches with emphases on recovering aircraft schedule, crew schedule, passenger schedule, and those integrating two or the three of these problems.

I.6 Summary and Thesis Orientation

The problem that is targeted concerns aircraft continuous operational dependability improvement whilst coping with failures.

As described in section I.4.2.2, aircraft are operated in compliance with operational requirements depending principally on the current functional state of the aircraft systems components and the intended mission profile. Since the operations are planned in advance, the occurrence of unexpected events may cause significant disruptions in the planning. Indeed, even if most of aircraft systems are fault tolerant, offering thereby the possibility to operate with some components failed, there are still some failures or some external events that can prevent them from being operational. The disruptions result mainly into operational interruptions, i.e., delays, flight cancellations, in-flight turn-backs and diversions. These interruptions may lead to heavy economic losses due, for instance, to inoperability, unscheduled maintenance cost and compensation given to passengers.

Therefore, a continuous attention must be paid to the aircraft, regarding the operational requirements fulfillment. The approaches to the operability improvement, based on design improvement and disruption management, are not sufficient. The missions and maintenance activities must be adjusted regularly. Paying a continuous attention to the effects of the aircraft system components failures helps in the successful achievement of the missions. Failures that may disturb the achievement of the aircraft mission have to be handled with adequate corrective actions. Furthermore, the ability to promptly cope with these failures depends on the location where they occur, as maintenance facilities are not the same at all airports. Generally, airlines have more facilities at their main base than at the other airports. Efficiently adapting maintenance resources availability to the aircraft missions will be definitely worth for the airline activities. The issue is to develop an assessment method, to evaluate the operational dependability that one can use as support for the assignment of a suitable mission to a given situation, and for the accommodation of maintenance activities.

The approach in this dissertation consists in assessing, during the aircraft's operations, its ability to satisfy the operational requirements, in the presence of failures, and the ability to undertake an adapted action to prevent adverse situations. There is a need to have a good control on the aircraft's behavior while in service, and to be able to predict, with a good level of precision, the events that may be encountered. This includes an ability to cope with random events and a capacity to ensure that the solutions adopted are suitable. Model-based dependability assessment is well suited to support this process.

I.6.1 Role of model-based dependability assessment

Using models offers the advantages of sketching the situation without using the real system. In addition, stochastic models are very convenient to represent the stochastic

aspects of the events. For our particular case, the model is to be used essentially to evaluate the probability to successfully achieve missions or the risk of encountering an adverse situation. The estimated value will be used in decision-making processes to determine whether proactive actions should be taken or not. The model can be used while planning the missions and during their achievement.

To plan the mission, the model can be used to estimate the period of time during which the aircraft system can be operated without reaching an adverse state. It will help in determining the mission profile the aircraft can be assigned.

Once a mission is assigned to the aircraft, the model can be used during its achievement, both on ground and while in flight, to assess the ability to succeed in continuing the remaining part of the mission, or re-adapt it if necessary. The dispatch decision may take into account the ability to continue until the next airport where there are enough facilities to correct the problems that may be encountered. All along the mission achievement, after the occurrence of (or the detection of) major events that may affect the operability, the operational dependability assessment may be performed to provide an indication on the dependability of the remaining part of the mission. The outcome may be used to support the decision, by the operations control centre, to continue the mission or to revise the planned mission. To do so, the model uses runtime information about the aircraft operational state and the missions.

In case of a decision to divert the flight, the model can also be used to determine a convenient diversion airport. In case of emergency (due for example, to problems that may affect safety), the model is used once a diversion airport is selected, to re-assess the operational dependability.

Finally, the result may help in selecting the most appropriate maintenance or operation planning actions in order to improve the ability to achieve the whole mission. Assessing the success of a mission is a means for evaluating the concordance of maintenance planning with the mission. Hence, different maintenance strategies can be compared considering various alternatives for performing the component maintenance. The best strategy is then selected based on the estimated probability of mission accomplishment without operational disruption.

In order to implement the model-based assessment, the major characteristics of the modeling and assessment problems must be analyzed.

I.6.2 The modeling and assessment problem characteristics

The modeling problem concerns aircraft system representation, considering its missions and potential maintenance actions, for the purpose of dependability assessment.

Aircraft systems are characterized by a prominent complexity with multiple redundancies and reconfiguration scenarios, as mentioned in section I.4.1. The modeling approach has to manage this complexity. The ultimate purpose of the model consists in the dependability assessment during the aircraft missions' achievement, so as to support decision-making regarding the missions planning and achievement, as well as maintenance processes. Therefore, the model must allow for the consideration of the

current operational information in order to provide an adapted and accurate result. This goes beyond the classical dependability modeling and assessment during design, for which a dependability specialist is available to calibrate the model for the relevant scenarios.

As suggested in [Malek 2008], and applied in [Masci et al. 2011] to support interoperability between networked systems, dependability assessment should encompass the need to monitor and perform online assessment, considering online data for short term prediction in specific situations, instead of only the design phase assessment which considers the whole operational life for an average behavior prediction. Thus, for each mission that is to be achieved, one must consider its specificity and calibrate the model accordingly, before evaluating the likelihood of mission success. The actual possible scenarios of failures must be encompassed in order to provide suitable evaluations. Sensitivity with regard to non-critical combinations of failures that were not even evident during design could be observed in some situations.

Therefore, the model will be integrating runtime information about the aircraft system component states and behavior, and the characteristics of the missions. The assessment will be mainly done by the aircraft operators, who are not necessarily dependability-modeling specialists. As the developer of the model will not be always present at runtime, the dependability modeling and assessment framework must provide means to support the integration of the information.

I.7 Conclusion

The objective of the dissertation is to develop a model-based assessment framework, for aircraft operational dependability evaluation during its operations. Dependability concepts and evaluation techniques have been mainly developed to support critical systems construction and operation. Quite generic, but efficient techniques have been developed to assess or forecast the attainment of required level of service delivery. These techniques are widely applied during the design of aircraft in order to cover safety issues that may arise during their operation.

During its operation, an aircraft has to achieve missions consisting of predetermined sequence of flights. Each flight is achieved considering requirements and procedures that are aimed at reducing the risk of critical malfunction during the flight. Scheduled maintenance activities that consist principally of various periodic checks, are also conducted to preserve the aircraft operational state. However, due to their nature, there are still some unforeseen failures leading to the necessity to accomplish unscheduled maintenance activities. The non-accommodation of an unscheduled maintenance with the planned mission results in an operational interruption, which may cause other disruptions in the airline activities. This is an important issue in today's aviation industry, which is asking for competitiveness.

Several works related to aircraft operational dependability assessment have been found in the literature, but they are rather concentrated on specific aspects that do not cover the need to continuously assess aircraft operational dependability in service, so as to enhance the delivery of service. Some of them are aimed at dealing with safety concerns. Safety represents an aspect that is regulated and is ensured through constraints included in the

operational requirements that must be satisfied during operation. The problem of avoiding unscheduled maintenance resulting in an interruption is also considered during the design of the aircraft. The approaches principally concern the dispatch reliability prediction and improvement. The corresponding predictions have to cover the whole operational life of the aircraft as scope.

The operational research community has also been dealing with the reactive action to contain disruptions. Their researches are mainly aimed at resuming the initial operation schedule as quickly as possible, whilst minimizing the cost. Numerous approaches have been proposed, but these are still insufficient to meet today's concerns. They are limited by the great complexity of the problem. The increasing demand in the aviation industry and the call for competiveness requires a support for the continuous analysis of aircraft operational dependability so as to take proactive measures. Model-based operational dependability analysis should play an important role in providing means for continuous assessment and objectives adjustment.

Hence, our research study targets the development of a stochastic state-based model that allows for the assessment of aircraft operational dependability, during the aircraft operation, whenever it may be needed. During flight and maintenance planning, an assessment is done so as to support the selection of the solution that is in accordance with the operational capability of the aircraft and with other impacting conditions. After the occurrence of major events during a mission achievement, such as failures of presumed operational relevant components, the dependability is re-evaluated to assess the impact of the new conditions on the mission's achievement.

The work includes thus, as essential aspect of the construction of the model on which the assessment is based, the consideration of the assessment circumstances, which may require an adaptation of the model in order to take into account the characteristics of the current situation. This aspect concerns how to manage the model at run-time so as to obtain useful results.

II Modeling Approach and Assessment Framework

This chapter aims at presenting the global modeling approach that is proposed in order to enable the aircraft dependability assessment during operation. The model construction requires an appropriate analysis that helps to cope with aircraft specificities and the issues related to the assessment during missions taking into account the information recorded during operation. As a result, the model construction process and the assessment in operation involve i) the development of a stochastic dependability model, based on a meta-model, and ii) the use of this stochastic dependability model to obtain a model integrating the current operational information and which is ready to be used for the assessment. We choose to base the construction of the model on a meta-model in order to provide a common basis standardizing models that may differ due to the particularities of the aircraft systems. The meta-model also represents a reference for getting an insight into the model at a high level of abstraction.

The necessary basis for developing the modeling approach is firstly established. It presents the measures that are considered, the information to include in the model, and the dependability assessment framework. The approach to update the model so as to take into account changes during operation is also presented, as well as the model construction process. Subsequently, the model content specification is proposed, together with the main variables whose initial value may be affected by changes following an update of the model. Finally, we present the meta-model that represents the common basis for the construction of models corresponding to different aircraft.

II.1 Establishing the Model

Modeling complex systems dependability such as aircraft operational dependability needs to be handled with efficient techniques. This section establishes the basis of our modeling approach. For this purpose, the kinds of assessments the model can be used for are firstly defined. Subsequently, the categories of information that should be taken into account, in order to make the model representative of the aircraft operational situations, are identified. Based on the categories of information, a first model structure is derived. We consider an approach based on the concept of separation of concerns that consists in segregating the mission related information and the systems description. Major changes that may happen during operation, concerning the information, which must be taken into account in the model, are indentified. As shown in Figure II.1, these changes should result in an update of the operational dependability model, and a re-assessment of the dependability should be initiated if the change is relevant. The framework to monitor the changes, to capture the update data and to initiate the re-assessment is also given. The approach to update the model and the model construction process are discussed in the end of the section.

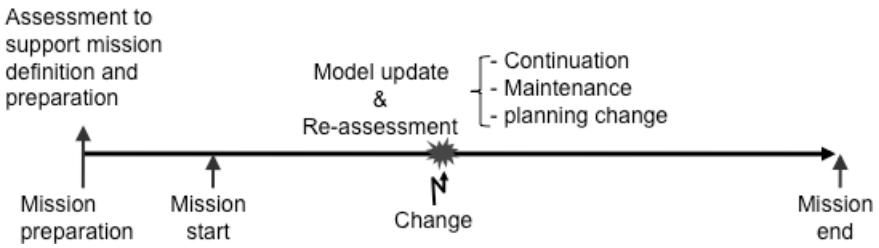


Figure II.1 Changes and re-assessment during missions

II.1.1 Measures to evaluate

The ultimate use of the model will be during the aircraft operation. However, the model is intended to be built by the aircraft manufacturer during the aircraft development process. Our aim is also to make it possible to use the model for some development phase dependability analyses. The analyses generally carried out by an aircraft manufacturer concern principally failure effect analysis based on stepwise discrete event simulations ([Kehren et al. 2004] for instance). The analyses also include the generation of sequences and cut-sets of events that may lead to the loss of given functions. The reliabilities of these functions are also evaluated. Therefore, the proposed model will allow for the evaluation of the reliability and availability measures of the high level functions of the aircraft system.

For the dependability assessment considering the in-operation conditions, the objective is to evaluate the probability of occurrence of the adverse events that may lead to an operational interruption. Such events principally result from the non-fulfillment of the requirements that must be satisfied in order to achieve the missions. Therefore, we adopt an approach based on the operational requirements fulfillment.

The aircraft has to fulfill specific dispatch and in-flight requirements in order to succeed the achievement of the mission. We particularly distinguish:

- *The minimal system requirements (Min_Sys_R)* that are independent of the mission profile and that must be fulfilled to operate the aircraft whatever the mission. Min_Sys_R is principally made of the requirements necessary to avoid the MEL conditions that may result in a No-Go (I.4.2.2).
- *The mission profile requirements (M_Prof_R)* that are specific to given mission profiles.

Based on these defined requirements, we distinguish two principal reliability measures:

- While planning a mission, the aircraft system reliability (SR) is evaluated with regard to Min_Sys_R in order to determine the maximum number of flight hours that can be achieved without maintenance, whilst ensuring a given minimum reliability threshold. This is used to determine the length of the mission or to plan maintenance activities.
- Once a mission is assigned to the aircraft and during its achievement, the reliability measure (MR) which corresponds to the probability to achieve the mission without the occurrence of a technical event leading to an operational interruption, is

evaluated with regard to Min_Sys_R and M_Prof_R in order to determine whether a preventive action must be initiated or not.

Additionally, in order to obtain quick results in some situations, one may analyze only the reliability of a given aircraft function that is identified to be critical for the mission. The next subsection presents an overview of the relevant information to be considered in the model so as to assess these measures.

II.1.2 Model content

To cover the aircraft operational dependability issues, one has to identify the relevant types of information involved in the aircraft operability. We consider design phase information together with the necessary information in service.

During the aircraft design phase, model-based safety analyses are conducted for the verification of the compliance with safety requirements and the establishment of the Master MEL. The analyses follow the aircraft functional decomposition in earlier design phases and the aircraft breakdown structure when the systems are completely designed. The analyses contain an important qualitative part. For that, the architectures of the subsystems supporting the aircraft functions are used together with results of FMEA (chapter I, section I.3.1.1) to build dependability models. The subsystems architectures provide the structural dependencies between their basic components and the inputs from FMEA provide the component fault models. The models constructed include fault trees and mostly models based on high-level description formalisms such as AltaRica. The models are used to analyze specific functional conditions that are safety relevant.

For an operational dependability assessment, all the functional conditions that may be required during the different phases of the missions must be taken into account in the model. The model must also incorporate maintenance information since maintenance activities may be possible during the missions' achievement. Therefore, the aircraft system, the mission profile and maintenance (scheduled and provisions for unscheduled) represent the main information sources that should be considered for the establishment of the model content. Requirements necessary for the correct achievement of the missions are also considered in order to define the concordance with the functional state of the system. One may additionally consider weather factors, but this type of information is rather dependent on the operator appreciation, than on the monitoring sources that will provide information to configure the model during operation. Moreover, the description of the impact of weather condition is beyond the scope of our work, and the related information needed in the model is simply reduced to whether it is favorable or not.

Figure II.2 gives an overview structuring the relevant categories of information considered for the model construction.

Mission profile: It is composed of information related to the succession of periods during which the aircraft is either flying or on ground. The mission profile information can be obtained from the flight scheduling process.

Requirements: they consist of the aggregation of the requirements from the potential contributors to the successful achievement and the continuity of the mission. These

requirements express the aircraft system functional availability and the constraints related to the mission considered. They can be obtained from the policies defined in the MEL, the Flight Crew Operating Manual (FCOM) and the Quick reference handbook (QRH).

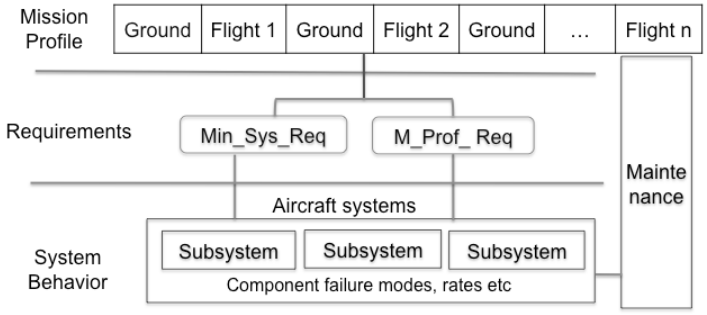


Figure II.2 Categories of information

System Behavior: It concerns the aircraft system components. The description of their failure scenarios and maintenance represents a fundamental point in the whole model construction. The aircraft is made of subsystems, integrating components, which support the accomplishment of different important functions as described in section I.4.1 of chapter I. The subsystems architectures and their FMEA will be considered together with the components maintainability information.

Maintenance: It concerns the maintenance possibilities at the various airports involved in the mission profile. The defined maintenance strategy, which determines the maintenance resources at the airports, has an impact on the maintenance time of the system components at a given stop.

The approach is to collect information from the different components that take part in the aircraft operational dependability in order to form a global model. The structure of Figure II.2 reflects thus the acquisition of data, from the different relevant components of the aircraft operability, as pieces of the model.

From a global viewpoint, the model is structured based on a separation of the concerns. Considering the assessment objectives, one can find out that the mission profile and maintenance information are not required in every case. The system reliability can be assessed with regard to minimal requirements without considering any mission dependent information. Therefore, two main parts are considered in the model construction:

- the system related information, referred to as the core model;
- the operational information, which integrates the mission profile and the maintenance strategy, referred to as mission dependent model.

The model is thus the composition of a ***core model*** that is based on the system components description, and a ***mission dependent model*** that is based on the mission profile related information and which may be removed or changed without affecting the ability to use the core model for dependability assessment. Figure II.3 shows the model structure from a global viewpoint.

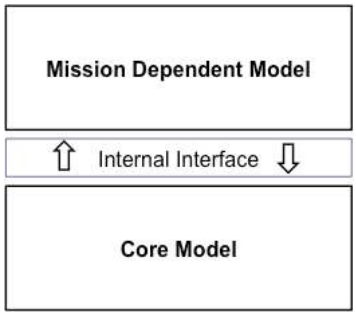


Figure II.3 The operational dependability model composition

The internal interface between the core and mission dependent models represents the requirements to be fulfilled during the mission, and the ongoing mission phase information that should be shared by the two parts of the model.

It is worth noting that the two main parts can also be decomposed into sub models. Both parts can be subject to updates following changes during operations.

II.1.3 Major changes to be accounted for

We have identified three kinds [Tiassou et al. 2011a; Tiassou et al. 2012c] of major changes that may take place during the achievement of an aircraft mission and which may require the operational reliability reassessment. They are summarized hereafter.

- **C1: Changes in the state of system components**: they correspond to the case where for example a system component has failed during the achievement of the mission (we assume that this failure does not impact safety, otherwise the mission is interrupted).
- **C2: Changes in failure distributions of the components**: they mainly concern the case where new failure rates or distributions have been prognosticated for the system components, during the duration of the mission.
- **C3: Changes in mission profile**: A mission profile is characterized by the number of flights, the ground periods, the flight phases and their sequencing. Due to external events such as weather conditions or failures in other aircraft, a mission profile that is completely different of the previous one may be assigned to the aircraft. The new mission profile may come with new requirements. Changing the mission profile also implies changes in maintenance facilities available at the various stops (or destinations), as well as in the mean time to repair the failed components or the repair time distribution itself.

Data from different sources will help to notify the changes. The notifications of the changes will lead to an update of the dependability model with the up-to-date information. The reliability measure is then re-assessed by processing the obtained model.

II.1.4 In operation assessment framework

For the integration of the up-to-date information into the model, our framework for the dependability assessment in operation consists in relying on data provided by on-board

modules that will be monitoring the aircraft system, and on data from the operations planning. An assessment manager will validate the data and determine its relevance before its integration into the model for the dependability assessment. Figure II.4 shows an overview of the proposed in-operation model-based dependability assessment framework.

A diagnosis module together with a prognosis module will be running during the aircraft operation in order to provide indication about the current state and the up-to-date failure distribution characterizing the future behavior of the components.

Diagnoses are based on analyses of the symptoms observed during the system operation. The analysis attempts to determine for each system component, the operational state that is coherent with the observed symptoms. The diagnosis is firstly performed locally, at each component level, then the resulting information is merged taking into account the interaction between the components [Ribot 2009]. The indication provided by the on-board diagnosis may not give firm information on the failure of the components. The failed components may be diagnosed with a certain uncertainty. The information must then be confirmed by further investigation of the maintenance crew. Nevertheless, the indication provided by the on-board diagnosis module can be used for a dependability assessment that provides information for preliminary decisions. The assessment will be refined after the troubleshooting task of the maintenance crew.

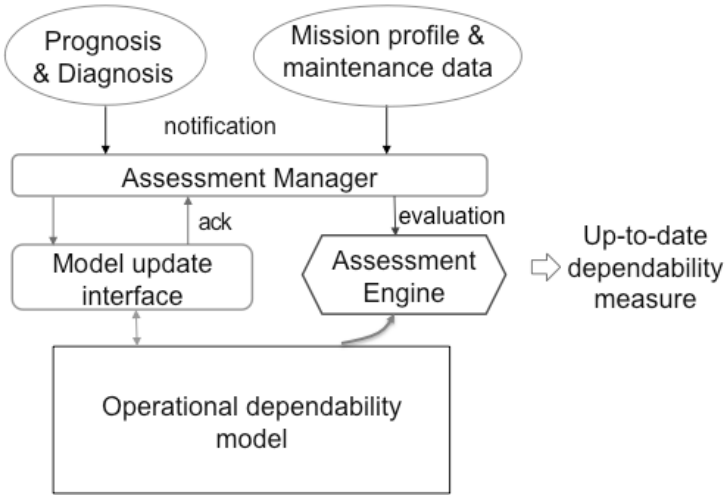


Figure II.4 In operation assessment framework

Prognoses are also based on analyses of a set of observations. According to [Ribot 2009], a prognosis consists in forecasting the system future sequence of states. The future behavior is described in terms of probability. It is worth to mention that most of the studies in the literature concentrate on the prediction of the Remaining Useful Life (RUL) of system components and equipments [Jardine et al. 2006]. The prognosis can also provide an estimation of the mean time to failure. For the purpose of the in-operation assessment, we will be using characteristics of the failure distribution functions that govern the system components' behavior.

The diagnosis and prognosis modules can be run either automatically by embedded control systems, or occasionally by the operator.

Data about the missions and maintenance planning is also accessible. In the operations scheduling process, there is primary an initial planning that defines the timing, the origins and destinations of the flights; and then the execution of the planning on a daily basis [Clarke and Naryadi 1995; Clausen et al. 2010]. Therefore, based on the initial planning, one can have the timeline giving general information about the aircraft mission, and detailed information can be gotten on day-to-day basis.

The assessment manager examines the data from these sources and requests the integration into the model via a model update interface. The assessment manager also determines the relevance of the notified changes of data in order to decide on the necessity to proceed to re-assessment, so as to obtain the up-to-date dependability measure. The assessment manager can correspond to a human operator or to an automatic algorithm (or probably to a mix of both). The evaluation can be done either at the operator's request or automatically in order to notify the operator of changes that may affect the ability to continue the mission.

The operational dependability model should be flexible to cover the main characteristics of the aircraft system and operation conditions. It also has to be tractable enough for efficient processing during the missions' achievement. The model will be built based on high-level formalisms in order to benefit from their great expressiveness.

The assessment engine shall implement algorithms that are capable of processing the heterogeneity of contents that will result from the different cases of model update, especially the different stochastic processes that may characterize the components. Furthermore, offline extensions of the model can be considered and as the model will be covering as many situations as possible, it should be possible to define additional measures to be evaluated when needed. The assessment engine must be extensible so as to integrate the capability of processing extensions of the model, including new dependability measures definition.

Updating and processing the model can be performed either onboard or on ground, even while the aircraft is still in flight, without waiting for the aircraft landing, in order to obtain as early as possible the new assessment results (i.e., the dependability measure). Dedicated centers with high processing capabilities can also be considered for the prompt delivery of the assessment results. Globally, the update is possible by someone or a process that is informed that there are changes requiring a model update, and that is able to easily integrate the changes.

The in-operation assessment framework gives an overview of the interacting components involved in the extraction of up to date information for the assessment during the missions' achievement. Our work is more concerned by the model construction and how the model will integrate the current operational information. The final model obtained, which reflects the current situation in operation, is denoted as the *up-to-date model*.

II.1.5 Model construction and update process

The dependability model should allow for an easy and efficient model update and processing. The easiness of the update depends on the model content and the means provided for the adaptation to the current situation. Therefore, the approach to update the

operational dependability model should be determinant in the model construction approach. We propose an update approach that should require only basic actions. The model construction process is established accordingly.

II.1.5.1 The approach to the model update

Model update and adaptation to an operational situation may involve parameters tuning, model composition based on sub-models or model reconstruction. Parameter tuning concerns the modification of the initial value of basic variables. Model composition based on sub-models involves the selection of sub-models that should be combined in order to obtain the model adapted to the current operational situation. Model reconstruction is needed when the adaptation to the operational situation requires a deep modification of the model content, or includes behavior representations that are not considered in the previous existing model. However, a deep modification of the model content may require a specialized modeling analysis that could not be carried out in real time. Therefore, model reconstruction should be considered only if it is possible to carry out the specialized analysis in real time.

For our particular case, the update of the model should not require the presence of modeling specialists. It should be possible to update the model from outside, without necessitating a deep knowledge of the model. The model update should not require the knowledge of the underlying modeling technique and formalism used. Also, the model should be tractable enough to provide quick results whilst covering as many situations as possible.

As a consequence, the approach consists in building and validating a stochastic model for which the essential operational information can be entered based on parameter tuning. Indeed, the model content cannot be deeply modified during the update, due to the fact that it may require an analysis by a specialist. Only basic actions should be done for the update and adaptation of the model. Nevertheless, model composition based on predefined sub-models can be considered if necessary, since the composition process can be automated. Therefore, if some operational situations cannot be reflected by tuning the stochastic dependability model, predefined sub-models may be used to capture them. The stochastic dependability model is configured for a default situation.

II.1.5.2 The model construction process overview

The determinant aspect of the model construction process, which has been dealt with above, concerns the update and adaptation of the model to the current situation during operation. Accordingly, the model construction consists in developing a stochastic dependability model (also referred to as stochastic model), configured for a default situation and which can capture other situations based on parameter tuning.

Furthermore, since the same type of model will be used for supporting operation planning for several families of aircraft from the same manufacturer, it is important to follow an harmonized procedure to build the stochastic model for several families of aircraft in order to provide a unified view, to facilitate both the model construction process and the model tuning process.

Therefore, we decided to establish a common basis to support the model construction, based on a general specification (i.e., a **meta-model**) consisting in a structured representation of the information that will compose the stochastic model. The meta-model is an abstracted representation of the final models, by means of analyzing the relevant elements to be modeled. It structures the information about the aircraft system, its possible missions and maintenance policies.

Figure II.5 presents the overall model construction and update process. A specialized dependability analyst, with the help of aircraft system manufacturers and specialists of aircraft operations and maintenance, establishes the meta-model.

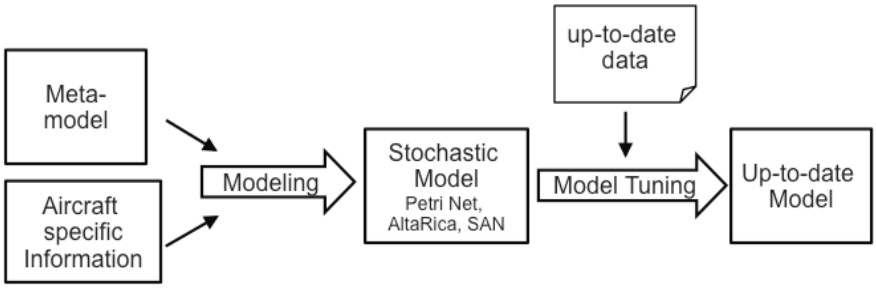


Figure II.5 Operational dependability model construction process

The meta-model is used to build the stochastic model using the aircraft specific information. The meta-model can facilitate the model construction since models can be partly generated automatically based on meta-models. In addition to supporting the model construction, it also represents a basis, for the non-specialists that will use the model during operation, to have an insight into the general content of the model.

Before the presentation of the meta-model, the model is firstly specified based on a description of its content as presented in [Tiassou et al. 2012a]. The specification gives an insight into the model content as traditionally described by dependability modeling formalisms. We also highlight the variables that may be affected by changes during operation.

II.2 Model Content Specification

The specification concerns the two major parts of the model: the core model and the mission dependent model. The main variables, whose values may be affected in case of changes in operation, are highlighted at the end of the section.

II.2.1 The Core model

The core model, mainly dedicated to the aircraft systems description, is specified considering:

- their components,
- the interdependencies between the components,
- the functions delivered by the system,
- the applicable dependability requirements.

The simplified hydraulic power supply subsystem presented in Figure II.6 is considered to illustrate the description.

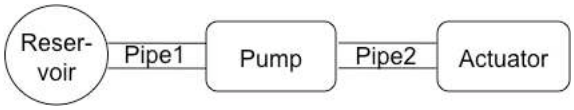


Figure II.6 Simplified hydraulic power supply subsystem

It is composed of a reservoir of fluid, a pressurization pump and an actuator, connected by pipes.

System component (SysComponent): Each system component C_l is characterized by:

- its identifier (*id)* ; it will help in managing the component update.
- its state *Variables* (*stateVars*).
- the *Events* it may be subjected to.

A s*tate* of a component C_l, let's say C_lS, may take different values identified as its domain C_lSD. C_lSD can be decomposed into two domains C_lSD = Operational (C_lSO) ∪ Failed (C_lSF).

Usually, at least two distinct states are associated to each component: $C_lSD = \{ok, failed\}$. The pump, for example, is characterized by a `health state` (which can be `ok`, `lost` or `leaking`) and an `actuation state` (which is either `ON` or `OFF`).

There may be several kinds of events *Ei* related to a system component but the main events for our case are *Failure* and *Maintenance* events. The *Failure* and *Maintenance* events lead to a change of the state variable value from C_lSO to $C_{l,}SF$, and from C_lSF to C_lSO respectively.

For instance, the pump can be subject to the failure event *loss* that changes the *health state* from *ok* to *lost*, and the failure event *leak* that changes the *health state* from *ok* to *leaking*.

The events are also characterized by the durations until their occurrences, using a probabilistic distribution. Accordingly,

- a distribution *Fdistr* characterizes the failure event occurrence,
- a distribution *Mdistr* characterizes the maintenance activities duration.

Generally, exponential distributions are assumed for these events, characterized respectively by failure rate λ and repair rate μ, which are *Parameters* used in the distribution description. For instance, the failure rate of the pump loss can be 10^{-6} per flight hour.

Functions and dependencies:

The system provides *Functions* that are necessary for the aircraft operation. The availability of each *Function* is derived from the analysis of the availability of the system components contributing to its implementation. Concretely, apart from its identification

name or number, a function is characterized by its state, which is defined by a conditional function based on the system components state and other basic function states.

Hence, one can use the following expression for the definition of the basic functions provided by the system:

$$f_{b=} = g\ (C_1S, C_2S, \ldots C_{nk}S);$$

where C_1S, C_2S, … $C_{nk}S$ are the variables representing the state information of the components involved in the accomplishment of f_b, g is a conditional function formulating the *relation* between the components states and the function f_b.

The state information of high-level functions, depending on other functions can be formulated as follows:

$$f_{h.} = g\ (C_1S, C_2S, \ldots f_{b1}, f_{b2}, \ldots).$$

A component behavior description, especially the *Failure* event, may use the state information of a *Function* in its description. It may also use other components' state information in its behavior description. The information provided represents *Dependency* information that may be used in the component description. The *Dependency* information is characterized similarly to *Functions,* with

$$stateInfo = g\ (C_1S, C_2S, \ldots f_1, f_2, \ldots).$$

The dependency information may be obtained from qualitative data such as FMEA data and failure propagation scenarios that are produced during design phase analyses. The FMEA identifies the effects of the failure events, which determine dependencies between the components. For example, as consequence of a *leak* in Pipe1, the pump may have no input fluid and will be `lost` if its `actuation state` is `ON`.

Requirements:

Based on the functions and the components' states one can express requirements associated with the dependability measure to assess. The requirements can be expressed using *Boolean expressions*, based on state variables, representing the combination of functions or system components that need to be available for the attainment of the assessment goal.

Min_Sys_R is defined accordingly. The requirements, defined by a Boolean expression, are determined by the availability of some required functions $f_1, f_2, \ldots f_{nf}$:

$$Min\_Sys\_R = f\ (f_1, f_2, \ldots f_{nf}).$$

II.2.2 The Mission dependent model

The mission dependent model is intended to capture the *Mission profile,* its specific requirements and the maintenance achievement information.

Mission profile

We describe the sequence of flights composing the mission profile, and their appropriate decomposition into phases.

The actual execution of a flight is subjected to the success of its prior preparation activities. Indeed, the decision to dispatch may be conditioned by the success of required maintenance activities as described in section I.4.2.2 of chapter I. Moreover, as the ultimate goal is to help in reducing operational disruption due to technical problems, the ability to achieve the activities and get ready for the execution of the flight should be considered. Therefore, the complete successful achievement of a flight consists of the success of *Ground period* activities, prior to the flight, and the success in executing the *flight* itself. Figure II.7 illustrates the decomposition into *Ground period* and *flight period*.

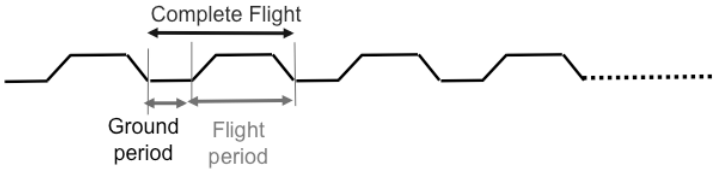


Figure II.7 Mission profile representation

Let *Gp* denote a *Ground period* and *Fp* the *Flight* execution *period*. The couple *CF =(Gp, Fp)* or $CF = Gp \bullet Fp$ represents the *Complete flight* achievement process, from the flight preparation activities to the flight end. In the following, the operator "•" symbolizes a succession of activities or periods.

A mission profile *Mp* composed of *n* flights is formulated as follows: $Mp = \bullet_{i=1..n} CF_i = \bullet_{i=1..n} (Gp_i, Fp_i)$.

Each flight period *Fp* is decomposed into *Phases* that are distinguished by the system functionalities required for their success. Assuming that a given number *p* of *Phases* is identified for each flight, we have $Fp = Ph_1 \bullet Ph_2 \bullet \ldots Ph_p$. The order of the *Phases* in the notation is important in the sense that the *Phases* are achieved successively. Each *Phase* Ph_j has a duration determined by a *Duration distribution* DPh_j that may be deterministic.

Let *I* denote the occurrence of an interruption during the mission. *I* might occur during a *Flight period* or a *Ground period*.

The interruption of a flight is defined as the interruption of one of its *Phases*. A *Phase* interruption is defined as the loss of its *Requirements* fulfillment when the phase is ongoing. The requirements fulfillment of a *Phase* Ph_i is represented by a Boolean variable RPh_i, which indicates whether the requirements are fulfilled or not.

At each *Ground period*, it should be ensured that the aircraft meets the *Requirements* in order to achieve the next flight, and some activities should complete before departure time; otherwise a delay may occur. A ground period can consist of: i) scheduled maintenance (SM) activities and other ground activities (OGA) or ii) scheduled maintenance activities extended by unscheduled maintenance (UM) activities, followed by

the other ground activities. It is noteworthy that the duration of an unscheduled maintenance period UM corresponds to the time needed to perform the maintenance activities, which are taken into account in the core model as events associated to the affected components.

Accordingly, each ground period can be represented by the succession of the corresponding activities: *Gp=SM • UM • OGA*. SM and OGA have an associated duration determined by a distribution function, which depends on the considered location. The extension of an SM activity with an UM activity depends on the operational state of the system and the necessity to undertake maintenance activities. An UM activity generally takes place when the dispatch requirements (denoted *DR*) are not met. It is generally dedicated to the repair of the critical system components that are needed to perform the flight. A ground period has a deterministic planned duration *pd(Gp)*, which indicates the maximum time beyond which a delay or a cancellation of the flight is considered.

Mission requirements

The requirements to be satisfied for the successful achievement of the missions take into account the decomposition of the mission profile into successive flight and ground periods. The successful accomplishment of the phases Ph_j of a flight is conditioned by the availability of a group of functions $f_1, f_2, \ldots f_{nj}$ delivered by the aircraft system, defined in the core model. Thus, the availability of these functions corresponds to the requirements to be satisfied during each phase in order to ensure the successful evolution of the flight. These requirements, denoted as RPh_i, can also be expressed through the identification of the combination of function losses that would lead to the interruption of the flight phase.

The requirements associated to a given mission composed of *n* flight cycles result from the aggregation of the requirements associated to each flight of the mission. For a given flight CF_i, DR_i and $RPh_{j=1..p}$ are the requirements related to ground and flight phases. For dispatch requirement DR_i, the required functions are almost the same whatever the flight to be achieved. The required combinations of functions, needed for every flight, are defined by *Min_Sys_R*. $M\_Prof\_DR_i$ expresses the additional requirements that are specific to the mission profile under investigation. These mission profile specific requirements can be related to the availability of some functions and to the achievement of the maintenance activities (*Ma*) required to dispatch the flight, if any. Accordingly,

$$M\_Prof\ \_DR_i = f(f_1, f_2, \ldots f_{nf}, Ma);$$

$$DR_i = Min\_Sys\_R \wedge M\_Prof\ \_DR_i.$$

Maintenance achievement:

Maintenance activities at each ground period are characterized by the resources available such as spares and technicians. A logistic delay function LDF_{Gp} can be used to determine the additional delay that the ability to have the facilities necessary for the maintenance tasks at the considered ground period Gp, may add on the nominal maintenance duration. A maintenance strategy should also be defined with a priority level associated to each component in order to determine the order of the maintenance activities when several components are failed.

The model content specification also helps in highlighting the major variables that may be affected by changes, without referring to the specificity of the ultimate formalism that will be used to build the model.

II.2.3 Main variables that may be affected

The main variables, whose values may be affected by changes, are presented considering the categories of information presented in Figure II.2 and which were fundamentally used to establish the model content.

Mission profile: From the mission dependent model description, it appears that the definition of a mission profile requires the knowledge of the number *n* of flights composing the mission, the phases composing each flight and their durations, the ground period activities durations, and the sequencing of the ground and flight phases. The duration of the ground period activities, beyond which the flight is interrupted, is also necessary. The changes may then concern the number *n* of flights, the parameters of each flight, and the ground period parameters. The parameters of a flight are the duration distribution of the phases DPh_1, DPh_2, … DPh_p. For a ground period, the parameters concern the total duration allocated to the ground activities pd_{Gp}, the estimation of scheduled maintenance activities (SM) and the other activities (OGA) durations. These durations are obtained using estimated values given whether by an operator, or based on historical data. These changes only concern basic parameter changes.

Requirements: When the mission profile changes, the functions required for the achievement of the flights in the new mission profile may change. That is for each flight *Fp* in the new profile, the required functions allowing it and those required for each of its phases ($RPh_{j=1..p}$) may be different. The specification of the requirements consists in defining combinations of predefined aircraft functions. The requirements can be specified by selection of predefined ones or combination established by the operator, based on well-known aircraft functions. It will be, however, very unlikely to change the requirements initially expressed in the model.

Maintenance: Changing the mission profile also implies changes in maintenance facilities available at the various stops (or destinations). The changes concern the logistic delay functions LDF_{Gp} related to the ground periods involved in the mission profile. These may be obtained based on historical maintenance data related to the ground periods.

System behavior: It appears from the core model description that the relevant characteristics that define a system component are its state, its failure and maintenance duration distributions. Thus, the changes may concern the components initial state C_lS, and their failure and maintenance distribution ($Fdistr_l$, $Mdistr_l$). For the state of the components, it consists in changing its current initial value to another value of its domain C_lSD. The impact on the global model is just the change of its initial configuration. For the failure and maintenance distributions, it consists in considering a new distribution function or new values for the distribution function parameters, in order to have a better fitting of the current event occurrence distribution.

Changes affecting the system concern the core model update and those related to the mission profile, including the requirement and the maintenance facilities, concern the mission dependent model.

Concluding comment

It is worth remembering that the model is intended to be used during the aircraft operation for dependability assessment. Accordingly some information in the model may need to be updated in order to consider up to date information for the assessment. This constraint greatly impacts the granularity of the model specification.

Based on the model content description, which helps to highlight the main variables that may be affected by changes from an up-to-date model to another one, we develop the meta-model, which will be used as a basis for the construction of the operational dependability model. The meta-model is an abstraction that structures the data to be included in the model.

Finally, it is worth noting that the dependability assessment requires concrete models using an appropriate formalism. The concrete model may include more details than the information presented in the specification. Especially, the internal decomposition into atomic model components and the representation of the dependencies, so as to facilitate the model processing, depend on the modeling formalism and, thus, may be differently implemented. The meta-model will highlight the elements that are needed to implement the concrete model, but will keep a sufficient level of abstraction with respect to the targeted modeling formalisms.

II.3 The Meta-model

Meta-modeling consists in abstracting a given category of models. It consists in defining the type of data that must be included in the models, their structuring, and the constraints that they must comply with. In the context of our work, the meta-model structures the elements defined in the model specification to give a high-level overview of their relationship.

II.3.1 Meta-modeling means

Meta-modeling has been the purpose of an Object Management Group (OMG) standard, Meta-Object Facility – MOF [OMG 2010]. MOF is a framework that provides means for meta-models representations, and interface specifications for the corresponding models manipulation. MOF is based on a hierarchical meta-modeling paradigm [Karagiannis and Kühn 2002]. The hierarchy is structured into four layers, listed hereafter from the higher level to the lower level.

- M3: meta-meta-model layer
- M2: meta-model layer
- M1: model layer
- M0: real objects or run-time objects layer

The underlying philosophy is that every meta-model is after all a model and, thus, can be abstracted to another meta-model which becomes a meta-meta-model for the initial model. In this hierarchy established for both models and meta-models representation, the elements of a given layer are represented using the features provided by its immediate higher layer. MOF is designed to be the top layer (M3) in the meta-modeling hierarchy. As MOF belongs to the top layer, it is a self-describing formalism. There are two definitions of MOF: a basic definition called the Essential MOF (EMOF) and a complete definition called Complete MOF (CMOF).

The open source project Eclipse has also developed for a purpose of model driven engineering support, a modeling framework, named Eclipse Modeling Framework (EMF) [Griffin 2003], that includes means (Ecore) for meta-modeling. EMF is conceptually very similar to MOF [Gerber and Raymond 2003]. Both are based on the concepts of classes with attributes and operations. They principally use UML class diagram features for the meta-model representation. MOF is a meta-modeling standard and EMF can be seen as a platform specific implementation [Gardner 2003].

Our meta-model is developed based on EMF since it is implicitly supported by the existing and well-known platform Eclipse. The main features of EMF are presented in Figure II.8, which shows their graphical representation.

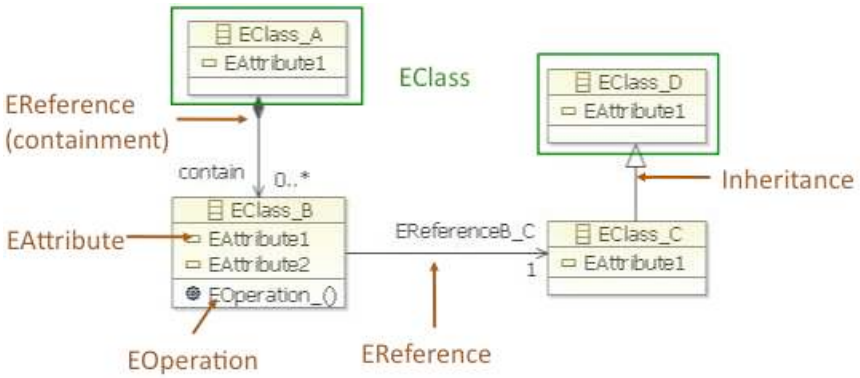


Figure II.8 Ecore features

EClass: represents an abstraction of a model element, characterized by some given attributes and operations, respectively named EAttribute and EOperation.

EReference: represents an oriented relationship between two objects. In a software-modeling viewpoint, it represents the fact that the objects of the source EClass have properties that are references to objects of the destination EClass. In our case, the orientation indicates that the source object uses information of the destination object. An EReference can be declared containment, expressing the fact that the destination object is completely part of the source object. An EReference is graphically represented by an arrow.

Inheritance: represents a relationship between two EClass, defining the source EClass as a particular subtype of the destination EClass. An Inheritance is graphically represented by an open-headed arrow.

The following presents the meta-model proposed for the operational dependability model construction. The description is based on the two major model parts.

II.3.2 The core meta-model

The meta-model for the system behavior description is shown in Figure II.9.

The system components representations are abstracted with the EClass *SysComponent*. As presented in the formal description, the representation of a system component is characterized by its identifier (*id*), its state variables (*stateVars*) and the *events* it may be subjected to.

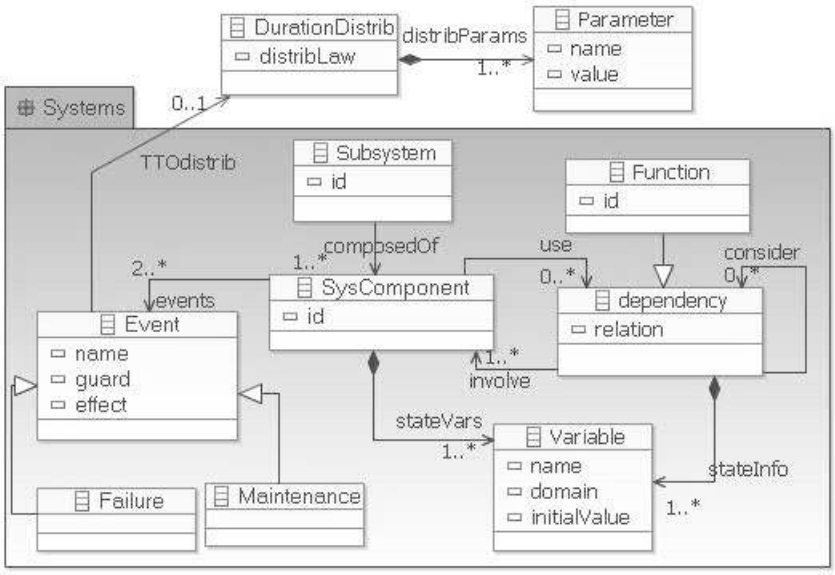


Figure II.9 The System meta-model

Several state variables may be used in the description of a component. An EClass *Variable* is defined to abstract them. A *Variable* has a *name*, a *domain*, which defines the possible values that results from the changes of the component state, and a value (*initialValue*) that will correspond to its initialization in the model.

For the events related to a component, *Failure* and *Maintenance* events are distinguished as they represent the main events. An *Event* is described by its *name*, the *guard* representing the condition under which it may happen, the *effect* representing the changes in the state of the system after its occurrence, and the occurrence delay distribution (time to occurrence distribution: *TTOdistrib*). *TTOdistrib* is an object of the EClass *DurationDistrib*, which is aimed at representing the distribution law followed by the duration spent in a given situation. The distribution is described by the name of the distribution law (*distribLaw* - e.g., exponential, weibull, deterministic) and its *distribParams,* which are abstracted based on the EClass *Parameter*. A distribution *Parameter* is characterized by its *name* and a *value*.

As mentioned in the specification of section II.2.1, it is necessary to express dependencies between the components. Concretely, a component may have to use other component

attribute values in its behavior description. In this case, either the attribute value is directly accessible, or the value is obtained via an intermediate object, which makes the link between the two components. This is abstracted by the EClass *Dependency*, which is intended to combine information from different components, based on a conditional function (*relation*). The resulting information (*stateInfo*) corresponds to the dependency information that can be used in the dependent component behavior description. The *Functions* delivered by the system components are also specified as a particular case of *Dependency*, since the state information of a *Function* is also derived based on the states of its contributing components.

The functions, as well as the system components state information, are used to define requirements related to the system. Figure II.10 shows the features for the requirements specification.

A requirement is a constraint related to a part of the system or the mission profile, which must be satisfied in order to succeed in achieving a given part of the mission. It is abstracted by the EClass *Requirement* and is characterized by i) a *reference* that is an identifier, which may be used to reference the same requirement in different situations, ii) a Boolean variable *status* that will indicate whether the requirement is satisfied or not, and iii) a constraint or a Boolean *expression*. The Boolean *expression* formulates a condition, involving the system components and function states, that needs to be satisfied otherwise an interruption may occur during the mission achievement. It can be based on *combination* of other requirements.

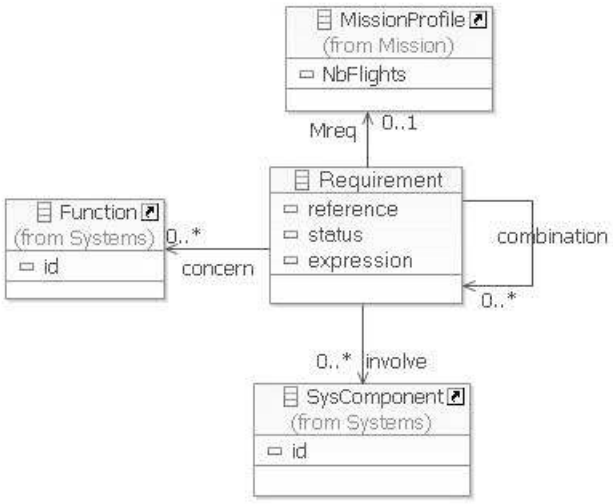


Figure II.10 Feature for requirement specification

Requirements that do not require any particular information from the mission profile can be expressed as part of the core model. Also every kind of requirements or system information that might be necessary in defining a mission profile is represented so as to facilitate other specific requirements expression when needed.

The resulting core model can be used for a stepwise simulation of the system behavior. The corresponding model in AltaRica can also be used to generate cut sets corresponding to events that may lead to the loss of a given function or the non-fulfillment of some

defined requirements. There is no need to set the time to occurrence distribution (*TTOdistrib*) objects associated to the events for the cut set generation. They are however needed if one wants to compute quantitative measure. The core model, set with these objects, can be used to evaluate the probability to operate the system, according to some given requirements, during a given time, known as reliability measure.

II.3.3 The mission dependent meta-model

Figure II.11 shows the meta-model for the representation of the mission profile information. A mission profile is defined by a given number (*NbFlights*) of sequenced flights to be achieved, and the related maintenance strategy.

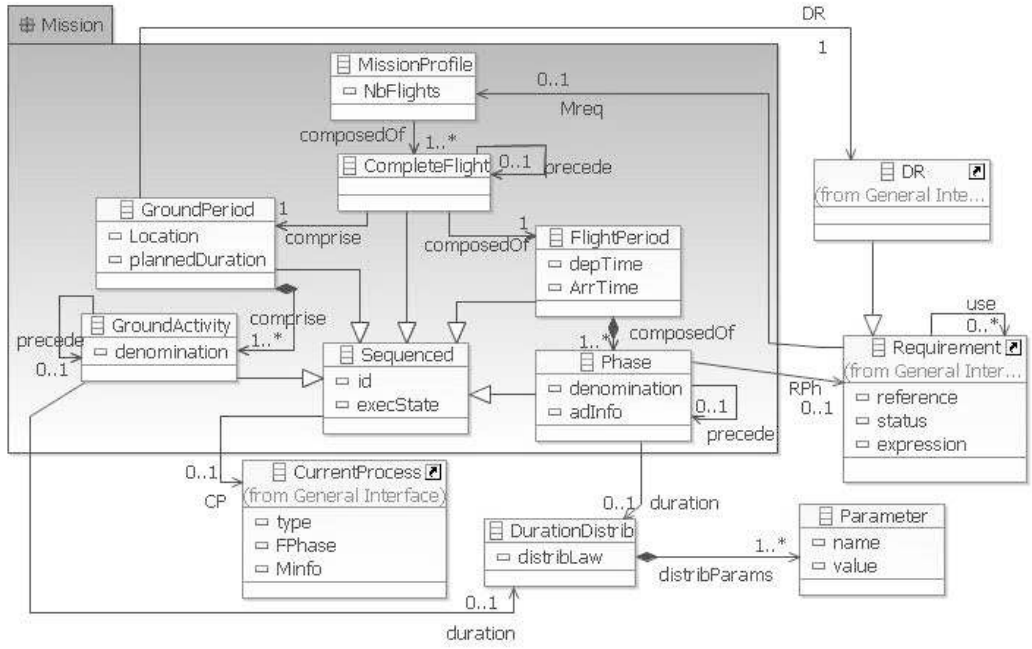


Figure II.11 Mission profile meta-model

For the achievement of each flight, as presented in the specification of subsection II.2.2, a *Ground Period* to get ready for the flight, and a *Flight Period* that consists in the actual execution of the flight are distinguished. The whole process to achieve the flight is named a *CompleteFlight*.

The ground period precedes the flight period and is composed of the description of the activities (*GroundActivity*) that are achieved on ground until departure clearance. The ground activities are characterized by their *denomination* and *duration*. The duration may be probabilistically distributed.

The *Flight Period* is decomposed into different *Phases* based on the *requirement* that must be fulfilled. A flight *Phase* is also characterized by a *denomination*, a *duration* and additional information (*adInfo*) that might be necessary in the requirements definition.

As the mission profile is decomposed into a sequence of periods and phases, an Eclass *Sequenced* is defined to represent their common characteristics, which are the identifier

(*id*) and the attribute *execState*. The attribute *execState* will indicate whether the corresponding part of the mission is in its achievement state. The information of the mission part that is being achieved is transmitted to the core model based on the Eclass *CurrentProcess*. The information concerns the *type* of mission part (ground period or flight phase), the flight phase identifier (*FPhase*) if it is a flight phase, the maintenance authorization information if it is a ground period. *CurrentProcess* will be used to create an interfacing object between the core and the mission dependent models.

On ground the operational state of the system is tested against dispatch requirements (DR), which are the synthesis of the MEL. *DR* corresponds to *Min_Sys_R* if there is no specific mission requirement. The maintenance activities are such that they cannot be considered completed if the dispatch requirements related to the system state are not met. The success of a ground period is determined by the completion of its activities within its planned duration (*plannedDuration*). The success of a flight is determined by the success of its phases' achievement. A phase is successfully accomplished if its related requirements are met during its achievement.

The achievement of the maintenance activities depends on the availability of adequate logistics in the considered ground period. A maintenance *Station* is associated to each ground period and the dependence is taken into account by considering a probability distribution characterizing the time needed to get ready for the corresponding maintenance activities. For example a logistic delay function (*LDF*) is specified to take into account the delay in supporting the activities. *LDF* is a subclass of *DurationDistrib* (Figure II.9). *Types* of maintenance station, such as main-base and outstation, can be used to categorize them. The tasks are done considering a *Prioritization* in the repair of the failures. Figure II.12 shows the corresponding features.

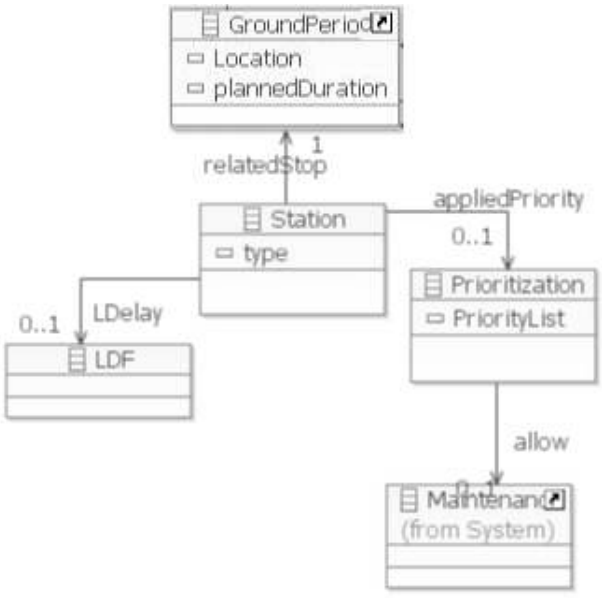


Figure II.12 Meta-model features for maintenance related information representation

The complete instantiation of the mission dependent meta-model, as a model, requires in-operation information. Different scenarios of flight decompositions or maintenance strategy may be considered. A flight is basically considered as a sequence of phases.

Flights integrating possible choices of phases (contingency phases to cope with degradations for example) resulting in a complex profile may be considered. The mission profiles may also be similar. The mission dependent model obtained is composed with the core model based on the requirements to fulfill during the mission achievement.

II.3.4 Concluding comment

The meta-model is an abstraction aimed at defining features for the model construction. The attributes of the classes defined may be enriched with other specific information. The graphical representation, provided based on the Ecore meta-modeling means, is very convenient for an easy overview of the model content. The corresponding models will consist in a representation using instances of the components described and considering their relationship. The meta-model can also be used to build models for different purposes. The models can be built only based on a particular subsystem, or it can be built considering several subsystems.

II.4 Conclusion

The modeling approach concerns aircraft operational dependability assessment during its operation. The model is expected to deal with the dynamic evolution of the system behavior, its mission profile and maintenance activities, so as to produce accurate results for short-term operations. Given the complexity of current aircraft systems and the challenge related to the model update and dependability re-assessment during operation, it is necessary to carefully consider establishing a convenient approach to the operational dependability assessment.

We have determined the relevant categories of information to be considered, and the changes that should require an update and adaptation of the model in order to make it consistent with the current situation in operation. The update of the model should not require the presence of modeling specialists.

The adopted modeling approach consists in constructing a stochastic dependability model, based on a meta-model that represents a common basis for the construction of models corresponding to different types of aircraft. The model that is adapted to the current situation in operation, referred to as up-to-date model, is obtained by a parameterization of the stochastic model.

Based on the relevant categories of information that must be considered, two major parts have been identified in the model; the model content is specified respecting these parts. The overall content is described together with the meta-model, which highlights the model components and structure. The major changes from an up-to-date model to another one are also analyzed.

The proposed modeling approach encompasses the use of the model for safety analyses as well as for operational dependability analyses. The core model is especially designed to capture qualitative data such as the FMEA results that are involved in safety analyses. In this chapter, we emphasize on the use of the model during operation for dependability

assessment, as it represents a new application context of aircraft dependability assessment, that deserves a particular attention.

The stochastic model can be built based on an appropriate dependability modeling formalism. The AltaRica formalism and the stochastic activity network formalism, two suitable formalisms, are examined in the next chapter.

III Model Construction Based on AltaRica and Stochastic Activity Network Formalisms

The previous chapter addressed the specification of the detailed content of the model, which is to be built using a formalism that can be processed to obtain the assessment results. Different formalisms are used in the dependability analysis field to support model construction. State space based formalisms are the most appropriate to model complex systems thanks to their great expressiveness, especially their capacity in dealing with dependencies among components. Therefore it is worth choosing a state-space-based formalism to support the modeling tasks in our framework. Since Airbus and its collaborators have been successfully using the AltaRica formalism to support model-based safety analysis, it was chosen as a supporting formalism for the construction of the model. However, AltaRica and its available supporting tools are still more focused on qualitative analysis than quantitative analysis, and the stochastic analysis tool dedicated to our assessment framework, called EPOCH, is still under development [Teichteil-Königsbuch et al. 2011]. In order to proceed to quantitative evaluation during the implementation of the modeling approach in case studies, we have selected the Stochastic Activities Network (SAN) formalism that is well known for dependability and performability analyses. The SAN formalism intrinsically integrates stochastic behavior description features and is supported by an available tool that allows for quantitative evaluations.

We present in this chapter a comparative analysis of the SAN and AltaRica formalisms, which shows their main characteristics and the modeling aspects on which they are interchangeable. Both formalisms claim their generalization of Petri nets, essentially the states/transitions mechanisms. They are developed to make model construction more flexible and easier to manage. The SAN formalism is closer to Petri Nets since it is based on the same concepts and graphical representation, whereas AltaRica is designed to make the system architecture more transparent through the model, by means of node connection mechanisms. Therefore, the two formalisms are compatible on their basic features, and model transformation from each one to the other one can be considered.

A presentation of the two formalisms is given firstly. Then, we present a comparative overview of their characteristics. Methods to transform models from one of them to the other one are examined at the end.

III.1 Presentation of the two formalisms

The two formalisms are presented based on an example. We consider a simple system S composed of two components C1 and C2, which deliver a function f1. We assume that the components are either delivering correct service (state ok) or are failed. The function is delivered when at least C1 or C2 is ok.

We present firstly the SAN formalism, then the AltaRica formalism.

III.1.1 Stochastic Activity Network (SAN)

SAN is a stochastic modeling formalism using the basic notions of place, marking and transition of Petri nets. They can be seen as a natural extension of stochastic Petri nets with the distinction of instantaneous and timed transitions (activities), the use of input and output gates to model transitions preconditions and post conditions, as well as the possibility to model choices upon the firing of a transition by associating cases. Figure III.1 shows the different features of SAN.

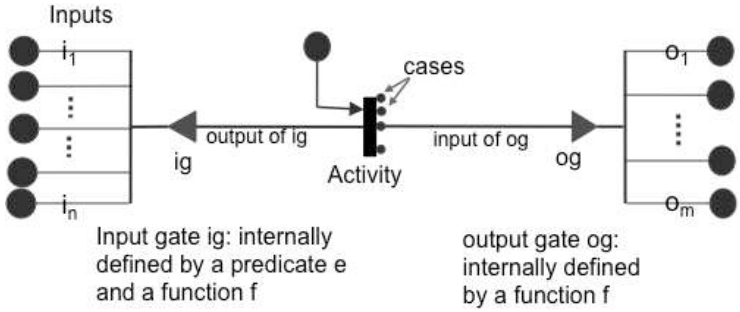


Figure III.1 SAN input gate definition

The transitions are called *activities* and when an activity fires, it is said completed. A distribution function is associated to each timed activity. The distribution function determines the completion time of the activity when it is enabled. When an enabled activity act1 is preceded by another activity act2 in completion time, act1 may be reactivated, i.e. a new completion time may be determined for act1, depending on the new marking after the completion of act2.

Input and *output gates* are introduced in SAN to offer greater flexibility in defining "activities enabling" and "completion rules".

An *input gate* has:

- a finite set of inputs, each one associated to a place, and
- one output, associated to an activity.

It is internally defined by

- an n-ary computable predicate, called enabling predicate, and
- an n-ary computable partial function, called input function and defined for all values for which the enabling predicate is true.

An *output gate* has:

- one input, associated to an activity, and
- a finite set of outputs, each one associated to a place.

It is defined by an n-ary computable function over the markings of its output places, called the output function.

Cases represent uncertainty at the completion of an activity. Different configurations may be possible when an activity completes, and only one of these configurations must be chosen. Cases are used to represent these possible choices. A probability distribution is associated to the cases of each activity. The distribution defines the probability for each case to be chosen at the completion of the activity.

Figure III.2 shows a SAN model of S. Places *C1* and *C2* represent the state of the components. Associated activities $Faili_{i=1,2}$ and $Mai_{i=1,2}$ model their failure and maintenance.

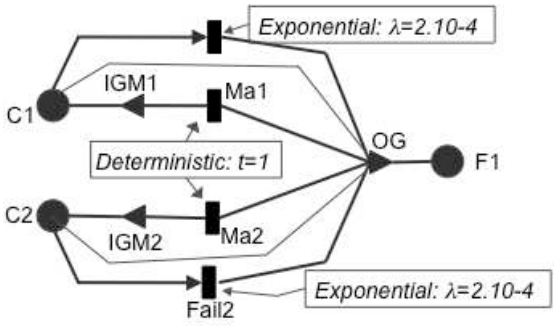


Figure III.2 Example of SAN model

Place *F1* represents the state of function f1. *IGM1* and *IGM2* are input gates expressing the condition to activate maintenance activities and the markings update function. The output gate *OG* models the availability of function F1 depending on the marking of C1 and C2. The function is unavailable when both components are failed. The specifications of the distribution function associated to the activities are shown in Figure III.2. The internal specifications of the gates are given in Table III.1. The predicates and the functions of the gates are expressed in C++ in the Möbius tool.

	IGM1	IGM2	OG
Predicate	*C1->Mark()==0*	*C2->Mark()==0;*	--
Function	*C1->Mark()=1*	*C2->Mark()=1;*	*If(C1->Mark()==0 && C2->Mark()==0)* *F1->Mark()=0;* *Else F1->Mark()=1;*

Table III.1 Predicates and functions of the gates in the example of SAN model

In the previous example, none of the activities has multiple cases associated. To illustrate the use of cases, let us consider that the components C1 and C2 integrate a fail-safe mechanism that disconnects the component when it is failed, otherwise the function is not delivered even if the second component is ok. Also, let us assume that following a failure there is 90% of chance for the component to be disconnected and 10% that it remains connected. This is represented using cases associated to activities *Fail1* and *Fail2,* as shown in Figure III.3.

Two cases are distinguished for each of the activities *Fail1* and *Fail2*. The first one leads to the nominal determination of the function delivery, expressed based on the output gate OG1, and the second one to which is associated the output gate OG2, is the case in which

the component is not disconnected systematically, leading to the non delivery of the function f1. Probabilities of selection are associated to each of the cases.

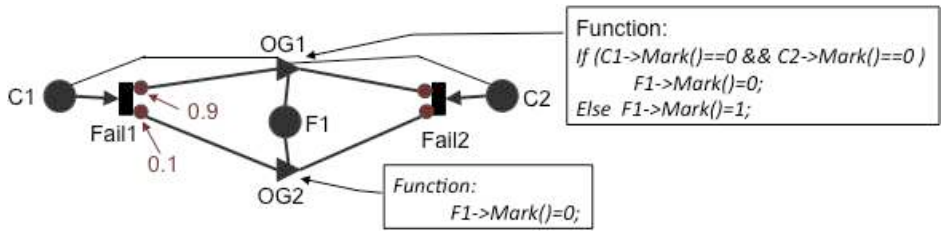


Figure III.3 Example of SAN model with cases

The dynamic of the net is that an activity is activated when the predicates of all the associated input gates hold, and at the completion of the activity, the functions of the associated input gates are executed together with the functions of the output gates associated to the selected case.

One can see that SAN models are very compact and manageable thanks to input and output gates. A single gate can be used to define predicates and functions involving several places.

- **Formal definition**

SAN is formally defined based on Activity Networks (AN) definition. Let P denotes the set of all the places of the network.

If S is a set of places ($S \subseteq P$), a **marking** of S is a mapping $\boldsymbol{\mu: S \rightarrow \mathbb{N}^2}$**.** The set of possible markings of S is the set of functions $M_S = \{\mu \mid \mu: S \rightarrow \mathbb{N}\}$.

An **input gate** is defined as a triple **(G, e, f)** such that:

- $G \subseteq P$ is the set of input places associated to the gate;
- $e: M_G \rightarrow \{0, 1\}$ is the enabling predicate of the gate;
- $f: M_G \rightarrow M_G$ is the input function of the gate.

Similarly, an **output gate** is a pair, **(G, f)**, where $G \subseteq P$ is the set of the output places associated to the gate and $f: M_G \rightarrow M_G$ is the output function of the gate.

An **Activity Network** is an eight-tuple $\mathbf{AN = (P, A, IG, O, \gamma, \tau, \iota, o)}$ such that :

- P is a finite set of places;
- A is a finite set of activities;
- IG is a finite set of input gates;
- O is a finite set of output gates;

[2] The markings are considered to be positive integers but any type can be used provided that the functions defined in the gates are coherent with them.

- γ: A $\rightarrow \mathbb{N}^+$ defines the number of cases for each activity;
- τ: A $\rightarrow$ {Timed, Instantaneous} defines the type of each activity;
- ι: I $\rightarrow$ A maps input gates to activities;
- o: O $\rightarrow$ {(a, c) | a $\in$ A and c $\in$ {1, 2, . . . γ (a)}} maps output gates to cases of activities.

It is noteworthy that the net structure is defined by the functions ι and o.

A **stochastic activity network** is a five-tuple **SAN = (AN, μ_0, C, F, H)** such that:

- AN is an activity network;
- μ_0 is the initial marking;
- C is the case distribution assignment, such that for any activity *act*, in a marking μ, the assignment of values over the set {1, . . . γ (act)} of cases is a probability distribution called the case distribution of activity *act* in μ;
- F is the activity time distribution function assignment, an assignment of continuous functions to timed activities such that for any timed activity *act* and marking μ in which activity *act* is enabled, F_{act} (μ, ·) is a continuous probability distribution function;
- H is the reactivation function assignment, an assignment of functions to timed activities such that for any timed activity *act* enabled in μ, H_{act} (μ, .) is a set of markings in which the activity *act* is reactivated if one of them is reached before activity *act* completes.

AN is considered stabilizing in the initial marking μ_0, i.e., there is no reachable marking leading to the firing of an infinite sequence of instantaneous activities.

More details about the SAN formal definition can be found in [Sanders and Meyer 2001].

III.1.2 AltaRica language

In contrast to SAN, AltaRica is a language with a set of keywords and a syntax. AltaRica is developed to facilitate the joint use by dependability analysis specialists and system developers. It was developed with the objective of helping create models that are very close to the system functional architecture. A system is considered composed of components connected to one another. An AltaRica model is a hierarchical description consisting in the system global functional architecture, which is detailed step by step considering its lower level components. The model is composed of nodes and instructions that are used to initialize and connect the nodes. A node may be composed of sub-nodes that can also be subdivided into lower level sub-nodes. The nodes are detailed using mode automata [Rauzy 2002][3] based on state variables and transitions. Flow propagation instructions are used to represent structural and functional links between the nodes

[3] A new version of AltaRica [PERROT et al. 2010] with considerable modifications of the version described here is being developed.

according to the system structure. The following gives, in AltaRica, a description of the two components system, presented at the beginning of this section.

```
node S
 flow
    F1:bool:out;
 state
    C1:{failed, ok};
    C2:{failed, ok};
 event
    Fail1, Fail2, Ma1, Ma2;
 trans
    (C1 = ok) |- Fail1 -> C1 := failed;
    (C1 = failed) |- Ma1 -> C1 := ok;

    (C2 = ok) |- Fail2 -> C2 := failed;
    (C2 = failed) |- Ma2 -> C2 := ok;
assert
    F1 =((C1 = ok) or (C2 = ok)) );
  init
    C1 := ok,
    C2 := ok,
extern
   law <event Fail1 > = exponential(2.0E-4);
   law <event Ma1 > = Dirac(1);

   law <event Fail2 > = exponential(2.0E-4);
   law <event Ma2 > = Dirac(1);
edon
```

The keyword *flow* is used to declare the variables that represent the input and output information of the node. These variables can be linked to flow variables of other nodes; output variables of a given node are linked to input variables of other nodes. In this example, the variable *F1* represents the availability of the function f1, provided by S and which can be used as input for other nodes at a higher level. Keyword *state* introduces the state variables representing the components states. *Event* is used to declare the events that may affect the system. Keyword *trans* introduces the section where the dynamics of the system (transitions enabling guards and state changes after the transitions fire) is described. The guard is expressed based on input variables and state variables. The transitions change only the value of the state variables. The corresponding changes in the flow variables (output variables) are done based on the statements of the *assert* part of the node. The *assert* statements are also used to make links between nodes. The *init* part is used to initialize the state of the components. The *extern* part is used to provide additional information devoted to assessment tools. The distribution laws of the events are provided in the *extern* part.

The previous AltaRica representation of S is developed in such a way to be similar to the SAN model. The different components of S are not distinguished in the model. In fact, AltaRica is rather a component oriented modeling language. A convenient representation of S in AltaRica should consist in using sub-nodes for the component C1 and C2, and composing them according to the combination logic that leads to the availability of function f1. As the components are similar, a generic node *Component* should be created to represent the components C1 and C2.

```
node Component
 flow
  stateOk : bool : out ;
 state
  status : {ok,failed} ;
 event
  Fail, Ma ;
 init
   status := ok ;
trans
  status=ok |-Fail -> status:=failed;
  status=failed |-Ma ->  status:=ok;
assert
   stateOk=(status=ok);
extern
   law <event Fail > = exponential(2.0E-4);
   law <event Ma > = Dirac(1);
edon
```

The global model is constructed using "instances" of *Component*, as follows.

```
node S
 flow
    F1 : bool : out;
  sub
    C1: Component;
    C2: Component;
assert
    F1=( C1.stateOk or C2.stateOk );
edon
```

Doing so, the different structural components of the system are distinguishable and one can associate a graphical representation that is more meaningful for non-specialists. Figure III.4 shows a graphical representation corresponding to the model of S. Each box represents a sub-node and the links between the boxes are represented based on input and output variables using *assert* statements.

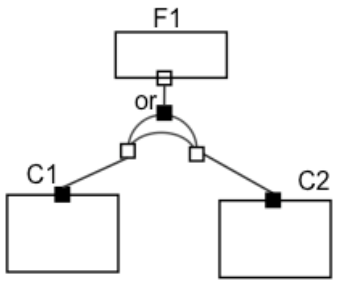


Figure III.4 Example of AltaRica model representation

The features shown in the representation are not formalized; they are informal elements that are defined by the model developer. The AltaRica language itself has, however, formal bases.

- **Formal definition**

Several versions of AltaRica have been defined. We present here the formal definition of the AltaRica dataflow.

An AltaRica node is formally defined based on mode automata. A **mode automaton** A is a 9-tuple $\mathbf{A = (D, S, F^{in}, F^{out}, dom, E, \delta, \sigma, I)}$ such that:

- **D** is a finite or an infinite domain;
- **S**, $\mathbf{F^{in}}$ and $\mathbf{F^{out}}$ are sets of variables (respectively state variables, input flow variables, and output flow variables);
- $\mathbf{dom : V \rightarrow 2^D}$ such that $\forall v \in V = S \cup F^{in} \cup F^{out}$, $dom(v) \neq \emptyset$ determines the value domain of v. For any subset $U \subseteq V$, we denote by VAL(U) the Cartesian product of its elements domains : $\mathbf{VAL(U) = \prod_{v \in U} dom(v)}$; VAL(U) is the set of all the possible assignments of values to the variables of U.
- **E** is a set of event names;
- $\mathbf{\delta: VAL(S) \times VAL(F^{in}) \times E \rightarrow VAL(S)}$, is a partial function that specifies the transitions corresponding to the events;
- $\mathbf{\sigma: VAL(S) \times VAL(F^{in}) \rightarrow VAL(F^{out})}$ is a total function that defines assertions over the variable values;
- $\mathbf{I \in VAL(S)}$ defines the initial state of the automaton.

More details about mode automata in the scope of AltaRica formalism definition are provided in [Point and Rauzy 1999; Rauzy 2002].

III.2 Characteristics and Comparison of the Two Formalisms

The differences between the two formalisms are principally their aims. SAN was defined in the early eighties to facilitate systems with complex stochastic aspects modeling. AltaRica was defined in the mid-nineties to model fault propagation in complex systems. We examine firstly the basic features used to model systems.

III.2.1 Basic features

Both formalisms provide features to represent information related to the system state and events that affect this information.

In SAN, places are used to represent the various parameters that characterize the system state. Places are basically marked with integer value but one can define extended places to represent any type of variable, especially when using the SAN version supported by the tool Möbius.

AltaRica distinguishes two types of information, the internal state information (state variables) and flow data that represent information exchanged between the components. This distinction is well suited for the enhancement of the model's modularity but it

complicates the representation of resources shared by several components. A module cannot modify the internal state variable of other modules and, therefore, the representation of shared resource, whose state is modifiable by several modules, is not straightforward.

Concerning the representation of events that may modify the system state, both formalisms are based on the transitions firing schema: predicate - event - post condition. The predicate is a Boolean condition that must be fulfilled in order to allow the event occurrence.

For the transition firing mechanism, the SAN formalism is mainly dedicated to stochastic modeling for performance and dependability assessment. The firings of the transitions are based on the notion of activities that complete after given durations when activated (predicate fulfilled). Instantaneous activities, which complete in a negligible amount of time, are thus distinguished from timed activities whose durations impact the system's ability to perform its functions. SAN ability to model stochastic system is based on the assignment of time distribution functions to activities. Furthermore, the time distribution functions can be state dependent.

Concerning AltaRica, it was not initially aimed at building stochastic models. It does not natively integrate the notion of time to fire when the transition is enabled. An AltaRica node is first of all an automaton aimed at exploring the possible states without a real notion of time duration. However, the possibility to associate probabilistic distribution laws to the events has been given via the use of statements introduced by the keyword *extern*. Priorities can be defined between AltaRica transitions in order to determine the firing order when several ones are in concurrence. Several events concerning different components can also be synchronized so that their occurrences happen simultaneously.

For the events post conditions modeling, SAN provides a feature that can be used to make a probabilistic choice upon the firing of a transition: cases. That is, when there is an uncertainty about the outcome of an activity, one can represent all the possibilities and associate a probability of selection to each one.

III.2.2 Modularity

The formal definition of SAN does not include the management of models' modularity. However some operators (Replicate and Join) have been defined in order to support the construction of composed models. The composition of the sub-models is based on a mechanism of place sharing i.e., the sub-models use jointly some places. The design of the different components is left to the model builder's appreciation. The Replicate operator is used to create copies of models and to combine them into a single model. The Join operator is used to combine different models into one model.

AltaRica intrinsically manages the modularity via the structuring of the model using instances of nodes. The definition of a node distinguishes the state variables, which are local to the node representing a component, from the flow variables, which are defined as connection points, forming the interface of the component with regard to other components. Every node manages locally its behavior and communicates with its environment via input and output variables. The mechanism of events synchronization

also facilitates the modular construction of the model. Synchronizations explicitly group and trigger simultaneously subsets of basic events. The synchronized events belong to different nodes and therefore, it is possible to separate two components that can be subject to effects of the same event into two nodes.

III.2.3 Compactness

One of the drawbacks of Petri Nets and their extensions is their complexity, especially the difficulty to model conditions and operations, integrating for example if-then-else operators, using the system variables. Therefore, it is worth to have a look at how SAN and AltaRica deal with this problem.

For SAN, the condition enabling activities are expressed using input gates, which can integrate any kind of functions, defined using the markings of the nets places, in the condition expression. The change of the variables at the completion of an activity can also be expressed using predefined functions in the input gates and output gates. The Möbius tool allows the use of any function from the C/C++ libraries.

As a language, AltaRica also allows the use of predefined functions in the expression of conditions and transitions, based on the variables. Functions can be defined as nodes of the model. The input variables of the node are used as parameters and the returned value is represented as an output variable. However, the AltaRica definition doesn't include any condition-control loop statement, such as the "while" or "do – while" statements that are part of the C/C++ language, and thus, can be used in SAN.

III.2.4 Robustness of the model

While constructing a model, it may happen that the developed model presents some aspects that make it technically not solvable. Occurrences of infinite loops and non-determinisms are some of the problems usually encountered. The features provided by the modeling formalism are the main sources where one can check their origin and their solution.

The SAN formalism integrates the use of instantaneous activities, which can be at the origin of infinite loops, since there may be hidden sequences leading to a cyclic activation. There is no way to automatically detect all these kinds of situation but to process the model and check the processing traces. Actually, the problem doesn't formally concern SAN models, since the formal definition states that the network must be stabilizing, but it is still not decidable whether an activity network is stabilizing [Sanders and Meyer 2001]. Non-determinism can also occur in a SAN model due to the possible presence of conflict between instantaneous activities. A solution is to define a priority order among them.

AltaRica uses the priority mechanism to cope with conflicts between transitions. There is still the problem of loops, which is complicated by the use of flow variables. Indeed, two kinds of loops can happen. First, assertions can lead to a cyclic definition of the flow variables. This is not allowed in the so-called data-flow version of AltaRica. It can be detected by a preliminary model analysis before the execution. Then, one can built oscillating models with instantaneous urgent transitions, which are not stabilizing.

The non-stabilizing problem can be prevented by good modeling practices. It can be detected at run-time by the SAN modeling tool Möbius and some of the AltaRica modeling tools.

III.2.5 Supporting tools and facilities

Modeling formalisms are interesting insofar as they offer means to describe systems. However, a formalism is useful only if it is supported by a modeling tool. The following discusses the proposed tools to support SAN and AltaRica, together with the facilities that they offer.

III.2.5.1 Supporting tools

Several tools have been proposed to support modeling with SAN. METASAN [Sanders and Meyer 1986] was the first tool. Then UltraSAN [Couvillion et al. 1991; Sanders 1995] was developed to improve the model solving run-time complexity encountered with METASAN. The Möbius tool [Daly et al. 2000], designed as a multi-formalism supporting platform, firstly integrated the modeling facilities of UltraSAN. These tools are mainly devoted to quantitative analyses. The SharifSAN tool and its evolution SANBuilder [Azgomi and Movaghar 2005] were developed to support new extensions of SAN and integrate model checking based on temporal logic facilities.

AltaRica models can also be developed and analyzed by various tools. The AltaRica project team proposes ARC and MEC [LABRI-Tools 2011] for AltaRica models checking. Dassault Aviation has developed an AltaRica model editor, called Cecilia Ocas, which integrates tools for stepwise simulation as well as failures sequences and cut-sets generation. Cecilia Ocas has a commercial version called BPA-DAS at Dassault System. These tools are mainly dedicated to qualitative analyses. The quantitative analysis is usually done based on the cut-sets generation and using for example the tool Aralia to compute the measure of interest. Other tools [Signoret et al. 2004] can be used to perform stochastic simulation based on AltaRica models. Finally, the tool SIMFIA from APSYS can be used to develop and analyze, inter alia, AltaRica models.

Additionally to the formalism's facilities, the supporting tools can provide means to facilitate the modeling tasks. Modeling tasks with Möbius and Cecilia Ocas, major supporting tools respectively for SAN and AltaRica, are presented in the following.

III.2.5.2 Möbius and Cecilia Ocas

Models construction with Möbius consists in the development of atomic models that can be composed together. Since the tool is designed to support multiple formalisms, sub-models from different formalisms can be composed. Reward variables are then defined upon the models to specify the dependability measures. During the construction of the model, global variables can be used to parameterize the model. These variables are used to create studies, which are groups of evaluations corresponding to different assignments of values to them. Möbius provides model solvers based on two classes of solution techniques: discrete event simulation and state-space-based analytical/numerical techniques. Any model can be solved using simulation.

The tool Cecilia Ocas provides means to construct AltaRica models with an association of graphical representation to each AltaRica node. The modeling task consists of the construction or selection of the components, gathered in a library as reusable components, and their connection following the hierarchical architecture of the system. Then a variable, called *observer*, describing a given combination of the components outputs, is created to represent the global state to analyze. According to the initial values of the state variables, one can generate sequences and cut-sets of events that may lead to the state described by the observer. A step-by-step events simulation can also be done. It consists in a display of the transitions and the state variables, where one can select and fire the enabled transitions successively in order to analyze the model's behavior. It is used for "what if" analyses.

III.2.6 Summary

Despite the difference in their form of presentation, the two formalisms are compatible on the essential modeling features.

- The two formalisms are based on states and transitions firing mechanisms. SAN is still very close to stochastic Petri nets but defines new features that facilitate the expression of complex predicates and actions after events occurrences. AltaRica uses variables that can also be easily combined to express predicates and actions.
- Distribution functions can be associated to events while using the two formalisms. It is systematic in SAN, while distribution functions are considered as external properties of the model in AltaRica.
- AltaRica defines input/output variables and assertions that are used to build hierarchical models. Hierarchical models can be built in SAN using the Join operator and shared places.

However, the possibility of associating priority to AltaRica events is not currently present in the definition of SAN. On the other side, the mechanism expressed by cases in SAN is not present in AltaRica, and it is not straightforward to represent shared data modifiable by several components as modeled by shared places in SAN. In addition, the expressiveness of AltaRica language is limited compared to the C/C++ language that is used to specify the gates in SAN.

We analyze in the following, how models transformation can be achieved from each of the two formalisms to the other one whilst respecting as much as possible the model structure. The objective is to show that the features of the formalism, whether it is AltaRica or SAN, that will be used to experiment the modeling approach proposed in chapter II, can be equivalently found in the other formalism, and therefore the quantitative analysis can be performed using SAN, which is supported by an available and accessible quantitative analysis tool. Furthermore, as shown in Figure III.5, based on the transformation, equivalent models can be used to compare evaluation results derived from the Möbius tool that supports SAN, and the EPOCH tool that is being developed to process the AltaRica model, in order to validate the latter tool.

The Möbius tool is an academic tool. Ultimately, it will be interesting to confirm the results obtained using the Möbius tool by analyses using other AltaRica supporting tools like those presented in [Signoret et al. 2004] to which we don't currently have access.

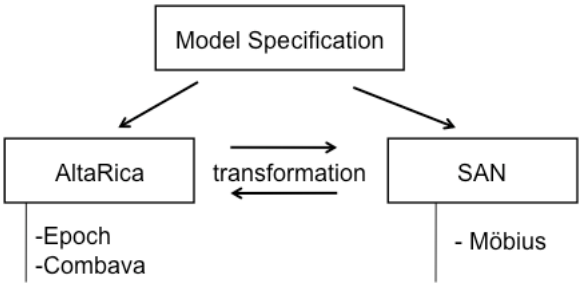


Figure III.5 Model transformation from AltaRica to SAN and vice versa

We consider firstly model transformation from AltaRica to SAN, then from SAN to AltaRica.

III.3 From AltaRica to SAN

The transformation of the basic elements such as state variables and transitions from AltaRica to SAN is straightforward since they are part of both formalisms. An AltaRica state variable can be represented by a place, and guarded transitions by activities and input/output gates. The places can be defined with the same data as in AltaRica since it is possible to use user-defined types while modeling with the Möbius tool. A correspondence can be made between AltaRica discrete data types and integer representation for the place marking in SAN. The initialization of the state variables is merely the definition of the initial marking with the values stated in the AltaRica model. The basic elements, which cannot be directly represented, are the flow variables.

Output variables are set by assertions stating an invariable relationship between the variable and a set of state and other input variables: $outv = r\ (sv_1, \ldots sv_k, inv_1, \ldots inv_l)$. That is, if the value of sv_i or inv_i changes, the value of *outv* must automatically be updated to keep the relationship valid. The variable *outv* can be represented by a place, which must be updated in the gates of the transitions affecting each of the variable sv_i and inv_j. However, this does not keep the separation that AltaRica makes between the component behavior description and the calculation of output variables values. Instantaneous activities can also be used to represent the invariable relationship. The instantaneous activity will fire each time the relationship becomes invalid and the outcome will consist in updating the variable *outv*.

The modularity of the model can be taken into account considering the same hierarchical structure and using the Replicate and Join operators. This is to remain close to the AltaRica model. In practice, it may be more useful to compact several components sub-models into a single sub-model as the systematic transformation may generate a considerable number of small sub-models. AltaRica also integrates the notion of events synchronization between several nodes. A synchronization is a unification of several transitions. It can be represented by a unique activity combining the specifications of the transitions and using the variables (represented as shared places) involved.

AltaRica allows the definition of priority between transitions. The notion of priority is not formally defined in SAN. It is also worth noting that only AltaRica with events distribution laws specified can be transformed into SAN model; the SAN definition

requires the definition of the quantitative attributes of the activities. However the transformation can still be applied if the model exploitation doesn't require the quantitative aspect; e.g., to support qualitative analyses based on model checking as carried out with the SANBuilder tool.

The guidelines for applying the transformation are concretely given, together with examples, in the following subsections. The transformation of an AltaRica node without sub-node is considered firstly. Then, we present the transformation of composed nodes.

III.3.1 Basic features transformation

Table III.2 gives the correspondences for transforming the basic AltaRica features into SAN.

AltaRica element	SAN correspondence	Comments
State variable	Place	A correspondence is made between discrete data type and integer marking; extended places are used for the other variable type.
Flow (in, out) variable ($v \in F^{in} \cup F^{out}$)	Shared Place	
Event	SAN activity	All the activities are timed except those corresponding to events that have Dirac(0) as distribution law.
Transition specification (guard *grd*, event *evt*, post-condition *post*)	Input gate, with predicate *grd* and a function specified by the expression of the post-condition *post*, associated to the activity corresponding to event *evt*.	An output gate can be used to express the post-condition but the function of the input gate is sufficient
Assertion	Input gate associated to an instantaneous activity	
Initialization	Place marking	
Sub node	SAN sub model	

Table III.2 AltaRica to SAN correspondence

It is worth paying an attention to assertions as they are specific to AltaRica. An assertion stated in the form of $outv = r(sv_1, \ldots sv_k, inv_1, \ldots inv_l)$ is represented by an input gate and an instantaneous activity such that:

- the predicate of the input gate holds when the assertion becomes non-valid, so as to enable the update of the output variable value *outv*:

  predicate:

  $$outv\text{->}Mark() \mathrel{!}= r(sv_1\text{->}Mark(), \ldots sv_k\text{->}Mark(), inv_1\text{->}Mark(), \ldots inv_l\text{->}Mark())$$

- the function of the input gate reassigns the correct value to outv ->Mark():

  outv->Mark() = r(sv_1->Mark(), … sv_k->Mark(), inv_1->Mark(), … inv_l->Mark());

  the instantaneous activity is used to trigger the update of the output flow outv.

Figure III.6 shows an example of a basic component transformation from AltaRica to SAN, following the rules given previously.

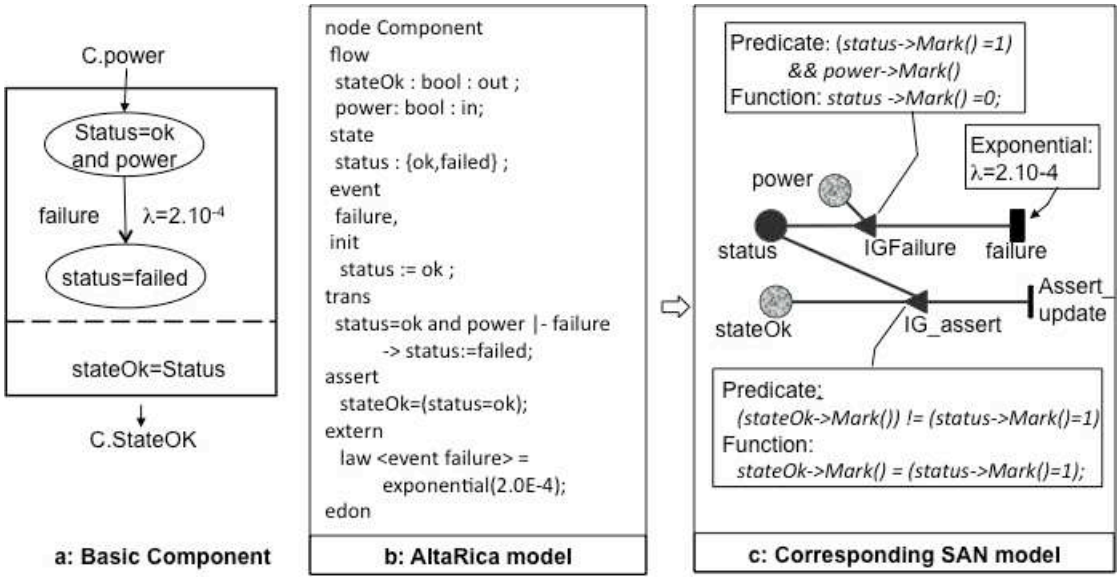


Figure III.6 Basic transformation from AltaRica to SAN

The models represent a component that, if powered, can be subject to a failure event. The current state information of the component is produced as output. Each of the flow and state variables is transformed into a place, and the event *failure* has given rise to an activity. The places *power* and *stateOk* are to be shared with the model of the components that set or use the value of their corresponding AltaRica variable. The particular data type consisting in the set of values *{failed, ok}* that is used to define the state variable *status* is replaced by the integer values {0,1} corresponding to the marking of this place. The input gate *IG_assert* and its corresponding instantaneous activity *Assert_update* are used to represent the assertion as described previously.

III.3.2 Formal definition of the transformation

The transformation can be defined formally using the formal definitions of SAN and AltaRica presented in section III.1. Each AltaRica node described by the 9-tuple (D, S, F^{in}, F^{out}, dom, E, δ, σ, I^S) is transformed into an activity network AN= (P, A, IG , O, γ , τ , ι, o) such that:

- $S \cup F^{in} \cup F^{out} \equiv P$, each AltaRica variable corresponds to a place in the SAN.
- $E \equiv A^E \subseteq A$; each event in AltaRica is transformed into an activity in SAN. A^E is the set of SAN activities corresponding to events in AltaRica. $A^E = A$ if no extra instantaneous activity is defined in the transformation of the assertions.

In the followings, the variables of $S \cup F^{in} \cup F^{out}$ are assimilated to their corresponding places and the events to their corresponding activity. The assignments of values to the variables of $S \cup F^{in} \cup F^{out}$ are also assimilated to their corresponding SAN markings.

- δ: $VAL(S) \times VAL(F^{in}) \times E \rightarrow VAL(S)$ defines input gates that must be associated to the activities of A^E. For each event $evt \in E$, the restriction δ^{evt} of δ on $VAL(S) \times VAL(F^{in}) \times \{evt\}$ defines its corresponding transition. δ^{evt} is transformed using an input gate $ig^{evt} = (G^{evt}, e^{evt}, f^{evt})$ such that:
  - $G^{evt} \equiv S \cup F^{in}$;
  - $val^S \in VAL(S)$, $val^{in} \in VAL(F^{in})$, the concatenation $val^S \bullet val^{in}$ corresponds to a marking of G^{evt} and:
    - *if* $\delta(val^S, val^{in}, evt)$ is defined and $\delta(val^S, val^{in}, evt) \neq val^S$ then $e^{evt}(val^S \bullet val^{in}) = 1$ and $f^{evt}(val^S \bullet val^{in}) = \delta(val^S, val^{in}, evt) \bullet val^{in}$,
    - *otherwise* $e^{evt}(val^S \bullet val^{in}) = 0$;
- σ: $VAL(S) \times VAL(F^{in}) \rightarrow VAL(F^{out})$ defines an input gate $ig^{\sigma} = (P, e^{\sigma}, f^{\sigma})$ associated to an instantaneous activity that triggers the update of the output variables. e^{σ} and f^{σ} are defined as follows:
  - $val^S \in VAL(S)$, $val^{in} \in VAL(F^{in})$, $val^{out} \in VAL(F^{out})$,
    $e^{\sigma}(val^S \bullet val^{in} \bullet val^{out}) = 1$ *if* $\sigma(val^S, val^{in}) \neq val^{out}$;
    $e^{\sigma}(val^S \bullet val^{in} \bullet val^{out}) = 0$ *otherwise*;

    The predicate of the input gate ig^{σ} holds so as to enable the update when the assertion no longer holds.
  - $f^{\sigma}(val^S \bullet val^{in} \bullet val^{out}) = val^S \bullet val^{in} \bullet \sigma(val^S, val^{in})$;

The resulting activity network AN is obtained such that:

- $IG = \{ig^{evt}, evt \in E\} \cup \{ig^{\sigma}\}$.
- $O = \emptyset$; output gates are not needed; the functions of input gates are sufficient to express changes after transitions firings.
- $\gamma(act) = 1$, $\forall$ act $\in A$; there is no multiple choices at the completion of an activity. Cases are not needed since the notion is not present in AltaRica and output gates are not used in the transformation.
- τ is defined based on the distribution property that is not part of the definition of AltaRica nodes as modes automata. Events distributions are considered as an external property that should be managed by quantitative evaluation tools.
- The mapping ι of the input gates to the activities is straightforward since each input gates is created based on an activity.

A resulting SAN = (AN, μ_0, C, F, H) is obtained considering $\mu_0{}^s \equiv I^s$, which defines the initial marking of the places corresponding to the AltaRica state variables. The marking of the flow variables are deduced based on their relationship with the state variables. F is defined by associating the probability distribution functions that are defined as external parameters in AltaRica. There is no need to define C since the transformation does not

integrate any multiple choices. Reactivation scenarios are not specified for the events in AltaRica. Additional information is needed to define H. A default reactivation policy consisting in not reactivating the activities can be considered.

The transformation rules just presented deal with the content of a basic node that is not composed of sub-nodes. The transformation of a global model must cope with hierarchical compositions that integrate dependencies between basic components.

III.3.3 Dealing with composed nodes

An AltaRica node may be made of other sub-nodes with dependencies between them. A composed node uses instances of other nodes as sub-nodes and the dependencies are expressed based on assertions that connect the output variables with the input variables. We distinguish simple connections and complex connections.

Figure III.7-a shows a simple connection of two AltaRica sub-nodes SN1 and SN2 based on the output variable *outV* of SN1 and the input variable *inV* of SN2.

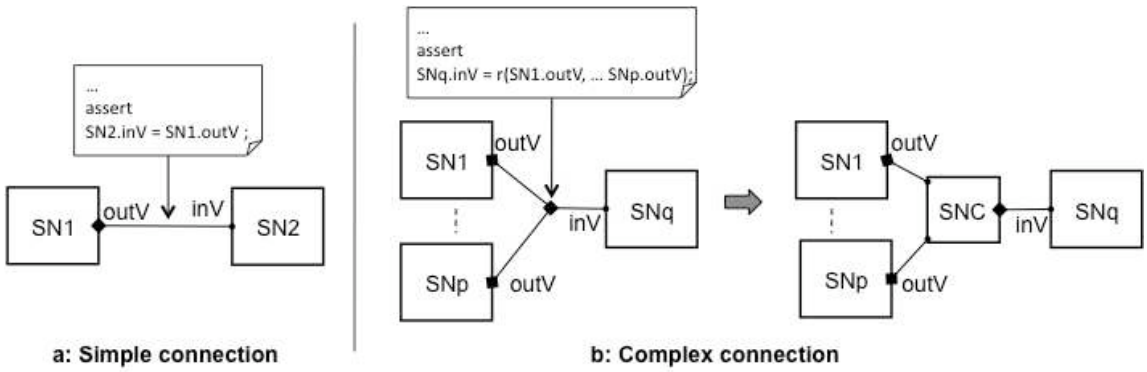


Figure III.7 AltaRica nodes connection

The mechanism merely corresponds to SAN sub-models composition based on places sharing. The places corresponding to *outV* and *inV* must only be joined together into a shared place during the composition using the SAN's Join operator. The only difference is that SN1 sets the values of the variable *outV*, which becomes an input for SN2, whereas both sub-models can, but not necessarily, modify the shared variable in case of SAN composition. Therefore, the SAN model corresponding to a node, composed of sub-nodes connected based on simple connections, is obtained by simply using the Join operator and specifying the connected variables as shared places.

In case of a composition based on a complex relationship as shown in Figure III.7-b, an additional sub-model SNC is considered so as to break the complexity into simple compositions. SNC takes as input, all the variables $SNj_{j=1..p}.outV$ that should be combined to define the input of SNq. The content of the corresponding SAN sub-model consists in the transformation of the assertion that represents the complex connection.

All complex connections transformations can be gathered in the same sub-model, which becomes the composer of the other sub-models. An AltaRica node N containing n sub-nodes with complex connections between the sub-nodes is thus transformed into n+1 SAN

sub-models, whose composition forms the global SAN model corresponding to N. N = ⟨SN1, . . . SNn, SNC⟩, SNC represents the composition assertions.

As an example, let us consider the component of Figure III.6. Assuming that its input power is provided by two different suppliers, we obtain an AltaRica model integrating three sub-nodes. We consider that the power suppliers can fail to deliver power at a given rate. The supplier 1 is described in AltaRica as follows.

```
Node Supplier1
  flow
    power:bool:out;
  state
    status:{loss, ok};
  event
    loss;
  trans
    (status = ok) |- loss -> status := loss;
  assert
    power = (status = ok);
  init
    status := ok;
  extern
    law <event loss> = exponential(2.0E-6);
edon
```

The description of the other supplier is the same except that the loss event occurrence rate is considered equal to 2.0E-4. The composition of these sub-nodes into a global model is given as follows.

```
node main
  sub
    s2:Supplier1;
    s1:Supplier2;
    C: Component;
  assert
    C.power = (s1.power or s2.power);
edon
```

The SAN model corresponding to this model is presented in Figure III.8. Each sub-node is transformed into a sub-model. As the composition relationship between the sub-nodes is not a simple connection relationship, an additional sub-model *Composer* is created to express the composition relationship between them. Using this intermediate sub-model, the composition can be done based on simple connections. The places *power* of the sub-models *supplier1, supplier2* and *Component* are specified respectively to be the same with the places s1_power, s2_power, C_power of the sub-models composer.

The transformation is done in such a way to respect the AltaRica model's composition. In practice, it may be more tractable to compact some of the sub-models into a single sub-model, as their contents are not so complex. This depends on the modeler appreciation.

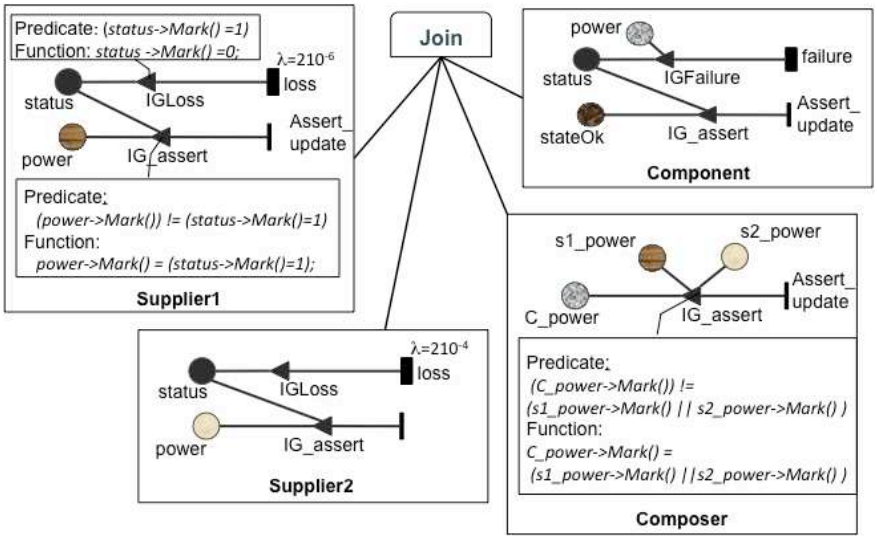


Figure III.8 SAN model corresponding to the AltaRica composed model

III.4 From SAN to AltaRica

The transformation from SAN to AltaRica can also be made by transforming each sub-model to an AltaRica sub-node. The composition is done taking into account the SAN model structure. It is worth mentioning that it is possible to define a parameter n specifying a number for replicating a SAN sub-model while using the Möbius tool. Such possibility is not integrated in AltaRica.

For the transformation of a sub-model's content, each place can be represented by either a state variable or an input variable. The place corresponds to a state variable if its marking can be modified by an activity in the model, otherwise, it corresponds to an input variable. Additional output variables can be defined for shared places. An activity can be declared as an AltaRica event to which must be associated a distribution law. The corresponding transition in AltaRica will have as guard, the conjunction of all the predicates of the activity's related input gates, and as post condition, the composition of the activity's input and output gates functions. In SAN, activities are explicitly specified as instantaneous or timed. Instantaneous transitions are specified in AltaRica by associating *Dirac(0)* as distribution.

For activities with multiple cases, the notion of uncertainty in the action to choose after the occurrence of an event is not defined in AltaRica. However, from a technical viewpoint, an activity *act* to which *n (n>1)* cases are associated can be transformed to *n* activities *act_1, … act_n*, each activity corresponding to a case of *act*. Their distribution functions correspond to the products of the case probabilities and the distribution function of the activity *act*: $F_{act\_i}(\mu, \cdot) = C_{act}(\mu, i) * F_{act}(\mu, \cdot)$, $i=1, \ldots n$. Given this possibility of removing multiple cases in the model, the activities of the SAN model considered for the transformation are assumed to be without multiple cases.

Shared places between the sub-models can be either represented using flow variables and assertions, or events synchronization. When the shared place's marking is subject to modifications in only one of the sub-models (which correspond to information

communication to the other sub-models), the place is represented as a state variable and an output variable in the corresponding sub-node, and as input variable in the sub-nodes corresponding to the other sub-models. If the shared place can be modified by several of the sub-models, it becomes complicated to use input and output variables, due to the risk of occurrences of loops in the model. Therefore, one must consider using events synchronization. The place is represented by a state variable in a dedicated sub-node, and auxiliary transitions, synchronized with the transitions that may affect it, are used to set its value. The value of the variable is transmitted as input to the sub-nodes. It is worth noting that this is still fairly complicated since one has to identify all the transitions that may modify the shared place.

Finally, it is worth mentioning that extended places representing arrays of variables can be defined while using the Möbius tool to build SAN models. Variables of type array are not defined in AltaRica, and thus, are not considered for the transformation. Furthermore, the possible control flow statements that can be taken into account in AltaRica are the "if-then-else" and "switch/case" statements. Therefore, SAN models using functions that include statements such as the "while" or "for" statements of the C/C++ language are not taken into account.

The transformation rules are presented in the following. We consider firstly the transformation of the basic features, then the features involved in composed models.

III.4.1 Basic features transformation

Table III.3 gives the correspondence for transforming SAN features into AltaRica. The correspondence deals with the content of non-composed models.

SAN Feature	AltaRica correspondence	Comments
Place	**State variable** if the place's marking can be changed by an activity in the model, **Input variable** otherwise	For composed models, shared places require the consideration of all the sub-models involved, and output variables must be defined for the composition.
Place marking	Value assignment to the variable	
Activity	Event	Instantaneous activities are represented by associating Dirac(0) as distribution law of the event.
Input gates (Predicate Λpre, function f^t), and output gate (function g^t) associated to activity act	Transition specification (guard Λpre, event evt, post-condition $g^t \circ f^t$)	evt is the event corresponding to act, Λpre is the conjunction of the input gates predicates, f^t and g^t are respectively the compositions of all the input gates and output gates functions.
Activity distribution	Extern property law	
Sub model	Sub node	

Table III.3 SAN to AltaRica correspondence

Figure III.9 illustrates the application of the transformation considering a basic component that can be subject to a failure event and maintenance activities. Figure III.9-a presents the SAN model and Figure III.9-b the corresponding AltaRica model.

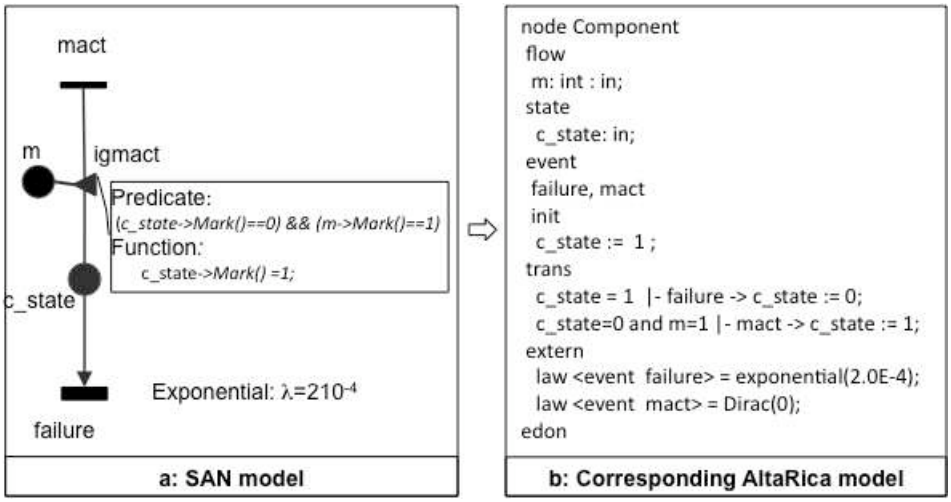


Figure III.9 Basic transformation from SAN to AltaRica

Place *c_state* marking is affected by the completion of the failure and maintenance activities. Therefore, it is transformed into a state variable. As the marking of place *m* is not affected by the activities, it represents an input variable. Integer values are used to define the variables type since they are not extended places and their makings are of integer type. Actually, the marking of the places are bounded by 0 and 1. An event is created for each of the two activities, and their corresponding transitions are specified using their associated input gate predicates and functions. The *failure* event predicate consists in testing the presence of a token in place *c_state*, and it resets the place to zero at completion. The input gate *igmact* specifies the transition corresponding to the activity *mact*. The distributions of the activities are taken into account in the *extern* section of the AltaRica node.

III.4.2 Formal definition of the transformation

A SAN is formally defined as the tuple (AN, μ_0, C, F, H) with AN = (P, A, IG, O, γ, τ, ι, o) and an AltaRica node is defined as the tuple (D, S, F^{in}, F^{out}, dom, E, δ, σ, I^S). The function τ that distinguishes timed activities from instantaneous activities, the distribution assignment function F, and the reactivation function H are not considered since these aspects are not included in the AltaRica formal definition. They have to be managed by the assessment tool, which must integrate them as additional parameters.

The transformation is such that:

- D is made of i) the integer set, to which belong the nominal places markings, and ii) the user-defined types that are used to create extended places.
- $S \cup F^{in} \equiv P$, $S \equiv P \setminus P^{in}$, $F^{in} \equiv P^{in}$, P^{in} is the set of places whose markings are not affected by any activity in the SAN. Each place has a corresponding variable in AltaRica.

- dom is implicitly defined, based on the association of the place marking type to the corresponding variable as its domain.
- $VAL(S) \times VAL(F^{in}) \equiv M_P$, the set of the possible markings of P corresponds to the set of the possible assignment of values to the variables of $S \cup F^{in}$. A marking μ_P of P defines an assignment of values to the variables of $S \cup F^{in}$, $\mu_P \equiv val^S \bullet val^{in}$.
- $E \equiv A$, each activity corresponds to an AltaRica event.
- δ: $VAL(S) \times VAL(F^{in}) \times E \rightarrow VAL(S)$ is defined based on the activities, the input and output gates mapping functions ι, o. $evt \in E$, $evt \Leftrightarrow act \in A$, $\delta(.,.,evt)$ is the composition of all the functions of the input gates and output gates associated to act, provided that all the predicates of the associated input gates hold.

  PRE= $\{e \mid \exists SP \subset P, \exists f: M_{SP} \rightarrow M_{SP}$, and $ig=(SP, e, f) \in \iota^{-1}(act)\}$, the predicates of the input gates associated to act;
  IF= $\{ f \mid \exists SP \subset P, \exists e: M_{SP} \rightarrow \{0,1\}$, and $ig=(SP, e, f) \in \iota^{-1}(act)\}$, the functions of the input gates associated to act;
  OF=$\{ f \mid \exists SP \subset P$, and $og=(SP, f) \in o^{-1}(act,1)\}$, the functions of the output gates associated to the unique case of act.
  The functions of these sets are defined on the markings of subsets of P. We must extend them to the markings M_P of P in order to compose them.
  Every marking μ_P of P defines a marking μ_{SP} on any subset SP of P. Therefore,
  $SP \subset P$, $e: M_{SP} \rightarrow \{0,1\}$: $ext(e) : \mu_P \rightarrow e(\mu_{SP})$, the extension of the predicate e holds in $\mu_P \Leftrightarrow$ e hold in μ_{SP}.
  Also, the change of markings, by a function f on SP, represents a change of marking on P for which the markings remain identical on the complement (P \ SP) of SP in P.
  $SP \subset P$, $f: M_{SP} \rightarrow M_{SP}$, if $f(\mu_{SP})$ is defined, $ext(f) : \begin{smallmatrix} M_P \rightarrow M_P \\ \mu_P \rightarrow f(\mu_{SP}) \bullet \mu_{P \setminus SP} \end{smallmatrix}$.
  Let consider $e^{tt}=\prod_{e \in PRE} ext(e)$, the conjunction of the predicates of the input gates, and let $f^{tt}=(\circ_{f \in OF} ext(f)) \circ (\circ_{f \in IF} ext(f))$, the composition of the functions of the input gate and output gate associated to act;
  if $val^S \in VAL(S)$ and $val^{in} \in VAL(F^{in})$ correspond respectively to values assignments to the variables of S and F^{in}, $val^S \bullet val^{in}$ corresponds to a marking of P,

  $\delta(val^S, val^{in}, evt) = f^{tt}(val^S \bullet val^{in})$ *if* $e^{tt}(val^S \bullet val^{in})=1$,
  otherwise $\delta(val^S, val^{in}, evt)$ is not defined (δ is a partial function).
- I^s is defined by μ_0, which defines the initial marking of P and, therefore, an assignment of values to S.

σ is defined by the sub-models composition mechanisms based on shared places, since the data exchange mechanism used for SAN is based on shared places that are not formally specified in the SAN definition. The shared places also define the output variables (F^{out}) necessary for the sub-nodes interconnections.

III.4.3 Dealing with composed models and shared places

The objective is to transform a model composed of n sub-models into an AltaRica model composed of sub-nodes corresponding to these sub-models. The composition of SAN sub-models is based on shared places that are the unifications of groups of places belonging to

different sub-models. Let us consider a place *Psh* shared by *k* sub-models $SM^{w}_{i,\ i=1,\ \ldots\ k}$, which can modify its marking, and *l* sub-models $SM^{r}_{j,\ j=1,\ \ldots\ l}$ that only use it as input. According to Table III.3, the place is represented as a state variable in the transformation of $SM^{w}_{i,\ i=1,\ \ldots\ k}$, and as an input variable in the transformation of $SM^{r}_{j,\ j=1,\ \ldots\ l}$. One has to connect these representations of *Psh*. In the composition of $SM^{w}_{i,\ i=1,\ \ldots\ k}$ and $SM^{r}_{j,\ j=1,\ \ldots\ l}$, if:

- *k=0*, the marking of *Psh* is used as an input value in all the sub-models, then its corresponding variable represents just an input variable in the global node that integrates the AltaRica sub-nodes corresponding to the sub-models.
- *k=1*, the marking of *Psh* is controlled by SM^{w}_{1} and it is used as input in the other sub-models. Therefore, *Psh* is represented in the AltaRica sub-node corresponding to SM^{w}_{1} by a state variable. A corresponding output variable *Psh_out* is created to make the connection with the other sub-nodes in which *Psh* corresponds to an input variable.
- *k>1*, *Psh* represents a resource that is affected by activities in several sub-models. The changes affecting its corresponding variable are managed in a dedicated sub-node SN0. The state variable corresponding to *Psh* is created in the sub-node SN0, and it is replaced by an input variable $Psh\_in_{i,\ i=1,\ \ldots\ k}$ in the sub-nodes corresponding to the sub-models $SM^{w}_{i,\ i=1,\ \ldots\ k}$. An output variable Psh_out_{0} is also created in the sub-node SN0 to make the connection with the other sub-nodes. To take into account the marking changes that affect *Psh* following activities completion in the sub-models $SM^{w}_{i,\ i=1,\ \ldots\ k}$, for each activity *act* that can affect *Psh*, an auxiliary event *auxi_evt* is created in the sub-node SN0, which is synchronized with the event *evt* corresponding to *act*, so as to update *Psh* when *evt* fires. For that, an output variable $Psh\_chg\_out^{evt}$ is created in the sub-node of *evt*, to report the new value of *Psh* to SN0.

To illustrate the transformation, let us consider the hypothetical SAN model shown in Figure III.10. It is composed of three sub-models.

The sub-model1 describes a resource-consuming scenario in which the amount represented by *P1* is decreased from time to time by activity *act1a*. *P1* is reset by activity *act1b* based on the supplies represented by place *Psh*, respecting a reference maximum capacity represented by place *Ref*. *Psh* is shared by the three sub-models.

The quantity represented by *Psh* is set in sub-model 2. Activity *act2b* of sub-model 2 increases the quantity represented by *Psh* while activity *act1b* of sub-model1 absorbs the quantity by using it to increase the quantity represented by *P1*. Activity *act2b* of sub-model2 can fire only if *P2* is marked, representing the availability of the component necessary for the delivery of the supplies represented by *Psh*. P2 can be affected by activity *act2a* that nullifies its marking. When activity *act2b* fires, the marking of *Psh* is incremented by 1 if it was not null; otherwise the increment depends on the duration since the marking has become null. This duration is managed in sub-model 3, in which the activity *act3a* is enabled when the marking of *Psh* is null.

The activity *act3a* of sub-model 3 fires periodically until *Psh* is set again, representing the delivery of alerts on the shortage of supplies. Place *Pal* counts the number of times *act3* is

triggered and is used in sub-model 2 to set *Psh* as follows: *Psh->Mark() = Psh->Mark() + 1 + Pal->Mark()*.

The three sub-models are composed based on place *Psh* which is shared by all of them, and place *Pal*, which is shared by sub-model 2 and sub-model 3.

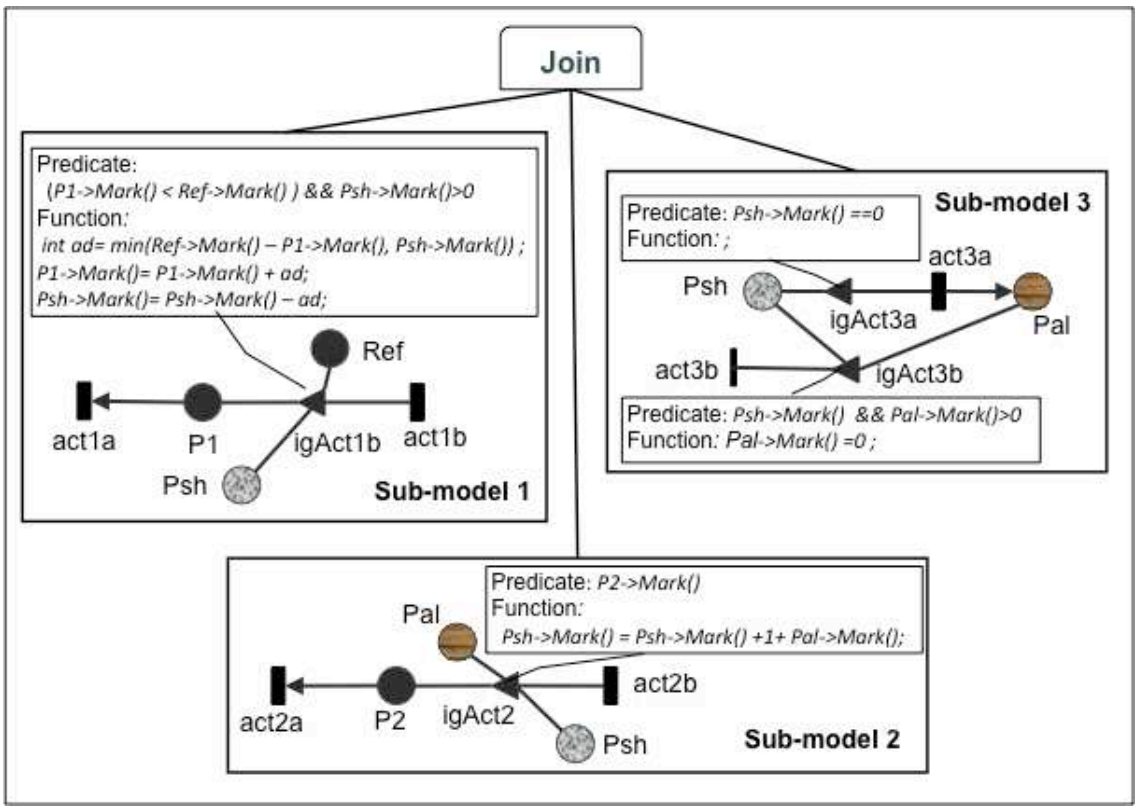


Figure III.10 Composed SAN model

Considering the places in the sub-models, we have for:

Sub model 1: *P1* whose marking can be modified and therefore corresponds to a state variable; *Ref* whose marking is not affected by any activity in the sub-model and then corresponds to an input variable; and *Psh* that is shared and which can be affected by activities in sub-model1 and sub-model 2 (k=2). Thus, the state variable corresponding to *Psh* is to be managed apart in a sub-node SN0. A corresponding input variable must also be declared in the sub-node to get its values from SN0. The change of *Psh* marking that results from the completion of activity act1b must be represented as an output variable that is to be used to set *Psh* in node SN0.

Sub model 2: *P2* whose marking can be modified and therefore corresponds to a state variable; *Pal* whose marking is not affected by any activity in the sub-model and then corresponds to an input variable; and *Psh* that is to be managed as in the case of sub-model1.

Sub model 3: *Psh* whose marking is not affected by any activity in the sub-model and then corresponds to an input variable, and *Pal,* which is shared with sub-model 2, but not affected by any activity in sub-model 2 (k=1).

The corresponding AltaRica sub-nodes are given in Figure III.11, together with the sub-node 0 dedicated to the management of the shared place *Psh*. Apart from the transformation related to the shared places *Pal* and *Psh*, the content of each sub-model is transformed considering the correspondences presented in Table III.3.

For place *Pal*, an output variable *Pal_out* is created in the sub-node 3 so as to transmit its value to the sub-node 2 in which it is defined as an input variable. For place *Psh,* the sub-node 0 is created to centralize its changes. It is represented as a state variable only in this sub-node. Its value is transmitted to the other sub-nodes as input. The changes that affect its value in the sub-nodes 1 and 2 are reported to the sub-node 0 based on output variables *(Psh_chg_out)*.

Sub-node 1

```
Node SN1
 flow
  Psh_in : int : in ;
  Psh_chg_out : int : out ;
  Ref : int : in ;
 state
  P1 : int ;
 event
  act1a, act1b ;
 init
   P1 := 1 ;
trans
   P1>0 |-act1a -> P1:=P1-1;
  (P1 < Ref) and Psh_in>0  |-act1b ->
    P1:=P1+min(Ref-P1,Psh_in);
Assert
 Psh_chg_out=Psh_in-(min(Ref-P1,Psh_in));
extern
 law <event act1a> = exponential(0.2) ;
edon
```

Sub-node 2

```
Node SN2
 flow
  Psh_in : int : in ;
  Psh_chg_out : int : out ;
  Pal : int : in ;
 state
  P2 : int ;
 event
  act2a, act2b ;
 init
   P2 := 1 ;
trans
   P2=1 |-act2a -> P2:=0;
   P2=1 |- act2b -> ;
assert
 Psh_chg_out=Psh_in+1+Pal;
extern
 law <event act2a>=
    exponential(2.0E-4);
edon
```

Sub-node 3

```
Node SM3
 flow
  Psh : int : in ;
  Pal_out : int : out ;
 state
  Pal : int ;
 event
  act3a, act3b ;
 init
   Pal := 0 ;
trans
  Psh=0 |-act3a -> Pal:=Pal+1;
 (Psh > 0) and Pal>0  |-act3b -> Pal:=0;
assert
  Pal_out=Pal;
 extern
  law <event act3a> = Dirac(4.0) ;
  law <event act3b> = Dirac(0.0) ;
edon
```

Sub-node 0

```
Node SM0
 flow
  Psh_out : int : out ;
  Psh_chg_in2 : int : in ;
  Psh_chg_in1 : int : in ;
state
  Psh : int ;
 event
  act1b_auxi, act2b_auxi ;
 init
   Psh := 0 ;
trans
 Psh>=0 |-act1b_auxi -> Psh:=Psh_chg_in1;
 Psh>=0 |-act2b_auxi -> Psh:=Psh_chg_in2;
assert
Psh_out=Psh;
edon
```

Figure III.11 AltaRica model corresponding to the composed SAN model

The sub-node 0 contains in addition to the state variable *Psh*, an output variable *Psh_out* that is used to transmit the values of *Psh* to the other sub-nodes. The events *act1b_auxi* and *act2b_auxi* are respectively used to set the change that should affect *Psh* when the events *act1b* of sub-node 1 and *act2b* of sub-node 2 fire. The input variables *Psh_chg_in1* and *Psh_chg_in2* are respectively used to get from the sub-nodes 1 and 2, the new value that must be given to *Psh* when *act1b* and *act2b* fire.

In the sub-nodes 1 and 2, *Psh* is replaced by the input variable *Psh_in* in the instructions that require its values. Values assignments to *Psh,* in the specification of the transitions corresponding to the events *act1b* and *act2b,* are transformed into assertions that define the output variables *Psh_chg_out,* which are used to make the connection with the input variables *Psh_chg_in1* and *Psh_chg_in2* of sub-node 0.

The main node describing their composition is presented as follows:

```
node main
  event
    SincAct1b,
    SincAct2b;
  sub
    sn0:SN0;
    sn1:SN1;
    sn2:SN2;
    sn3:SN3;
  assert
    sn0.Psh_chg_in2 = sn2.Psh_chg_out,
    sn0.Psh_chg_in1 = sn1.Psh_chg_out,
    sn1.Psh_in = sn0.Psh_out,
    sn2.Psh_in = sn0.Psh_out,
    sn2.Pal = sn3.Pal_out,
    sn3.Psh = sn0.Psh_out;
  sync
    <SincAct1b , sn0.act1b_auxi , sn1.act1b>,
    <SincAct2b , sn0.act2b_auxi , sn2.act2b>;
  extern
    law <event SincAct1b> = Dirac(12.0);
    law <event SincAct2b> = exponential(0.15);
edon
```

The events *act1b_auxi* and *act2b_auxi* of sub-node 0 (sn0) are respectively synchronized with the events *act1b* of sub-node 1 (sn1) and *act2b* of sub-node 2 (sn2) so as to update *Psh* exactly at the firing time of the events *act1b* and *act2b*.

III.5 Conclusion

In this chapter, we have presented a comparative overview of Stochastic Activities Network (SAN) and AltaRica, two formalisms used in dependability modeling. Both formalisms are based on states and transitions firing mechanisms. The SAN formalism was developed in the early eighties and still provides strong features to model todays' systems performances. The AltaRica formalism development started in the mid-nineties and was mainly aimed at systems qualitative behavior analyses. The AltaRica formalism, issued as a language, has been upgraded to support stochastic events modeling. Its main

characteristics are the flow propagation mechanism and the intrinsic modularity support. It supports events synchronization that consists in allowing several transitions to fire jointly. The language is currently used to develop models to which can be associated graphical representations so as to have a presentation close to the system architecture, thanks to its supporting tool Cecilia Ocas. The SAN formalism, which is a direct extension of Petri Nets, is almost completely based on graphical representation. Its main characteristics are the intrinsic stochastic modeling and its quantitative analyses oriented aspects; the dynamics of the model is completely specified probabilistically. It supports the definition of cases, which represent probabilistic choices of actions at the firing of an activity. It is also possible to specify places belonging to different sub-models as the same and unique place.

Despite the specificities of each one of them, models can be transformed between each other. We have considered transformations that preserve, as much as possible, the model structure. Both formalisms incorporate the basic features that are states, and transitions, with firing enablement determined by predicates. Based on these basic features, one can transform the content of basic models that serve as sub-models in composed models. However, the distribution functions, which are integrally part of the activities specification in SAN, are taken into account in AltaRica as external parameters of the model, and thus cannot be state dependent. The two formalisms also manage differently model composition. Sub-models are composed in SAN using the Join operator based on shared places. AltaRica uses assertions, based on input and output variables, and events synchronizations. Nonetheless, the composition features of each one give place to the composition features of the other one thanks to some adjustments.

Considering the transformation from AltaRica to SAN, respecting the AltaRica model structure, the SAN model obtained may be composed of very small sub-models due to the fact that every system component leads to the creation of a sub-node in the AltaRica model. It may be more interesting to aggregate some of them into sub-models corresponding to intermediate levels in the AltaRica model, in order to reduce the number of sub-models obtained.

The SAN formalism is used in the next chapter to develop case studies experimenting the modeling approach presented in the previous chapter. The implementation using AltaRica is given in annex.

IV Case Studies

This chapter presents two case studies to illustrate the implementation of the modeling approach presented in chapter II, using the SAN formalism presented in chapter III. The first case study concerns the A340 rudder control subsystem [Bernard et al. 2007]. A corresponding model is presented in [Tiassou et al. 2011b] which highlights the different categories of information to be included in the models. The second case study concerns the A320 electric supply subsystem, which is characterized by several reconfiguration mechanisms. Examples of evaluation results are presented in each case study. Also, based on the model corresponding to the rudder control subsystem, we investigate the need of updating the model during operation, especially the impact of changes on the assessments. We consider the major changes identified in chapter II, and proceed to hypothetical experimentation of their impact on the mission reliability.

The chapter is presented as follows. The necessary information for modeling the rudder control system is given firstly; we present the subsystem description together with its related operational requirements, and an informal description aimed at giving detailed information for the description with a modeling formalism. We then present the SAN corresponding model. We set the parameters of the model to give examples of evaluation results. The examples of evaluation results are extended by the study of the valuable role of re-assessing the operational dependability after major changes during missions' achievement. Finally, the case study corresponding to the electric supply subsystem is presented.

IV.1 Modeling the Rudder Control Subsystem

The rudder is a movable surface located at the rear of the aircraft. It is used to control the movement of the aircraft around its vertical axis. The following describes the subsystem that is used to command its movement.

IV.1.1 Presentation of the rudder control subsystem

The subsystem is, as illustrated in Figure IV.1, composed of three primary computers (P1, P2, P3), a secondary computer S1, three servo-controls (ServoCtrl_G, ServoCtrl_B and ServoCtrl_Y), a backup control module (BCM) and two backup power supplies (BPS_B and BPS_Y).

The computers are connected to the servo-controls, which move the rudder. S1 and P1 are connected to the servo-control ServoCtrl_G, P2 is connected to ServoCtrl_B, and P3 is connected to ServoCtrl_Y. The connection between a computer and a servo-control form a control line that can act on the rudder. Thus, four control lines can be distinguished:

- P1 control line (PL1): formed by the connection between P1 and ServoCtrl_G,
- P2 control line (PL2): formed by the connection between P2 and ServoCtrl_B,
- P3 control line (PL3): formed by the connection between P3 and ServoCtrl_Y,

- S1 control line (SL): formed by the connection between S1 and ServoCtrl_G.

We have also a backup control line (BCL), which is based on BCM, BPS_B, BPS_Y, ServoCtrl_Y and ServoCtrl_B.

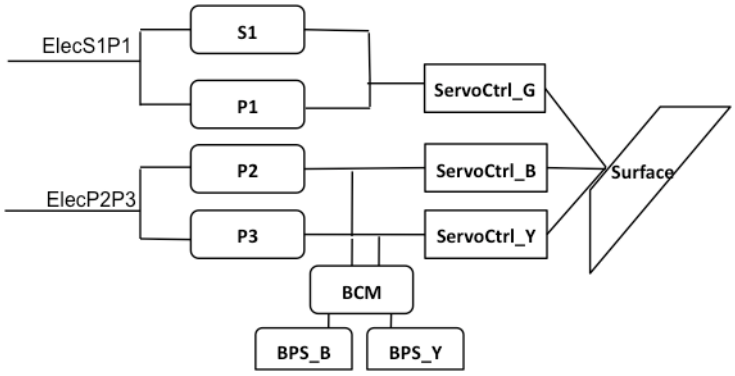


Figure IV.1 The rudder control system

Initially the secondary computer S1, the backup control module BCM and the backup power supplies BPS_B and BPS_Y are inhibited. The rudder is then controlled by the three primary control lines (PL1, PL2, PL3). When the three primary control lines fail, S1 is activated and the system switches to SL. If the latter also fails, BCM, BPS_B and BPS_Y are activated enabling the backup control. Therefore, three control modes can be distinguished: the primary control (PC), the secondary control (SC) and the backup control (BC). Figure IV.2 summarizes the control modes.

Figure IV.2: The control modes and associated control lines

The subsystem uses electric and hydraulic power to control the rudder. The computers use electric power from two different distribution sources (ElecP1S1 and ElecP2P3) as shown in Figure IV.1. Each of the three servo-controls uses hydraulic power from a different distribution line, namely hyd_G, hyd_B, hyd_Y. The hydraulic power inputs are not shown on Figure IV.1 since their behaviors are not analyzed. They are considered always available.

Related operational requirements: According to [MMEL 2008][4],

- OR1: the failure of any component among P2, ServoCtrl_G, ServoCtrl_Y, ServoCtrl_B, BCM, BPS_B or BPS_Y leads to "No-Go" status.
- OR2: the failure of any component among P1, P3 and S1 is "Go-If":
    - P3 and S1 must be operational if P1 is failed

[4] [MMEL 2008] is actually a Master MEL (MMEL). MELs result from the completion of MMELs with airline specific policies and are not public documents. MMELs are established by the aircraft's manufacturer.

- P1 and S1 must be operational if P1 is failed.
- P1, P2 and P3 must be operational if P1 is failed[5].

There is no mission profile requirement related to the subsystem in the Flight Crew Operating Manual (FCOM) that we have consulted.

Based on the system description, one can build the core model that represents the basic part of the stochastic model. We present in the following an informal description highlighting the kinds of data that should be used while developing the model using the meta-model.

IV.1.2 The core model specification

The meta-model that serves as basis for the core model development is defined in section II.3.2 of chapter II. This specification uses its features.

All the components have similar behaviors and are represented using the features related to the Eclass *SysComponent*. As identifier (*id*), the component name or an identification number can be defined. A state variable with a domain defined by the set *{ok, failed}* is considered; the initial value is ok.

For the related *events*, we consider a *failure* event, which changes the state from "ok" to "failed", and a *maintenance* event that restores the state to "ok". We assume that the failure event occurs while in flight, since it is usually characterized by a rate per flight hour. For this, we use an interfacing object *CP* (instance of *CurrentProcess* - Figure II.11*)*, between the core and the mission dependent models, which represents the period of the mission profile that is being achieved. For the expression of the guard of the maintenance event, authorization information (*CP_M*) from the mission dependent model is needed to enable the event. For example, the events are defined as follows for P1:

Failure event:
Name: *P1_failure*
guard: *state=ok and CP-type=flight;* effect *state=failed*
TTOdistrib: *distribLaw=exponential, parameter: lambda=2E-4*

Maintenance activity:
Name: *maintainP1*
guard: *state= failed and id ∈CP-M;* effect *state=ok*
TTOdistrib: *distribLaw=deterministic*, parameter: *t=1*

For the secondary computer and the backup control components, the activation and deactivation scenarios are represented. The activation and deactivation depend on the state of the primary control lines (PL1, PL2, PL3). The primary control lines are represented using instances of *Dependency*. Instances *PL1, PL2,* and *PL3* represent respectively the state of the connection between P1 and ServoCtrl_G, the state of the connection between P2 and ServoCtrl_B, and the state of the connection between P3 and ServoCtrl_Y. The

[5] These are not actually the full conditions, we only consider the conditions related to the components involved in the subsystem described.

resulting state variables, named *PL1-state, PL2-state* and *PL3-state,* have the set {*ok*, *failed*} as domain and their values are determined by the following combination function (*relation*), using PL1 as example.

$$\begin{aligned} &PL1\text{-}state = ok \\ &\quad if \quad P1\text{-}state=ok \quad and \quad ServoCtrl\_G\text{-}state=ok \quad and \\ &\quad\quad ElecP1S1 \text{ and } hyd\_G \\ &otherwise\ PL1\text{-}state = failed \end{aligned} \tag{IV-1}$$

ElecP1S1 and *hyd_G* are input data from other subsystems. As we are considering the rudder control subsystem alone, *ElecP1S1* and *hyd_G* are considered always available.

The requirements expressed at the end section IV.1.1 are not dependent on any mission profile. They are part of the minimum requirements (*Min_Sys_R*), to which may be added requirements specific to a given mission profile. They are expressed using the features defined in the meta-model of Figure II.10.

The attribute *reference* is not used here since it is used only during model updates in operation. The requirements are considered initially satisfied, i.e., *status=satisfied*. The expression of *Min_Sys_R* is formulated as the conjunction of OR1 and OR2 (see section IV.1.1):

OR1: The condition related to the "no go" components is as:

$$P2 = ok \wedge ServoCtrl\_G = ok \wedge ServoCtrl\_Y = ok \wedge ServoCtrl\_B = ok \wedge BCM = ok \wedge BPS\_B = ok \wedge BPS\_Y = ok$$

OR2: The operational conditions related to the "go if" components are expressed as follows:

- $(P1=ok) \vee (S1=ok \wedge P3=ok);$
- $(P3=ok) \vee (S1=ok \wedge P1=ok);$
- $(S1=ok) \vee (P1=ok \wedge P2=ok \wedge P3 =ok).$

The conjunction of the conditions of OR2 and the expression of OR1 gives *Min_Sys_R*.

$$\begin{aligned} Min\_Sys\_R = \{ & P2=ok \wedge ServoCtrl\_G =ok \wedge ServoCtrl\_Y =ok \wedge ServoCtrl\_B=ok \\ & \wedge BCM=ok \wedge BPS\_B=ok \wedge BPS\_Y =ok \wedge \\ & (P1 =ok \vee (S1 =ok \wedge P3 =ok)) \wedge \\ & (P3 =ok \vee (S1 =ok \wedge P1 =ok)) \wedge \\ & (S1 =ok \vee (P1 =ok \wedge P2=ok \wedge P3 =ok)) \} \end{aligned} \tag{IV-2}$$

The conditions are actually derived from high-level constraints based, in our case, on the ability to control the rudder in case of failure during a flight. They are rather dependent on the availability of the control lines. Using the control lines, we have:

$$Min\_Sys\_R = (PL2 = ok \wedge (PL1 = ok \vee (PL3 = ok \wedge SL = ok)) \wedge (PL3 = ok \vee (PL1 = ok \wedge SL = ok)) \wedge BCL = ok \wedge (SL = ok \vee (PL1 = ok \wedge PL3 = ok))). \quad \text{(IV-3)}$$

This means that the requirements are satisfied as long as (PL2, BCL and at least two of PL1, PL2 and SL) are operative.

The core model is to be composed with a mission dependent model in order to take into account the mission profile characteristics. For this, one has to define a mission profile.

IV.1.3 A mission dependent model

Different mission dependent models may be defined to represent different mission profiles. Different ways of decomposing the flights into phases can be considered, with each scenario leading to a different flight profile. For the case studies, we consider a parametric mission dependent model. It can be tuned to obtain different scenarios of mission profile.

A mission is composed of a sequence of a number (*NbFlights)* flights, as defined in the meta-model (Figure II.11). Each of them is represented based on *CompleteFlight*, which is in turn represented by instances of *GroundPeriod* and *FlightPeriod*. For their identification (*id*), they are numbered.

All the ground periods have the same structure, i.e., composed of 3 instances of *GroundActivity*: scheduled maintenance, unscheduled maintenance and the other activities during the flight preparation. A duration p*gd* is considered as associated *planned duration*. The ground period comprises a period of scheduled maintenance activities whose duration *SM_Time* is considered deterministic. The unscheduled maintenance is an extension of the scheduled maintenance. It takes place when the dispatch requirements are not satisfied. The other activities are considered to have a given duration *oad*.

The flight periods are divided into three phases denominated *Taxing_to_Takeoff*, *In_Flight* and *Landing*. They are characterized as follows:

Taxing_to_Takeoff
duration: distribLaw=deterministic, parameter: t=ttd

In_Flight
duration: distribLaw=deterministic, parameter: t=ifd

Landing
duration: distribLaw=deterministic, parameter: t=ld

The variables *tttd, ifd* and *ld* are the estimated duration of these phases. For illustration purpose, we consider that Min_Sys_R is the requirement related to these phases.

Maintenance policy: We assume that the maintenance activities take place at the *main-base-station*, and therefore no logistic delay time is considered. For the prioritization of maintenance, "no-go" components are considered primarily during unscheduled maintenance. A predefined ordered list of failed components is considered for scheduled maintenance. The *priorityList* is thus a simple ordered list of the *id* of the components to maintain.

The following section presents an implementation of this description as a concrete model using the SAN formalism.

IV.2 The Model Using SAN Formalism

We consider the Stochastic Activity Network formalism and the associated Möbius tool, which provide compositional operators that are convenient to master the complexity of the model.

IV.2.1 The core model

The core model consists of the subsystem and its related requirements representation. In AltaRica, the model distinguishes each basic system component by using a sub-node for each of them. In SAN, the representation of a basic system component is too small to be considered as a sub-model. Figure IV.3 shows for example the representation of the primary computer P2, in AltaRica and SAN. The complete AltaRica model of the case study is given in annex. Considering a sub-model for each basic component will lead to a considerable number of very small sub-models to manage separately.

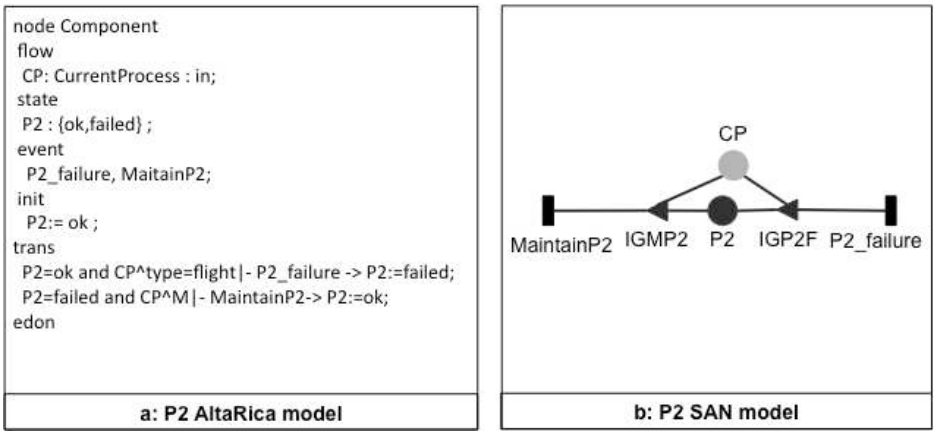


Figure IV.3 Primary computer P2 model

To simplify the presentation, we consider a decomposition into three sub models corresponding to the control modes given in Figure IV.2. Figure IV.4 shows the corresponding core model structure. The subsystem interface is composed of the control lines states information that is used to explicitly express requirements, as represented in Figure IV.8.

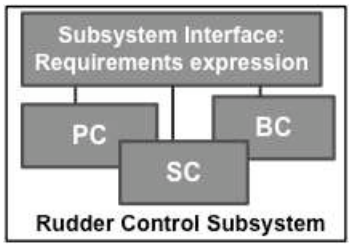


Figure IV.4 The core model structure of the rudder control subsystem

Also, to ease the understanding of the model, the representations of the system components are based on their name as presented in Figure IV.1, rather than using a particular identification number. The marking of their corresponding places represents their state value (ok or failed).

Activities named *xxx_failure* represent failure events. Activities *Maintainxx* represent maintenance activities and are enabled based on the extended place *CP*. *CP* provides information about the phase of the mission that is being achieved. *CP* is essentially used to determine whether a flight is ongoing or whether maintenance activities are allowed. It is controlled by the mission dependent model. Places *ElecP1S1, ElecP2P3, hyd_G, hyd_B,* and *hyd_Y* represent the electric and hydraulic power supply inputs to the model.

For clarity purpose, some places involved in the predicate or function of the input gates are not explicitly linked to them; this is allowed by the modeling tool Möbius.

Primary control (PC) model is given in Figure IV.5.

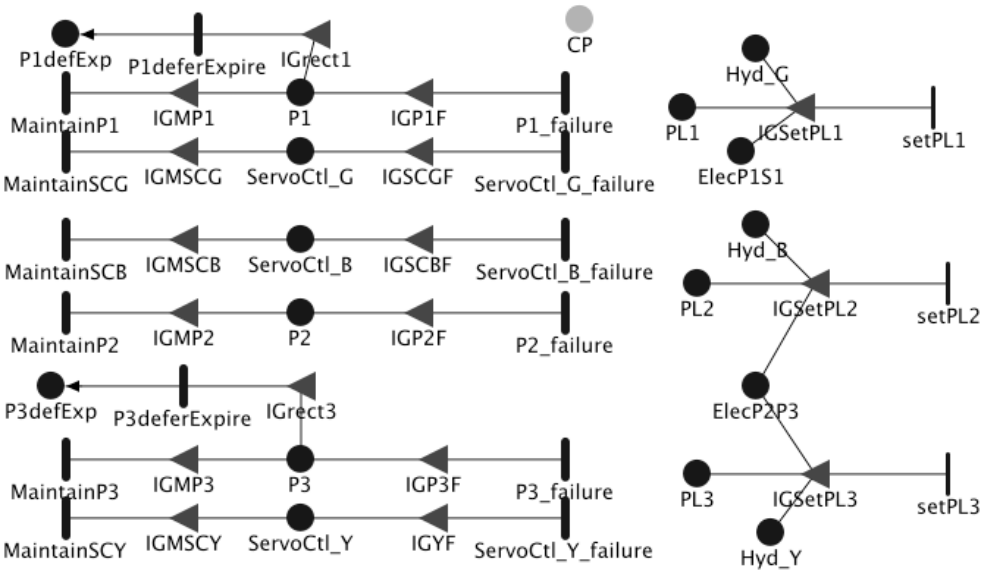


Figure IV.5: PC sub model

The transitions representing the maintenance activities (*Maintainxx*) are at the left side and the failure events (*xxx_failure*) at the right side of the places representing the basic components. Their associated input gates control their firings according to the guards and effects specified in subsection IV.1.2. Transition *P1(3)deferExpire* represents the expiration of the deadline before which the computer must be maintained after being

failed. This doesn't concern P2 since its maintenance is not deferrable ("no go" component).

Places *PLi* represent the state of the lines PLi. The places *PLi* are set according to expression (IV-1), given as example for PL1 in subsection IV.1.2. *CP* is used in the predicates of the input gates to enable the failure and maintenance activities as explained above.

Secondary control (SC) is represented in Figure IV.6. Place *S1Active* represents the activation state of S1. That is when PC fails, the instantaneous activity *S1_active* fires in order to mark place *S1Active,* representing the failover to SL. *S1_inhib* models the inhibition event. It fires when one of *PL1*, *PL2* and *PL3* becomes marked again, removing the token from *S1Active*.

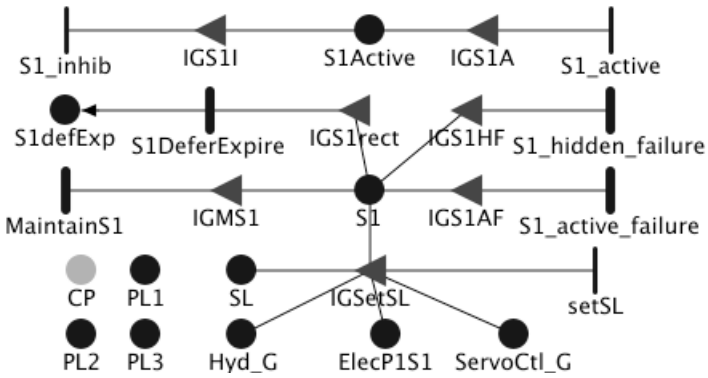


Figure IV.6: SC sub model

PL1, *PL2* and *PL3* are shared with the PC sub model, which controls their makings. They are only used in the predicates of *IGS1A* and *IGS1I* to express whether PC is failed or not. *S1_hidden_failure* and *S1_active_failure* model respectively the failure events of S1 while inhibited and activated.

SL represents the functioning state of the secondary control line. It holds when *S1, ServoCtrl_G, hyd_G* and *ElecP1S1* hold. *ServoCtrl_G* is shared with PC sub model.

The Backup control (BC) model is depicted in Figure IV.7. *BPS_BActive* and *BPS_YActive* describe the inhibition and the activation of BPS_B and BPS_Y. That is, when PL1 and SL are inoperative, BPS_B and BPS_Y are activated to supply power to BCM. They are inhibited when PL1 or SL is operative. *BPS_BActive* and *BPS_YActive* are updated by their associated instantaneous transitions, which fire according to the marking of *PL1* and *SL* as described above.

ActivateBCM represents the use of the BCM to control the surface; when none of the primary and secondary control lines is operative and BPS_B or BPS_Y supply the BCM with electric power, the BCM is activated to attempt to control the surface via ServoCtrl_Y or ServoCtrl_B. *B_YCoutput* and *B_BCoutput* represent respectively the use of power from BPS_Y and BPS_B.

BCL represents the fulfillment of the requirements concerning the components of the line. It is marked when *BCM*, *BPS_B*, *BPS_Y*, *ServoCtrl_B*, *ServoCtrl_Y*, *hyd_B* and *hyd_Y* are marked. Places *PL1*, *PL2*, *PL3*, *ServoCtrl_B* and *ServoCtrl_Y* are shared with PC sub model; and *SL* with SC sub model. Their marking are used as input to the BC sub model as they are involved in the activation and inhibition of the BC.

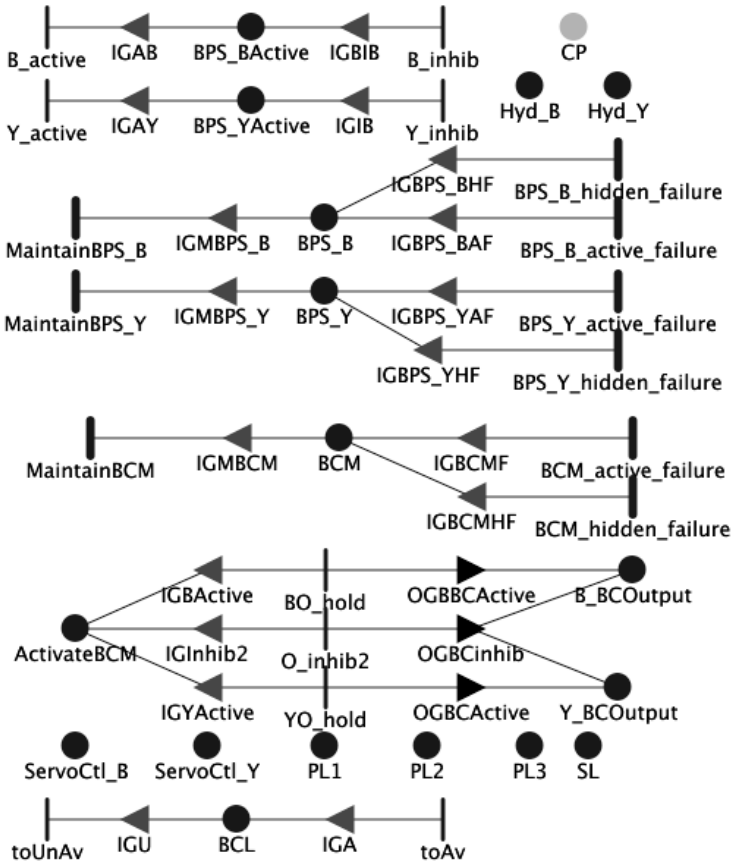


Figure IV.7: BC sub model

As only one subsystem is considered in this case study, the composition of PC, SC and BC sub models correspond to the system description. The place representing the states of the control lines are used to express the related requirements. Figure IV.8 shows the core model with an explicit representation of the requirements expression.

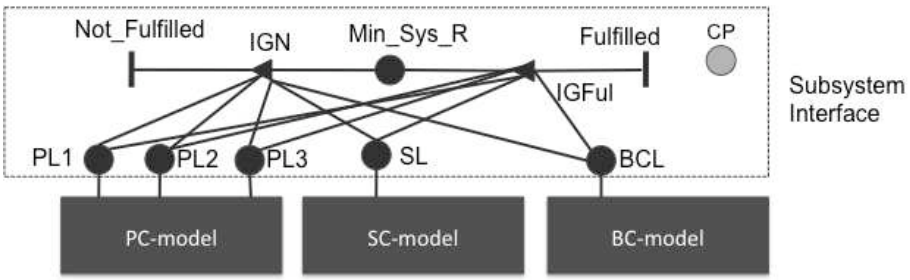


Figure IV.8: The core model with an explicit representation of the requirements expression

Place *Min_Sys_R* models the fulfillment of the requirements. The firings of the instantaneous activities *Fulfilled* and *Not_ Fulfilled* update the place according to the satisfaction of the expression (IV-3).

The underlying principle of the requirements expression is to use the state information of the different system functions and components to model the different constraints related to the mission without having to change the core model content. Therefore, other requirements can be expressed similarly, provided the corresponding Boolean expression on which will be based the predicate of the gates is given.

The core model is the basic part of the initial model. The updates that will affect it during operation concern the component states changes, the failures distribution and the maintenance activities duration's distribution.

Min_Sys_R is used to make the connection with the mission dependent model that is described in the following.

IV.2.2 The mission dependent model

As the supporting tool Möbius allows the construction of parametric models, the mission dependent model is based on a generic structure that is to be set by specifying the number of flights for each mission and the duration parameters of each flight.

The mission dependent model is shown in Figure IV.9. It is composed of two parts. The upper part represents a flight and the lower part represents the activities on ground at a stop.

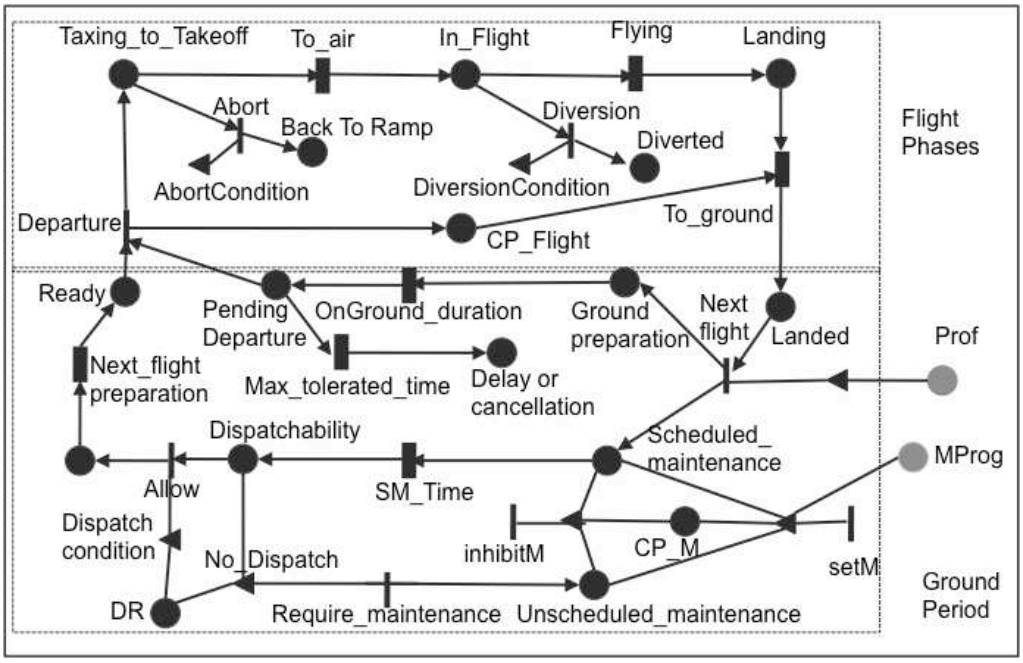


Figure IV.9: The generic mission dependent model

A flight is represented by three phases *Taxing_to_TakeOff*, *In_Flight* and *Landing* as defined in section IV.1.3. During the *Taxing_to_TakeOff* the flight can be aborted and it

can be diverted during the *In_Flight* phase. The input gates *AbortCondition* and *DiversionCondition* represent the conditions under which these interruptions can occur (in-flight requirements fulfillment). The conditions are stated using the marking of *Min_Sys_R,* which is assumed to be the related requirement for illustration purpose. Place *CP_Flight* indicates whether a flight is ongoing or not.

The sub model of a ground period consists of the representation of the preparation for the next flight and the readiness for departure on time. The beginning of the preparation for the upcoming flight is represented by the marking of places *Ground_Preparation* and *Scheduled_maintenance,* stating that the scheduled ground period is ongoing and the system is under scheduled maintenance. When the scheduled maintenance is finished (activity *SM_Time* fires), the place *Dispatchability* then holds and the instantaneous activity *Allow* can fire if the dispatch requirements, stated in the predicate of *Dispatch_condition*, are fulfilled. Otherwise the instantaneous activity *Require_maintenance* fires if the corrective action requires maintenance tasks (stated by the predicate of *No_Dispatch*), place *Dispatchability* still holds until the corrective action succeeds (predicate of *Dispatch_condition* becomes true) and the flight is allowed.

The management of the maintenance is that the extended place *MProg* is made of lists that determine, for each ground period, the failed components to consider firstly for the maintenance. Place *CP_M* is set with a number that identifies the component to maintain and it is used in the core model to allow the corresponding maintenance activities.

In the current illustration, the dispatch requirements fulfillment consists of testing the marking of *DR*, which is assumed to be the same as Min_Sys_R. The non-fulfillment of the dispatch requirements may require longer duration for the accomplishment of the ground activities since unscheduled maintenance activities may be required. Until the end of the ground activities, the scheduled ground duration may have elapsed (firing of activity *OnGround_duration* moving the token to place *Pending_Departure*) and the tolerable delay (*Max tolerated time*) may be running out. A delay or cancellation occurs if the tolerated time to dispatch is exceeded. The timed transition *Next_flight_preparation* represents the other activities (passengers and baggage processing …) that may consume time, causing delay. Place *Prof* (at right) is an extended place representing the parameters of the list of flights to be achieved. The input gate linked to this place indicates whether there is a next flight to achieve or not (end of the mission or not).

The global model results from the composition of the core and the mission dependent models, based on *Min_Sys_R* and *CP* (CP is represented by *CP_M* and *CP_Flight* in the mission dependent model). The obtained model is used to assess hypothetical scenarios whose results are presented in the following.

IV.2.3 Example of assessment results

To process the model and get the evaluation results, one has to set the initial markings and the parameters such as the distribution laws of the timed activities, using data collected during operation. In order to provide examples of evaluation results, hypothetical values are assumed for the parameters. We assume that all the failure events corresponding to the system components have exponential distributions. The default failure rates used are given in Table IV.1. It is worth mentioning that, in order to preserve the industrial

confidentiality, the values used have been selected to form a consistent set, without disclosing industrial properties. These values are used to set the core model by default. Changes of the rates or the failure distributions will be given by the prognosis module in operation.

	Rate (per flight hour)
ServoCtrl_B(G)(Y)_failure	$\lambda_{SC}=2.10^{-5}$
P1(2)(3)_failure	$\lambda_{p}=2.10^{-4}$
S1_hiden_failure	$\lambda_{Shf}=2.10^{-5}$
S1_active_failure	$\lambda_{Saf}=10^{-4}$
BPS_B(Y)_hidden_failure	$\lambda_{bps}=10^{-6}$
BPS_B(Y)_active_failure	$\lambda_{bps}=5.10^{-5}$
BCM_hiden_failure	$\lambda_{bcm}=4.10^{-6}$
BCM_active_failure	$\lambda_{bcm}=5.10^{-5}$

Table IV.1 Default failure rates for the rudder control subsystem

All the system components are also considered operational by default. The failed components will be identified by the diagnosis module during operation.

IV.2.3.1 System reliability

The system reliability concerns only the core model. SR corresponds to the probability that Min_Sys_R holds during a given period. It can be used to characterize the duration of the mission to assign to the aircraft. Curve 1 of Figure IV.10 shows the corresponding reliability curve computed with 95% of confidence level. All the system components are initially operative. This evaluation can be used while defining the mission to assign to the aircraft. From the evaluation, the maximum number of flight hours that can be achieved, with the reliability remaining above a given threshold can be determined. For example, considering 0.98 as the reliability threshold, the mission duration, without maintenance activities, is about 80 flight hours.

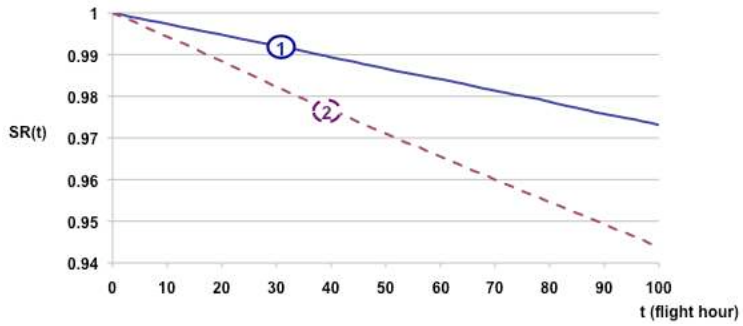


Figure IV.10: System Reliability

Case of a failed component: Curve 2 of Figure IV.10 shows the system reliability considering the failure of the primary computer P1; P1 state is set failed in the initial state of the model.

Curve 2 together with curve 1 of Figure IV.10 illustrate a situation where for example one has to decide on whether it is preferable to maintain the primary computer before considering a new mission or not. For a mission of 80 flight hours for example, the reliability decreases from 0.98 to 0.955 in case of P1 failed. To have a reliability of 0.98 in case of P1 failed, the maximum number of flight hours is about 35.

The SR measure helps to estimate the length of mission to assign. To deal with the actual ability to succeed a given mission, its reliability MR must be evaluated taking into account the mission profile. After for example a failure, the MR measure of the remaining part of the mission can also be evaluated to ensure that the mission can successfully be completed without causing any disruption for the subsequent mission. Examples of mission reliability measures are presented in the following.

IV.2.3.2 Mission reliability

For the parameters of the mission dependent model, we consider a mission of 4 flights per day over a week. We assume that the timed activities of the mission dependent model have deterministic durations. Table IV.2 gives the default values used to set the mission dependent model.

Activities	Duration (in hour)
To_air	0.5
Flying	2
To_ground	0.5
OnGround_duration (Ground Period)	8.25 - end of day 1.25 – otherwise
Max_tolerated_time	0.25
SM_Time	0.5
Next_flight_preparation	0.75

Table IV.2 Mission dependent model default parameters

It is noteworthy that different distributions can be specified. Each flight takes 3 hours. The planned duration *gpd* of a ground period is 1.25 hours during the day and 8.25 hours at the end of the day (after 4 flights).

The mission reliability *MR* is the probability to have no tokens in places *Delay_Or_Cancellation*, *Back_to_Ramp* and *Diversion* of Figure IV.9. We assume that, as long as *MR* is larger than a threshold, referred to as Minimum *MR* Requirement (*MMRR*), the mission can be continued. *MMRR* is to be set by the airline company, in agreement with the aircraft manufacturer.

Figure IV.11 shows the mission reliability considering two initial states of the primary computer P1: P1-OK (P1 is ok at the starting of the mission), and P1-KO (P1 is failed at the starting of the mission), the other components are assumed OK at the start of the mission.

Based the results, the time from which the reliability becomes lower than a given threshold can be determined. For example, considering 0.975 as *MMRR*, one has to consider strengthening its ability to maintain after 120h (end of the 5th day) in case of P1-

KO. The curves also illustrate a situation where one has to decide on whether it is preferable to defer the maintenance of computer P1, knowing that there is one week remaining mission to achieve. With the assumed parameters, the reliability of the one-week mission (t = 168h) will increase from 0.952 to 0.978 if P1 is repaired before achieving the mission.

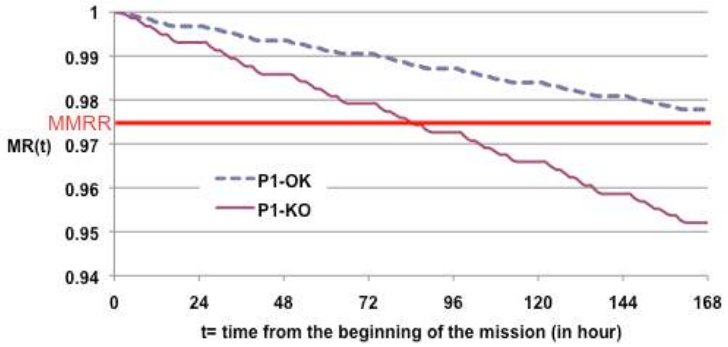


Figure IV.11: Mission Reliability

Maintenance during the mission achievement: In the case that P1 is failed (P1-KO) before the mission, one can consider scenarios of maintenance during its achievement. Figure IV.12 shows the reliability measure corresponding to two scenarios of the maintenance deferment:

- M1: Accomplishment of the maintenance the first day at night,
- M2: Accomplishment of the maintenance the second day at night.

These are taken into account in the model, through the authorization of the maintenance at the corresponding stations. Curve No-M corresponds to the case without maintenance authorization.

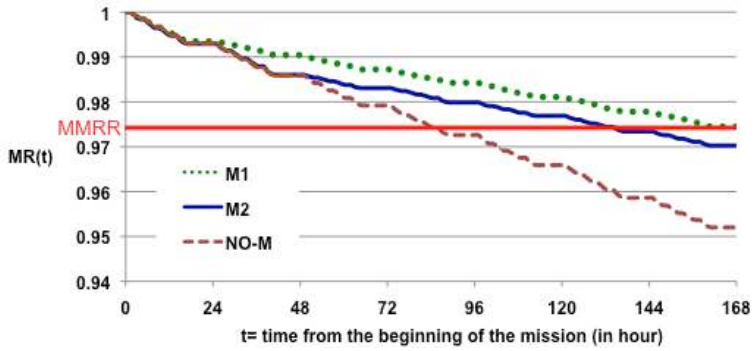


Figure IV.12 Maintenance during the mission

The reliability threshold of 0.975 is fulfilled in case M1 and not in case of M2. However, these are still estimations that are close, and since the magnitude of the difference is not so much considerable, one can decide to defer the maintenance if other factors like the cost are to be privileged.

The examples show assessment results that analyze whether the reliability requirement of a given mission during a given period of time can be achieved or not. The assessment is

intended to be done again each time a major event occurs during the mission achievement. We examine in the next section the valuable role and the impact of the re-assessment in regard to the previous initial assessment. We also show how to use the assessment results to manage the mission and its associated maintenance activities.

IV.3 Impact of Re-Assessments During Missions

As the system will be continuously monitored, diagnosis and prognosis information will be notifying major changes in the system components' functional state, their failure rates and distributions. For instance, the failure distribution of a computer may be initially following an exponential law and prognosis may denote, during the mission achievement, an increasing likelihood of failure, suggesting a failure distribution that is following a Weibull law. As these changes may affect the predicted mission reliability, the model must be updated with the new failure distribution in order to reevaluate the reliability.

The major changes that we consider are the changes in the state of system components, the changes in components' failure rates and the changes in mission profile. We firstly consider the impact of a failure occurrence during the mission, and show how the assessment results will help to determine when to repair this component. Then we show the impact of a failure distribution change before analyzing the impact of mission profile changes. We consider that $MMRR = 0.975$.

IV.3.1 Component failure occurrence

The single failures of P1 or S1 do not affect safety and do not prevent mission achievement. However, the failures of both components lead to a "No-Go" condition. Therefore the mission reliability can be assessed considering both cases.

IV.3.1.1 Failure of primary computer P1

Curve 0 of Figure IV.13-a shows the mission reliability, *MR*, as assessed before the beginning of the mission, with the assumption that all components are OK at the starting of the mission. It can be seen that at the end of the mission, *MR* is above *MMRR*.

Curve 1 of Figure IV.13-b corresponds to the case where P1 has been diagnosed as inoperative at the end of day 4. *MR* is thus re-assessed, considering i) as initial time (t=0) day 5, and ii) P1 is inoperative at t=0. *MR* is thus equal to 1 for each re-assessment, as the system is in an operative global state at the time the model execution is performed (the system is inspected and the global operational state is ensured). It can be seen that the new assessed measure is still above *MMRR* at the end of the whole planned mission. The mission can be continued without maintenance until its end, unless a new event occurs, in which case a new re-assessment will be needed.

Curve 2 of Figure IV.13-c corresponds to the case where P1 has been diagnosed as inoperative at the end of day 2. As for the previous case, *MR* is re-assessed, considering i) as initial time the next day (i. e., day 3), and ii) P1 is inoperative at t=0. It can be seen that *MR* is below *MMRR* from day 5. This result shows that P1 has to be repaired no later than day 5 to satisfy the *MMRR* requirement.

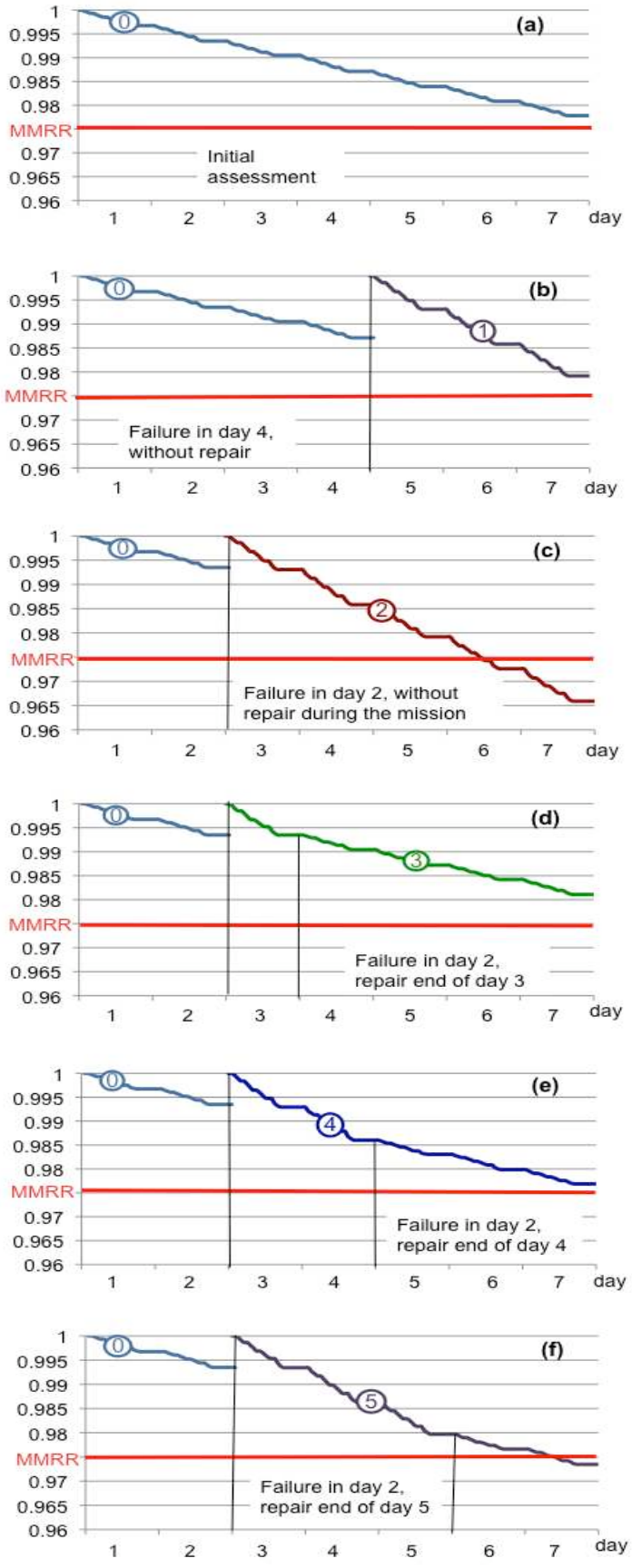


Figure IV.13 Impact of P1 failure during mission achievement

Of course, the earlier P1 is repaired the earlier the remaining mission reliability will be improved. However, spares are not available at all destinations, and one has to find the right time to repair P1, according to resource availability, while ensuring an *MR* above *MMRR*. Three situations are possible at this stage, considered below:

- S3: P1 is repaired at the end of day 3,
- S4: P1 is repaired at the end of day 4,
- S5: P1 is repaired at the end of day 5.

Figure IV.13-d, Figure IV.13-e and Figure IV.13-f correspond respectively to the three above situations. Curve X, X = {3, 4, 5}, is related to situation SX. It corresponds to the result of *MR* re-assessment, at end of day 2, assuming that P1 will be repaired at the end of day X. It can be seen that for S3 and S4, *MR* is above *MMRR* for the whole mission, while S5 leads to an *MR* below *MMRR*, at day 7. S5 improves *MR* but not enough to avoid an *MR* below the threshold. This means that P1 should be repaired either in day 3 or day 4, depending where and when the maintenance can take place.

It can be seen that curves 1 and 2 have the same slope but shifted in time, which is not surprising, as they have been obtained from the same model, with the same initial states of the components and the same exponential distributions. Indeed, we have assumed exponential distributions for all components to show that the operational changes will induce perceptible changes in the results.

With the modeling approach used and the available tools, it is possible to consider other distributions and to take into account the age of the other components involved in the analysis. However, aging is a long-term variation process, the granularity of changes is much larger than one day or one week (the duration of a mission). In addition, very small variation of the failure rates of the components during a mission induces a non-perceptible variation in the reliability curves. However, we will see in the subsection IV.3.2 that changes in components' distributions can have a significant impact.

IV.3.1.2 Failure of secondary computer S1

Curves 6 and 7 of Figure IV.14 show the re-assessment of *MR* after the secondary computer S1 failure, respectively during day 4 and day 2. These curves are to be compared to curves 1 and 2 of Figure IV.13-b and Figure IV.13-c. Curve 0 is the same for all figures.

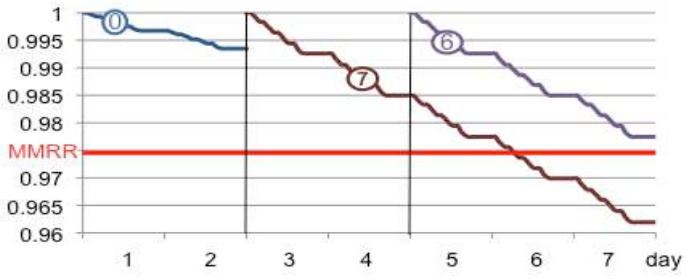


Figure IV.14 Impact of S1 failure during mission achievement

Curve 6 is below curve 1 and Curve 7 is below curve 2. This means that S1 has a more negative impact on the remaining mission reliability than P1. This is due to the fact that P1 failure rate is greater than S1 failure rate. The requirements are that one of computers P1 and S1 must be operative in order to achieve the mission. Therefore the risk of interrupting the mission is higher when S1 is inoperative than when P1 is inoperative.

IV.3.2 Changes in failure distribution

The prognostic is based on long-term observations of specific parameters during the whole life of multiple aircraft of the same family. It is based on statistics as well as on other approaches that provide an acceptable confidence. Prognostic results are usually made available from time to time (that can be of the order of magnitude of few months to few years), without any synchronization with mission achievements.

The aim of this analysis is to check the impact of the distribution change when the notification is received. This does not mean that the distribution has suddenly changed during the mission. This corresponds to the case where the notification of the distribution change takes place during the mission.

The impact of the primary computer P2 failure on mission reliability is larger than the impact of P1 or S1 failures, because the failure of P2 leads directly to an inoperative state. Let us assume that, based on the various observation means used by the prognostic process:

- P2 failure has been first identified as following an exponential distribution, with a mean time to failure, MTTF0 = 5000 flight hours.
- During day 2, a new distribution is notified.

For purpose of illustration, we consider two possible distributions for the failure rate of P2, D1 and D2:

- D1: is a conditional Weibull distribution with shape parameter α=2.5 (α>1 represents an increasing failure rate), scale parameter β=5635 and elapsed time Te=5000.
- D2: is also an exponential distribution, with a mean time to failure, MTTF1 = MTTF0 / 2 (on average 4 failures per year instead of 2 failures per year).

Figure IV.15 shows the impact of the distribution change from exponential (curve 0) to Weibull (curve 8). Curve 8 corresponds to a prognostic issued at day 2 of the mission. The rapid reliability decrease of curve 8 compared to curve 0 results from the increasing failure rate of computer P2.

Curve 8 shows in particular that with a Weibull distribution, *MR* is almost equal to *MMRR* at the end of the mission.

The curve corresponding to distribution D2 is very similar to curve 8. It is slightly below this curve from day 5, due to the fact that the failure rate of D2 is on average higher than

that of D1. It is in particular below *MMRR* for day 7. As a result, the mission should be modified, most probably shortened, to satisfy the *MMRR* condition.

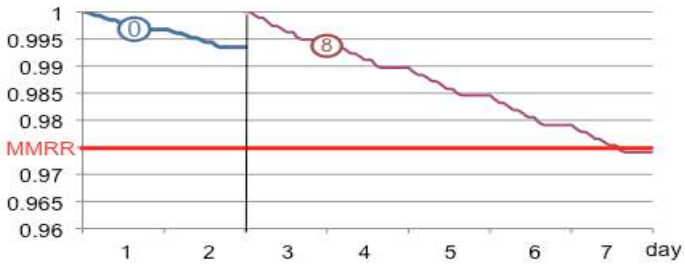


Figure IV.15 Failure distribution change, notified and integrated at day 2

These results show that for some impacting parameters, taking into account the newly identified distribution, as soon as it is notified, is very important, while it is less important for some other parameters. Such analyses should be performed during the building of the model to identify the most sensitive parameters for which mission reliability re-assessment is recommended as soon as a new distribution is notified.

IV.3.3 Change in the mission profile

Aircraft operations depend on various external factors. In particular, some unforeseen events, that do not necessarily affect directly the aircraft itself, may lead to a change of the initial mission. For example, an aircraft may be assigned new flights with different durations, or additional flights that were initially assigned for another aircraft that should undergo a repair. Such changes require the re-assessment of the mission reliability.

To illustrate the impact of changes in mission profile, we have considered four profiles, presented in Figure IV.16:

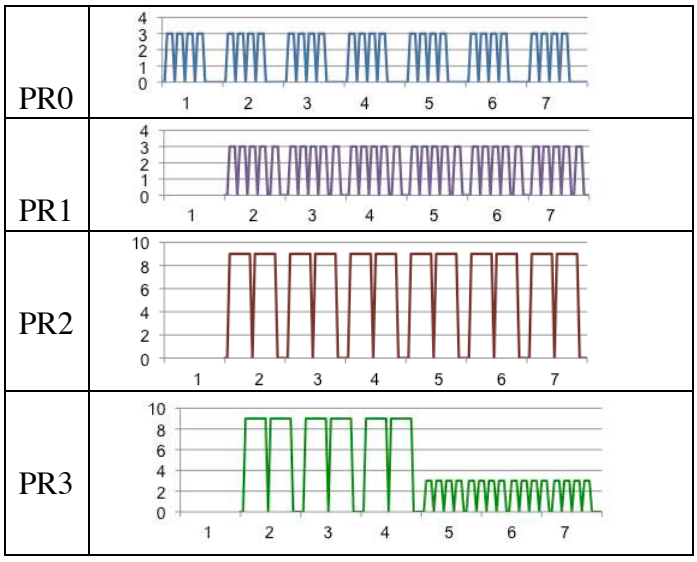


Figure IV.16 Mission profiles (number and duration of flights per day)

- PR0: the initial assignment, 4 flights per day during 7 days, the duration of each flight is 3 hours. PR0 corresponds to the case considered for the previous assessments.
- PR1: 5 flights per day from day 2 (corresponds to a mission change after day 1), same durations of flights.
- PR2: 2 flights per day from day 2, the duration of each flight is 9 hours.
- PR3: 2 flights per day from day 2 to day 4, the duration of each flight is 9 hours, then again 4 flights per day, 3 hours each, from day 5.

Figure IV.17 gives *MR* for PR0 and PR1. One can see that the reliability values for PR1 is lower than the values for PR0 after 6 days. However, the minimal mission reliability requirement (*MMRR* = 0.975) is still satisfied.

Figure IV.17 Mission changes from PR0 to PR1

Figure IV.18 gives *MR* for PR0, PR2 and PR3. For PR2, *MR* becomes lower than *MMRR*. One can consider adjusting this new profile in order to improve the mission reliability. A possible mission adjustment could correspond to PR3. The mission reliability with the adjusted profile PR3 becomes approximately the same as the initial one, and the *MMRR* is again satisfied.

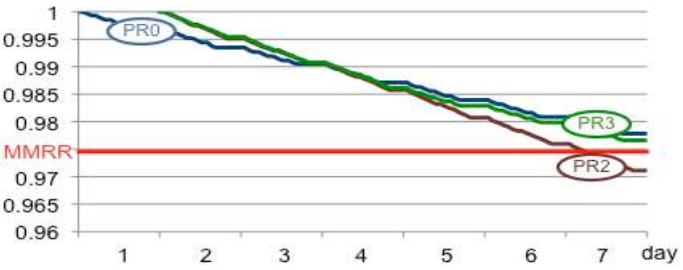


Figure IV.18 Mission adjustment from PR2 to PR3

These examples show the magnitude of online changes impact on the assessment, using the rudder control subsystem. In the next section, we consider another subsystem, the electrical power supply subsystem.

IV.4 Modeling the Electric Supply Subsystem

In the case study previously presented, we have assumed that the electrical power provided through *ElecS1P1* and *ElecP2P3* in Figure IV.1 is always available. In this section, we build the core model corresponding to the electric supply subsystem and compose it with the generic mission dependent model proposed in Figure IV.9. The composition of the electric supply subsystem and the rudder control subsystem has also been considered; its resulting analysis is briefly described in the assessment results. The following subsection presents the description of the electric supply system.

IV.4.1 Description of the electric supply subsystem

The electrical subsystem, which is presented in Figure IV.19, includes generators, bus bars, contactors, Transformers/Rectifiers Units and junctions.

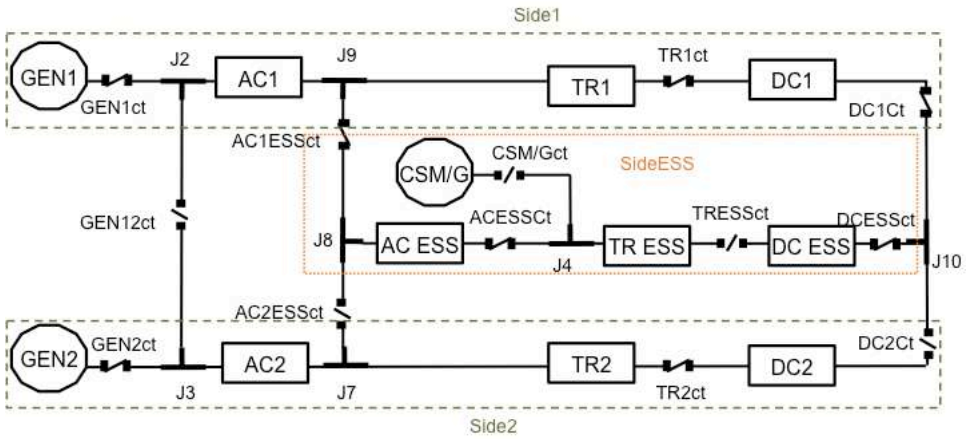


Figure IV.19 Electric supply subsystem

The delivery of the electric power is based on two main nominal generators GEN1, GEN2, and an emergency generator CSM_G, which is automatically deployed in case of the main generators loss. The subsystem includes essential components (xx_ESS) based on which the electricity is still supplied when the emergency generator CSM_G is deployed. The electricity is supplied to the electrical loads through four nominal distribution bus bars AC1, DC1, AC2, DC2 and two essential bus bars AC_ESS and DC_ESS. AC1, AC2 and AC_ESS are alternative current distribution bars. DC1, DC2 and DC_ESS are direct current distribution bars. The alternative current is converted to direct current by the transformers TR1, TR2 and TR_ESS. Based on the segregation of the subsystem components, three electrical supply lines can be identified in the system:

- Side1, composed of the generator GEN1, the distribution bus bars AC1 and DC1, and the transformer TR1.
- Side2, composed of the generator GEN2, the distribution bus bars AC2 and DC2, and the transformer TR2.
- SideESS, composed of the generator CSM_G, the distribution bus bars AC ESS and DC ESS, and the transformer TR ESS. It is the essential line.

These lines are connected to each other by junctions (J*xx*) so as to enable various reconfigurations. Various contactors (xxct), installed between the components, are used to control the reconfigurations corresponding to different scenarios of closing and opening the contactors, implemented by a controller. Any nominal generator (GEN1 or GEN2) can be used to provide electricity to all the distribution bus bars. This case study focuses on failure-based reconfigurations that can be automatically triggered.

Initially, GEN1 supplies the components of Side1 and SideESS with current. GEN2 supplies side2. If GEN1 fails, the contactors GEN12ct and AC2ESSct, initially opened, are closed and GEN2 is used to supply all the bus bars. Similarly, GEN1 supplies the whole system if GEN2 fails. If both nominal generators are lost, the emergency generator CSM_G only provides power to the essential bus bars AC_ESS and DC_ESS.

The failure of AC1 blocks the transmission of the power provided by GEN1 to the rest of the system. GEN2 is used in this case to supply the whole system. In the case of GEN2 failure, contactor GEN12ct, which makes the connection between the two nominal lines, is closed so as to use the power provided by GEN1 to supply the system, through AC2. The failure of AC2 is similarly managed by a switch to Side1. In case of the failure of both AC1 and AC2, contactor CSM_Gct is closed so as to use power from the emergency generator CSM_G. Only AC_ESS and DC_ESS are supplied in that case.

The failure of a nominal transformer is managed by the use of power from the other line to feed the corresponding DC. The failure of both of them leads to the use of the essential transformer TR_ESS, which is not used in nominal case, to supply the direct bus bars. The power is then transmitted from the nominal generators via AC_ESS and J4 to TR_ESS.

Concerning the requirements, all the components of the emergency line SideESS are No-go. For the nominal part of the system, the Go-If requirement can be summed up as electricity distribution must be possible via *DC1* or *DC2*.

IV.4.2 The model of the electric supply subsystem

The characteristic of the electrical system is that the transmission of the power introduces dependencies between the components. A component can deliver its function only if it is powered, and it can be powered only if the intermediate transmitting components are functional. Therefore the power transmitting mechanism must be fully represented. The links ensuring the transmission between the components may be bidirectional, due to the various possible reconfigurations. For instance, the current is directed from DC1 to DC2 when GEN2 is not operational and it is the inverse when GEN1 is not operational. The approach to model such behavior is based on representing, in addition to the components state, the status of the links between them. This was already implemented [Kehren et al. 2004] to build a corresponding model for safety analysis, using the AltaRica formalism, which naturally manages flow propagation mechanisms. The following concerns a representation using the SAN formalism.

IV.4.2.1 Basic features of the subsystem model

The data specification for the represented electric supply subsystem is similar to the case of the rudder control subsystem (in subsection IV.1.2). The basic components (generators,

bus bars, and transformers) are represented as instances of the meta-model Eclass *SysComponent*, using state variables and activities corresponding to the maintenance and failure events. For the representation of the status of the links, instances of meta-model's Eclass *Dependency* are used. Each portion of link, between the basic components, the contactors and the junctions, is represented as a variable (or two if both directions are possible) representing the presence of the current. Figure IV.20 gives the principle, taking as example the transformer TR1 and the link with the contactor TR1ct, which bears its output.

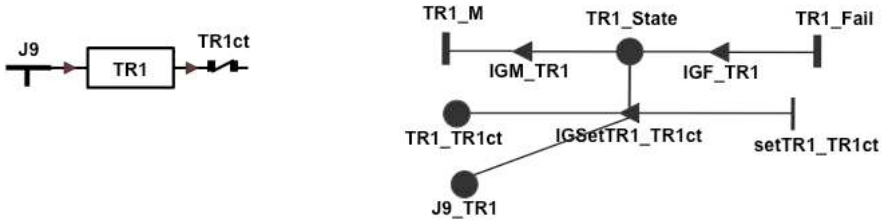


Figure IV.20 Example of component and link representation

The portion of the system is shown at the left side of Figure IV.20 and the model of the component TR1 together with the state of the link TR1-TR1ct is at the right side.

Place *TR1_State* represents the state the component TR1. *TR1_M* and *TR1_Fail* represent respectively the maintenance and the failure activities. Place *TR1_TR1ct* represents the status of the link between TR1 and TR1ct i.e., whether power is being transmitted on this portion of link or not. The link TR1-TR1ct is powered if TR1 is powered and still operational, i.e., the link J9-TR1 is powered and TR1 state is ok. The instantaneous activity *setTR1_TR1ct* is used to update place *TR1_TR1ct* according to changes of TR1 state and the powering of J9-TR1. The use of a separate instantaneous activity to update *TR1_TR1ct* is necessary, otherwise one has to cope with the complexity of determining the events that are involved in the powering of J9-TR1, in order to reflect changes. It is worth noting that *TR1_TR1ct* represents a directed transmission from TR1 to TR1ct. For some links, depending on the configuration that is being used, the power transmission may use the inverse direction. The states of both directions are represented in those cases.

Figure IV.20 shows the representation of a basic component and the use of its input link status to represent the status of the link that bears its output. The status of the links related to the junctions and the contactors are depicted using the same principle. The difference from a junction is that there is no state variable and two input links statuses are used to determine the status of an output link. The difference from the case of a contactor is that the state variable represents whether the contactor is closed or not, and it is determined based on the reconfiguration policy.

Figure IV.21 gives an example of a contactor representation considering the contactor DC1ct. Place *DC1ctClose* represents whether DC1ct is closed or not, and it is set based on the reconfiguration policy specified as a condition in the input gate *IGSetDC1ctClose*. For this particular case of failure-based reconfiguration, DC1ct is closed only when i) at least one of the nominal generators (GEN1 or GEN2) is operational, and ii) the bus bar DC1 is powered. Place *DC1ct_J10* represents the status of the output link of the contactor. It holds when *DC1ctClose* and *DC1-DC1ct* hold.

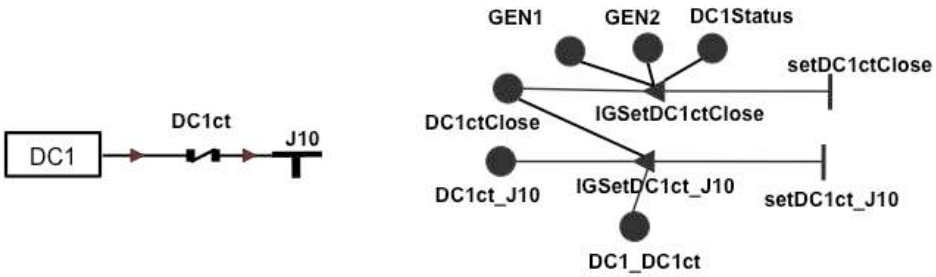


Figure IV.21 A contactor output representation

Figure IV.22 shows an example of a junction output representation. The current flow is supposed to be directed from DC1ct or DC2ct to J10, and the output of the junction, supported by J10-DCESSct. Place *J10_DCESSct* is thus updated by the instantaneous activity *setJ10_DCESSct* when the relation *J10_DCESSct = DC1ct_J10 or DC2ct_J10* no longer holds. The verification of this relation is controlled by the input gate *IGSetJ10_DCESSct,*

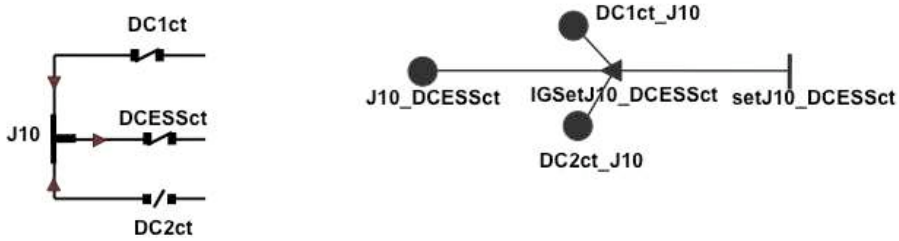


Figure IV.22 Output of a junction

These examples describe the basic principle to model the subsystem. The overall model corresponding to the electric supply subsystem is presented in the following subsection.

IV.4.2.2 The overall model

In order to simplify the construction of the core model, it is decomposed based on the three lines presented in section IV.4.1. Figure IV.23 shows the overall structure of the core model.

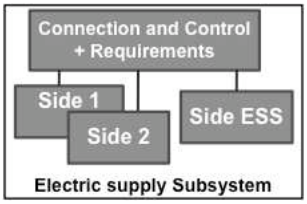


Figure IV.23 Structure of the core model corresponding to the electric subsystem

Side 1 and Side 2 are derived from the same sub model, since they are similar. A sub-model is devoted to SideESS as it behaves differently. An additional sub model is used to represent the components that make the connection between the lines, and the conditions to close and open the contactors so as to reconfigure the system as a result of a component failure. It also expresses the requirements related to the subsystem. The SAN sub-models

are presented in the following. For clarity, all the links between the input gates and their input places are not shown. However, all the links with places, whose markings are controlled by the input gate, are shown.

Figure IV.24 shows the sub-model associated to Side1 and Side2. The component names used are rather appropriate to side1. For the correspondence with side2, J2 and J9 must be replaced respectively by J3 and J7.

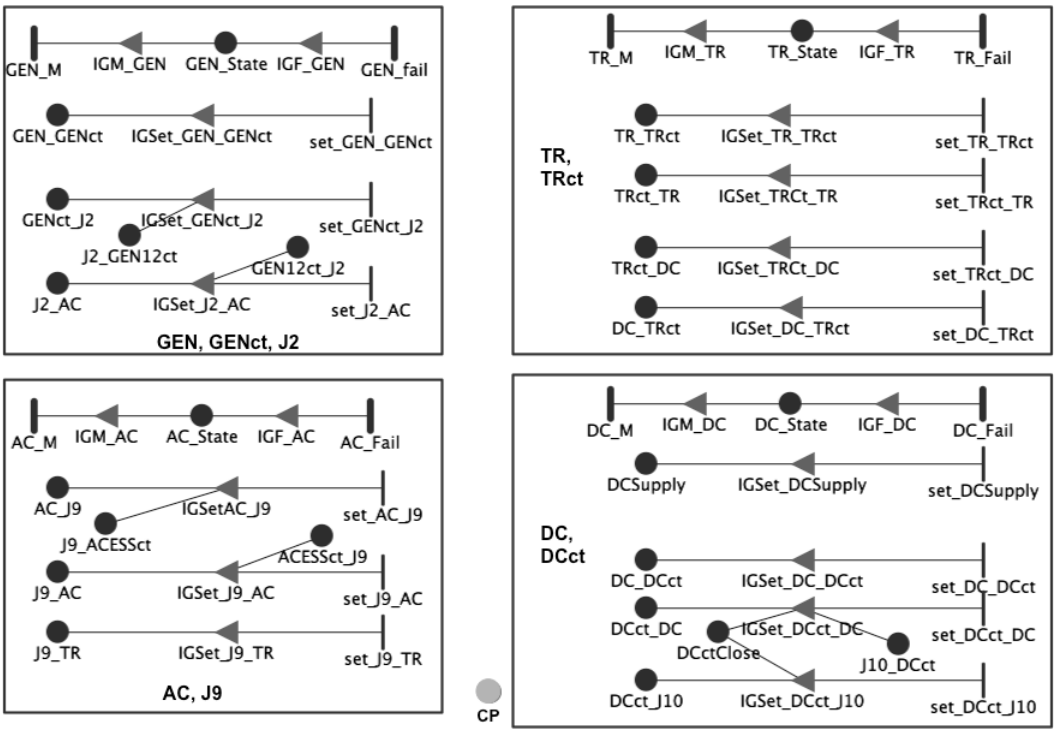


Figure IV.24 Basic model for Side1 and Side2

The understanding of the items contained in the sub-model is already given through the examples presented previously. The squares are intended to help distinguish the related components. Place *DCSupply* in the bottom right square represents whether the component DC1 (DC2 for side2) can distribute current or not. It is especially used in requirement verification as functional status of the line. It will be also used in the composition with the model of other subsystem.

SideESS sub-model is shown in Figure IV.25. Place *LineESS* represents the global state of the line. It holds when all the components of the line are operational. The other standalone places at the bottom are input to the model and are thus shared places.

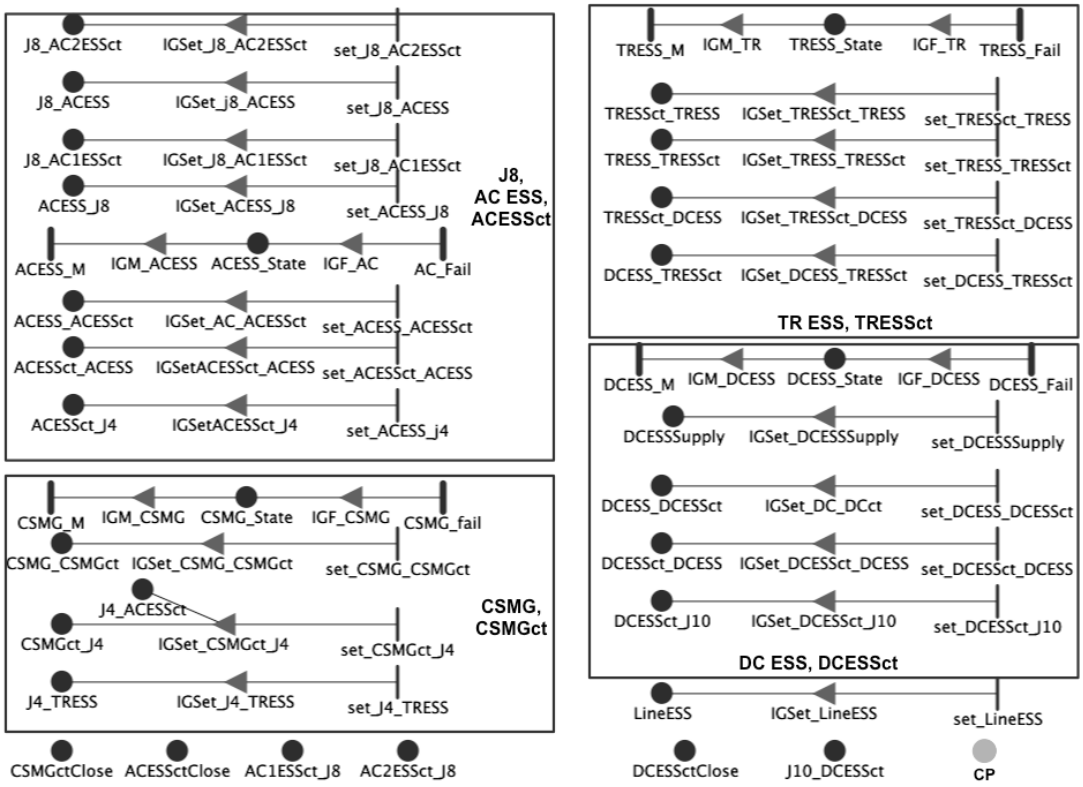


Figure IV.25 SideESS sub-model

The sub-model representing the connection points and the control of the reconfiguration is shown in Figure IV.26.

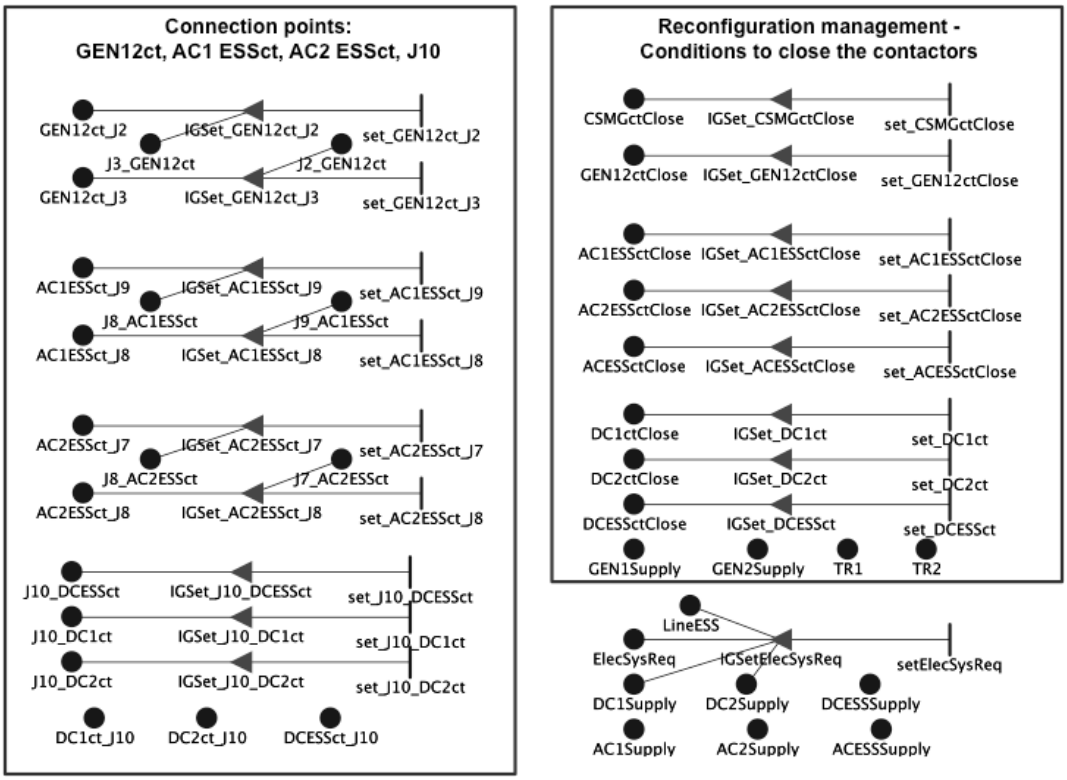


Figure IV.26 Connection and control sub-model

GEN12ct makes the connection between Side1 and Side2, AC1ESSct between Side1 and SideESS, AC2ESSct between Side2 and SideESS, and J10 between the three lines.

Figure IV.26 also shows, at its bottom, place *ElecSysReq*, which models the satisfaction of the system requirement related to the whole electrical supply system. This place is set based on the global state of SideESS (all its components have to be operational), represented by place *LineESS*, and the ability to supply power via DC1 or DC2, represented by places *DC1Supply* and *DC2Supply*. *LineESS* is shared with SideESS sub-model. *DC1Supply* and *DC2Supply* are shared respectively with Side1 and Side2 sub-models.

ElecSysReq is used in the composition with the mission dependent model, as a shared place, via an internal interface as shown in Figure IV.27. *ElecSysReq* corresponds to Min_Sys_R as the electric supply subsystem is considered alone in the core model.

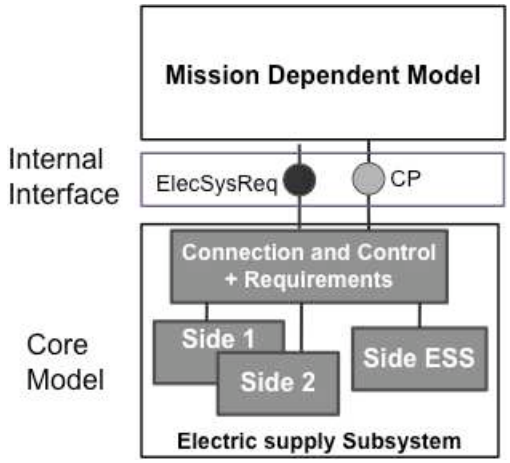


Figure IV.27 Connection with the mission model

IV.4.3 Example of assessment results

As for the rudder control system, we consider a coherent set of values, presented in Table IV.3, as default failure rates for the electric supply subsystem components.

Failure event	Rate (per flight hour)
GEN1(2)_Fail, CSMG_Fail	$\lambda_{gen}=10^{-6}$
AC1(2)(ESS)_Fail	$\lambda_{p}=2.10^{-7}$
DC1(2)(ESS)_Fail	$\lambda_{p}=2.10^{-7}$
TR1(2)(ESS)_Fail	$\lambda_{p}=2.10^{-5}$

Table IV.3 Default failure rates for the electric supply subsystem components

We firstly use the core model to analyze the system reliability. Then, we consider the mission reliability assessment.

IV.4.3.1 System Reliability Assessment

Curve 1 of Figure IV.28 shows the system reliability corresponding to the electric subsystem. The reliability is above 0.99 for the horizon of 100 flight hours considered.

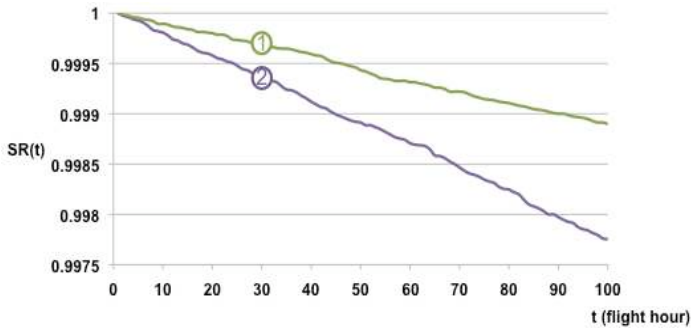


Figure IV.28 System reliability

Curve 2 of Figure IV.28 shows an example of result in case of failure, especially TR1 failure. TR1 is selected because missions can be achieved with TR1 failed, provided that TR2 is operational. The reliability decreases of about 10^{-3} when TR1 is failed. The failure of TR2 gives the same results.

Compared to the assessment with the rudder control subsystem, the reliability with regard to the electric supply subsystem is significantly better. The unreliability for 100 flight hours is in the range of 10^{-3} whereas it is 10^{-2} for the rudder control subsystem. This is due to the fact that the failure rates of the electric supply subsystem components are lower, and the subsystem includes several reconfiguration mechanisms that make it possible to keep the required DC supply available in different failure situations. Therefore, it may not be necessary to consider reliability with regard to the electric supply subsystem in the case that the failure rates of the rudder control subsystem components are significantly higher than those of the electric supply subsystem components.

Considering failure rates of the same order of magnitude ($\lambda'=\lambda*10$ for the electric supply subsystem components) as for the rudder control subsystem leads to a notable decrease of the reliability. Curve 2 of Figure IV.29 shows the corresponding results; curve 1 corresponds to the normal condition as in Figure IV.28. However, the reliability is still better compared to the rudder control subsystem, thanks to the reconfiguration mechanisms.

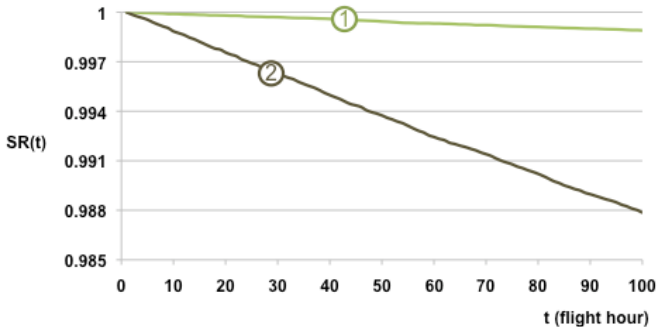


Figure IV.29 SR with higher failure rates for the electric supply subsystem components

IV.4.3.2 Mission Reliability Assessment

We consider the same basic mission profile parameters given in subsection IV.2.3.2. Figure IV.30 shows results of the mission reliability assessment based on the electric supply subsystem.

Curve *All-OK* of Figure IV.30 shows the mission reliability considering that all the subsystem components are operational at the beginning of the mission. The mission reliability is above 0.999. Considering a case of failure, especially the case of TR1 failure represented by curve *TR1-KO* of Figure IV.30, the reliability is decreased of about 10^{-3}, but is still higher than a threshold of 0.99, for example.

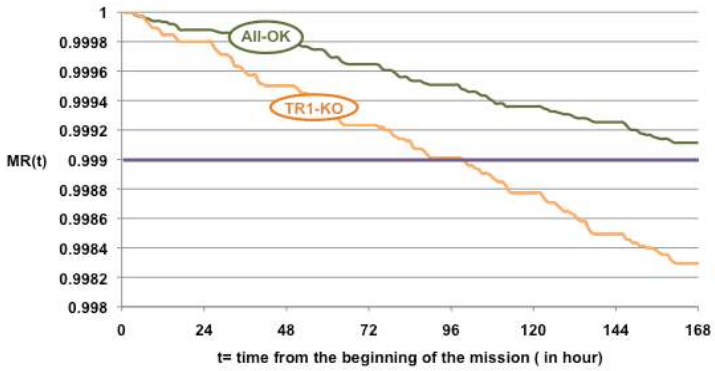


Figure IV.30 Mission reliability

TR1 maintenance scenarios: Figure IV.31 shows the reliability curves while considering the maintenance of TR1 during the mission achievement.

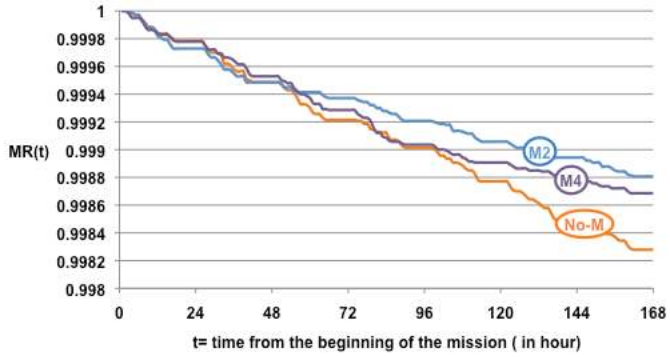


Figure IV.31 MR considering TR1 maintenance during the mission

Curve M2 shows the results for TR1 maintenance after two days, and curve M4 shows the results for the maintenance after 4 days of mission. Curve *No-M* corresponds to the case without maintenance. The difference in the mission reliabilities is not high (10^{-4} of magnitude) between the two cases of maintenance. Therefore, the maintenance can be further deferred, from case *M2* to case *M4*, without significantly impacting the resulting mission reliability.

It is worth mentioning that, as the electric supply subsystem is more reliable compared to the rudder control subsystem, the reliability results, obtained while composing their

corresponding models, do not significantly change in regard to the results corresponding to the latter subsystem. Nonetheless, notable decreases of the reliability are observed while increasing the failure rates of the electric supply subsystem components.

IV.5 Conclusion

The aims of this chapter are to illustrate the modeling approach that is presented in chapter II, to experiment scenarios of reliability assessments and to consolidate the need of re-assessing, during missions achievement, the operational dependability after major events. The modeling approach that has been proposed is intended to support both qualitative and quantitative analyses. In this chapter, we concentrated on the quantitative analysis aspect to build the operational dependability model, using the SAN formalism. Several model-based safety analysis studies, aimed at design improvement, have already experimented qualitative analysis aspects, by means of events cut-set generations and stepwise simulation.

The quantitative analysis consists of system and mission reliability measures assessment. We have used at first the rudder control subsystem to build the model. The subsystem is composed, inter alia, of computers and servo-controls that form different lines capable of controlling the rudder. The different components and their dependencies are represented considering the features defined in the meta-model proposed in chapter II. A resulting core model is obtained. It corresponds to the basic part of the stochastic dependability model. The core model is to be composed with a mission dependent model that is best suited to the mission to achieve. We have defined a generic mission profile and have developed a corresponding mission dependent model that is parametric and allows for the analysis of different scenarios based on parameter tuning. The composition of this mission dependent model with the core model corresponds to the stochastic model.

In order to provide quantitative results, the obtained stochastic model is set with illustrative parameters, and examples of numerical results have been obtained. The examples illustrate the valuable role that stochastic model-based dependability can play in the context of aircraft operation. Through more elaborated examples of illustration, we have given insights about the impact of reported changes on the reliability measure for the remaining mission time, that should be useful to adjust the mission if needed.

We have secondly considered the electric supply subsystem. The corresponding core model is developed and composed with the generic mission dependent model previously developed. The obtained results shows higher reliability trends compared to the case of the rudder control subsystem, even while considering failure rates of the same order of magnitude. The electric supply subsystem includes several reconfiguration scenarios that ensure the power delivery in different failure cases.

Considering the composition of the two models, the impact of the electric supply subsystem is significant only when it is degraded (components failure or increased failure rates). This consolidates the idea of considering only the subsystems that are affected by relevant degradations, instead of all the subsystems, when the assessment is subjected to a high time constraint.

Finally, it is worth noting that, in addition to the reliability measure, other criteria such as the cost may be determinant in the selection of the best solution for maintaining failed components. Nevertheless, the solution should comply with the reliability target defined. The solution may involve an adaptation of the mission profile so as to include an airport that is best suited for achievement of the maintenance activities. Modifying a mission profile may involve the reassignment of flights to other aircraft, extending the problem to the fleet level.

Conclusion and Perspectives

Dependability assessment, during aircraft design, is of common practice, and has been very useful for defining appropriate architectures satisfying the dependability requirements set for a given system. The assessments are based on techniques, such as model-based analyses, that capture the essential aspects of the system for forecasting its behavior. Nevertheless, the studies performed during the design phase are led by considerations that cover the whole operational life of the system, and therefore, are based on assumptions addressing the average behavior of the system. Additional assessments in service, considering the current online functional state and operational scenarios, for short-term behavior characterization, become necessary for getting results more suitable to system operation. In the aircraft operation context, the development of efficient means for updating the dependability models and assessment results during the aircraft mission is clearly needed since the stakeholders are searching for a continuous improvement in the service delivery.

In this dissertation, we have addressed the problem of aircraft operational dependability modeling for an assessment while in service, so as to support mission and maintenance planning as well as their successful achievement. The ultimate goal is to contribute to aircraft operability improvement by reducing economical losses due to failures and unsuitable planning for promptly accomplishing maintenance activities. The challenge related to the problem is the complexity of aircraft systems and the fact that the model must be suitable to the systems actual states and the specific operational situation. The assessment is to be done by the aircraft operation team, which is not necessarily familiar with sophisticated modeling formalisms and tools.

The main contribution of this thesis concerns the development of a dependability assessment approach based on stochastic state-space-based models that can be easily updated during the aircraft operation, considering the information related to the current specific situation.

We have identified the system behavior description, the mission profile information, the related requirements, and the maintenance accomplishment information as the relevant types of information to consider in the model. The model adaptation to the situation online is managed by updating the information in the model. The update is the result of an event or a change during the aircraft operation. Indeed after a major change, one has to check if its impact is significant or not. A re-assessment of the operational dependability is consequently required so as to have the up-to-date result. The aircraft will be monitored by prognosis and diagnosis modules that will provide information on the states of the system components and on the trends in their likelihood of failure. The mission profile and maintenance information is also accessible.

Based on the analysis of the major changes that can happen during mission achievement, we have proposed an approach to manage the model update. Accordingly, the dependability model is designed to be a generic and parametric, with default operational information. Thus, the updates consist of tuning the model with the data resulting from the

changes. Model updates should not require any modeling task that the aircraft operator, who will perform the dependability assessment, is not qualified for.

The stochastic dependability model is intended to be developed by the system designer, who is the only entity knowledgeable about the system. The proposed model results from the composition of two main parts: a core model dedicated to the description of system behavior and its system minimal requirements, and a mission dependent model based on the mission profile. To support the construction of these models, we have provided a meta-model that specifies their content at a high-level of abstraction. The meta-model provides common features for the construction of models corresponding to different types of aircraft. Moreover, other modules, like the prognosis and the diagnosis modules that are involved in the assessment framework, will be communicating data for model update. The proposed meta-model features define the kind of data needed.

Following the model content specification, we have investigated two formalisms to implement the model: AltaRica and Stochastic Activity Network (SAN). The AltaRica formalism is widely used in the industrial context in France, including at Airbus for safety analysis, and thus has been selected for the ultimate construction of the operational dependability model. It provides features for fault propagation analysis. It can be used for stochastic model construction, but its supporting tools are still limited for stochastic analyses. The SAN formalism is more known in academia and provides features for complex stochastic behavior modeling. For our modeling problem, the SAN formalism and its supporting tool Möbius have been considered as experimental means for the illustration of the modeling approach. The operational dependability model will be used for quantitative assessment during aircraft operation.

Examining the two formalisms, we have shown that they are interchangeable on the basic features and even most of the model aspects represented using their specific features can be described using the features of the other formalism. We have proposed guidelines for transforming models between the two formalisms whilst respecting as much of the model structure as is possible. These transformation guidelines should be useful in doing comparative analyses while validating quantitative model processing tools for AltaRica. A prototype tool is being developed to support the implementation of the ultimate operational dependability model that will be developed in AltaRica.

Meanwhile, we have used the SAN formalism and its associated tool Möbius to model firstly the A340 rudder control subsystem, and secondly, the electric supply subsystem. We have built the core model corresponding to the subsystems, and composed it with a mission dependent model that has been defined for the illustration purpose. Examples of hypothetical evaluation results have been presented.

Since the proposed approach to the assessment consists in re-evaluating the operational dependability after relevant changes have occurred so as to obtain an up-to-date result, we have investigated the impact of the re-assessment in regard to the previous initial assessment. The results show that, for example, a failure of a primary computer, whose availability is not mandatory for the achievement of a flight, can significantly decrease the reliability of a 7-day mission. The analyses also indicate how to use the assessment results to manage the mission and its associated maintenance activities.

The performed quantitative analyses represent a preliminary experimentation of the approach, which should be extended by other experimentation using the dedicated tool that is under development. Also, as the input data processed by the models may be provided with a given impreciseness, the assessment result will be provided with an indication of the induced uncertainty. A research study [Jacob et al. 2011; Jacob et al. 2012] is being carried out to provide methods for the computation of such uncertainties.

Regarding the future development of the research study carried out, and which is reported in this dissertation, several orientations are possible.

The model is to be developed using the features defined by the meta-model. However, the model construction task is still fully manual, unless means are provided to facilitate the task. Indeed, the model or parts of the model could be generated based on the meta-model. Meta-models are considerably used in model driven engineering for model and software source codes generation. Therefore it will be very useful to investigate the automatic generation of the model based on the meta-model. The operational dependability model construction can be fully integrated in the aircraft development process, and information about the aircraft system could be automatically used to generate the model.

Concerning the transformation of models between AltaRica and SAN, it would be very appropriate to develop a dedicated tool to automate the transformation.

The dependability assessment considering several aircraft represents also an extension of our approach. Indeed, airlines manage a fleet of aircraft and the unavailability of an aircraft can be dealt with by replacing it with other aircraft. Furthermore, the aircraft may share maintenance resources, such as spare components and technicians. It can be valuable to assess the ability to achieve the global airline mission planning with regards to failures and maintenance issues.

We have focused on the aircraft operation context to develop the modeling approach. However, dependability assessment in service could concern any system whose service delivery is subject to a significant attention. Other systems operational context could be targeted and investigated. Automotive and telecommunication systems could also benefit from dependability assessment, in real time, during their operation. The development of generalized modeling features for online assessment will be definitely worth.

The achieved work represents, thus, an important basis for the development of new means that should be essential in ensuring success during aircraft operation. Moreover, the proposed approach could be adapted to the operation of other systems.

Appendix A: The AltaRica model based on the rudder control subsystem

This appendix presents the AltaRica model resulting from the implementation of the modeling approach, based on the rudder control subsystem. The model is developed using the AltaRica data flow version. A mission of 4 flights is considered for simplification purpose. Figure A.1 gives a global overview of the model. The model is composed of a core part and a mission dependent part as proposed in chapter II.

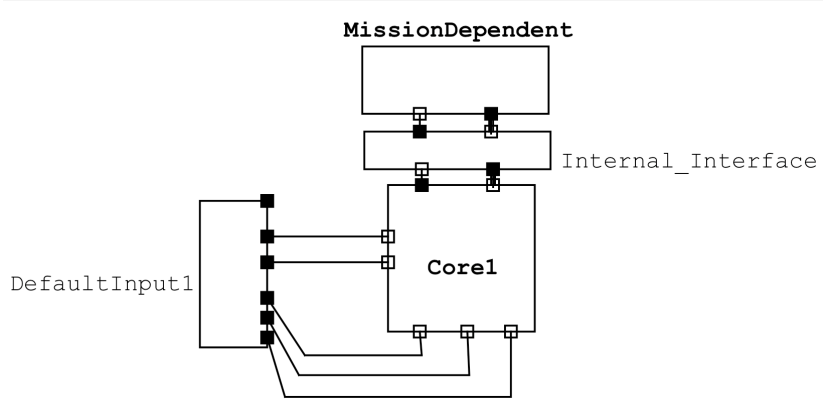


Figure A.1 The AltaRica model global representation

An additional component (*DefaultInput*) is used to set the default inputs of the core model. These are principally the electric and hydraulic power input.

The reminder of this appendix provides the overall AltaRica code of the different nodes composing the model.

○ General data types defined

```
domain DispatchStatus = {no_m, no_o, ok};

domain PeriodType = {flight, none, onground};

domain FlightState = {backToRamp, diverted, done, inFlight, landing,
                      pending, taxingToTakeOff};
```

○ The core model

The AltaRica nodes corresponding to the different system components are developed together with the node that manages the dependency information between them.

A first version of the models of the system components was developed to support the safety assessment of the rudder control subsystem. The interested reader may found more details about the modeling approach for failure modes and failure events in [Bernard et al. 2007]. In the core model, the novelty is the introduction of repair events to enable the connection of the system model with the maintenance model, and the expression of the operational requirements.

In all the nodes, the variable *M* represents maintenance authorization, *flightOn* indicates a flight period, and *Status* represents the component state.

- **The components included in the core model**

```
node PrimaryComputer
       /* A primary computer representation. The node includes an input
       (ServoCtlStatus) that is a feedback about the operational status
       from the servo-control to which it will be connected. This
       feedback is included for safety related mechanism analysis based
       on stepwise simulation. It is use to express the related control
       line status (LineStatus) that is provided as output of the primary
       computer. */
  flow
    Output:bool:out; // send to the servo-control
    ElecPower:bool:in;
    M:bool:in;            // maintenance authorization
    ServoCtlStatus:{failed, ok}:in;
    LineStatus:{failed, ok}:out;
    flightOn:bool:in;    // indication of a flight period
  state
    Status:{failed, ok};  // state of the component
  event
    failure,
    Maintain;
  trans
    ((Status = ok) and flightOn) |- failure -> Status := failed;
    ((Status = failed) and M) |- Maintain -> Status := ok;
  assert
  Output = ( if (((Status = ok) and (ServoCtlStatus = ok)) and ElecPower)
              then true else false),
  LineStatus  =  (if  (((Status  =  ok)  and  (ServoCtlStatus  =  ok))  and
                  ElecPower) then ok else failed);
  init
    Status := ok;
  extern
    law <event failure> = exponential(Lbd_PC);
    law <event Maintain> = Dirac(MT_PC);
edon


node ServoCtl
       /*   A   Servo-control   representation.   The   output   variable
     (StatusFeedBack) represents the operational status of the servo-
     control taking into account the availability of hydraulic power. */
  flow
    input:bool:in; // an input command from a control computer
```

```
    StatusFeedBack:{failed, ok}:out;
    output:bool:out;  // output command to the surface
    hydSource:bool:in;
    M:bool:in;
    flightOn:bool:in;
  state
    status:{failed, ok};
  event
    failure,
    Maintain;
  trans
    ((status = ok) and flightOn) |- failure -> status := failed;
    ((status = failed) and M) |- Maintain -> status := ok;
  assert
    StatusFeedBack = case { ((status = ok) and hydSource) : ok,
                            else failed
                            },
    output = (((status = ok) and input) and hydSource);
  init
    status := ok;
  extern
    law <event failure> = exponential(Lbd_SCtl);
    law <event Maintain> = Dirac(MT_SCtl);
edon


node Cmd_mixer
      /* Used to represent the command input of the servo-control, since
    each servo-control has input from two sources(two control computers
    or a control computer and the backup control module. */
  flow
    input1:bool:in;
    input2:bool:in;
    output:bool:out;
  assert
    output = (input1 or input2);
edon


node PrimaryControl_Off
       /* Used to represent the dependency information necessary in the
     representation of the activation/deactivation scenarios of the
     secondary computer. It takes as input the 3 primary lines status and
     provides an output indicating whether they are all failed or not. */
  flow
    Status1:{failed, ok}:in;
    Status2:{failed, ok}:in;
    Status3:{failed, ok}:in;
    PC_Off:bool:out;
  assert
    PC_Off = (((Status1 = failed) and (Status2 = failed))
                and (Status3 = failed));
edon
```

```
node SecondaryComputer
      /* Secondary computer representation. The output variable
    LineStatus represents the secondary control line status. */
  flow
    Output:bool:out;  // output command to the servo-control
    ElecPower:bool:in;
    ServoCtlStatus:{failed, ok}:in;
    PrimaryControl_Off:bool:in; // true when primary control failed
    M:bool:in;
    flightOn:bool:in;
    LineStatus:{failed, ok}:out;
  state
    Status:{hs, ok};
    defermentExpired:bool;
  event
    active_failure,
    hidden_failure,
    Maintain,
    defermentExpiration;
  trans
  ((Status = ok) and PrimaryControl_Off and flightOn) |- active_failure -
          > Status := hs;
  ((Status  =  ok)  and  (not  PrimaryControl_Off)  and  flightOn)  |-
          hidden_failure -> Status := hs;
  ((Status = hs) and M) |- Maintain -> Status := ok;
  ((Status  =  hs)  and  (not defermentExpired)) |- defermentExpiration ->
          defermentExpired := true;
  assert
   Output = ((((Status = ok) and PrimaryControl_Off) and ElecPower) and
                (ServoCtlStatus = ok)),
    LineStatus = case {
          (((Status = ok) and ElecPower) and (ServoCtlStatus = ok)) : ok,
          else failed
          };
  init
    Status := ok,
    defermentExpired := false;
  extern
    law <event active_failure> = exponential(LbdA_SC);
    law <event hidden_failure> = exponential(LbdH_SC);
    law <event Maintain> = Dirac(MT_SC);
edon


node Backup_Activation
       /* Used to represent the dependency information necessary in the
     representation of the activation/deactivation scenarios of the
     backup components. It takes the 2 primary lines status as input and
     indicates an activation output when both lines are failed. */
  flow
    Input1:bool:in;
    Input2:bool:in;
    Activation:bool:out;
  assert
    Activation = (not (Input1 or Input2));
edon
```

```
node BPS
      /* Backup power supplier. Takes hydraulic power as input and
    provides electric power to the backup control module when it is
    activated. */
  flow
    ElecPower:bool:out;
    M:bool:in;
    HydPower:bool:in;
    BPSActivation:bool:in;
    flightOn:bool:in;
  state
    Status:{failed, ok};
  event
    active_failure,
    hidden_failure,
    Maintain;
  trans
  ((Status = ok) and BPSActivation and flightOn) |- active_failure ->
          Status := failed;
  ((Status = ok) and (not BPSActivation) and flightOn) |- hidden_failure
          -> Status := failed;
  ((Status = failed) and M) |- Maintain -> Status := ok;
  assert
    ElecPower = (if (Status = ok) then BPSActivation else false);
  init
    Status := ok;
  extern
    law <event active_failure> = exponential(LbdA_BPS);
    law <event hidden_failure> = exponential(LbdH_BPS);
    law <event Maintain> = Dirac(MT_BPS);
edon


node BCM
      /* Backup control module. It provides command input to either
    ServoCtrl_B or ServoCtrl_B when it is activated. B_BCOutput and
    Y_BCOutput are respectively the corresponding output. */
  flow
    Y_BCOutput:bool:out;
    B_BCOutput:bool:out;
    Activation_order:bool:in;
    B_BPS_On:bool:in; // indicates whether the corresponding BPS is
    Y_BPS_On:bool:in; // delivering power or not
    M:bool:in;
    flightOn:bool:in;
  state
    status:{failed, ok};
  event
    active_failure,
    hidden_failure,
    Maintain;
  trans
  (flightOn and (status = ok) and Activation_order and (B_BPS_On or
          Y_BPS_On)) |- active_failure -> status := failed;
  (flightOn and (status = ok) and (not (Activation_order and (B_BPS_On or
          Y_BPS_On)))) |- hidden_failure -> status := failed;
```

```
  ((status = failed) and M) |- Maintain -> status := ok;
  assert
    Y_BCOutput = case {
      ((status = ok) and Activation_order and Y_BPS_On) : true,
      else false
    },
    B_BCOutput = case {
    ((status = ok) and Activation_order and (not Y_BPS_On) and B_BPS_On)
          : true,
      else false
    },
  init
    status := ok;
  extern
    law <event active_failure> = exponential(LbdA_BCM);
    law <event hidden_failure> = exponential(LbdH_BCM);
    law <event Maintain> = exponential(MT_BCM);
edon


node Rudder_Req_Fulfillment
    /* Express Min_Sys_R based on the control lines, as expressed in
    chapter IV, section IV.1.2. */
  flow
    PL2:{failed, ok}:in;
    PL3:{failed, ok}:in;
    SL:{failed, ok}:in;
    Min_Sys_R:{fulfilled, not_fulfilled}:out;
    BCL:{failed, ok}:in;
    PL1:{failed, ok}:in;
  assert
    Min_Sys_R = case {
           ( (PL2 = ok) and ((PL1 = ok) or ((PL3 = ok) and (SL = ok)))
          and (BCL = ok)) and ((PL3 = ok) or ((PL1 = ok) and (SL = ok)))
          and ((SL = ok) or ((PL1 = ok) and (PL3 = ok)))
          ) : fulfilled,
      else not_fulfilled
    };
edon



node Rudder_Movement
     /* Represents the possibility to control the Rudder surface, i.e.,
       whether one of the three servo-controls can control the rudder. It
       is used during the analysis of the global functional state of the
       subsystem during the design phase. */
  flow
    Input1:bool:in;
    Input2:bool:in;
    Input3:bool:in;
    Output:bool:out;
  assert
    Output = ((Input1 or Input2) or Input3);
Edon
```

- **The core model composition**

```
  /* Creation and connection of the different components of the core model.
  The synchronizations of the maintenance events are intended to upgrade
them to the global model level so as to synchronize them with the next
scheduled maintenance event authorization that is managed in the mission
dependent model. */
  node CoreModel
    flow
      ElecP2P3:bool:in; // electric power input for P2 and P3
      cp^type:PeriodType:in; // whether the aircraft is on ground or flying
      cp^FPhase:int:in;
      cp^M:int:in;
      cp^CFID:int:in;  // Current flight id
      cp^Interruption:int:in; // whether the mission is interrupted
      ElecS1P1:bool:in; // electric power input for S1 and P31
      hyd_B:bool:in;
      hyd_Y:bool:in;
      hyd_G:bool:in;
      BCLine:{failed, ok}:private; // Backup line status: local variable
      Min_Sys_R:{fulfilled, not_fulfilled}:out;
    event
       /* upgraded maintenance events for synchronization */
      P1Maintenance,
      P2Maintenance,
      P3Maintenance,
      S1Maintenance,
      SCGMaintenance,
      SCBMaintenance,
      SCYMaintenance,
      BPS_BMaintenance,
      BPS_YMaintenance,
      BCMMaintenance;
    sub
      Req_Fulfillment:Rudder_Req_Fulfillment;
      BPSActivation:Backup_Activation;
      ActivateBCM: Backup_Activation;
      or1:Cmd_mixer;
      or3:Cmd_mixer;
      or2:Cmd_mixer;
      BPS_Y:BPS;
      BPS_B:BPS;
      BCM:BCM;
      S1Activation: PrimaryControl_Off;
      ServoCtl_Y:ServoCtl;
      ServoCtl_B:ServoCtl;
      ServoCtl_G:ServoCtl;
      S1:SecondaryComputer;
      P3:PrimaryComputer;
      P2:PrimaryComputer;
      P1:PrimaryComputer;
      Rudder:Rudder_Movement;
    assert
      BCLine = case {
        ((BCM.status = ok) or (BPS_B.Status = ok) or (BPS_Y.Status = ok)) :
```

```
    ok,
   else failed
},
Min_Sys_R = Req_Fulfillment.Min_Sys_R,
Req_Fulfillment.PL2 = P2.LineStatus,
Req_Fulfillment.PL3 = P3.LineStatus,
Req_Fulfillment.SL = S1.LineStatus,
Req_Fulfillment.BCL = BCLine,
Req_Fulfillment.PL1 = P1.LineStatus,
BPSActivation.input1 = S1.Output,
BPSActivation.input2 = P1.Output,
ActivateBCM.Input1 = P3.Output,
ActivateBCM.Input2 = P2.Output,
or1.input1 = S1.Output,
or1.input2 = P1.Output,
or3.input1 = P3.Output,
or3.input2 = BCM.Y_BCOutput,
or2.input1 = P2.Output,
or2.input2 = BCM.B_BCOutput,
BPS_Y.M = (cp^M > 0),
BPS_Y.HydPower = hyd_Y,
BPS_Y.BPSActivation = BPSActivation.output,
BPS_Y.flightOn = (cp^type = flight),
BPS_B.M = (cp^M > 0),
BPS_B.HydPower = hyd_B,
BPS_B.BPSActivation = BPSActivation.output,
BPS_B.flightOn = (cp^type = flight),
BCM.Activation_order = ActivateBCM.Activation,
BCM.B_BPS_On = BPS_B.ElecPower,
BCM.Y_BPS_On = BPS_Y.ElecPower,
BCM.M = (cp^M > 0),
BCM.flightOn = (cp^type = flight),
S1Activation.Status1 = P1.LineStatus,
S1Activation.Status2 = P2.LineStatus,
S1Activation.Status3 = P3.LineStatus,
ServoCtl_Y.input = or3.output,
ServoCtl_Y.hydSource = hyd_Y,
ServoCtl_Y.M = (cp^M > 0),
ServoCtl_Y.flightOn = (cp^type = flight),
ServoCtl_B.input = or2.output,
ServoCtl_B.hydSource = hyd_B,
ServoCtl_B.M = (cp^M > 0),
ServoCtl_B.flightOn = (cp^type = flight),
ServoCtl_G.input = or1.output,
ServoCtl_G.hydSource = hyd_G,
ServoCtl_G.M = (cp^M > 0),
ServoCtl_G.flightOn = (cp^type = flight),
S1.ElecPower = ElecS1P1,
S1.ServoCtlStatus = ServoCtl_G.StatusFeedBack,
S1.PrimaryControl_Off = S1Activation.PC_Off,
S1.M = ((cp^M = 14) or (cp^M = 1)),
S1.flightOn = (cp^type = flight),
P3.ElecPower = ElecP2P3,
P3.M = ((cp^M = 13) or (cp^M = 1)),
P3.ServoCtlStatus = ServoCtl_Y.StatusFeedBack,
P3.flightOn = (cp^type = flight),
```

```
    P2.ElecPower = ElecP2P3,
    P2.M = (cp^M > 0),
    P2.ServoCtlStatus = ServoCtl_B.StatusFeedBack,
    P2.flightOn = (cp^type = flight),
    P1.ElecPower = ElecS1P1,
    P1.M = ((cp^M = 11) or (cp^M = 1)),
    P1.ServoCtlStatus = ServoCtl_G.StatusFeedBack,
    P1.flightOn = (cp^type = flight),
    Rudder.Input1 = ServoCtl_G.output,
    Rudder.Input2 = ServoCtl_B.output,
    Rudder.Input3 = ServoCtl_Y.output;
  sync
    <P1Maintenance , P1.Maintain>,
    <P2Maintenance , P2.Maintain>,
    <P3Maintenance , P3.Maintain>,
    <S1Maintenance , S1.Maintain>,
    <SCGMaintenance , ServoCtl_G.Maintain>,
    <SCBMaintenance , ServoCtl_B.Maintain>,
    <SCYMaintenance , ServoCtl_Y.Maintain>,
    <BPS_BMaintenance , BPS_B.Maintain>,
    <BPS_YMaintenance , BPS_Y.Maintain>,
    <BCMMaintenance , BCM.Maintain>;
edon
```

○ The mission dependent model

The mission dependent model represents the sequence of flights to achieve, as a sequence of CompleteFlight instances. CompleteFlight is composed of an instance of GroundPeriod and an instance of a FlightPeriod. It also includes a component that manages the order of maintaining the failed components during scheduled maintenance. Four instances of CompleteFlight are considered here for simplification reason.

The mission dependent model also includes a component *Filter* that is used to synthesize, from the instances of CompleteFlight, the current mission achievement information (instance *CP* of CurrentProcess) that is necessary in the core model.

▪ The mission dependent model components

```
node Ground_Period  /* Represent Ground Periods*/
  flow
    start:bool:in; // triggers the ground period achievement process
    Maintenance_Period:{no, scheduled, unscheduled}:out;
    GA_done:bool:out; // indicates the end of the ground period
    DR:DispatchStatus:in;
    DelayedOrCancelled:bool:out;
    Running:bool:out; // indicates whether it is being achieved

 state
    Unscheduled_maintenance:{OFF, ON};
    Scheduled_maintenance:{OFF, ON};
    execState:{Delay_or_Cancellation, Ground_Preparation, done, pending,
              pending_Departure};
```

```
  Ready:bool;
  Dispatchability:bool;
event
  begin,
  Next_Flight_preparation,
  Max_tolerated_time,
  OnGround_duration,
  SM_Time,
  Require_maintenance,
  departure;
trans
(start and (execState = pending)) |- begin -> execState :=
        Ground_Preparation, Scheduled_maintenance := ON;
(Scheduled_maintenance = ON) |- SM_Time -> Scheduled_maintenance :=
        OFF, Dispatchability := true;
(Dispatchability and (DR = no_m)) |- Require_maintenance ->
        Unscheduled_maintenance := ON;
(((not (execState = Delay_or_Cancellation)) and Dispatchability) and
        (DR = ok)) |- Next_Flight_preparation -> Ready := true,
        Unscheduled_maintenance := OFF, Dispatchability := false;
(execState = Ground_Preparation) |- OnGround_duration -> execState :=
        pending_Departure;
(execState = pending_Departure) |- Max_tolerated_time -> execState :=
        Delay_or_Cancellation;
((execState = pending_Departure) and Ready) |- departure ->
        execState := done;
assert
  Maintenance_Period = case {
    (Scheduled_maintenance = ON) : scheduled,
    (Unscheduled_maintenance = ON) : unscheduled,
    else no
  },

  GA_done = case {
    (execState = done) : true,
    else false
  },

  DelayedOrCancelled = case {
    (execState = Delay_or_Cancellation) : true,
    else false
  },

  Running = (not ((execState = pending) or (execState = done)));

init
  Unscheduled_maintenance := OFF,
  Scheduled_maintenance := OFF,
  execState := pending,
  Ready := false,
  Dispatchability := false;
extern
  law <event begin> = Dirac(0.0);
  law <event OnGround_duration > = Dirac(gpd);
  law <event SM_Time > = Dirac(SMT);
  law <event Next_Flight_preparation > = Dirac(oad);
```

```
    law <event Require_maintenance> = Dirac(0.0) ;
    law <event Max_tolerated_time > = Dirac(mtt);
    law <event departure> = Dirac(0.0);
edon


node Prioritization
     /* Used to allow the maintenance activities according to an order
       that must be respected during scheduled maintenance. 3 components
       to be maintained are considered for illustration purpose. Item1,
       Item2 and Item3 represent the id of the components. Order: Item1,
       then Item2, and then Item3. The event activityDone (common cause
       sync) is synchronized with the maintenance events in the core model
       so as to set the next item to maintain. */
  flow
    ToRepair:int:out;
    NextItem:int:private;
    MPeriod:{no, scheduled, unscheduled}:in;
  state
    Item1:int;
    Item2:int;
    Item3:int;
  event
    activityDone;
  trans
  (MPeriod = scheduled) |- activityDone -> Item3 := (if ((Item1 = 0) and
          (Item2 = 0)) then 0 else Item3), Item2 := (if (Item1 = 0) then
          0 else Item2), Item1 := 0;
  assert
    NextItem = case {
      (Item1 # 0) : Item1,
      (Item2 # 0) : Item2,
      (Item3 # 0) : Item3,
      else 2
    },
    ToRepair = case {
      (MPeriod = scheduled) : NextItem,
      (MPeriod = unscheduled) : 1,
      else 0
    };

  init
    Item1 := 0,
    Item2 := 0,
    Item3 := 0;
edon



node Flight_Period /* Represent Flight Periods*/
  flow
    F_end:bool:out;  // indicates the completion of the flight
    start:bool:in; // trigger the achievement process
    execState:FlightState:out;
    diversionCondition:bool:in;
    abortCondition:bool:in;
```

```
  state
    In_Flight:{OFF, ON};
    Landing:{OFF, ON};
    Taxing_to_TakkeOff:{OFF, ON};
    Diverted:bool;
    Back_To_Ramp:bool;
    FStatus:FlightState;
  event
    To_air,
    Flying,
    ToGround,
    departure,
    diversion,
    Abort;
  trans
  (start and (FStatus = pending)) |- departure -> Taxing_to_TakkeOff :=
          ON, FStatus := taxingToTakeOff;
  ((Taxing_to_TakkeOff = ON) and abortCondition) |- Abort -> FStatus :=
          backToRamp, Taxing_to_TakkeOff := OFF;
  ((Taxing_to_TakkeOff = ON) and (not abortCondition)) |- To_air ->
          Taxing_to_TakkeOff := OFF, In_Flight := ON, FStatus :=
          inFlight;
  ((In_Flight = ON) and diversionCondition) |- diversion -> FStatus :=
          diverted, In_Flight := OFF;
  ((In_Flight = ON) and (not diversionCondition)) |- Flying -> In_Flight
          := OFF, Landing := ON, FStatus := landing;
  (Landing = ON) |- ToGround -> Landing := OFF, FStatus := done;
  assert
    F_end = (FStatus = done),
    execState = FStatus;
  init
    In_Flight := OFF,
    Landing := OFF,
    Taxing_to_TakkeOff := OFF,
    Diverted := false,
    Back_To_Ramp := false,
    FStatus := pending;
  extern
    law <event To_air> = Dirac(ttd);
    law <event Flying> = Dirac(ifd);
    law <event ToGround> = Dirac(ld);
    law <event departure> = Dirac(0.0);
    law <event diversion> = Dirac(0.0);
    law <event Abort> = Dirac(0.0);
edon



node CompleteFlight
      /* Represents the complete flight achievement process composed of a
    ground period and a flight period. It also includes an instance of
    Prioritization that manages the scheduled maintenance activities. */
  flow
    Req:DispatchStatus:in;
    next:bool:out; // indicates its completion to the next instance
    perform:bool:in; // trigger the achievement process
```

```
    execInfo^Maintain:int:out; // id of the component allowed to undergo
                               // maintenance, 0 if none
    execInfo^Phase:int:out; // current flight phase, 0 if none
    execInfo^Interruption:int:out;
    execInfo^ExecState:PeriodType:out;
  event
    NextScheduledItem;
  sub
    Prioritization:Prioritization;
    Ground_Period:Ground_Period;
    Flight_Period:Flight_Period;
  assert
    next = Flight_Period.F_end,
    execInfo^Maintain = Prioritization.ToRepair,
    execInfo^Phase = case {
      (Flight_Period.execState = taxingToTakeOff) : 1,
      (Flight_Period.execState = inFlight) : 2,
      (Flight_Period.execState = landing) : 3,
      else 0
    },
    execInfo^Interruption = case {
      Ground_Period.DelayedOrCancelled : 1,
      (Flight_Period.execState = backToRamp) : 2,
      (Flight_Period.execState = diverted) : 3,
      else 0
    },
    execInfo^ExecState = case {
          Ground_Period.Running : onground,
         ((Flight_Period.execState # pending) and
                (Flight_Period.execState # done)) : flight,
         else none
    },
    Prioritization.MPeriod = Ground_Period.Maintenance_Period,
    Ground_Period.start = perform,
    Ground_Period.DR = Req,
    Flight_Period.start = Ground_Period.GA_done,
    Flight_Period.diversionCondition = (Req # ok),
    Flight_Period.abortCondition = (Req # ok);
  sync
    <NextScheduledItem , Prioritization.activityDone>;//upgrade the event
             //to the mission dependent model level
edon



node Filter
      /* Synthesizes mission achievement information, from the 4
    instances of CompleteFlight that compose the mission profile, as an
    instance (CP) of CurrentProcess, to be used in the core model. */
  flow
    F4^Maintain:int:in;
    F4^Phase:int:in;
    F4^Interruption:int:in;
    F4^ExecState:PeriodType:in;
    CP^type:PeriodType:out;
```

```
    CP^FPhase:int:out;
    CP^M:int:out;
    CP^CFID:int:out;   // current flight id
    CP^Interruption:int:out;
    F3^Maintain:int:in;
    F3^Phase:int:in;
    F3^Interruption:int:in;
    F3^ExecState:PeriodType:in;
    F2^Maintain:int:in;
    F2^Phase:int:in;
    F2^Interruption:int:in;
    F2^ExecState:PeriodType:in;
    F1^Maintain:int:in;
    F1^Phase:int:in;
    F1^Interruption:int:in;
    F1^ExecState:PeriodType:in;
  assert
    CP^type = (if (F1^ExecState # none) then F1^ExecState
                else (if (F2^ExecState # none) then F2^ExecState
                else (if (F3^ExecState # none) then F3^ExecState
                else (if (F4^ExecState # none) then F4^ExecState
                else none)))),
    CP^FPhase = (if (F1^ExecState # none) then F1^Phase
                else (if (F2^ExecState # none) then F2^Phase
                else (if (F3^ExecState # none) then F3^Phase
                else (if (F4^ExecState # none) then F4^Phase else 0)))),
    CP^M = (if (F1^ExecState # none) then F1^Maintain
                else (if (F2^ExecState # none) then F2^Maintain
                else (if (F3^ExecState # none) then F3^Maintain
                else (if (F4^ExecState # none) then F4^Maintain
                else 0)))),
    CP^CFID = (if (F1^ExecState # none) then 1
                else (if (F2^ExecState # none) then 2
                else (if (F3^ExecState # none) then 3
                else (if (F4^ExecState # none) then 4
                else 0)))),
    CP^Interruption = (if (F1^ExecState # none) then F1^Interruption
                else (if (F2^ExecState # none) then F2^Interruption
                else (if (F3^ExecState # none) then F3^Interruption
                else (if (F4^ExecState # none) then F4^Interruption
                else 0))));
edon
```

```
node StartMission /* trigger the mission achievement process */
  flow
    go:bool:out;
  assert
    go = true;
edon
```

- **The mission dependent model composition**

```
node MissionDependent
      /* Creation and sequencing of the 4 instances of CompleteFlight.
     Connection with Filter1 that derives the mission achievement
     information that should be shared with the core model. */
  flow
    CP^type:PeriodType:out;
    CP^FPhase:int:out;
    CP^M:int:out;
    CP^CFID:int:out;
    CP^Interruption:int:out;
    Req:DispatchStatus:in;
  event
    ScheduledItemDone;
  sub
    Start:StartMission;
    Filter1:Filter;
    CompleteFlight4:CompleteFlight;
    CompleteFlight3:CompleteFlight;
    CompleteFlight2:CompleteFlight;
    CompleteFlight1:CompleteFlight;
  assert
    CP^type = Filter1.CP^type,
    CP^FPhase = Filter1.CP^FPhase,
    CP^M = Filter1.CP^M,
    CP^CFID = Filter1.CP^CFID,
    CP^Interruption = Filter1.CP^Interruption,
    Filter1.F4^Maintain = CompleteFlight4.execInfo^Maintain,
    Filter1.F4^Phase = CompleteFlight4.execInfo^Phase,
    Filter1.F4^Interruption = CompleteFlight4.execInfo^Interruption,
    Filter1.F4^ExecState = CompleteFlight4.execInfo^ExecState,
    Filter1.F3^Maintain = CompleteFlight3.execInfo^Maintain,
    Filter1.F3^Phase = CompleteFlight3.execInfo^Phase,
    Filter1.F3^Interruption = CompleteFlight3.execInfo^Interruption,
    Filter1.F3^ExecState = CompleteFlight3.execInfo^ExecState,
    Filter1.F2^Maintain = CompleteFlight2.execInfo^Maintain,
    Filter1.F2^Phase = CompleteFlight2.execInfo^Phase,
    Filter1.F2^Interruption = CompleteFlight2.execInfo^Interruption,
    Filter1.F2^ExecState = CompleteFlight2.execInfo^ExecState,
    Filter1.F1^Maintain = CompleteFlight1.execInfo^Maintain,
    Filter1.F1^Phase = CompleteFlight1.execInfo^Phase,
    Filter1.F1^Interruption = CompleteFlight1.execInfo^Interruption,
    Filter1.F1^ExecState = CompleteFlight1.execInfo^ExecState,
    CompleteFlight4.Req = Req,
    CompleteFlight4.perform = CompleteFlight3.next,
    CompleteFlight3.Req = Req,
    CompleteFlight3.perform = CompleteFlight2.next,
    CompleteFlight2.Req = Req,
    CompleteFlight2.perform = CompleteFlight1.next,
    CompleteFlight1.Req = Req,
    CompleteFlight1.perform = Start.go;
  sync
   /* Broadcast of a component maintenance completion so that the next
      system component to maintain can be set.*/
```

```
  <ScheduledItemDone:         CompleteFlight1.NextScheduledItem         or
     CompleteFlight2.NextScheduledItem                                  or
     CompleteFlight3.NextScheduledItem                                  or
     CompleteFlight4.NextScheduledItem>;
edon
```

- **The internal Interface node**

```
node InternalInterface
       /* Interface between the core and the mission dependent models. */
  flow
    Min_Sys_R:{fulfilled, not_fulfilled}:in;
    DR:DispatchStatus:out;
    CP_out^type:PeriodType:out;
    CP_out^FPhase:int:out;
    CP_out^M:int:out;
    CP_out^CFID:int:out;
    CP_out^Interruption:int:out;
    CP_in^type:PeriodType:in;
    CP_in^FPhase:int:in;
    CP_in^M:int:in;
    CP_in^CFID:int:in;
    CP_in^Interruption:int:in;
  assert
    DR = case {
      (Min_Sys_R = fulfilled) : ok,
      else no_m
    },
    CP_out^type = CP_in^type,
    CP_out^FPhase = CP_in^FPhase,
    CP_out^M = CP_in^M,
    CP_out^CFID = CP_in^CFID,
    CP_out^Interruption = CP_in^Interruption;
Edon
```

- **Component to set the default of the core model**

```
node DefaultInput
       /* Set electric and hydraulic power input of the core model,
     considered always available, since their corresponding systems
     behaviors are not described here. It also provides default mission
     information input to the core model when it is analyzed alone */
  flow
    hyd_B:bool:out;
    hyd_Y:bool:out;
    hyd_G:bool:out;
    ElecP2P3:bool:out;
    ElecS1P1:bool:out;
    CP^type:PeriodType:out;
    CP^FPhase:int:out;
    CP^M:int:out;
```

```
    CP^CFID:int:out;
    CP^Interruption:int:out;
  assert
    hyd_B = true,
    hyd_Y = true,
    hyd_G = true,
    ElecP2P3 = true,
    ElecS1P1 = true,
    CP^type = flight,
    CP^FPhase = 0,
    CP^M = 0,
    CP^CFID = 0,
    CP^Interruption = 0;
Edon
```

- **The model composition – main node**

```
node main
/* The overall model composition */
  event
    P1Maintenance,
    P2Maintenance,
    P3Maintenance,
    S1Maintenance,
    SCGMaintenance,
    SCBMaintenance,
    SCYMaintenance,
    BPS_YMaintenance,
    BPS_BMaintenance,
    BCMMaintenance;
  sub
    /* Main components of the overall model as shown in Figure A. */
    Compositon:InternalInterface;
    MissionDependent:MissionDependent;
    Core1:CoreModel;
    DefaultInput1:DefaultInput;
  assert
    Compositon.Min_Sys_R = Core1.Min_Sys_R,
    Compositon.CP_in^type = MissionDependent.CP^type,
    Compositon.CP_in^FPhase = MissionDependent.CP^FPhase,
    Compositon.CP_in^M = MissionDependent.CP^M,
    Compositon.CP_in^CFID = MissionDependent.CP^CFID,
    Compositon.CP_in^Interruption = MissionDependent.CP^Interruption,
    MissionDependent.Req = Compositon.DR,
    Core1.ElecP2P3 = DefaultInput1.ElecP2P3,
    Core1.cp^type = Compositon.CP_out^type,
    Core1.cp^FPhase = Compositon.CP_out^FPhase,
    Core1.cp^M = Compositon.CP_out^M,
    Core1.cp^CFID = Compositon.CP_out^CFID,
    Core1.cp^Interruption = Compositon.CP_out^Interruption,
    Core1.ElecS1P1 = DefaultInput1.ElecS1P1,
    Core1.hyd_B = DefaultInput1.hyd_B,
    Core1.hyd_Y = DefaultInput1.hyd_Y,
    Core1.hyd_G = DefaultInput1.hyd_G;
```

```
  sync
    <P1Maintenance : Core1.P1Maintenance or
                     MissionDependent.ScheduledItemDone>,

    <P2Maintenance : Core1.P2Maintenance or
                     MissionDependent.ScheduledItemDone>,
    <Core1.P2Maintenance>,
    <MissionDependent.ScheduledItemDone>,
    <P3Maintenance : Core1.P3Maintenance or
                     MissionDependent.ScheduledItemDone>,
    <S1Maintenance : Core1.S1Maintenance or
                     MissionDependent.ScheduledItemDone>,
    <SCGMaintenance : Core1.SCGMaintenance or
                     MissionDependent.ScheduledItemDone>,
    <Core1.SCGMaintenance>,
    <SCBMaintenance : Core1.SCBMaintenance or
                     MissionDependent.ScheduledItemDone>,
    <Core1.SCBMaintenance>,
    <SCYMaintenance : Core1.SCYMaintenance or
                     MissionDependent.ScheduledItemDone>,
    <Core1.SCYMaintenance>,
    <BPS_YMaintenance : Core1.BPS_YMaintenance or
                     MissionDependent.ScheduledItemDone>,
    <Core1.BPS_YMaintenance>,
    <BPS_BMaintenance : Core1.BPS_BMaintenance or
                     MissionDependent.ScheduledItemDone>,
    <Core1.BPS_BMaintenance>,
    <BCMMaintenance : Core1.BCMMaintenance or
                     MissionDependent.ScheduledItemDone>,
    <Core1.BCMMaintenance>;
extern
  ccf <event P2Maintenance> = true;
  withCCF <event Core1.P2Maintenance> = {<event P2Maintenance>};
  withCCF <event MissionDependent.ScheduledItemDone> =
          { <event BCMMaintenance>, <event BPS_BMaintenance>,
            <event BPS_YMaintenance>, <event SCYMaintenance>,
            <event SCBMaintenance>, <event SCGMaintenance>,
            <event P2Maintenance>};
  ccf <event SCGMaintenance> = true;
  withCCF <event Core1.SCGMaintenance> = {<event SCGMaintenance>};
  ccf <event SCBMaintenance> = true;
  withCCF <event Core1.SCBMaintenance> = {<event SCBMaintenance>};
  ccf <event SCYMaintenance> = true;
  withCCF <event Core1.SCYMaintenance> = {<event SCYMaintenance>};
  ccf <event BPS_YMaintenance> = true;
  withCCF <event Core1.BPS_YMaintenance> = {<event BPS_YMaintenance>};
  ccf <event BPS_BMaintenance> = true;
  withCCF <event Core1.BPS_BMaintenance> = {<event BPS_BMaintenance>};
  ccf <event BCMMaintenance> = true;
  withCCF <event Core1.BCMMaintenance> = {<event BCMMaintenance>};
edon
```

Appendix B: Résumé en Français

Sommaire

Introduction

Le transport aérien est devenu un moyen de déplacement usuel pour un nombre de plus en plus croisant de personnes. Avec cet accroissement d'intérêt et le besoin de compétitivité dans le secteur aéronautique, il est essentiel pour les avionneurs et les compagnies aériennes de développer des moyens pouvant aider à une exploitation optimale des avions. De nouvelles approches doivent être développées pour aider dans la réalisation des opérations, de façon à atteindre et à maintenir un niveau élevé de prestation de services, réduire les interruptions, et éviter les pertes économiques dues à l'indisponibilité des avions et à l'insatisfaction des clients.

Le projet @MOST (Airbus Operations Maintenance Solutions & Technologies) a été mis en place par Airbus pour examiner de nouvelles approches, à l'opération et à la maintenance des avions, qui peuvent aider à une meilleure exploitation des avions et à la réduction des coûts. Le projet a pour principal objectif le développement de moyens de prévision assurant une gestion efficace de la réalisation des opérations, et capable d'aider à effectuer les activités de maintenance juste au bon moment.

Dans le contexte de @MOST, le projet DIANA (Decision Impact ANAlysis) vise à développer un cadre d'évaluation de sûreté de fonctionnement, basé sur les modèles, qui peut être utilisé pour analyser et assurer la fiabilité opérationnelle des avions, et ainsi, assurer le succès et l'efficacité dans la réalisation des opérations vis-à-vis des perturbations liées aux défaillances. Pour cela, les approches actuelles d'analyse de sûreté de fonctionnement en phase de conception des avions ne peuvent être considérées comme suffisantes. L'analyse en phase de conception ne concerne généralement que l'évaluation de la sécurité innocuité et, lorsque la fiabilité opérationnelle est abordée, l'analyse porte sur le gain en fiabilité perceptible d'une technologie à une autre. En outre, dans le cadre de DIANA, l'évaluation de la sûreté de fonctionnement aura lieu lorsque l'avion est en service, de sorte à prendre en compte les informations courantes pour pouvoir obtenir des résultats de fiabilité adaptés à la situation. A notre connaissance, l'évaluation de la sûreté de fonctionnement basée sur les modèles, en temps réel, en cours d'opération, n'a pas encore été beaucoup abordée. Pourtant, le constant besoin en terme de meilleur rendement et le caractère évolutif des systèmes actuels requièrent l'utilisation de données plus précises en phase opérationnelle, pour obtenir une évaluation adaptée, afin d'aider à faire de meilleurs choix.

Cette thèse fait partie du projet DIANA et concerne l'élaboration du cadre d'évaluation de sûreté de fonctionnement basé sur les modèles. Elle se concentre en particulier sur la construction du modèle et son utilisation pendant la réalisation des missions. L'objectif de la thèse est donc la mise en place du modèle de sûreté de fonctionnement sur lequel peut se baser l'évaluation de la sûreté de fonctionnement en opération.

Pour identifier le rôle que peut jouer l'évaluation de sûreté de fonctionnement lors de l'opération des avions, et comprendre son importance dans la planification de la maintenance, il est nécessaire d'avoir un aperçu des différentes façons dont les opérations d'avion peuvent être réalisées, et connaitre les principes de base des pratiques de maintenance. La construction du modèle de sûreté de fonctionnement nécessite, en plus des techniques d'analyse de sûreté de fonctionnement, d'avoir un aperçu sur l'opération et la maintenance des avions, ainsi que sur la façon dont elles sont organisées. Les

connaissances de base pour pouvoir effectuer le travail sont données dans le chapitre I. Nous présentons les concepts et méthodes d'analyse de sûreté de fonctionnement, donnons un aperçu sur l'opération des avions et présentons les principes sous-tendant les politiques de maintenance des avions. Nous présentons également une revue des travaux visant à améliorer l'opérabilité des avions, des travaux de modélisation qui contribuent au renforcement de la sécurité innocuité, aux travaux portant sur la gestion des perturbations opérationnelles, en passant par les études s'appuyant sur la conception des systèmes et la planification de la maintenance pour améliorer la fiabilité opérationnelle. L'objectif est de mettre en exergue la place que l'évaluation de sûreté de fonctionnement basée sur des modèles aura dans le processus d'amélioration de l'opérabilité des avions et de réduction des interruptions opérationnelles.

Le champ d'application de l'évaluation de sûreté de fonctionnement basée sur des modèles est présenté à la fin du chapitre I. Comme différents types de missions avec des exigences différentes peuvent être attribués à un avion donné, l'outil d'évaluation de sûreté de fonctionnement donnera des estimations de la probabilité de réussir les missions ou le risque de rencontrer une situation indésirable, en considérant les informations opérationnelles courantes. L'évaluation porte sur l'adaptation du profil de la mission et des activités de maintenance planifiées à l'état opérationnel courant de l'avion. Les missions et les planifications de maintenance peuvent être ajustées en utilisant l'évaluation. L'évaluation peut être effectuée suite à l'occurrence d'événements majeurs en opération.

Le deuxième chapitre vise à présenter l'approche de modélisation mise en place. Les difficultés liées à la mise en place du modèle y sont traitées. Outre la complexité des systèmes avion, la prise en compte des informations en cours d'opération, afin d'avoir un modèle adapté à la situation courante, doit aussi être traitée. Par ailleurs, comme différents types d'avions existent, et afin d'harmoniser la construction des modèles correspondants, il convient d'établir une base commune pour leur construction. Nous avons établi la base commune pour la construction des modèles en utilisant la méta-modélisation.

L'approche de modélisation est développée en se basant sur les techniques de modélisation stochastique à espace d'état. Ainsi, tout formalisme de modélisation stochastique basé sur les techniques à espace d'état peut être utilisé pour construire le modèle. Le langage AltaRica, utilisé par Airbus pour développer des modèles pour des analyses sécurité innocuité, a donc été considéré pour la construction du modèle. Cependant, étant donné qu'AltaRica et ses actuels outils de traitement ont été principalement développés pour faire des analyses qualitatives, nous avons également choisi les réseaux d'activités stochastiques (SAN) pour mettre en œuvre les aspects d'analyse quantitative. Le projet DIANA envisage de développer un outil pour prendre en compte le traitement du modèle en AltaRica pour obtenir des résultats quantitatifs. Nous présentons une analyse comparative des deux formalismes dans le chapitre III. Des méthodes de transformation de modèles entre les deux formalismes sont également examinées.

Les formalismes sont utilisés pour développer des cas d'études basés sur des sous-systèmes d'avion. Nous présentons ces cas d'études, développés en utilisant le formalisme SAN, dans le chapitre IV. La mise en œuvre de l'approche de modélisation à l'aide d'AltaRica est donnée en annexe.

Nous avons considéré premièrement le sous-système de contrôle de la gouverne de direction. En se basant sur les exigences, nous avons analysé différents scénarios de défaillances, de changements de distribution de défaillance à cause de l'usure, et de modifications de profil de mission. Nous avons ensuite considéré le sous-système d'alimentation électrique pour analyser son impact sur la réussite des missions.

Les résultats de l'évaluation montrent que les événements, comme les défaillances qui n'empêchent pas l'autorisation d'un vol, peuvent avoir un impact significatif sur la probabilité de succès d'une mission.

Nous avons conclu en rappelant la problématique abordée, et nos principales réalisations. De possibles orientations, pour le futur développement de la recherche menée, sont également présentées.

Nous présentons dans ce qui suit l'essentiel du contenu de chaque chapitre.

I Contexte et Notions de Base

Ce chapitre présente le contexte général du travail de thèse. La thèse porte sur l'évaluation de la sûreté de fonctionnement dans le contexte de l'opération des avions, en utilisant des modèles. Les concepts et les techniques d'analyse de sûreté de fonctionnement sont présentés, de même que les principes de base de l'opération d'un avion, de sorte à donner une base de connaissances appropriée au sujet et à mettre en exergue le champ d'application de l'évaluation de la sûreté de fonctionnement. L'évaluation cible les défaillances et les problèmes de maintenance qui peuvent entraîner des perturbations d'opération. Les principes fondamentaux de la maintenance des avions sont aussi présentés.

I.1 Concepts et méthode d'évaluation de la sûreté de fonctionnement

Les concepts de la sûreté de fonctionnement sont définis dans [Avizienis et al. 2000; Laprie et al. 1995]. La sûreté de fonctionnement est définie comme la propriété d'un système permettant à ses utilisateurs de placer une confiance justifiée dans le service qu'il leur délivre. Elle englobe trois classes de concepts : les *attributs*, les *entraves* et les *moyens* pour atteindre. Les principaux attributs sont la *fiabilité*, la *disponibilité*, la *sécurité*, l'*intégrité*, et la *maintenabilité*. Les entraves se déclinent en *fautes*, *erreurs* et *défaillances*. Les moyens sont classés en quatre grandes catégories : la *prévention des fautes*, la *tolérance aux fautes*, l'*élimination des fautes* et la *prévision des fautes*.

La thèse concerne la prévision des fautes au cours de l'utilisation des systèmes, en particulier dans le cadre de l'opération des avions. La prévision des fautes est réalisée en effectuant une évaluation du comportement du système par rapport à l'occurrence et à l'activation des fautes. L'évaluation peut consister en une analyse qualitative ou quantitative, et peut être effectuée en utilisant un modèle. La construction du modèle est de nos jours effectuée en utilisant des formalismes de haut niveau permettant la prise en compte de comportements complexes. Ce sont les formalismes AltaRica [Arnold et al.

1999; LABRI 2010] et SAN [Meyer et al. 1985; Sanders and Meyer 2001] qui ont été considérés dans le cadre de cette thèse.

L'approche développée dans cette thèse concerne la construction de modèles en considérant les systèmes avion, leur opération et leur maintenance.

I.2 Les systèmes avion, réalisation des opérations et maintenance

Le travail de modélisation fait intervenir des connaissances sur les systèmes avion, sur la planification et la réalisation des opérations, et sur la maintenance en ligne.

I.2.1 Les systèmes avion

Un avion est constitué de plusieurs sous-systèmes, qui assurent ses fonctions de haut niveau. Les sous-systèmes sont conçus de manière à éviter les défaillances catastrophiques. En général, leur conception détaillée présente plusieurs redondances et des scénarios de reconfiguration complexes qui visent à éviter les défaillances catastrophiques et à assurer la délivrance continue des fonctions. Les sous-systèmes sont constitués de composants, remplaçables en service (Line Replaceable Unit, LRU), qui interagissent pour assurer les fonctions nécessaires à la réalisation des misions.

I.2.2 Réalisation des opérations

Une mission d'avion consiste en une liste prédéfinie de vols à effectuer sous certaines conditions opérationnelles et de maintenance. La réalisation de la mission est telle que chaque vol est suivi par une escale où l'avion est apprêté pour le prochain vol. A chaque escale, l'avion est inspecté et les anomalies observées lors du vol précédent sont examinées. Si une défaillance est détectée, une décision doit être prise quant à l'aptitude à effectuer le prochain vol. Les agents se réfèrent à un document (appelé Minimum Equipment List, MEL) où les composants sont répertoriés avec le statut Go, Go-If ou No-Go.

Le statut Go représente le cas où l'avion peut voler avec le composant défaillant. Pour le statut Go-If, le vol peut être effectué à condition qu'un certain nombre d'autres composants soient opérationnels et que certaines procédures opérationnelles ou de maintenance soient possibles. Le statut No-Go empêche l'avion de voler. Dans ce cas, la défaillance doit être réparée avant tout vol. Le vol est autorisé s'il n'y a pas de No-Go et si toutes les conditions Go-If sont réalisables. Lorsqu'il est autorisé, le vol peut être interrompu ou dérouté si l'aptitude de l'avion est considérablement dégradée. Des procédures décrites dans le manuel de vol servent de support pour déterminer si le vol peut continuer ou doit être dérouté.

Les situations indésirables pendant la réalisation d'une mission entraînent des interruptions de mission telles que les retards de vol, les annulations, les demi-tours et les déroutements. Notre travail prend en compte principalement les interruptions causées par des défaillances système ou l'incapacité d'effectuer la maintenance corrective dans un délai acceptable.

I.2.3 Maintenance

Un programme de maintenance est requis pour l'exploitation de tout avion et doit être approuvé par l'autorité de régulation. Le programme de maintenance définit les politiques de maintenance à appliquer aux divers composants système de l'avion. On distingue diverses tâches de maintenance préventive qui doivent être effectuées sur des bases régulières. Du point de la réalisation des travaux, il y a, entre autres, les activités réalisées en ligne, pendant lesquelles l'avion reste en service (au sol, non pas en vol). Ces activités concernent les inspections aux escales, les inspections journalières et les activités de réparation ne nécessitant pas l'utilisation d'un hangar de maintenance.

La thèse considère les activités de maintenance en ligne, puisque l'objectif est d'aider à gérer les problèmes opérationnels au cours de la réalisation des missions.

I.3 Les travaux connexes

Plusieurs études ont été menées dans le contexte abordant les problèmes de sécurité innocuité, la prédiction de fiabilité principalement pendant la phase de conception, la planification et l'optimisation de la maintenance, et la gestion des perturbations opérationnelles. Cependant, elles sont plutôt concentrées sur des aspects spécifiques, comme l'amélioration de la sécurité innocuité et la l'établissement de programmes de maintenance, qui ne couvrent pas la nécessité d'évaluer continuellement la sureté de fonctionnement en opération. Les travaux les plus proches concernent une analyse globale portant sur toute la période de vie du système

I.4 Orientation de la thèse

L'approche adoptée dans cette thèse consiste à évaluer, au cours des opérations de l'avion, sa capacité à satisfaire les exigences opérationnelles en présence de fautes, et l'aptitude à entreprendre des actions adaptées aux situations indésirables. Il y a besoin d'avoir un bon contrôle sur le comportement de l'avion en service, et d'être en mesure de prédire, avec un bon niveau de précision, les situations indésirables qui peuvent être rencontrées. La solution requiert une capacité à gérer des événements aléatoires et la capacité à faire en sorte que les solutions adoptées soient appropriées. L'évaluation de la sûreté de fonctionnement basée sur un modèle peut bien aider dans ce processus.

Ainsi, notre travail de recherche vise le développement d'un modèle qui permette l'évaluation de la fiabilité opérationnelle des avions, à tout moment où cela peut être nécessaire. Lors de la planification des missions et de la maintenance, une évaluation est réalisée afin d'aider à choisir la meilleure solution en conformité avec l'aptitude opérationnelle de l'avion, en tenant compte d'autres conditions l'impactant. A l'occurrence d'un événement majeur au cours de la réalisation d'une mission, la fiabilité est réévaluée afin d'évaluer l'impact de la nouvelle situation sur la réussite de la mission.

Le travail comprend donc, comme élément essentiel de la construction du modèle sur lequel est basé l'évaluation, la prise en compte de la situation au moment de l'évaluation. Ce qui peut nécessiter une adaptation du modèle afin de prendre en compte les caractéristiques de la situation courante. Cet aspect concerne la manière de gérer le modèle au moment de l'exécution afin d'obtenir des résultats adaptés.

II Approche de Modélisation et Contexte d'Evaluation

Ce chapitre vise à présenter l'approche de modélisation globale proposée. La construction du modèle nécessite une analyse qui aide à traiter les spécificités des avions et les problèmes liés à l'évaluation au cours des missions.

Le processus de construction du modèle et de l'évaluation en opération implique: i) le développement d'un modèle stochastique en utilisant un méta-modèle, et ii) l'utilisation de ce modèle stochastique pour obtenir un modèle intégrant les informations courantes, et qui est prêt à être utilisé pour l'évaluation. Nous avons choisi de baser la construction du modèle sur un méta-modèle afin de fournir une base commune standardisant des modèles pouvant être différents à cause des particularités des systèmes avions. Le méta-modèle représente également une référence donnant un aperçu du contenu du modèle à un haut niveau d'abstraction.

II.1 Elaboration du modèle

Cette section établit les éléments de base de notre approche de modélisation. Pour les besoins de l'évaluation en opération, nous avons défini deux types de mesures: la mesure de fiabilité système (SR) qui est à utiliser pendant la définition d'une mission pour déterminer la durée de mission la plus appropriée, et la mesure de fiabilité mission (MR), qui correspond à la probabilité d'accomplir la mission sans interruption.

Nous considérons les informations relatives au profil de mission, aux exigences opérationnelles, aux systèmes avion, et à la maintenance comme principales informations devant être prises en compte dans le modèle. Les informations contenues dans le modèle seront sujettes à des changements en opération. Les changements majeurs pouvant avoir lieu en opération et qui doivent être pris en compte sont les changements dans les états des composants système, les changements dans les distributions de défaillance et les changements dans le profil de la mission. Ces changements devraient donner lieu à une mise à jour du modèle stochastique, et une réévaluation de la fiabilité devrait être effectuée. La mise à jour est principalement basée sur l'ajustement de paramètres de modèle. Nous considérons que les systèmes avion ne subissent pas de modification pouvant entrainer une modification de la structure initiale du modèle. Quant aux profils de mission, Nous proposons une structuration du modèle en deux grandes parties : une partie de base (appelée *core model*) liée aux systèmes avion, et une partie liée au profil de mission. La partie liée au profil de mission vient se greffer à la partie de base de sorte à pouvoir être déclinée sous plusieurs versions prenant en charge d'éventuelles mises à jour structurelles que pourrait entrainer un changement de profil de mission.

Concernant la construction du modèle stochastique, comme le même type de modèle sera utilisé pour plusieurs familles d'avions d'un même fabricant, il est important de suivre une procédure harmonisée pour construire les modèles correspondant aux diverses familles d'avions, de sorte à obtenir une vue unifiée facilitant à la fois le processus de construction et la mise à jour. Ainsi, nous avons décidé d'établir une base commune pour la construction du modèle, en mettant en place un méta-modèle. Le méta-modèle est une représentation abstraite des modèles finaux grâce à l'analyse des éléments pertinents qui doivent être représentés.

II.2 Le méta-modèle

Le méta-modèle est développé en utilisant les éléments de méta-modélisation défini d'EMF (Eclipse Modeling Framework) [Griffin 2003], plus précisément les notations de Ecore. Nous présentons, dans ce qui suit, les différentes parties correspondant aux catégories d'information à inclure dans le modèle.

La partie pour la description des informations relatives aux composants système est présentée à la Figure 32. Les composants du système sont abstraits par l'EClass *SysComponent*. La représentation d'un composant système est caractérisée par son identifiant (*id*), ses variables d'état (*stateVars*) et les événements pouvant l'impacter.

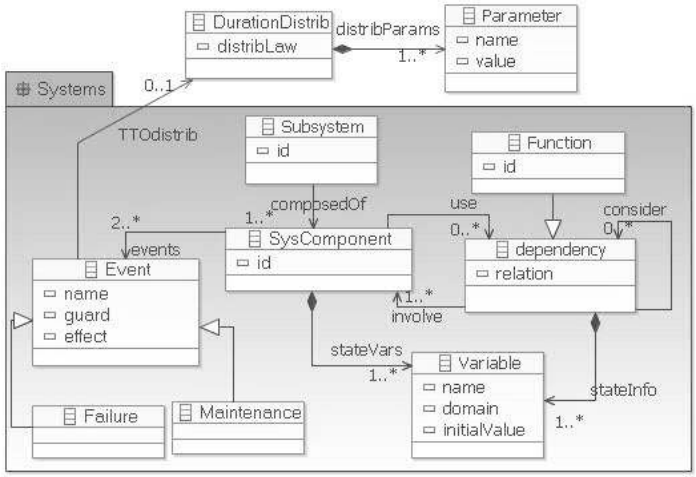


Figure 32 Méta-modèle pour la représentation des éléments système

Plusieurs variables d'état peuvent être utilisées dans la description d'un composant. Une EClass *Variable* est définie pour les abstraire. Une variable possède un nom (*name*), un domaine qui définit les valeurs possibles que peut prendre l'état, et une valeur initiale (*initialValue*) qui correspondra à son initialisation dans le modèle.

Pour les événements liés au composant, on distingue les événements de défaillance et de maintenance car ils représentent les principaux événements à considérer. Un événement est décrit par son nom (*name*), la garde conditionnant son occurrence, l'effet exprimant le changement dans l'état du système suite à son occurrence, et la distribution caractérisant le temps jusqu'à son occurrence (*TTOdistrib*). *TTOdistrib* est un objet de *DurationDistrib*, qui vise à représenter la loi de distribution du temps passé dans une situation donnée. La distribution est décrite par le nom de la loi de distribution (*distribLaw* - par exemple, exponentielle, Weibull, …) et ses paramètres *distribParams*, qui sont de l'EClass *Parameter*. Un paramètre de distribution est caractérisé par son nom (*name*) et une valeur *value*.

Il est important d'exprimer les dépendances entre les composants. Concrètement, un composant peut avoir à utiliser les valeurs d'attributs d'autres composants dans la description de son comportement. Dans ce cas, soit la valeur d'attribut est directement accessible, soit la valeur est obtenue par l'intermédiaire d'un objet, qui décrit les liens entre les composants. C'est le rôle de *Dependency*, qui vise à combiner les informations

provenant de différents composants, en utilisant une fonction conditionnelle (dénommée ici *relation)*. La valeur retournée (*stateInfo*) par la fonction *relation* correspond à l'information de dépendance, qui peut être utilisée dans la description de comportement du composant dépendant des autres. Les fonctions (*Function*) délivrées par les composants du système sont également représentées comme un cas particulier de dépendance, car la valeur de la variable d'état d'une fonction est également déterminée par la combinaison des variables d'état des composantes qui contribuent à sa délivrance.

Les fonctions, ainsi que les valeurs des variables d'état des composants système, sont utilisées pour définir les exigences relatives au système. La Figure 33 présente les éléments intervenant dans la définition des exigences.

Une exigence est une contrainte liée à une partie du système ou au profil de la mission, et qui doit être satisfaite pour réussir dans la réalisation d'une partie donnée de la mission. Elles sont abstraites par *Requirement* et se caractérisent par: i) l'attribut *reference,* un identificateur qui peut être utilisé pour faire référence à la même exigence dans différentes situations, ii) une variable booléenne *status* qui indique si l'exigence est satisfaite ou non, et iii) une contrainte ou une *expression* booléenne. L'expression booléenne formule une condition, impliquant des variables d'états de composants système et de fonctions, qui doit être satisfaite, sans quoi, une interruption peut advenir au cours de la réalisation de la mission. Une exigence peut résulter de la combinaison d'autres exigences.

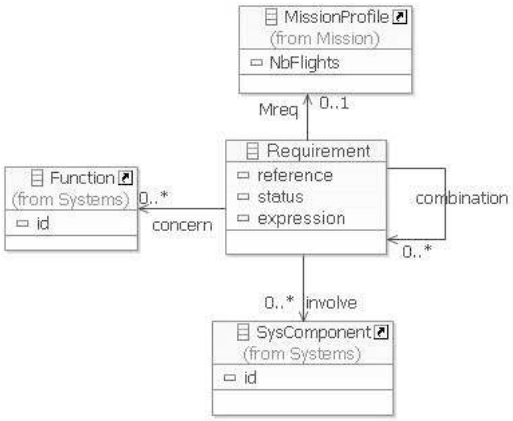


Figure 33 Méta-modèle concernant les exigences

Les exigences qui ne nécessitent pas d'information particulière liée au profil de la mission sont exprimées dans le modèle de base. Toute exigence ou information portant sur le système et qui pourrait être nécessaire dans la définition d'un profil de mission est représentée de façon à faciliter l'expression d'autres exigences spécifiques en cas de besoin.

La Figure 34 présente la partie du méta-modèle pour la représentation des informations concernant le profil de mission. Un profil de mission est défini par un nombre (*NbFlights*) de vols à effectuer en séquence.

Pour la réalisation de chaque vol, une période au sol (*GroundPeriod)* pour préparer le vol, et une période de vol *(FlightPeriod),* qui consiste en l'exécution proprement dite du vol,

sont distinguées. L'ensemble du processus pour réaliser le vol est dénommé *CompleteFlight*.

La période au sol *GroundPeriod* précède la période de vol et est composée de la description des activités (*GroundActivity*) réalisées au sol jusqu'à l'autorisation de départ. Les activités au sol sont caractérisées par leur dénomination (*denomination)* et leur durée (*duration)*. La durée peut être caractérisée par une distribution probabiliste.

La période de vol est décomposée en différentes phases en fonction des exigences qui doivent être satisfaites. Une phase de vol est caractérisée par sa dénomination (*denomination)*, sa durée (*duration)* et des informations additionnelles (*AdInfo*) qui pourraient être nécessaires dans la définition des exigences.

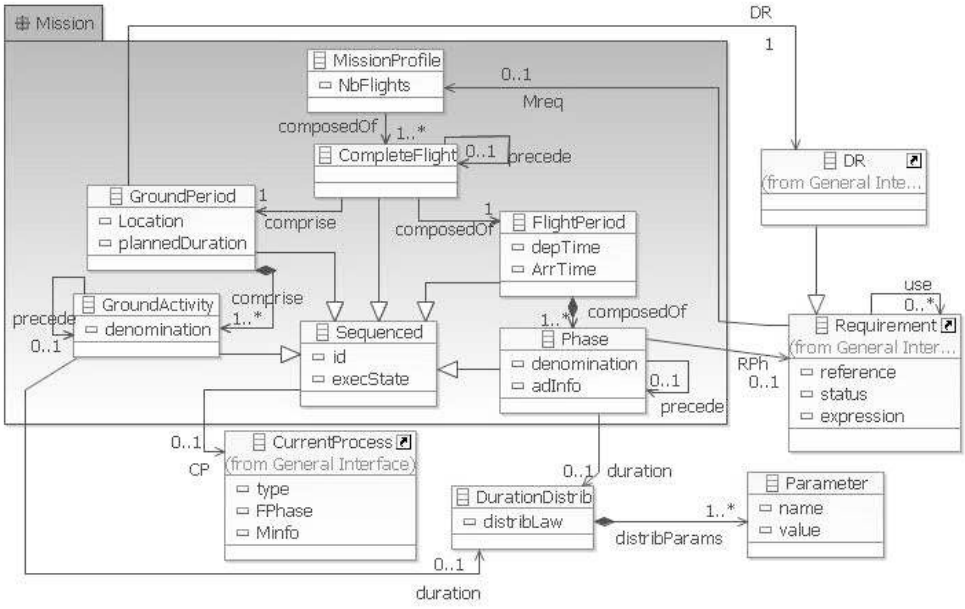


Figure 34 Méta-modèle pour la partie concernant la mission

Comme le profil de la mission est décomposé en une séquence de périodes et phases, une EClass *Sequenced* est définie pour représenter leurs caractéristiques communes, qui sont l'identifiant (*id*) et l'attribut *execState*. L'attribut *execState* indique si la partie correspondante de la mission est dans son état réalisation. L'information sur la partie de la mission en cours de réalisation est transmise à la partie de base du modèle en utilisant un objet de *CurrentProcess*. L'information concerne le type de la partie de mission (période au sol ou phase de vol), l'identifiant de la phase de vol (*FPhase*) s'il s'agit d'une phase de vol ; l'information d'autorisation de maintenance si c'est une période au sol. Les objets dérivés de *CurrentProcess* feront partie de l'interface entre la partie de base du modèle et la partie relative au profil de mission.

Avant tout vol, l'état opérationnel du système est testé par rapport aux exigences d'autorisation de vol (*DR*), qui représente une synthèse de la MEL. *DR* correspond à *Min_Sys_R* s'il n'y a pas d'exigence spécifique à la mission. Les activités de maintenance sont telles qu'elles ne peuvent pas être considérées comme terminées si les conditions d'autorisation du vol, liées à l'état du système ne sont pas satisfaites. La réussite de la

période au sol est déterminée par la réalisation des activités correspondantes durant la période de temps allouée. La réussite de la période de vol est déterminée par la réussite de ses différentes phases. Une phase est réalisée avec succès si les exigences correspondantes sont satisfaites durant la réalisation.

La réalisation des activités de maintenance dépend des moyens logistiques adéquats à l'escale considérée. Une station de maintenance est associée à chaque période au sol et la dépendance est prise en compte en considérant une distribution de probabilités caractérisant le temps nécessaire pour prendre en charge les activités correspondantes. Par exemple, une fonction *LDF* est définie pour prendre en compte le retard dans la prise en charge des activités. *LDF* est une sous-classe de *DurationDistrib* (Figure 32). Des types de station de maintenance, comme *base principale* et *hors base*, peuvent être utilisés pour les catégoriser. Les tâches sont effectuées en considérant un ordre de priorité dans la réalisation. La Figure 35 présente les éléments correspondants.

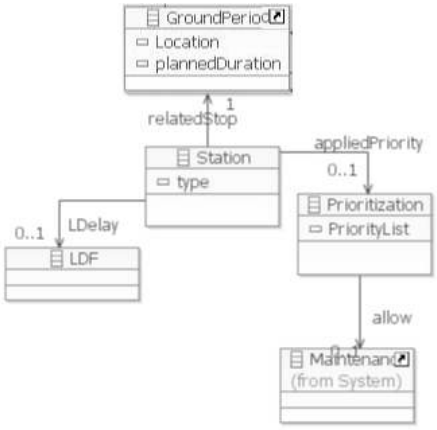


Figure 35 Partie pour la représentation des informations relatives à la maintenance

Le méta-modèle est une abstraction visant à définir les éléments pour la construction du modèle. Les attributs des classes définies peuvent être enrichis avec d'autres informations. Les représentations graphiques, basées sur les notations Ecore, sont très pratiques pour avoir un aperçu du contenu du modèle. Le modèle correspondant consistera en une représentation utilisant des instances des éléments décrits. Le méta-modèle peut également être utilisé pour construire des modèles à différentes fins. Le modèle peut être construit en se basant sur un sous-système particulier, ou il peut être construit en considérant plusieurs sous-systèmes.

III Modélisation avec les Formalismes AltaRica et Réseaux d'Activités Stochastiques

Divers formalismes sont utilisés dans le contexte de l'analyse de sûreté de fonctionnement pour aider à la construction des modèles. Les formalismes basés sur les méthodes à espace d'état sont les plus appropriés pour modéliser les systèmes complexes car ils offrent une grande expressivité. Ainsi, notre choix porte sur un formalisme à espace d'état pour la construction des modèles stochastiques. Comme Airbus et ses collaborateurs utilisent avec succès le formalisme AltaRica pour construire des modèles destinés à des analyses de sécurité, AltaRica a été choisi comme formalisme de support. Cependant, AltaRica et ses

outils de support disponibles sont encore très focalisés sur l'analyse qualitative, l'outil d'analyse quantitative (appelé EPOCH) prévu dans le cadre du projet est encore en cours de développement [Teichteil-Königsbuch et al. 2011]. Afin de procéder à des évaluations quantitatives lors de la mise en œuvre de l'approche de modélisation, nous avons choisi le formalisme des réseaux d'activités stochastiques (SAN) qui est bien connu pour effectuer les analyses quantitatives. Le formalisme SAN intègre intrinsèquement des éléments pour la description de comportement stochastiques et est supporté par un outil disponible (Möbius) qui permet de faire les évaluations quantitatives.

Nous analysons les caractéristiques des deux formalismes considérés et proposons des règles de transformation de modèles entre eux.

III.1 Caractéristiques et comparaison des deux formalismes

Le formalisme SAN a été développé dans les années quatre-vingt et permet de modéliser des systèmes pour des fins d'analyse de sûreté de fonctionnement et de performance. Le développement du formalisme AltaRica a commencé au milieu des années quatre-vingt-dix et concernait principalement des analyses qualitatives de systèmes.

Les deux formalismes sont basés sur les mécanismes d'états et de transitions. Le formalisme SAN reste très proche des réseaux de Petri stochastiques, mais définit de nouveaux éléments qui facilitent l'expression des prédicats et des actions liés aux événements. AltaRica utilise des variables qui peuvent être facilement combinées pour exprimer les prédicats et actions.

Le formalisme AltaRica, qui est un langage, prend en compte les aspects de description stochastique comme attributs externes du modèle. Ses principales caractéristiques sont le mécanisme de propagation du flux et le support intrinsèque de la modularité. Il prend en charge la synchronisation des événements qui consiste à lier plusieurs transitions de façon à ce qu'elles soient tirées simultanément.

Le formalisme SAN est presque entièrement basé sur des représentations graphiques. Ses caractéristiques principales sont la prise en compte intrinsèque des aspects stochastiques et son orientation vers les aspects d'analyses quantitatifs; la dynamique du modèle est complètement décrite de façon probabiliste. Il prend en charge la définition des cas, qui représentent des choix probabilistes d'actions au franchissement d'une transition. Il est également possible de spécifier des places partagées, i.e., une place peut être définie comme appartenant à plusieurs sous-modèles.

L'analyse des deux formalismes montre qu'ils sont interchangeables sur les aspects essentiels de modélisation qui concernent cette étude. Il existe toutefois quelques spécificités à mentionner. La possibilité d'associer des priorités aux événements en AltaRica n'est pas actuellement présente dans la définition de SAN. De l'autre côté, le mécanisme exprimé par les cas en SAN n'est pas présent en AltaRica, et il n'est pas facile de représenter des données partagées, modifiables par plusieurs composants comme représentées par les places partagées en SAN. En outre, l'expressivité du langage AltaRica est limitée par rapport au langage C++ qui est utilisé pour spécifier les prédicats et les actions en SAN.

Nous avons proposé des règles de transformation de modèle entre les deux formalismes. L'objectif est de montrer que les éléments du formalisme, que ce soit AltaRica ou SAN, qui sera utilisé pour illustrer l'approche de modélisation proposée, peuvent être trouvés de façon équivalente dans l'autre formalisme, et donc l'analyse quantitative peut être effectuée en utilisant le formalisme SAN, qui est supporté par un outil d'analyse quantitative disponible. En outre, grâce aux règles de transformation, des modèles équivalents peuvent être utilisés pour comparer des résultats d'évaluation provenant de l'outil Möbius qui traite les modèles SAN, et de l'outil EPOCH qui est en cours de développement pour traiter les modèles AltaRica, dans le but de valider ce dernier.

III.2 De AltaRica vers SAN

La transformation est basée sur les correspondances respectives des variables et des événements AltaRica avec les places et les activités SAN. Une variable AltaRica peut être représentée par une place, et les transitions gardées par des activités et des portes d'entrée / de sortie. Une correspondance peut être faite entre les types de données discrètes AltaRica et le marquage de type entier des places SAN. Les autres types de données peuvent être pris en compte en utilisant des places étendues.

La modularité du modèle est prise en compte en respectant la structure hiérarchique du modèle et en utilisant les opérateurs « Replicate » et « Join ». Il s'agit de conserver le plus possible la structure du modèle AltaRica. Dans la pratique, il peut être plus utile de compacter plusieurs sous-modèles en un seul, plutôt que de procéder à la transformation systématique qui peut générer un nombre considérable de petits sous-modèles.

Le Tableau 1 donne les correspondances pour transformer les éléments essentiels de AltaRica vers SAN.

Élément AltaRica	Correspondance en SAN	Commentaire
Variable d'état	Place	Les variables de type non discret correspondent à des places étendues
Variable de flux ($v \in F^{in} \cup F^{out}$)	Place partagée	
Événement	Activité SAN	Toutes les activités sont temporisées, sauf celles correspondant à des événements ayant Dirac(0) comme loi de distribution.
Transition (garde *grd*, événement *evt*, post-condition *post*)	Porte d'entrée (prédicat grd et fonction spécifiée par l'expression post-condition) associée à l'activité correspondant à l'événement evt.	
Assertion	Porte d'entrée associée à une activité instantanée	
Initialisation	Marquage des places	
Sous nœud	Sous modèle SAN	

Tableau 1 De AltaRica vers SAN

Il convient de mentionner qu'AltaRica intègre la notion de synchronisation d'événements entre plusieurs nœuds. Une synchronisation est une unification de plusieurs transitions. Elle est représentée en SAN par une activité unique qui combine les caractéristiques des transitions impliquées.

III.3 De SAN vers AltaRica

La transformation de SAN à AltaRica peut également être faite en transformant chaque sous-modèle SAN en un sous-nœud AltaRica. La composition est faite en suivant la structure du modèle SAN.

Pour la transformation du contenu d'un sous-modèle, chaque place est représentée soit par une variable d'état, soit par une variable d'entrée. La place correspond à une variable d'état si son marquage peut être modifié par une activité du sous-modèle. Dans le cas contraire, elle correspond à une variable d'entrée. Des variables de sortie supplémentaires peuvent être définies pour représenter les places partagées. Une activité peut être déclarée comme un événement AltaRica auquel est associée une loi de distribution. La transition correspondante en AltaRica a comme garde la conjonction de tous les prédicats des portes d'entrée de l'activité, et comme post-condition, la composition des fonctions des portes d'entrée et de sortie de l'activité. En SAN, les activités sont explicitement spécifiées comme étant instantanée ou temporisée. Les transitions instantanées sont exprimées en AltaRica en y associant Dirac (0) comme loi de distribution.

Pour les activités avec de multiples cas, la notion de choix probabiliste de l'action à accomplir après le franchissement d'une transition n'est pas présente en AltaRica. Cependant, d'un point de vue technique, une activité *act* à laquelle *n* (n> 1) cas sont associés peut être transformée en *n* activités act_1, ... act_n, chaque activité correspondant à un cas donné. Leurs lois de distribution correspondent aux produits de la loi de distribution de l'activité *act* avec les probabilités associées aux cas: $F_{act\_i}(\mu, \cdot) = F_{act}(\mu, \cdot) * C_{act}(\mu, i)$, i = 1, ..., n.

Les places partagées entre les sous-modèles peuvent être représentées à l'aide de variables de flux. Lorsque le marquage de la place partagée ne peut être modifié que dans un seul des sous modèles (ce qui correspond à la communication d'informations aux autres sous modèles), la place est représentée par une variable d'état et une variable de sortie dans le sous-nœud correspondant. Elle est représentée comme une variable d'entrée dans les sous-noeuds correspondant aux autres sous modèles. Si le marquage de la place partagée peut être modifié par plusieurs sous modèles, il devient difficile d'utiliser uniquement des variables de sortie et d'entrée, en raison du risque d'apparitions de boucles dans le modèle. Il faut utiliser le mécanisme de synchronisation d'événements. La place est représentée par une variable d'état dans un sous-nœud dédié et des transitions auxiliaires, synchronisées avec les transitions qui peuvent l'affecter, sont utilisées pour mettre à jour sa valeur. La valeur de la variable est transmise comme entrée aux autres sous-nœuds. Il est à noter que cette transformation nécessite l'identification de toutes les transitions affectant la place.

Le Tableau 2 donne l'essentiel des règles de transformation de modèles SAN en AltaRica.

Il convient de mentionner que des places étendues représentant des tableaux de variables peuvent être définies en utilisant l'outil de Möbius pour construire le modèle SAN. Les

variables de type tableau ne sont pas définies en AltaRica. Elles ne sont donc pas considérées pour la transformation. D'autre part, les structures de contrôle possibles en AltaRica sont les structures « if-then-else » et « switch / case ». En conséquence, les modèles SAN qui utilisent des fonctions incluant les structures telles que « while » ou « for » du langage C/C++ ne sont pas pris en compte.

Des exemples illustrant la mise en œuvre des règles de transformation sont proposés dans le chapitre. A terme, un outil implémentant ces règles devrait faciliter l'application des transformations.

Élément SAN	Correspondance AltaRica	Commentaire
Place	**Variable d'état** si le marquage de la place peut être changé par une activité du modèle, **Variable d'entrée** sinon.	Pour les modèles composés, les places partagées requièrent la considération de tous les sous modèles concernés, et des variables de sortie doivent être définies pour la composition.
Marquage de place	Affectation de valeur à la variable correspondante	
Activité	Événement	Les activités instantanées sont spécifiées par l'association de Dirac(0) comme loi de distribution de l'événement.
Portes d'entrée (prédicat Λpre, fonction f^t), et portes de sortie de la porte (fonction g^t) associé à l'activité *act*	Spécification de transition (garde Λpre, événement *evt*, post-condition $g^t \circ f^t$)	*evt* est l'événement qui correspond à *act*, Λpre est la conjonction des prédicats des portes d'entrée, f^t et g^t sont respectivement les compositions des fonctions de toutes les portes d'entrée, et de toutes les portes de sortie, associées à *act*.
Distribution associée à une activité	Instruction « extern - law »	
Sous modèle	Sous nœud	

Tableau 2 De SAN vers AltaRica

IV Cas d'étude

Ce chapitre présente des études de cas pour illustrer la mise en œuvre de l'approche de modélisation présentée dans le chapitre II, en utilisant le formalisme SAN. Le premier cas d'étude concerne le sous-système de commande de la gouverne de direction de l'A340 [Bernard et al. 2007]. Le deuxième cas d'étude concerne le sous-système d'alimentation électrique de l'A320, qui se caractérise par plusieurs mécanismes de reconfiguration. Des résultats d'évaluation quantitative ont été obtenus dans chaque cas d'étude.

Grâce au modèle correspondant au sous-système de commande de la gouverne de direction, nous avons analysé la nécessité de mettre à jour le modèle en opération, en particulier l'impact des changements sur les évaluations. Nous avons considéré les

changements majeurs identifiés dans le chapitre II et avons procédé à d'hypothétiques expérimentations de leur impact sur la fiabilité de mission.

Les résultats obtenus dans le cas du sous-système d'alimentation électrique montrent des valeurs de fiabilité plus élevées par rapport au cas du sous-système de commande de la gouverne de direction, même en considérant des taux défaillance de même ordre de grandeur. Les changements par rapport aux événements de défaillance non critique ne sont pas significatifs. D'ailleurs la composition avec le modèle du sous-système de commande de la gouverne de direction donne un impact minime par rapport aux résultats obtenus avec le sous-système de commande de la gouverne de direction seul.

Nous présentons dans les paragraphes suivants, le cas d'étude sur le système de commande de la gouverne de direction et les résultats obtenus.

IV.1 Cas du sous-système de commande de la gouverne de direction

Le système de commande de la gouverne de direction, présenté en Figure 36, est composé de trois calculateurs primaires (P1, P2, P3), d'un calculateur secondaire S1, de trois servocommandes (ServoCtrl_G, ServoCtrl_B et ServoCtrl_Y), d'un module de contrôle secours (BCM) et de deux composants (BPS_B et BPS_Y) permettant d'alimenter le module de secours en énergie.

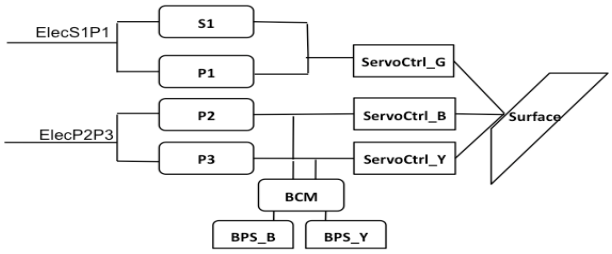


Figure 36 Sous-système de commande de la gouverne de direction

Les calculateurs sont connectés aux servocommandes, qui font mouvoir la surface. La connexion entre un calculateur et une servocommande forme une ligne de commande qui peut agir sur la surface. Il y a les lignes de commande primaire correspondant à P1, P2 et P3, la ligne secondaire correspondant à S1, et la ligne de commande secours correspondant à BCM.

Initialement S1, BCM, BPS_B et BPS_Y sont inhibés, et la surface est contrôlée par les trois lignes de commande primaire. Lorsque les lignes de commande primaire sont défaillantes, S1 est activé et la surface est contrôlée par S1. Si la commande secondaire devient défaillante, les composants BCM, BPS_B et BPS_Y sont activés pour contrôler la surface. Trois modes de contrôle peuvent ainsi être distingués : le contrôle par les calculateurs primaires (PC), le contrôle par S1 (SC) et le contrôle par les composants de secours (BC).

Exigences opérationnelles associées: Selon [MMEL 2008]:

- OR1: la défaillance d'un des composants P2, ServoCtrl_G, ServoCtrl_Y, ServoCtrl_B, BCM, BPS_B ou BPS_Y conduit à une situation de « No-Go ».

- OR2: les défaillances de P1, P3 et S1 sont « Go-If »:
  - P3 et S1 doivent être opérationnels en cas de défaillance de P1.
  - P1 et S1 doivent être opérationnels en cas de défaillance de P3.
  - P1, P2 et P3 doivent être opérationnels en cas de défaillance de S1.

IV.1.1 Construction du modèle

La Figure 37 montre la structure globale du modèle. Dans la partie relative aux composants système, nous distinguons des sous modèles correspondant aux trois modes de contrôle. Comme proposé dans l'approche de modélisation, le modèle est à construire en utilisant le méta-modèle. Nous présentons, en guise d'exemple, une spécification utilisant les éléments du méta-modèle pour décrire le contenu de la partie de base du modèle. Nous présentons ensuite un aperçu du modèle en SAN donnant plus détail sur la partie liée au profil de mission.

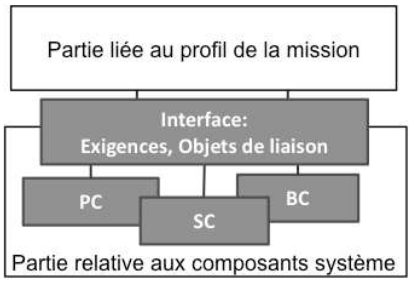


Figure 37 Structure du modèle

IV.1.2 Spécification de la partie de base du modèle

Tous les composants système ont des comportements similaires et sont représentés à l'aide des attributs relatifs à *SysComponent* (Figure 32).

Le nom de chaque composant est utilisé comme identifiant (*id*). Pour chaque élément x, une variable d'état est considérée, avec un domaine défini par l'ensemble {*ok, failed*}, la valeur initiale est « *ok* ».

Pour les événements à associer, nous considérons un événement de défaillance qui modifie l'état de « *ok* » vers « *failed* », et un événement de maintenance qui restaure l'état à « *ok* ». Nous supposons que l'événement de défaillance se produit en vol, étant donné qu'il est généralement caractérisé par un taux d'occurrence par heure de vol. Pour cela, nous utilisons un objet d'interface *CP* (instance de *CurrentProcess* - Figure 34) entre les deux parties du modèle, qui donne des informations sur la partie de la mission en cours d'exécution. Il est aussi utilisé dans l'expression de la garde de l'événement de maintenance pour spécifier l'information d'autorisation de la maintenance (*CP_M*). Par exemple pour le composant P1, les événements sont définis comme suit:

Evénement de défaillance :
name: *failure*
guard: *state=ok and CP-type=flight;* effect *state=failed*
TTOdistrib: *distribLaw=exponential, parameter: lambda=2E-4*

Evénement de maintenance:

name: *maintainP1*

guard: *state= failed and id ∈CP-M;* effect *state=ok*

TTOdistrib: *distribLaw=deterministic*, parameter: *t=1*

Pour le calculateur secondaire et les composants de commande de secours, les scénarios d'activation et de désactivation sont représentés. L'activation et la désactivation dépendent de l'état des lignes de commande primaires. Les lignes de commande primaires sont représentées en utilisant des instances de *Dependency*. *PL1*, *PL2*, *PL3* représentent respectivement l'état de la connexion entre P1 et ServoCtrl_G, l'état de la connexion entre P2 et ServoCtrl_B, et l'état de la connexion entre P3 et ServoCtrl_Y. Les variables d'état associées (*PL1-stateInfo*, *PL2-stateInfo* et *PL3-stateInfo*) ont l'ensemble {*ok, failed*} comme domaine et leurs valeurs sont déterminées par la fonction de combinaison suivante (*relation*), en utilisant *PL1* comme exemple :

PL1-stateInfo = ok si P1-state=ok et ServoCtrl_G-state=ok et ElecP1S1 et hyd_G
PL1-stateInfo = failed sinon

ElecP1S1 et *hyd_G* sont des entrées provenant d'autres sous-systèmes. Lorsqu'on considère le sous-système de commande de la gouverne de direction seul, *ElecP1S1* et *hyd_G* sont considérées comme étant toujours disponibles.

Les exigences opérationnelles exprimées à la fin du §IV.1 sont relatives aux composants système et sont applicables quel que soit le profil de mission. Nous les exprimons comme exigences minimales système (*Min_Sys_R*), auxquelles peuvent s'ajouter des exigences spécifiques au profil de la mission. Elles sont exprimées en utilisant les éléments définis dans la Figure 33.

L'attribut *reference* n'est pas utilisé ici car il est à utiliser uniquement dans le contexte des mises à jour du modèle. Les exigences sont considérées comme satisfaites initialement (*status=satisfied)*. L'expression de *Min_Sys_R* est formulée comme la conjonction de OR1 et OR2 (voir section IV.1). L'expression finale, basée sur les lignes de contrôle, est donnée comme suit :

$$\begin{aligned} Min\_Sys\_R = \{ \; & PL2 = ok \wedge BCL = ok \wedge \\ & (PL1 = ok \vee (PL3 = ok \wedge SL = ok)) \wedge \\ & (PL3 = ok \vee (PL1 = ok \wedge SL = ok)) \wedge \\ & (SL = ok \vee (PL1 = ok \wedge PL3 = ok)) \; \}. \end{aligned} \quad (1)$$

Min_Sys_R, exprimée dans la partie de base du modèle (*core model*) est utilisée pour faire la composition avec la partie liée au profil de mission.

IV.1.3 Le modèle global en SAN

La Figure 38 présente le modèle global en SAN. Nous décrivons la partie liée au profil de mission et sa connexion avec le modèle de base. Le marquage de la place *Min_Sys_R* est mis à jour par le tir des activités instantanées *Fulfilled* et *Not_Fulfilled* conformément à la condition exprimée par l'expression donnée en fin du §IV.1.2. Il est à noter que dans cet exemple, nous ne considérons pas les exigences spécifiques au profil de mission qui pourraient être associées à chaque vol.

Considérant la partie liée au profil de mission, elle est basée sur une structure générique qui doit être paramétrée en spécifiant le nombre de vols pour chaque mission et les paramètres de durée de chaque vol et des activités au sol.

La partie supérieure représente un vol et la partie inférieure représente les activités au sol, lors d'une escale. Un vol est représenté par trois phases *Taxing_to_TakeOff, In_Flight* et *Landing*. Pendant la phase *Taxing_to_TakeOff* le vol peut être annulé et il peut être dérouté pendant la phase *In_Flight*. Les portes d'entrée *AbortCondition* et *DiversionCondition* définissent les conditions conduisant à l'occurrence de ces interruptions.

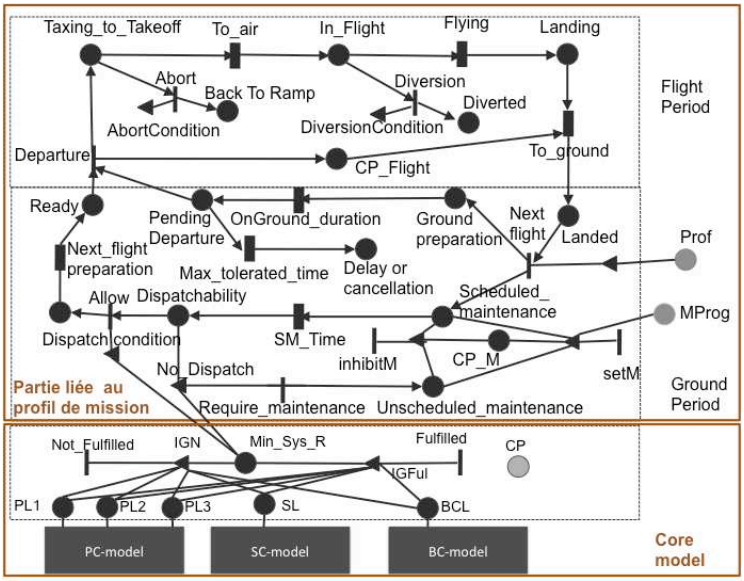


Figure 38 Le modèle globale en SAN

Le modèle de la partie au sol décrit la préparation pour le prochain vol. Les activités au sol (*SM_Time, Next_flight_preparation)* sont décrites en parallèle avec le respect du temps prévu pour la phase au sol *OnGround_duration*) et celui du délai de retard maximum tolérable (*Max_tolerated_time*). *Scheduled_maintenance* représente une période de maintenance planifiée et *Unscheduled_maintenance* représente le prolongement des activités de maintenance au cas où les exigences en disponibilité des composants système ne sont pas satisfaites (*No_Dispatch*). La gestion de la maintenance est telle que la place *MProg* représente des listes qui déterminent, pour chaque période au sol, les composants à réparer. La place *CP_M* identifie le composant à réparer et elle est utilisée dans le modèle de base pour autoriser la maintenance. *Next_flight_preparation* représente les autres activités (embarquement des passagers, traitement des bagages ...) qui peuvent consommer du temps, occasionnant un retard. La place *Prof* (à droite) est une place étendue représentant la liste des vols à effectuer. La porte d'entrée reliée à cette place indique s'il y a un prochain vol à effectuer ou non.

IV.2 Résultats d'évaluation et analyse de l'impact de la réévaluation en opération

L'approche de modélisation proposée dans le chapitre II est destinée à être utilisée pour faire des analyses qualitatives et quantitatives. Dans ce chapitre, nous nous sommes

focalisés sur l'analyse quantitative pour construire le modèle stochastique. Des études d'analyse de sécurité innocuité visant à renforcer la conception ont déjà mis en œuvre les aspects d'analyse qualitative.

L'analyse quantitative consiste en l'évaluation des mesures de fiabilité système et mission. Pour obtenir des résultats illustratifs, nous avons paramétré le modèle avec des valeurs hypothétiques.

IV.2.1 La mesure SR

La mesure de fiabilité du système, SR, correspond à la probabilité que la place Min_Sys_R soit marquée pendant une période de temps donnée. Elle ne concerne que le modèle de base. Cette mesure peut être utilisée pour aider à l'attribution d'une mission à l'avion, en considérant qu'elle ne doit pas être inférieure à un seuil acceptable donné.

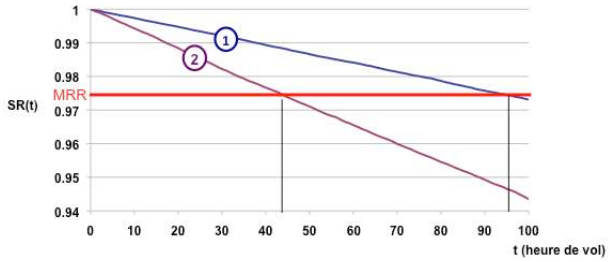


Figure 39 Mesure SR

La courbe 1 de la Figure 39 montre la fiabilité du système. Elle montre que la durée maximale de mission, sans les activités de maintenance, doit être inférieure à 95 heures de vol, pour respecter le seuil de 0,975. Cette évaluation suppose que tous les composants du système sont initialement opérationnels. La courbe 2 suppose que le calculateur P1 est défaillant au début de la mission. La seule défaillance de P1 n'empêche pas la réalisation d'une mission. La courbe montre que, pour respecter le seuil de 0,975, la durée maximale de la mission, sans activités de maintenance, doit être inférieure à 45 heures de vol.

IV.2.2 La mesure MR

Nous considérons donc une mission typique, d'une durée de 84h, composée de 4 vols identiques (3h chacun) par jour, durant 7 jours. La courbe 0 de la Figure 40-a montre la fiabilité de mission, MR, évaluée avant la mission, en supposant que tous les composants sont opérationnels au début. Nous supposons que, tant que MR est supérieure au seuil, dénommé MMRR, la mission peut être réalisée. On peut voir qu'à la fin de la mission, MR reste supérieure à MMRR.

Dans ce qui suit, nous analysons l'impact de la défaillance du calculateur P1 durant la réalisation de la mission, en considérant le jour de la défaillance. La courbe 0 est la même dans toutes les figures de 9-a à 9-d.

La courbe 1 de la Figure 40-b correspond au cas où P1 a été diagnostiqué comme défaillant à la fin du jour 2. MR est réévaluée en considérant i) comme temps initial le jour suivant (i.e., jour 3), et ii) P1 défaillant à l'instant t = 0. On peut voir que MR est inférieure

à MMRR à partir du jour 5. Ce résultat montre que P1 doit être réparé avant la fin de la mission afin de respecter l'exigence MMRR. Trois cas peuvent être envisagés: P1 est réparé à la fin du jour 3, à la fin du jour 4, ou à la fin du jour 5. La Figure 40-c correspond au cas où P1 est réparé à la fin du jour 4. On peut constater que MR reste supérieure à MMRR pour l'ensemble de la mission. Le cas où la réparation a lieu à la fin du jour 5 conduit à une MR inférieure à MMRR, en jour 7. Par conséquent, en cas de défaillance de P1 au jour 2, P1 doit être réparé avant le cinquième jour, en fonction du lieu et le moment où la maintenance peut avoir lieu.

La courbe 3 de la Figure 40-d correspond au cas où P1 a été diagnostiqué comme défaillant à la fin du jour 4. MR est donc réévaluée en considérant i) comme temps initial (t = 0), le jour 5, et ii) P1 défaillant à l'instant t = 0. On peut voir que la nouvelle évaluation est toujours supérieure à MMRR à la fin de la mission prévue. La mission peut être poursuivie sans maintenance jusqu'à son terme, à moins qu'un nouvel événement se produise, dans quel cas une nouvelle évaluation est nécessaire.

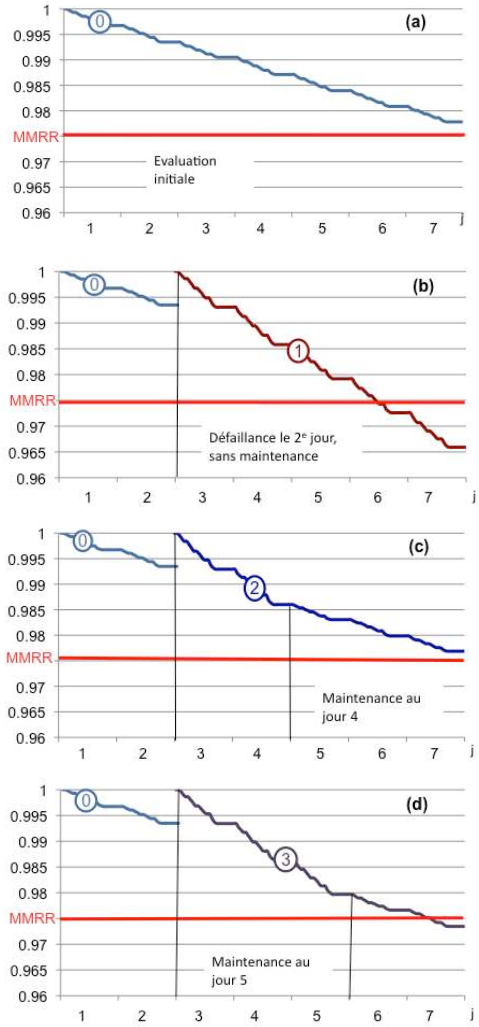


Figure 40 MR- Impact de la défaillance et de la maintenance de P1

Les résultats ci-dessus méritent deux commentaires majeurs:

- Nous avons supposé des distributions exponentielles pour tous les évènements de défaillances des composants pour montrer que les changements opérationnels induiront des changements perceptibles dans les résultats. Avec l'approche de modélisation utilisée et les outils disponibles, il est possible d'envisager d'autres distributions et de prendre en compte le vieillissement des composants impliqués dans l'analyse (voir [Tiassou et al. 2012b]). Cependant, le vieillissement est un processus changement sur un long terme ; la granularité des changements est beaucoup plus grande qu'un jour ou une semaine (la durée de la mission). En outre, de très faibles variations des taux de défaillance des composants lors d'une mission induisent une variation non perceptible dans les courbes de fiabilité.
- MR est égale à 1 au début de chaque nouvelle évaluation, car le système est complètement inspecté après la détection d'une défaillance d'un composant, et il est globalement dans un état opérationnel au moment où la mission est reprise.

Conclusion et Perspective

Dans cette thèse, nous avons abordé la problématique de modélisation de sûreté de fonctionnement des avions pour une évaluation en opération, de façon à aider à la planification des missions et de la maintenance, ainsi qu'à leur bonne réalisation. Le but ultime est de contribuer à l'amélioration de l'opérabilité des avions en réduisant les pertes économiques liées aux défaillances et aux planifications ne permettant pas de réaliser promptement les activités de maintenance. L'évaluation doit être faite par l'équipe menant les opérations, équipe qui n'est pas nécessairement familière avec les formalismes et les outils de modélisation.

La principale contribution de la thèse concerne le développement d'une approche d'évaluation basée sur les modèles stochastiques à espace d'état, qui peuvent être facilement mis à jour en cours d'opération, en tenant compte des informations relatives à la situation courante.

Nous avons identifié les types d'informations pertinentes à considérer dans le modèle. L'adaptation du modèle à la situation courante est gérée par la mise à jour de l'information correspondante dans le modèle. La mise à jour est le résultat de l'occurrence d'un événement ou d'un changement au cours de l'opération de l'avion. En effet, après tout changement majeur, on doit vérifier si son impact est significatif ou non. Une réévaluation de la sûreté de fonctionnement est par conséquent nécessaire afin d'avoir des résultats adaptés. L'avion sera surveillé par des modules de pronostic et de diagnostic qui fourniront des informations sur l'état des composants système et sur l'allure de la distribution caractérisant leurs défaillances. Des informations sur les profils de mission et la maintenance seront aussi accessibles.

En se basant sur l'analyse des principaux changements qui peuvent survenir pendant la réalisation des missions, nous avons proposé une approche pour gérer la mise à jour du modèle. Le modèle est conçu pour être générique et paramétrique avec des valeurs par défaut. Ainsi, la mise à jour consiste à ajuster les paramètres avec les données résultant des changements observés. La mise à jour du modèle ne devrait pas exiger des tâches de modélisation pour lesquelles l'opérateur de l'avion, qui effectuera l'évaluation n'est pas qualifié.

Le modèle est destiné à être construit en phase de conception des systèmes avion. Pour aider à la construction des modèles, nous avons proposé un méta-modèle qui spécifie leur contenu à un haut niveau d'abstraction. Le méta-modèle fournit les éléments pour la construction de modèles correspondant à différents types d'avions. Par ailleurs, d'autres modules, comme le pronostic et les modules de diagnostic qui sont utilisés dans le cadre de l'évaluation, devront communiquer des données pour la mise à jour du modèle. Le méta-modèle définit le genre de données à communiquer.

Après la spécification du contenu du modèle, nous avons étudié deux formalismes pour mettre en œuvre l'approche de modélisation: AltaRica et les réseaux d'activités stochastiques (SAN). Le formalisme AltaRica est très utilisé dans le contexte industriel en France, notamment chez Airbus pour réaliser des analyses de la sécurité. Il a donc été retenu pour la construction des modèles finaux. Il peut être utilisé pour la construction du modèle stochastique, mais ses outils de support restent limités en analyse stochastique. Le formalisme SAN est plus connu dans le milieu universitaire et fournit des éléments pour la modélisation de comportements stochastiques complexes.

En analysant les deux formalismes, nous avons montré qu'ils sont interchangeables sur les aspects de modélisation de base et que même la plupart des aspects représentés à l'aide de leurs caractéristiques spécifiques peuvent être décrits en utilisant les éléments de l'autre formalisme. Nous avons proposé des règles pour la transformation de modèles entre les deux formalismes. Ces règles de transformation devraient être utile dans la réalisation d'analyses comparatives lors de la validation des outils de traitement quantitatifs de modèles en AltaRica. Un prototype d'outil est en cours de développement pour traiter les modèles finaux qui seront développés en AltaRica.

En attendant, nous avons utilisé le formalisme SAN et son outil associé Möbius pour modéliser le sous-système de commande de la gouverne de direction de l'A340, et le sous-système d'alimentation électrique de l'A320. Des exemples illustratifs de résultats d'évaluation ont été donnés.

Etant donné que l'approche proposée pour l'évaluation consiste à réévaluer la fiabilité opérationnelle à chaque changement majeur, afin d'obtenir un résultat à jour, nous avons analysé l'impact que pourrait avoir la réévaluation par rapport à l'évaluation initiale. Les résultats montrent que, par exemple, la défaillance d'un calculateur primaire, dont la disponibilité n'est pas indispensable à la réalisation d'un vol, peut réduire considérablement la fiabilité d'une mission de 7 jours. Les analyses indiquent aussi comment utiliser les résultats de l'évaluation pour gérer la réalisation de la mission et des activités de maintenance.

Les analyses quantitatives effectuées représentent une validation préliminaire de l'approche. Elle devrait être complétée par d'autres analyses en utilisant l'outil dédié, qui est en cours de développement. D'autre part, comme les données en entrée des modèles finaux seront fournies avec une certaine imprécision, l'outil d'évaluation donnera une indication de l'incertitude induite sur le résultat obtenu. Des études [Jacob et al. 2011; Jacob et al. 2012] sont en cours pour développer les méthodes de calcul de ces incertitudes.

En ce qui concerne la suite de cette thèse, plusieurs orientations sont possibles.

Les modèles finaux vont être développés en utilisant les éléments du méta-modèle. Les modèles ou des parties de modèle pourraient être générés automatiquement en utilisant le méta-modèle. Ainsi, il serait intéressant de mener des études sur la génération automatique des modèles à partir du méta-modèle.

Concernant la transformation de modèles entre AltaRica et SAN, il serait très approprié de développer un outil dédié, qui ferait la transformation automatiquement.

L'évaluation de sureté de fonctionnement en tenant compte de plusieurs avions représente également une extension de notre travail. En effet, les compagnies aériennes opèrent des flottes d'avions et l'indisponibilité d'un avion peut être gérée en utilisant un autre avion disponible. En outre, les avions peuvent avoir des ressources de maintenance, telles que les composants de rechange et les techniciens, en commun. Il serait ainsi intéressant de couvrir cette dépendance en considérant tous les avions impliqués.

Nous nous sommes placés dans le contexte d'exploitation des avions pour développer notre approche de modélisation. Toutefois, l'évaluation de fiabilité opérationnelle en service pourrait s'appliquer à tout système dont l'opération nécessite une certaine surveillance. D'autres contextes opérationnels pourraient être ciblés et étudiés. Le développement de moyens génériques de modélisation pour l'évaluation en ligne ne peut qu'être utile.

Le travail réalisé représente, par conséquent, une base importante pour le développement de nouveaux moyens qui devraient être essentiels pour aider dans l'exploitation des systèmes.

References

AHMADI, A. AND SODERHOLM, P. 2008. Assessment of Operational Consequences of Aircraft Failures: Using Event Tree Analysis. *Proc. 2008 IEEE Aerospace Conf.*, 1–14.

AJMONE MARSAN, M., CONTE, G., AND BALBO, G. 1984. A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems. *ACM Transactions on Computer Systems 2*, 2, 93–122.

ARGÜELLO, M., BARD, J., AND YU, G. 1997. A Grasp for Aircraft Routing in Response to Groundings and Delays. *Journal of Combinatorial Optimization 1*, 3, 211–228.

ARNOLD, A., POINT, G., GRIFFAULT, A., AND RAUZY, A. 1999. The AltaRica formalism for describing concurrent systems. *Fundamenta Informaticae 40*, 2-3, 109–124.

ATA MSG. 1993. MSG-3 - Maintenance program development Document. http://www.7ts0.com/manuals/faamech/8300/Appendix5.pdf.

AVIZIENIS, A., LAPRIE, J.-C., AND RANDELL, B. 2000. Fundamental concepts of dependability. *Proceedings of the 3rd Information Survivability Workshop (ISW-2000)*, 7–12.

AVIZIENIS, A., LAPRIE, J.-C., RANDELL, B., AND LANDWEHR, C. 2004. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing 1*, 1, 11–33.

AZGOMI, M. AND MOVAGHAR, A. 2005. A modelling tool for hierarchical stochastic activity networks. *Simulation Modelling Practice and Theory 13*, 6, 505–524.

BALABAN, H.S., BRIGANTIC, R.T., WRIGHT, S.A., AND PAPATYI, A.F. 2000. A simulation approach to estimating aircraft mission capable rates for the United States Air Force. *Proceedings of the 2000 Winter Simulation Conference*, Society for Computer Simulation International, 1035–1042 vol.1.

BENNETTS, R.G. 1982. Analysis of Reliability Block Diagrams by Boolean Techniques. *IEEE Transactions on Reliability R-31*, 2, 159–166.

BÉOUMES, C., KANOUN, K., AGUERA, M., ET AL. 1993. SURF-2: A Program for Dependability Evaluation of Complex Hardware and Software Systems. *23th IEEE International Symposium on Fault-Tolerant Computing*, IEEE Comput. Soc. Press, 668–673.

BERNARD, R., AUBERT, J., BIEBER, P., MERLINI, C., AND METGE, S. 2007. Experiments in model-based safety analysis: flight controls. .

BETOUS-ALMEIDA, C. AND KANOUN, K. 1997. *Méthode des états fictifs dans les modèles de sûreté de fonctionnement*. LAAS, 65p.

BETOUS-ALMEIDA, C. AND KANOUN, K. 2004. Dependability modelling of instrumentation and control systems: A comparison of competing architectures. *Safety Science 42*, 5, 457–480.

BIEBER, P., BOUGNOL, C., CASTEL, C., ET AL. 2004. Safety Assessment with Altarica Lessons learnt based on two aircraft system studies. *IFIP International Federation for Information Processing*, 505–510.

BINEID, M. AND FIELDING, J.P. 2006. Development of an aircraft systems dispatch reliability design methodology. *The Aeronautical journal 110*, 1108, 345–352.

BISWAWS, P. AND SHRIMALI, S.C. 2001. Safety assessment of modern aircraft-a case study. *Reliability and Maintainability Symposium, 2001. Proceedings. Annual*, 365–371.

BOUISSOU, M. 1993. The FIGARO dependability evaluation workbench in use: Case studies for fault-tolerant computer systems. *Fault-Tolerant Computing, 1993. FTCS-23. Digest of Papers., The Twenty-Third International Symposium on*, 680 – 685.

BOUISSOU, M., BOUHADANA, H., BANNELIER, M., AND VILLATTE, N. 1991. Knowledge modeling and reliability processing: Presentation of the FIGARO language and associated tools. *Proc. Safety of computer control systems (SAFECOMP'91)*, 69–82.

BOWLES, J.B. 1998. The new SAE FMECA standard. *Reliability and Maintainability Symposium, 1998. Proc., Annual*, 48 –53.

CHENEVIER, P. 2001. Les programmes de maintenance aéronautique: méthodologie de création et cadre réglementaire. *La Jaune et la Rouge - Revue de la communauté polytechnicienne*. http://www.lajauneetlarouge.com/article/les-programmes-de-maintenance-aeronautique-methodologie-de-creation-et-cadre-reglementaire.

CHEW, S.P., DUNNETT, S.J., AND ANDREWS, J.D. 2008. Phased mission modelling of systems with maintenance-free operating periods using simulated Petri nets. *Reliability Engineering & System Safety 93*, 7, 980 – 994.

CHIOLA, G., FRANCESCHINIS, G., GAETA, R., AND RIBAUDO, M. 1995. GreatSPN 1.7: Graphical editor and analyzer for timed and stochastic Petri nets. *Performance Evaluation 24*, 1-2, 47–68.

CLARKE, L.W., HANE, C.A., JOHNSON, E.L., AND NEMHAUSER, G.L. 1996. Maintenance and Crew Considerations in Fleet Assignment. *Transportation Science 30*, 3, 249–260.

CLARKE, M.D.D. 1998. Irregular airline operations: a review of the state-of-the-practice in airline operations control centers. *Journal of Air Transport Management 4*, 2, 67–76.

CLARKE, M.D.D. AND NARYADI, Y. 1995. *The Airline Operation Control Centre: an overview of Garuda's Operation Control (EM) at Cengkereng Jakarta, Indonesia:*

final report to PT Garuda Indonesia. Massachusetts Institute of Technology, Flight Transportation Laboratory, Cambridge, Mass.

CLAUSEN, J., LARSEN, A., LARSEN, J., AND REZANOVA, N.J. 2010. Disruption management in the airline industry—Concepts, models and methods. *Computers & Operations Research 37*, 5, 809–821.

COUVILLION, J.A., FREIRE, R., JOHNSON, R., ET AL. 1991. Performability modeling with UltraSAN. *IEEE Software 8*, 5, 69–80.

DALY, D., DEAVOURS, D.D., DOYLE, J.M., WEBSTER, P.G., AND SANDERS, W.H. 2000. Möbius: An Extensible Tool for Performance and Dependability Modeling. *Proceedings of the 11th International Conference on Computer Performance Evaluation: Modelling Techniques and Tools*, Springer-Verlag, 332–336.

ECSS. 2001. *Failure modes, effects and criticality analysis (FMECA)*. European cooperation for space standardization, Noordwijk, The Netherlands.

ERICSON, C.A. 1999. Fault Tree Analysis – A History. *Proceedings of The 17th International System Safety Conference*, System Safety Society, 87–96.

FAA. 1978. AC 120-17A - Maintenance control by reliability methods. http://purl.access.gpo.gov/GPO/LPS108499.

FILAR, J., MANYEM, P., AND WHITE, K. 2001. How Airlines and Airports Recover from Schedule Perturbations: A Survey. *Annals of Operations Research 108*, 1, 315–333.

FOTA, N., KAANICHE, M., AND KANOUN, K. 1997. A modular and incremental approach for building complex stochastic Petri net models. *st International Conference on Mathematical Methods in Reliability (MMR'97)*, 151–158.

FOTA, N., KAÂNICHE, M., AND KANOUN, K. 1999. Dependability evaluation of an air traffic control computing system. *Performance Evaluation 35*, 3-4, 253–273.

GARDNER, T. 2003. Model-Driven Metadata Integration using MOF 2.0 and Eclipse. http://www.omg.org/news/meetings/workshops/MDA_2003-2_Manual/1-3_Gardner.pdf.

GERBER, A. AND RAYMOND, K. 2003. MOF to EMF: there and back again. *Proceedings of the 2003 OOPSLA workshop on eclipse technology eXchange*, ACM, 60–64.

GERMAN, R., KELLING, C., ZIMMERMANN, A., AND HOMMEL, G. 1995. TimeNET: a toolkit for evaluating non-Markovian stochastic Petri nets. *Performance Evaluation 24*, 1-2, 69–87.

GRANDEAU, S.C. 1995. *The process of airline operational control*. Massachusetts Institute of Technology, Flight Transportation Laboratory, Cambridge, MA.

GRIFFIN, C. 2003. Introduction to the Eclipse Modeling Framework. *MDA™ Implementers' Workshop - Succeeding With Model Driven Systems*.

GUPTA, P., BAZARGAN, M., AND MCGRATH, R.N. 2003. Simulation model for aircraft line maintenance planning. *Annual Reliability and Maintainability Symposium, 2003 Proceedings*, 387– 391.

HAMOUDA, O., KAANICHE, M., AND KANOUN, K. 2009. Safety modeling and evaluation of Automated Highway Systems. 73–82.

HUGUES, E., CHARPENTIER, E., AND CABARBAYE, A. 2002. Application of Markov processes to predict aircraft operational reliability. *Proc. 3rd European Systems Engineering Conference (EUSEC'2002)*, 231–235.

JACOB, C., DUBOIS, D., AND CARDOSO, J. 2011. Uncertainty handling in quantitative BDD-based fault-tree analysis by interval computation. *Proceedings of the 5th international conference on Scalable uncertainty management*, Springer-Verlag, 205–218.

JACOB, C., DUBOIS, D., AND CARDOSO, J. 2012. From Imprecise Probability Laws to Fault Tree Analysis. In: *Scalable Uncertainty Management*. Springer Berlin Heidelberg, 525–538.

JARDINE, A.K.S., LIN, D., AND BANJEVIC, D. 2006. A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing 20*, 7, 1483–1510.

JONES, J.A., WARRINGTON, L., AND DAVIS, N. 2002. Integrated modelling of system functional, maintenance & environmental factors. *Proceedings. Annual Reliability and Maintainability Symposium*, IEEE, 399–403.

KANOUN, K. AND BORREL, M. 1996. Dependability of fault-tolerant systems-explicit modeling of the interactions between hardware and software components. *Proceedings of the 2nd International Computer Performance and Dependability Symposium (IPDS '96)*, IEEE Computer Society, 252–261.

KARAGIANNIS, D. AND KÜHN, H. 2002. Metamodelling Platforms. In: K. Bauknecht, A.M. Tjoa and G. Quirchmayr, eds., *E-Commerce and Web Technologies*. Springer Berlin Heidelberg, Berlin, Heidelberg, 182–182.

KEHREN, C., SEGUIN, C., BIEBER, P., ET AL. 2004. Advanced simulation capabilities for Multi-systems with Altarica. *Proceedings of the 22nd International System Safety Conference*, International System Safety Society, 489–498.

KINNISON, H.A. 2004. *Aviation maintenance management*. McGraw-Hill, New York.

KOHL, N., LARSEN, A., LARSEN, J., ROSS, A., AND TIOURINE, S. 2007. Airline disruption management—Perspectives, experiences and outlook. *Journal of Air Transport Management 13*, 3, 149–162.

KUMAR, U.D. 2000. *Reliability maintenance and logistic support: a life cycle approach*. Kluwer Academic, Boston, Mass. USA.

KWIATKOWSKA, M., NORMAN, G., AND PARKER, D. 2009. PRISM: probabilistic model checking for performance and reliability analysis. *ACM SIGMETRICS Performance Evaluation Review 36*, 4, 40–45.

LABRI. 2010. AltaRica Language. http://altarica.labri.fr/forge/projects/altarica/wiki/AltaRicaLanguage.

LABRI-TOOLS. 2011. ALTARICA - Wiki - ALTARICA. http://altarica.labri.fr/forge/.

LAPRIE, J.-C., ARLAT, J., BLANQUART, ET AL. 1995. *Guide de la sûreté de fonctionnement*. Cépaduès, Toulouse.

LE, M., WU, C., ZHAN, C., AND SUN, L. 2011. Airline recovery optimization research: 30 years' march of mathematical programming—A classification and literature review. *2011 International Conference on Transportation, Mechanical, and Electrical Engineering (TMEE)*, 113–117.

LEE, W.S., GROSH, D.L., TILLMAN, F.A., AND LIE, C.H. 1985. Fault Tree Analysis, Methods, and Applications - A Review. *IEEE Transactions on Reliability R-34*, 3, 194 –203.

MALEK, M. 2008. Online Dependability Assessment through Runtime Monitoring and Prediction. *Proceedings of the 2008 Seventh European Dependable Computing Conference*, IEEE Computer Society, 181–181.

MARSAN, M. AND CHIOLA, G. 1987. On Petri nets with deterministic and exponentially distributed firing times. In: G. Rozenberg, ed., *Advances in Petri Nets 1987*. Springer Berlin / Heidelberg, 132–145.

MARSAN, M.A., BALBO, G., CONTE, G., ET AL. 1995. Modelling with Generalized Stochastic Petri Nets. *Series in Parallel Computing*, John Wiley & Sons Ltd.

MASCI, P., MARTINUCCI, M., AND DI GIANDOMENICO, F. 2011. Towards Automated Dependability Analysis of Dynamically Connected Systems. *Proceedings of the 2011 Tenth International Symposium on Autonomous Decentralized Systems*, IEEE Computer Society, 139–146.

MEYER, J.F. 1992. Performability: a retrospective and some pointers to the future. *Performance Evaluation 14*, 3-4, 139–156.

MEYER, J.F., FURCHTGOTT, D.G., AND WU, L.T. 1980. Performability Evaluation of the SIFT Computer. *IEEE Transactions on Computers C-29*, 6, 501–509.

MEYER, J.F., MOVAGHAR, A., AND SANDERS, W.H. 1985. Stochastic Activity Networks: Structure, Behavior, and Application. *Proc. International Workshop on Timed Petri Nets*, IEEE Computer Society, 106–115.

MIDKIFF, A.H., HANSMAN, R.J., AND REYNOLDS, T.G. 2004. *Air carrier flight operations*. MIT International Center for Air Transportation, Department of Aeronautics & Astronautics, Massachusetts Institute of Technology, Cambridge MA 02139 USA.

MMEL. 2008. Master Minimum Equipment List - AIRBUS A-340-200/300. http://fsims.faa.gov/wdocs/mmel/a340-200-300 original 05-30-08.pdf.

MOLLOY, M.K. 1982. Performance Analysis Using Stochastic Petri Nets. *IEEE Trans. Comput. 31*, 9, 913–917.

MOUDANI, W.E. AND MORA-CAMINO, F. 2000. A dynamic approach for aircraft assignment and maintenance scheduling by airlines. *Journal of Air Transport Management 6*, 4, 233–237.

MURA, I. AND BONDAVALLI, A. 2001. Markov regenerative stochastic petri nets to model and evaluate phased mission systems dependability. *IEEE Transactions on Computers 50*, 12, 1337–1351.

NICOL, D.M., SANDERS, W.H., AND TRIVEDI, K.S. 2004. Model-based evaluation: from dependability to security. *IEEE Transactions on Dependable and Secure Computing 1*, 1, 48 – 65.

OMG. 2010. Meta Object Facility (MOF) Core Specification. http://www.omg.org/spec/MOF/2.4.1/PDF.

PAPADOPOULOS, C. AND BERNARD, D. 2008. *Decision Impact Analysis Concept Description*. AIRBUS -EDYDSU.

PAPAKOSTAS, N., PAPACHATZAKIS, P., XANTHAKIS, V., MOURTZIS, D., AND CHRYSSOLOURIS, G. 2010. An approach to operational aircraft maintenance planning. *Decision Support Systems 48*, 4, 604–612.

PERROT, B., PROSVIRNOVA, T., RAUZY, A., AND SAHUT D'IZARN, J.-P. 2010. Introduction au nouveau langage de modelisation pour la surete de fonctionnement: AltaRica nouvelle generation. *Actes du congrès LambdaMu'17*, E. Fadier.

POINT, G. AND RAUZY, A. 1999. AltaRica: Constraint automata as a description language. *Journal Européen des Systèmes Automatisés 33*, 8–9, 1033–1052.

PRESCOTT, D.R. AND ANDREWS, J.D. 2005. Aircraft safety modeling for time-limited dispatch. *Proceedings of the Annual Reliability and Maintainability Symposium*, 139–145.

RAMESH, A., TWIGG, D., AND SHARMA, T. 2008. Advanced methodologies for average probability calculation for aerospace systems. *Proc. 26th international congress of the aeronautical sciences*, 1–9.

RAUZY, A. 2002. Mode automata and their compilation into fault trees. *Reliability Engineering & System Safety 78*, 1, 1–12.

RIBOT, P. 2009. Vers l'intégration diagnostic/pronostic pour la maintenance des systèmes complexes. http://thesesups.ups-tlse.fr/664/1/Ribot_Pauline.pdf.

RUGINA, A.-E., KANOUN, K., AND KAANICHE, M. 2011. Software Dependability Modeling Using AADL (Architecture Analysis and Design Language). *International Journal of Performability Engineering 7*, 4, 313–325.

RUGINA, A.-E., KANOUN, K., AND KAÂNICHE, M. 2008. The ADAPT Tool: From AADL Architectural Models to Stochastic Petri Nets through Model Transformation. *Proceedings of the 2008 Seventh European Dependable Computing Conference*, IEEE Computer Society, 85–90.

SACHON, M. AND PATÉ-CORNELL, E. 2000. Delays and safety in airline maintenance. *Reliability Engineering & System Safety 67*, 3, 301–309.

SAINTIS, L., HUGUES, E., BES, C., AND MONGEAU, M. 2009. Computing in-service aircraft reliability. *International Journal of Reliability, Quality and Safety Engineering 16*, 02, 91–116.

SANDERS, W. 1995. The UltraSAN modeling environment. *Performance Evaluation 24*, 1-2, 89–115.

SANDERS, W. AND MEYER, J. 2001. Stochastic Activity Networks: Formal Definitions and Concepts⋆. In: E. Brinksma, H. Hermanns and J.-P. Katoen, eds., *Lectures on Formal Methods and Performance Analysis*. Springer Berlin / Heidelberg, 315–343.

SANDERS, W.H. AND MEYER, J.F. 1986. METASAN: a performability evaluation tool based on stochastic activity networks. *Proceedings of 1986 ACM Fall joint computer conference*, IEEE Computer Society Press, 807–816.

SIGNORET, J.-P., BOITEAU, M., RAUZY, A., AND THOMAS, P. 2004. Disponibilité de production: les nouveaux outils sont arrivés. *Actes du congrès Lambda Mu'14*.

SRIRAM, C. AND HAGHANI, A. 2003. An optimization model for aircraft maintenance scheduling and re-assignment. *Transportation Research Part A: Policy and Practice 37*, 1, 29–48.

TEICHTEIL-KÖNIGSBUCH, F., INFANTES, G., AND SEGUIN, C. 2011. Lazy forward-chaining methods for probabilistic model-checking. In: C. Soares, ed., *Advances in Safety, Reliability and Risk Management*. CRC Press, 318–326.

TEODOROVIĆ, D. AND GUBERINIĆ, S. 1984. Optimal dispatching strategy on an airline network after a schedule perturbation. *European Journal of Operational Research 15*, 2, 178–182.

TIASSOU, K., KANOUN, K., KAÂNICHE, M., SEGUIN, C., AND PAPADOPOULOS, C. 2011a. Operational reliability of an aircraft with adaptive missions. *Proceedings of the 13th European Workshop on Dependable Computing - EWDC '11*, 9–14.

TIASSOU, K., KANOUN, K., KAÂNICHE, M., SEGUIN, C., AND PAPADOPOULOS, C. 2011b. Modeling aircraft operational reliability. *Proceedings of the 30th international conference on Computer safety, reliability, and security*, Springer-Verlag, 157–170.

TIASSOU, K., KANOUN, K., KAÂNICHE, M., SEGUIN, C., AND PAPADOPOULOS, C. 2012a. Online model adaptation for aircraft operational reliability assessment. *ERTS2 2012*.

TIASSOU, K., KANOUN, K., KAÂNICHE, M., SEGUIN, C., AND PAPADOPOULOS, C. 2012b. *Operational reliability re-assessment during aircraft missions*. 8p.

TIASSOU, K., KANOUN, K., KAÂNICHE, M., SEGUIN, C., AND PAPADOPOULOS, C. 2012c. Impact of Operational Reliability re-Assessment during Aircraft Missions. *2012 31st International Symposium on Reliable Distributed Systems*, IEEE, 219–224.

TOMASZEK, H. AND WAŻNY, M. 2009. Operation of an aircraft with risk of its loss. *Scientific Problems of Machines Operation and Maintenance: tribology, reliability, terotechnology, diagnostics, safety 44*, 2 (158), 45–57.

VESELY, W.E. AND ROBERTS, N.H. 1987. *Fault Tree Handbook*. US Independent Agencies and Commissions.

WEI, B.C. 1991. A unified approach to failure mode, effects and criticality analysis (FMECA). In: *Reliability and Maintainability Symposium 1991 Proceedings Annual*. 260–271.

YAN, S. AND YANG, D.-H. 1996. A decision support framework for handling schedule perturbation. *Transportation Research Part B: Methodological 30*, 6, 405–419.

ZIMMERMANN, A. 2010. Dependability evaluation of complex systems with TimeNET. *Proceedings of the First Workshop on DYnamic Aspects in DEpendability Models for Fault-Tolerant Systems*, ACM, 33–34.

Résumé

Lors de la conception des avions, il est courant que les constructeurs évaluent la sûreté de fonctionnement en utilisant des modèles stochastiques, mais l'évaluation de la fiabilité opérationnelle à l'aide de modèles en ligne, pendant la réalisation des missions, reste rarement effectuée. Souvent, l'évaluation stochastique concerne la sécurité des avions.

Cette thèse porte sur la modélisation de la fiabilité opérationnelle des avions, pour aider à la planification des activités de maintenance et des missions, ainsi qu'à la bonne réalisation de ces dernières. Nous avons développé une approche de modélisation, basée sur un méta-modèle qui sert de base : i) de structuration des informations nécessaires à l'évaluation de la fiabilité opérationnelle d'un avion et ii) pour la construction de modèles stochastiques pouvant être mis à jour dynamiquement. La mise à jour concerne l'état courant des systèmes avion, un profil de mission et les moyens de maintenance disponibles dans les diverses escales incluses dans le profil de la mission. L'objectif est de permettre l'évaluation de la fiabilité opérationnelle en ligne. Deux cas d'études, basés sur des sous-systèmes avions, sont considérés à titre d'illustration. Nous présentons des exemples de résultats qui montrent le rôle important de l'évaluation de la fiabilité opérationnelle pendant une mission d'avion.

Mots-clés : évaluation en opération - fiabilité - maintenance - modélisation stochastique - planification mission - Système avionique