Aircraft Operational Reliability-A Model-based Approach and a Case Study

Kossi Tiassou, Karama Kanoun, Mohamed Kaâniche, Christel Seguin, Chris Papadopoulos

Aircraft Operational Reliability — A Model-based Approach and a Case Study

Kossi Tiassou[1, 2], Karama Kanoun[1, 2], Mohamed Kaâniche [1, 2], Christel Seguin[3], Chris Papadopoulos[4]

[1] CNRS, LAAS, 7 Avenue du Colonel Roche, BP 54200, 31031 Toulouse Cedex 4, France

[2] Université de Toulouse, LAAS, BP 54200, 31031 Toulouse Cedex 4, France

{*firstname.lastname*}@laas.fr

[3] ONERA/DCSD/CD, 2 Avenue Edouard Belin, 31055 Toulouse Cedex 4, France

christel.seguin@onera.fr

[4]AIRBUS Operations Ltd., New Filton House, Golf Course Lane, Filton, Bristol, BS99 7AR, United Kingdom

Chris.Papadopoulos@Airbus.com

Abstract. The success of an aircraft mission is subject to the fulfillment of some operational requirements before and during each flight. As these requirements depend essentially on the aircraft system components and the mission profile, the effects of failures can be very severe if they are not anticipated. Hence, one should be able to assess the aircraft operational reliability with regard to its missions in order to be able to cope with failures. We address aircraft operational reliability modeling to support maintenance planning during the mission achievement. 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 tuned dynamically, in order to take into account the aircraft system operational state, 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. A case study, based on an aircraft subsystem, is considered for illustration using the Stochastic Activity Networks (SANs) formalism.

Keywords: operational reliability, model-based assessment, dependability re-assessment, maintenance planning, aircraft mission planning

1 Introduction

In the early days of aviation, dependability analysis for maintenance planning was not an essential concern, and maintenance was performed only when it was actually needed. Later, the need to improve the dependability of aircraft systems appeared and rigor- ous maintenance programs were required for aircraft operations. A time-based maintenance approach was first used to ensure safety. Needs for cost saving have then led to the new approaches based on reliability-centered maintenance [1]. The developed maintenance programs are aimed at ensuring the inherent design reliability of the aircraft systems [2]. However, there are still unforeseen failures in service, which disrupt the successful achievements of the missions. Yet the increasing demand in air transportation is continuously calling for an enhanced delivery of service. To avoid economical losses due not only to aircraft inoperability but also to customer dissatisfaction, airlines need to anticipate on the events that may disrupt the achievement of their aircraft missions.

Currently, decisions are taken concerning the ability to successfully achieve each upcoming flight. We intend to make it possible to take decisions concerning a full mission composed of multiple successive flights, based on stochastic modeling, without discarding the existing flight dispatch process.

Our work aims at developing an assessment approach, based on dependability modeling, that makes it possible to assess the ability of the aircraft to keep operating up to a given time or location. The model is to be used for planning a mission and during its achievement. To plan the mission, the model will be used to estimate the period of time during which the aircraft system can be operated without reaching adverse states. This assessment is aimed at providing support for the assignment of possible mission profiles to the aircraft (together with other operational criteria that are not discussed in this paper). Once a mission is assigned to the aircraft, the model will be used during the mission to assess the ability to succeed in continuing on the remaining part of the mission, in case of an unscheduled event occurrence. The model can also support maintenance planning by comparing the operational reliabilities related to different maintenance alternatives.

To this end, we develop a stochastic model that can be dynamically configured during the aircraft mission to represent the current state of the aircraft, with regard to the current mission, to enable an assessment of aircraft operational reliability on-the-fly.

The assessment results from the configuration and processing of a stochastic dependability model that is (and can only be) built and validated off-line in a way that makes it easily and very quickly configurable (i.e., tuned and updated). During operation, the configuration of the model should not require the knowledge of the underlying modeling techniques, and requires only a good knowledge of the system architecture and behavior. The model is to be configured by operators who have been trained to this purpose. However, the building and validation of the stochastic model requires both a deep knowledge of dependability modeling techniques, and of the

detailed system architecture and behavior in the absence and in the presence of faults, and under various conditions. Hence, the stochastic model can be built only by modeling specialists, with the support of the system manufacturer.

The stochastic dependability model can be built using classical dependability modeling formalisms such as Petri Nets and their off- springs, either directly [3,4] or based on model transformation from, e.g., Architecture Analysis and Design Language (AADL) [5,6], Stochastic Activity Networks (SANs) or the AltaRica language [7,8]. The latter is of common use at Airbus. The only requirement is that, once the model is built, it can be easily tuned from outside and processed efficiently. For the final usage, the model will be based on the AltaRica language. The model-processing module is proprietary and is still under development. To obtain quick results allowing us to check the validity of the proposed approach, before the end of the implementation of all the proprietary modules, we have used the SAN formalism, and its associated Möbius tool [9]. It is worth to mention that the two formalisms are equally expressive [10].

The stochastic model is to be built only once for a specific aircraft. Hence, a special attention is to be paid to the validation process, as it is not possible to support a thorough model validation during aircraft operation, due to time limitations (except some light checks, such as consistency checks). It is thus of prime importance to build the model following a well-defined process, accompanied by a thorough validation process. In our approach, we recommend to first develop a meta-model to structure the information needed to build the stochastic model (general information about an aircraft system, its possible missions, and maintenance policies). Moreover, considering the fact that this kind of assessment will be used for supporting maintenance planning for several families of aircrafts from the same manufacturer, the meta-model provides a common approach to support model construction for several systems, in a unified approach.

In this paper, we will first present the main elements of the meta-model. Then we show how it can be used for the specification of a model corresponding to a particular aircraft system. Then we will model this particular system using the SAN formalism, before presenting examples of evaluation results.

This paper extends our previous work presented in [11,12] by including in particular the meta-model for the specification and description of the dependability model used for aircraft operational reliability assessment. It is structured as follows. Section 2 presents the modeling and assessment context. Section 3 presents the meta-model. Section 4 proposes a case study and gives the basis for the construction of a corresponding model, based on the meta-model. A concrete implementation of the modeling approach is given in Section 5 using the Stochastic Activity Networks (SANs) formalism. Section 6 presents examples of evaluation results. Section 7 gives some related work. Finally, Section 8 concludes the paper.

2 Modeling and Assessment Context

A mission is composed of a given number of sequenced flights. The achievement of the mission is such that each flight is followed by a stop where the aircraft is prepared for next flight. The preparation consists of routine maintenance activities, cabin cleaning, catering, baggage and cargo processing, and passenger boarding. At each stop, the aircraft is inspected and the discrepancies that are reported during the previous flight are checked. If a component is found inoperative, a dispatch decision is taken regarding the next flight. Fig. 1 summarizes the possible outcomes of the dispatch decision.

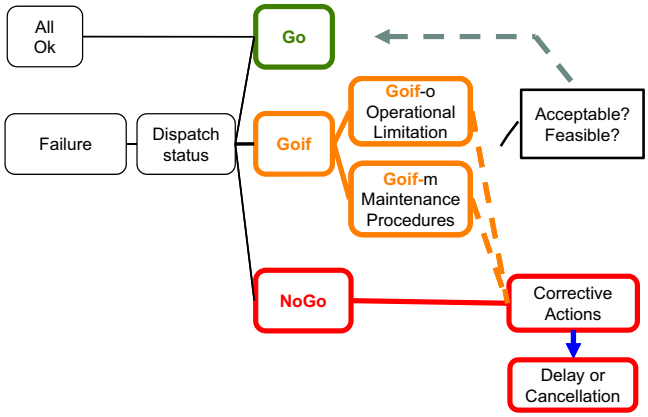

Fig. 1 Dispatch status outcomes

The flight captain refers to an approved document called Minimum Equipment List (MEL) where the components are listed with the status “go”, “go if” or “no go”:

- The “go” status is the case when the aircraft can fly with the failed component.
- The “go if” status allows the flight provided some conditions (on other components, operational performance or maintenance activities) are fulfilled. This includes a given deadline to repair the component.
- The “no go” status prevents the aircraft from flying. The failed component must be repaired before any flight.

The dispatch is allowed if there is no “no go” and all “go if” conditions are satisfied. When the aircraft does not meet the dispatch requirements following a failure, maintenance activities are initiated in order to solve the problem. The severity level of the failure depends thus on the ability to fix the problem at the considered location before the planned departure time. Actually, the flight is considered delayed only after exceeding a given tolerable time frame.

When the dispatch is allowed, the aircraft can depart after passenger, cargo and the other ground service processing. Then, the flight begins by the taxing of the aircraft to runway where the takeoff is initiated. During this period or even after the takeoff, the flight can be aborted as a result of a critical failure. The aircraft then returns back to

the departure airport. During the entire flight, it may be diverted if the aircraft capability is degraded. Procedures, stated in the Flight Manual (FM), the Flight Crew Operating Manual or the Quick Reference Handbook, are used to determine whether the flight must be diverted or not [13].
The adverse situations while operating an aircraft correspond to operational interruptions, consisting of flight delays, cancellations, flight turn-back and diversions. Delays and cancellations take place before the start of a flight, while turn-back and diversion occur during the flight.

2.1 Dependability Requirements and Associated Measures

Two categories of requirements are provided, related respectively to the system and to the mission:

- *Min_Sys_Req*, the minimal system requirements are given by the MEL. They are independent of the mission profile and must be fulfilled in order to operate the aircraft whatever the mission.
- *M_Prof_Req* the mission profile requirements. These include the mission dependent dispatch requirements and the in-flight requirements. They are composed of the requirements specific to the flights composing the mission.

These requirements are checked systematically after inspection of the aircraft and also during the flight after the occurrence of an unscheduled event such as the failure of a component. If these requirements are not satisfied, the next flight is not authorized. Currently, the dispatch decision concerns mainly the next flight. We aim at going further and make, in addition, decisions concerning the subsequent flights in order to anticipate maintenance activities. Thus, our objectives are (i) to assess how long the requirements will be satisfied and/or (ii) to assess the probability that they will still be satisfied at the end of the planned mission. If this probability is below an acceptable threshold, an action should be undertaken.
Hence, our modeling approach is aimed at assessing the probability of aircraft operation without operational interruptions due to the non-fulfillment of one of the above requirements. We have defined two reliability measures associated with the follow-ing requirements:

- the *system reliability* (*SR*): evaluated with regards to *Min_Sys_Req*. It is used while planning a mission, to determine the maximum number of flight hours that can be achieved without maintenance (or equivalently, during which the requirements will still be satisfied without any maintenance action). It can thus be used to help determine the length of the mission (or to plan maintenance activities).
- the *mission reliability* (*MR*) which corresponds to the probability to achieve the mission with success, i.e., the requirements will be satisfied all along the mission (or, in other terms, the mission will be accomplished without an operational interruption). It is evaluated with regards to *Min_Sys_Req* and *M_Prof_Req*. *MR* is to be used during the achievement of a mission, after the occurrence of a major unscheduled event, to determine whether a preventive action must be initiated or not, and when it should be done.

It is worth to mention that, in some situations, one may analyze only the reliability of a given aircraft function that is identified to be critical for the mission.

2.2 Essential Information Necessary for Building the Model

Modeling requires knowledge about the states of the subsystems, components and their associated functions, the mission of the aircraft, as well as the maintenance facilities. Fig. 2 gives an overview structuring the relevant categories of information considered for model construction.

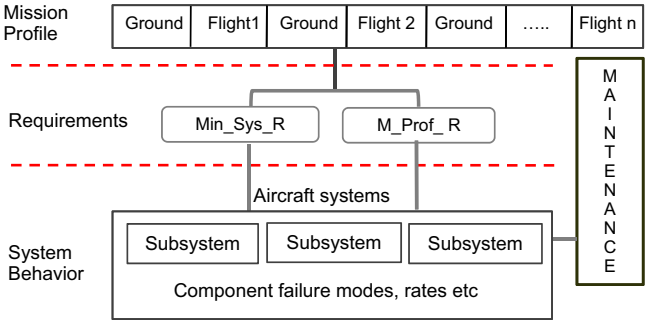

Fig. 2 Categories of information

The information considered is related to system behavior, mission profile, requirements and maintenance.

Mission profile: It is composed of information related to the succession of periods during which the aircraft is either flying or on ground (where maintenance can take place), as well as the number and duration of the flights included in a mission.

System Behavior: The aircraft system is composed of subsystems and atomic components, which provide different important functions. The description of components' failure and recovery scenarios, as well as the possible resulting function loss, represents a fundamental point in the model construction.

Requirements: These requirements are the representation of *Min_Sys_Req* and *M_Prof_Req*, usually formulated as complements of Boolean expressions, representing the different combinations leading to an operational interruption.

Maintenance: It concerns the maintenance possibilities (in terms of maintenance resources) at the various stops involved in the mission profile. This impacts directly the maintenance time of the system components at a given stop.

Our modeling approach consists in structuring and combining the above information to form a dependability model that will be processed to derive the two dependability measures, *SR* and *MR*.

2.3 The Model Construction Process

As stated in Section 1, our approach is to build first a meta- model to structure the information that will be used in the stochastic dependability model (referred to as stochastic model) that can be easily and quickly tuned and updated online to derive the up to date operational dependability model (referred to as up to date model). Fig. 3 summarizes the various steps for the up to date model construction and tuning processes.

Different types of changes may take place during the achieve- ment of an aircraft mission and lead to the need to update the dependability model in operation. Such changes can be related to (1) the state of system components, (2) the components' failure distributions, (3) the mission profile (i.e., the number of flights, their sequencing or duration), or (4) the 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.

In this paper, we will mainly address the building of the stochastic model, without going further into details concerning model tuning and update (more details are given in [12,10]).

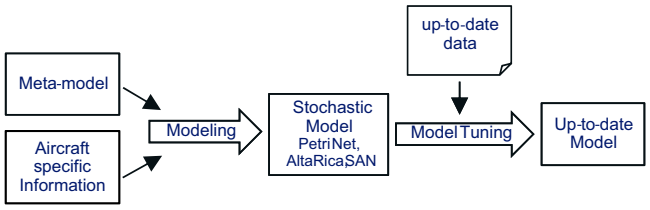

Fig. 3 Model Construction Process

2.4 Model Structure

As stated in Section 1, the stochastic model can be built only by a specialized team with the support of the aircraft manufacturer who is the only entity knowledgeable about the system. Model update should be possible by operators who do not necessarily have a stochastic modeling culture, and a deep knowledge about the internal functional dynamics of the system, and who have been trained for this purpose. They will make the update based on established procedures and a convenient interface, according to the occurrence of the unscheduled event(s). Hence, it is essential to structure the stochastic model in such a way that it will allow an easy and quick update.

The stochastic model is composed of two main parts: a core model that is dedicated to the modeling of system behavior, together with its system minimal requirements, and a mission dependent model that is associated with the mission profile and maintenance facilities, and which may be changed without affecting the ability to use the core model for dependability assessment. The dependency between the two parts

is to be clearly specified and modeled in an internal well-defined interface. The structure of the core model is mission independent, whereas all the changes in the model structure affect only the mission dependent model.

3 The Meta-model

The meta-model is an abstract representation of various stochastic models corresponding to the different families of air- craft. A meta-model is associated with each part of the depend- ability operational model (i.e., the core model and the mission dependent model).

We will first present the main notations used for the meta- model, then we will present successively the meta-models asso- ciated with the core model (system and system requirement representations) and the mission dependent model (mission-profile, maintenance, and mission requirements model).

3.1 Notations used for the Meta-model

In our approach, the meta-model is based on the Ecore features of the Eclipse Modeling Framework (EMF). Ecore is principally based on Unified Modeling Language (UML) class diagrams. Fig. 4 shows the graphical representation of the main 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.

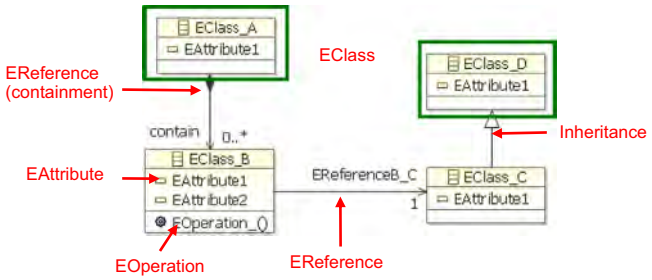

Fig. 4 Ecore features

3.2 The Core Meta-model

The core meta-model results from the composition of the system meta-model with the system requirement meta-model. For sake of effectiveness, the mission and the system requirements are represented following the same meta-modeling approach. After presenting the system meta-model we present the requirements meta-model.

The system to be modeled consists of elementary components which are Line Repairable Units (LRUs) that interact so as to provide functions that are required in the achievement of a flight as described in Section 2. Examples of LRU are flight control computers, hydraulic and electrical power generators. Generally, the LRUs are part of distinguishable subsystems (for example flight control subsystem and electrical supply subsystem) dedicated to high-level 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 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 system components. Therefore, the description of an LRU must take into account its dependencies on other LRUs.

While in service, the LRU is subject to errors and failure events that lead to changes of its 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.

To sum up, we have to consider system components characterized by their possible states, their dependencies, and the events that affect their states.

The computation of the dependability measures depends on qualitative data (e.g., description of the effects of events on the component states, and definition of the dependencies resulting from the failure propagation between components), as well as quantitative data (e.g., event occurrence rates).

The model will be using prognosis information and maintenance duration estimation to characterize failure and maintenance events. Generally these events are characterized by the duration until their occurrences, using probabilistic distribution.

Finally, it is worth pointing out that some model parts need updates during the aircraft operation and this constraint will impact the granularity of their definition.

Fig. 5 presents the proposed meta-model.

A system component is characterized by its identifier, its state variables, and the events it may be subjected to.

- The *identifier attribute id* will help in managing the component updates online.
- The state variables represent the different conditions in which the component may be. A state variable has a name, a domain, which defines the possible values

resulting from the changes of the LRU state, and an initial value that corresponds to its initialization in the model.

- The main *events* for our case are failures and maintenance events. An Event is described by i) its *name*, ii) the *guard* representing the condition under which it may happen, iii) the *effect* representing the changes in the state of the system after its occurrence (either on the component itself or on other components), iv) the occurrence time distribution (*TTOdistrib*). *TTOdistrib* is an object of the EClass *DurationDistrib,* which is aimed at representing the distribution law followed by the time duration spent in a given situation. The distribution is described by the name of the distribution law (e.g., exponential, Weibull, deterministic) and its parameters (each parameter has a name and a value).

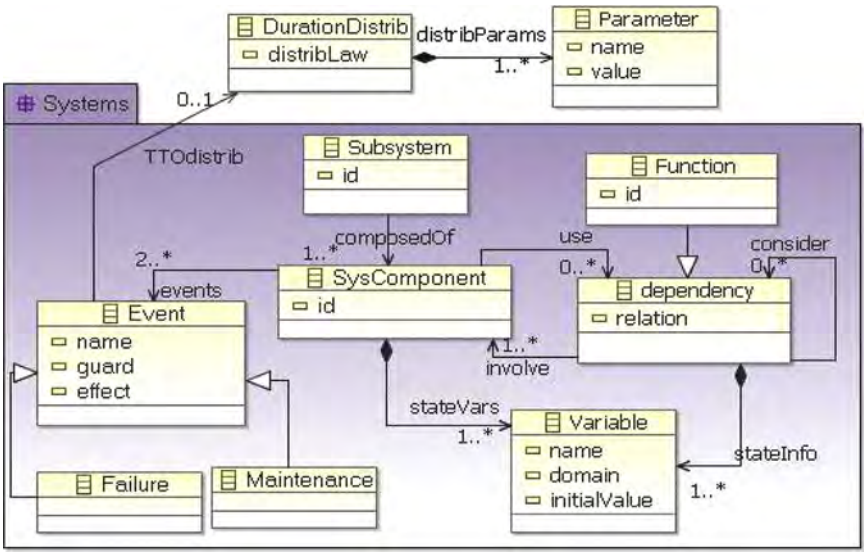

Fig. 5 The System meta-model

A system' component behavior may depend on other compo- nents. Concretely, the component may have to use other compo- nent's attribute value in its behavioral description. This is the purpose of the EClass Dependency, whose objects associate informa- tion from different components to produce, based on a combination function (relation), information as value of a variable (stateInfo) necessary in a component or a function description. It is also used to represent functions provided by the system components. These functions are used, in the same way as the system components state information, to define requirements related to the system. Fig. 6 illustrates the requirements representation.

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 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 need to be satisfied otherwise an interruption may occur during the mission achievement. It can be based on combination of other requirements.
System requirements do not need any particular information from the mission profile. They are expressed once and for all. For example, all no go components can be stated once and for all. 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 core model, set with these objects, can be used to evaluate the probability to operate the system, according to some system requirements, during a given time, i.e., to assess the system reliability (SR) measure defined in Section 2.

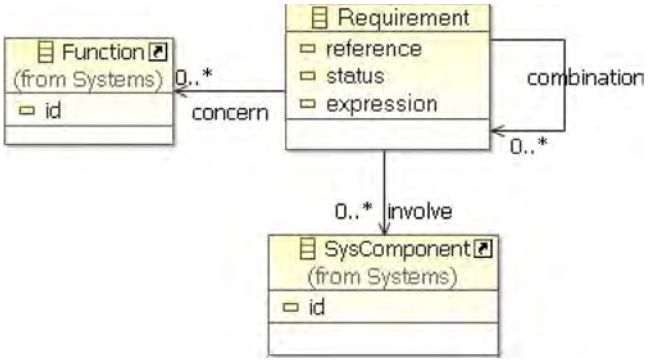

Fig. 6 System requirements meta-model

3.3 The Mission Dependent Meta-model

Fig. 7 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.
For the achievement of each flight, 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* represents an interfacing object between the core and the mission dependent models.

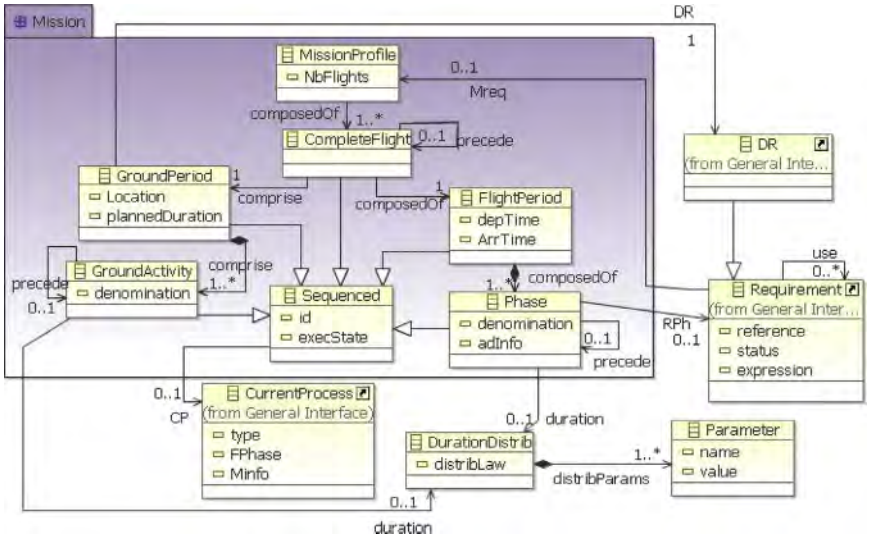

Fig. 7 Mission-profile meta-model

On ground, the operational state of the system is tested against dispatch requirements (*DR*), which are the synthesis of the *MEL* (see Section 2.1). *DR* corresponds to *Min_Sys_Req* 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 scheduled duration. 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 maintenance activities depends on the availability of adequate logistic in the considered ground period. A maintenance station is associated to each ground period and the dependence is taken into account by considering a logistic delay function (*LDF*), which represents the delay in supporting the activities. *LDF* is a subclass of *DurationDistrib.* The tasks are carried out considering a *prioritization* in the repair of failures. This is illustrated in Fig. 8.

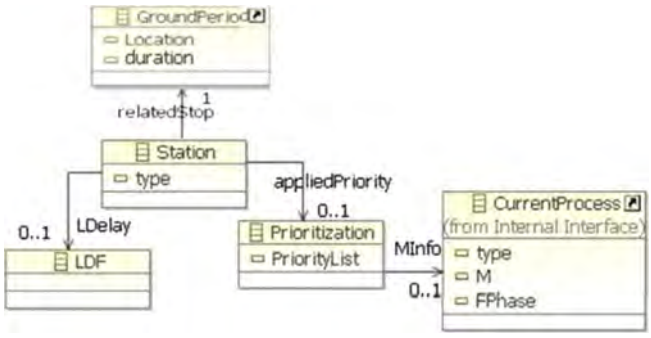

Fig. 8 Maintenance meta-model (considering the ground period)

3.4 Concluding comments

It is worth mentioning that the meta-model presented in this section is intended to show the philosophy of the approach. The attributes of the classes defined may be enriched with other specific information. Only the aircraft manufacturers have a complete knowledge of the system and of possible operations. The model can be built for a particular single subsystem, or for several subsystems grouped together, mainly if they are strongly dependent.

The main purpose of the meta-model is to support the construction of the sub-models and their integration. However, it can also be used for different purposes. In particular, it can be seen as a means for giving an insight into the general content of the model. Moreover, the meta-model can be used for training the teams that will be in charge of updating and tuning the stochastic model during aircraft operation.

The following section presents a case study based on a subsystem of an aircraft together with the specification of the core and mission meta models, that will be used for the construction of the corresponding stochastic model in Section 5, as an instance of the meta-model.

4 Case Study

The case study concerns the Control of the Movable Surfaces (CMS) subsystem of the aircraft [14]. In the following sections, the subsystem is described before presenting the specifications of the core model and of the mission dependent model following the meta modeling approach of Section 3.

4.1 Presentation of the CMS

Fig. 9 presents the subsystem that is 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 are in charge of moving the surface. The connection between a computer and a servo-control forms a control line that can act on the surface. We distinguish the following control lines:

- P1 control line (PL1): connects P1 and ServoCtrl_G,
- P2 control line (PL2): connects P2 and ServoCtrl_B,
- P3 control line (PL3): connects P3 and ServoCtrl_Y,
- S1 control line (SL): connects S1 and ServoCtrl_G.

There is also a Backup control line (BCL), which is based on BCM, BPS_B, BPS_Y, ServoCtrl_Y and ServoCtrl_B.

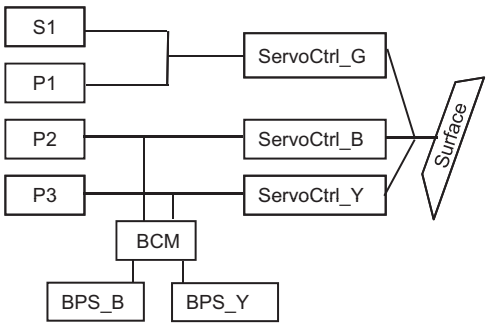

Fig. 9 The CMS subsystem

Initially the secondary computer S1, the backup control module BCM and the backup power supplies BPS_B and BPS_Y are inhibited. The surface 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 SL 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). Fig. 10 summarizes the control modes.

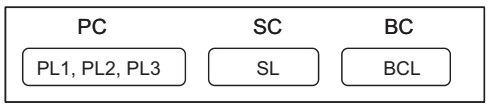

Fig. 10 The control modes and associated control lines

Related Operational Requirements: According to [15][1], the no-go and the go-if statuses are defined as follows:

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: P1, P3 and S1 are "go if" items with "go if" conditions stated as follows[2]:

- Conditions associated with P1: P3 and S1 are operational.
- Conditions associated with P3: P1 and S1 are operational.
- Conditions associated with S1: P1, P2 and P3 are operational.

There is no mission profile requirement related to the subsystem in the Flight Crew Operating Manual.

4.2 Specification of the CMS Core Model

This specification shows how to use the features of the core meta-model presented in Section 3 for the derivation of the core model.

All the components have similar behaviors and are represented using the features related to the Eclass SysComponent (Fig. 5).

The name of each component is used as its identifier (id).

For each component x, a state variable is considered, with a domain defined by the set {ok, failed}; 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 the guard expression of the maintenance event, author- ization information from the maintenance policies is needed to enable the event.

For example for P1, the events are defined as follows:

Failure event:

Name: P1_failure
guard: P1-state=ok; effect state=failed
TTOdistrib: distribLaw=exponential, parameter: lambda=1E-4

Maintenance event:

Name: MaintainP1
guard: state= failed and id ∈ CP_M-allowed; effect state=ok
TTOdistrib: distribLaw=exponential, parameter: lambda=1

For the secondary computer S1 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

[1] [15] is actually a Master MEL (MMEL) established by the aircraft's manufacturer. MELs result from the completion of MMELs with airline specific policies and they are not public.

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

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

PL1-stateInfo =ok if P1-state =ok and ServoCtrl_G-state =ok
otherwise PL1-stateInfo =failed.

It is worth noting that this is just an expression that determines the line state. In practice, and depending on the formalism chosen for the construction of the model, one may choose to set directly the value of *PL1-state* in the specification of the events that modify *P1-state* and *servo-control_G-state.*

The requirements expressed at the end section 4.1 are not dependent on any mission profile. They are part of the minimum requirements (*Min_Sys_Req*), to which may be added requirements specific to a given mission profile. They are expressed using the features defined in the meta-model of Fig. 5.

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_Req* is formulated as the conjunction of OR1 and OR2 (see section 4.1):

OR1: The condition related to the "no go" components is as:
P2 =ok ∧ ServoCtrl_G =ok ∧ ServoCtrl_Y =ok ∧ ServoCtrl_B =ok ∧ BCM =ok ∧ BPS_B =ok ∧ BPS_Y =ok

OR2: The operational conditions related to the "go if" components are expressed as follows
- (*P1=ok*) ∨ (*S1=ok ∧ P3=ok*);
- (*P3=ok*) ∨ (*S1=ok ∧ P1=ok*);
- (*S1=ok*) ∨ (*P1=ok ∧ P2=ok ∧ P3 =ok*).

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

$$\begin{aligned} Min\_Sys\_Req = \{ & P2{=}ok \wedge ServoCtrl\_G = ok \wedge ServoCtrl\_Y = ok \wedge \\ & ServoCtrl\_B{=}ok \wedge BCM{=}ok \wedge BPS\_Bok \wedge BPS\_Y = ok \\ & \wedge (P1 = ok \vee (S1 = ok \wedge P3 = ok)) \wedge \\ & (P3 = ok \vee (S1 = ok \wedge P1 = ok)) \wedge \\ & (S1 = ok \vee (P1 = ok \wedge P2{=}ok \wedge P3 = ok)) \} \end{aligned} \quad (1)$$

Using the control lines, whose states are derived from combinations of at least two basic components' states, the expression becomes:

$$\begin{aligned} Min\_Sys\_Req = \{ & PL2 = ok \wedge BCL = ok \wedge \\ & (\mathrm{PL1} = \mathrm{ok} \vee (\mathrm{PL3} = \mathrm{ok} \wedge \mathrm{SL} = \mathrm{ok})) \wedge \\ & (PL3 = ok \vee (PL1 = ok \wedge SL = ok)) \wedge \\ & (SL = ok \vee (PL1 = ok \wedge PL3 = ok)) \}. \end{aligned} \quad (2)$$

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

4.3 Specification of the CMS Mission Dependent Model

The mission dependent model is specified using the features defined in Fig. 7. For the mission profile, let us consider *p* flights per day, during *d* days. One has to create instances of *CompleteFlight* corresponding to these flights. For their identification (*id*), they can be numbered. For each of them, a ground period with *gpd* as planned duration is considered.
The ground period comprises a period of scheduled maintenance activities whose duration *SM_Time* can be considered deterministic with a value of *smd* hours.
The scheduled maintenance is extended by an unscheduled maintenance, which generally takes place when the dispatch requirements are not satisfied.
The other activities during the flight preparation are considered to have a given duration *oad*.
The flight periods following the ground 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 durations of these phases.

We assume that maintenance activities take place every night at the base stations of the airline company.

5 The Model Based on SAN Formalism

The SAN model is built based on the meta-model structured information presented in Section 4. A brief description of the SAN formalism is given in the following, before presenting the core and the mission dependent models.

5.1 SAN Formalism

Stochastic activity networks are an extension of Petri nets (PN). SANs consist of four primitive objects: places, activities, input gates, and output gates. Activities are the equivalent of transitions in PN. They are either timed or instantaneous. Timed activities have durations and a time distribution function. Instantaneous activities represent actions that complete immediately when enabled. Input gates are used to

control the enabling of activities and define the marking changes that will occur when an activity completes. Each input gate is defined with an enabling predicate and a function. Output gates are like input gates and are used to change the state of the system when an activity completes. An output gate is defined only with a function. The function defines the marking changes that occur when the activity completes. Input gates and output gates are represented graphically as triangles (see Fig. 11).
An activity is enabled when the predicates of all input gates connected to the activity are true, and all places connected to incoming arcs contain tokens, i.e., have non-zero markings. Once enabled, the activity samples its delay distribution function to determine the time delay before the activity fires. When the activity fires, the state of the model is updated by subtracting tokens from places connected by incoming arcs, adding tokens to places connected by outgoing arcs, and executing the functions in input and output gates.
The Möbius tool allows the construction of composed models. Indeed, for a large system, it may be helpful to compose the overall model based on sub-models that have less complexity. This is feasible using the Join and Replicate operators. The Join operator combines several models sharing some state variables. The Repli- cate operator is used to create copies of models; the copies are combined into a global model. The copies may hold some state variables in common. A Join node may have other Joins, Replicates, or other sub-models defined as its children.
The SAN model of the case study is presented in the following, considering first the core model then the mission dependent model.

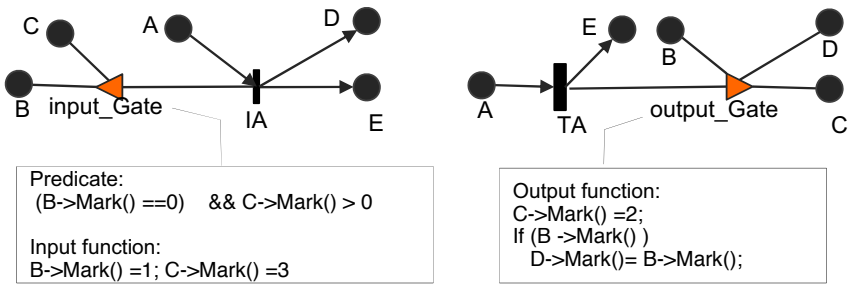

Figure 11 Input and output gates

5.2 The SAN Core Model

The core model consists of the CMS subsystem and its related requirements representation. The subsystem model is decomposed into three sub models corresponding to the control modes given in Fig. 10. Fig. 12 shows the core model structure. The subsystem interface is made of the control lines states, which are used to express explicitly the operational requirements.

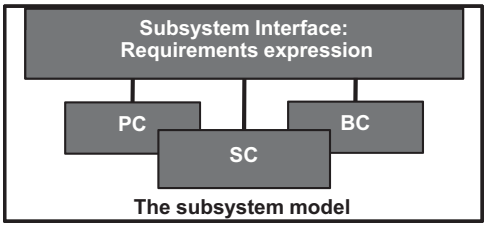

Fig. 12 Core model structure

Concerning the content PC, SC and BC sub models, as stated in Section 4.2, the subsystem components are identified by their name as presented in Fig. 9. The marking of their corresponding places represents their state value defined in Section 4.2 (*ok* or *failed*). Activities named *x_failure* represent failure events associated to component *x*. Their enabling is conditioned by the presence of a token in place *flight.* Activities *Maintainx* represent maintenance and their enabling is conditioned by the presence of a token in place *CP_M*. Place *flight* (respectively, *CP_M*) represent whether a flight (respectively, a maintenance) is ongoing or not. Their markings are controlled by the mission dependent 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.

The **Primary control (PC)** sub model is given in Figure 13.

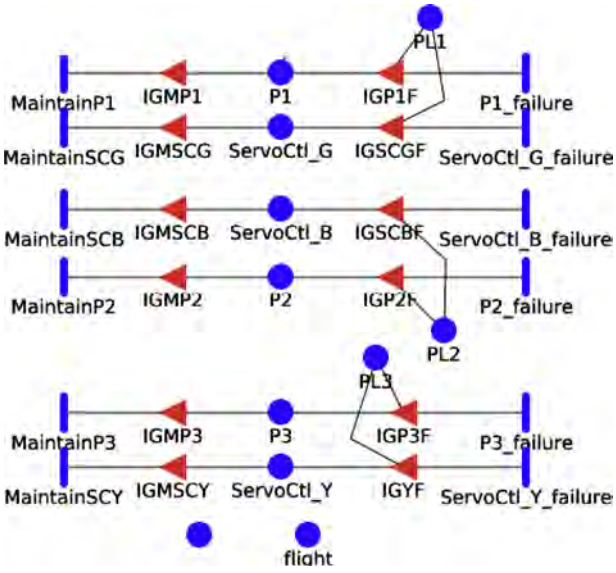

Fig. 13 PC sub model

The transitions representing the maintenance activities (*Maintainx*) are at the left side and the failure events (*x_failure*) at the right side of the places. Their associated

input gates control their firings. For example *IGP1F* and *IGMP1* are defined as follows:

IGP1F Predicate : P1->Mark() && flight->Mark() Function : P1->Mark()=0; PL1->Mark()=0;
IGMP1 Predicate : P1->Mark()==0 && CP_M->Mark() Function : P1->Mark()=1;
if (ServoCtrl_G->Mark()) PL1->Mark()=1;

Places *PLi* represent the state of the lines *PLi*. *PLi* is marked when *Pi* and the corresponding *ServoCtrl_x* in the line are marked. The markings of places *CP_M* and *flight* are used in the predicates of the input gates to enable the failure and maintenance activities as explained above.

The **Secondary control (SC)** sub model is represented in Figure 14. Place *S1Active* represents the activation state of S1. 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*. *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 is marked when S1 and ServoCtrl_G are operative. *ServoCtrl_G* is shared with PC sub model.

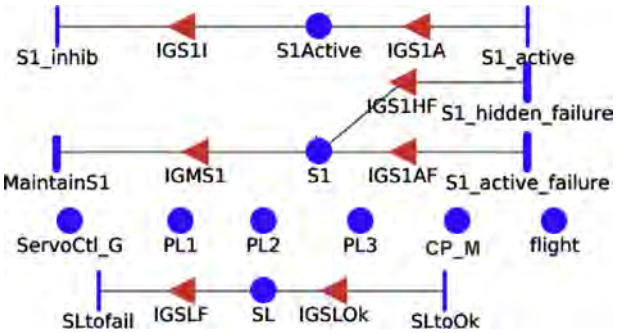

Fig. 14 SC sub model

The **Backup control (BC)** sub model is depicted in Figure 15. *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 based on the control line components states. It is marked when *BCM, BPS_B, BPS_Y, ServoCtrl_B* and *ServoCtrl_Y* are marked. Places *CP_M* and *flight* are shared with the mission dependent model; *PL1, PL2, PL3, ServoCtrl_B* and *ServoCtrl_Y* 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.

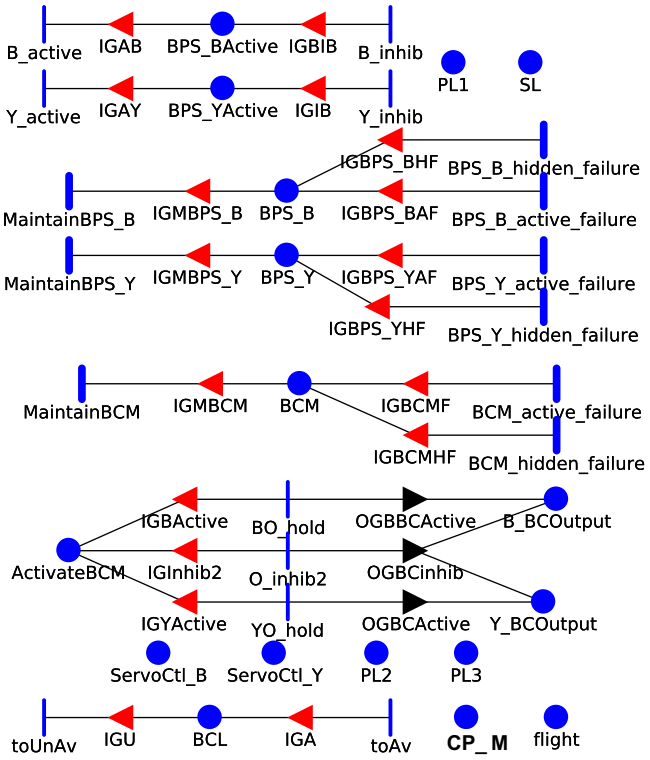

Fig. 15 BC sub model

The core model of the CMS subsystem results from the composition of PC, SC and BC sub models. The places representing the states of the lines are used to express the related requirements. Fig. 16 shows the portion of the model that represents the expression of these requirements.

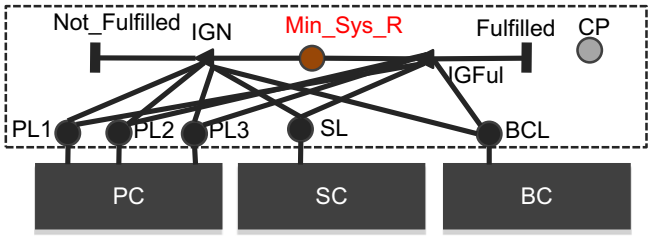

Fig. 16 System requirements expression

Place *Min_Sys_Req* models the fulfillment of the requirements. The firings of the instantaneous activities *Fulfilled* and *Not_ Fulfilled* update the place according to the satisfaction of Equation (2) derived in section 4.2. *Min_Sys_Req* is used to make the connection with the mission dependent model as shown in Fig. 17.

5.3 The SAN Mission Dependent Model

In this case study, it is assumed for the sake of illustration that the mission is composed of basic identical flights. The mission profile model is shown in Fig. 17, together with its connection with the core model via the sub-model of Fig. 16.
The upper part of the mission dependent model represents a flight and the lower part represents the activities on ground at a stop.
A flight is represented by the three phases defined in Section 4.3 (*Taxing_to_Takeoff*, *In_Flight* and *Landing)*. 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 *Diversion_Condition* represent the conditions under which these interruptions can occur (in-flight requirements fulfillment). The conditions are stated using the marking of *Min_Sys_Req,* which is assumed to be the related requirement for illustration purpose. Place *flight* indicates whether a flight is ongoing or not.

The sub model of a ground describes 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 (routine check for instance)[3]. 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. In the current illustration, the dispatch requirements

[3] These tasks are aimed at detecting failures, and not to repair any failed component.

fulfillment consists of testing the marking of *Min_Sys_Req*. Until then, the scheduled ground duration may have elapsed (firing of activity *GroundPeriod_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 *Profile* (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).

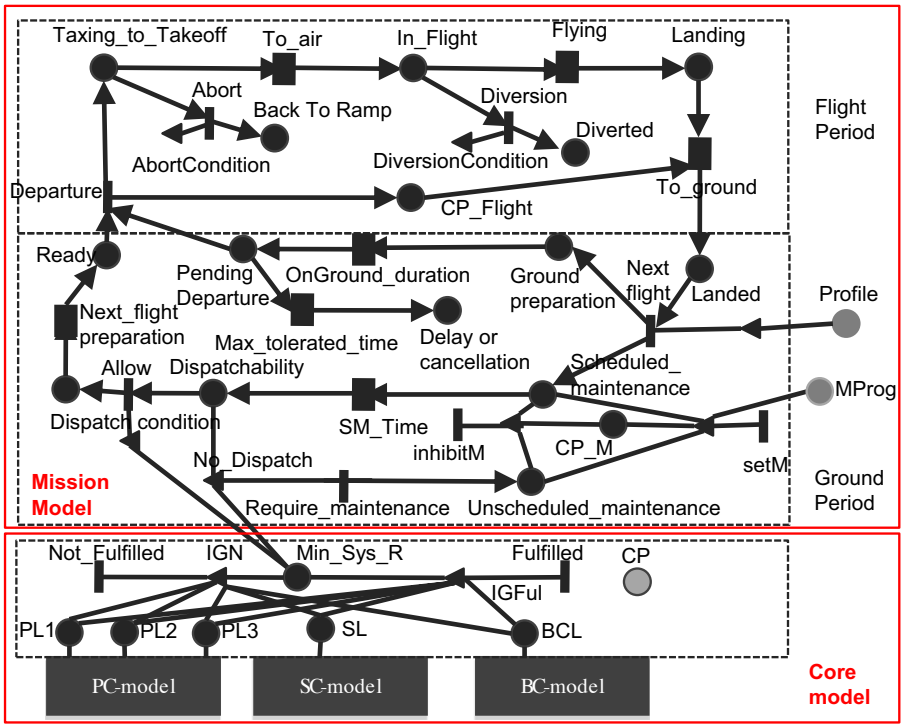

Fig. 17 Overview of the stochastic model

6 Example of Results

To process the model and get evaluation results for *SR* and *MR* measures, one has to set the initial markings of places and distribution laws of the timed activities. In this section, we assume that all the failure events corresponding of the system components have exponential distributions. To perform sensitivity analyses, failure rates between 10^{-4}/hour and 10^{-6}/hour are assumed.

6.1 System Reliability

The system reliability measure, *SR*, corresponds to the probability that *Min_Sys_Req* holds during a given period of time. *SR* measure concerns only the core model. This measure can be used to help assigning a mission to the aircraft, based on the fact that *SR* should not be below an acceptable threshold (referred to as the Minimal Reliability Requirement, *MRR*). *MRR* is set by the airline company, in agreement with the aircraft manufacturer. For the sake of illustration, we have considered *MRR* = 0.975. It is worth to mention that, in order to preserve the industrial confidentiality, the set of values used in this section have been selected to form a consistent set, without disclosing the industrial property.

Curve A of Fig. 18 shows the system reliability obtained from processing the core model, with 95% of confidence level. It shows that the maximum mission duration, without maintenance activities, should be less than 95 flight hours, to satisfy the considered value for *MRR*. This assessment assumes that all system components are initially operative (i. e., at the start of the mission).

Curve B of Fig. 18 assumes that computer P1 is in failure at the start of the mission. It shows that in order to satisfy *MRR*, the maximum mission duration, without maintenance activities, should be less than 45 flight hours.

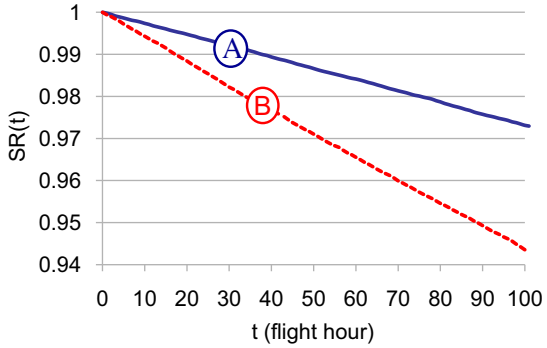

Fig. 18 System Reliability

6.2 Mission reliability

The mission reliability *MR* is evaluated taking into account both *Min_Sys_Req* and *M_Prof_Req*. *MR* corresponds to the probability to have no tokens in places *Delay_Or_Cancellation*, *Back_to_Ramp* and *Diversion* of Fig. 17. *MR* is usually evaluated at the start of a mission and after occurrence of a major event to check if a preventive action should be undertaken or not, and when.

For illustration purpose, we consider a typical mission composed of 4 identical flights per day over a week. We assume that the timed activities of the mission model have deterministic durations. It is noteworthy that different distributions can be specified. Each flight takes 3 hours. The planned duration of a ground period is of 1.5 hour during the day and 7.5 hours at the end of the day (after 4 flights).

We will first consider the impact of a failure occurrence during the mission to show how the re-assessment results help to determine when to repair this component. Then we show the impact of mission profile changes and how mission reliability re-assessment will help in mission re-assignment.

6.2.1 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 global "Nogo" state.

Curve 0 of Fig. 19-a shows the mission reliability, *MR*, as assessed before mission starting, assuming that all components are operative at the starting of the mission. It can be seen that, at the end of the mission, *MR* is above *MRR*.

Curve 0 is the same in all figures of 19-a to 19-e, and for Fig. 19.

Curve 1 of Figure 19-b corresponds to the case where P1 has been diagnosed as inoperative at the end of day 2. *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 *MRR* from day 5. This result shows that P1 has to be repaired before the end of the mission to satisfy the MRR requirement. Three cases can be considered: P1 is repaired at the end of day 3, at the end of day 4, or at the end of day 5.

Fig. 19-c and 19-d correspond respectively to the cases where P1 is repaired at the end of day 3 and day 4. It can be seen that for both cases, *MR* is above *MRR* for the whole mission. The case where the repair takes places at the end of day 5 leads to an *MR* below *MRR*, in day 7. The above results show that, in case of failure of P1 at day 2, P1 should be repaired either in day 3 or day 4, depending where and when the maintenance can take place.

Curve 4 of Fig. 19-e 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. It can be seen that the new assessed measure is still above MRR 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.

The above results deserve two major comments:

- We have assumed exponential distributions for all component failure 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 (see [16]). 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.
- *MR* is equal to 1 at the beginning of each re-assessment, because the system undergoes a major inspection after detection of a component failure and it is in an operative global state at the time the mission is resumed.

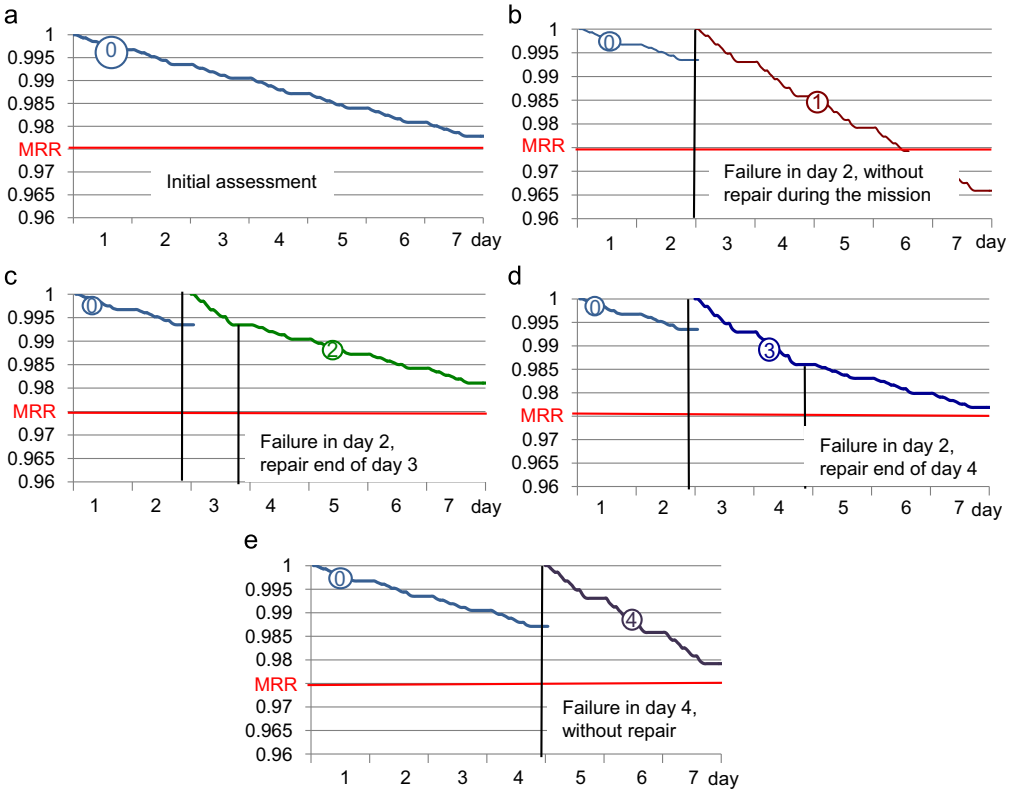

Fig. 19 Impact of P1 failure and repair on mission reliability

6.2.2 Failure of the Secondary computer S1

Curves 5 and 6 of Fig. 20 show the re-assessment of *MR* after the secondary computer S1 failure, respectively during day 2 and day 4. These curves are to be compared respectively to curves 1 and 4 of Fig. 19-b and 19-e.

Curve 5 is below curve 1 and Curve 6 is below curve 4. 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: 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.

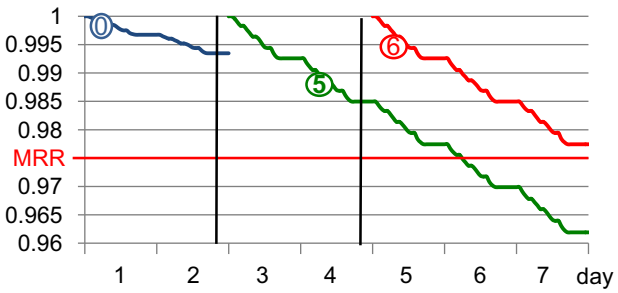

Fig. 20 Impact of S1 failure during mission achievement

6.2.3 Changes 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 changes in the initial mission. For example, an aircraft may be assigned new flights with different durations, or assigned 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 PR0 to PR3, defined in Fig. 21. PR1 to PR3 assume a mission re-planning from day 2.

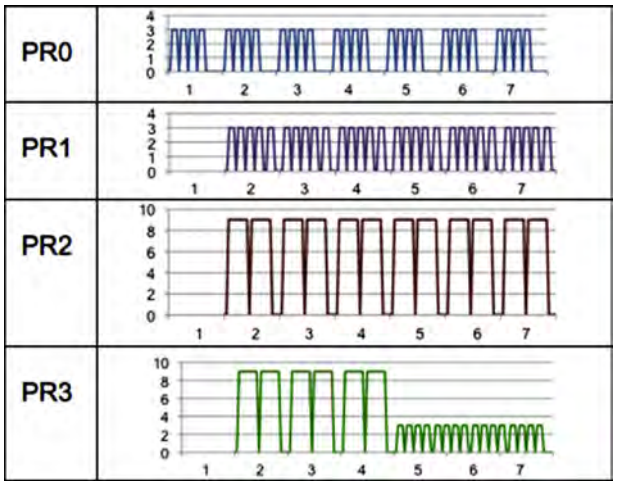

Fig. 21 Mission profiles

Fig. 22 shows that the reliability values for PR1 is lower than for PR0 after 6 days. However, the minimal reliability requirement (*MRR* = 0.975) is still satisfied. This means that the new mission is acceptable.

For PR2 (Fig. 23), *MR* becomes lower than *MRR*. This means that the new mission is not acceptable after day 6. 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, the *MRR* is again satisfied.

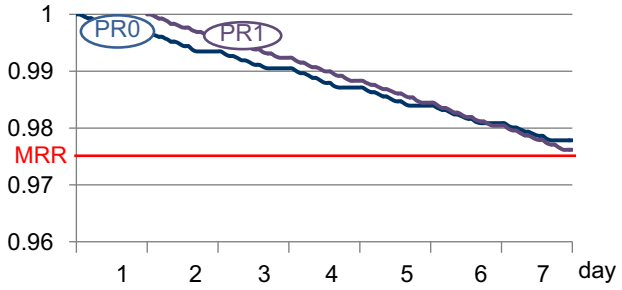

Fig. 22 Mission changes from PR0 to PR1

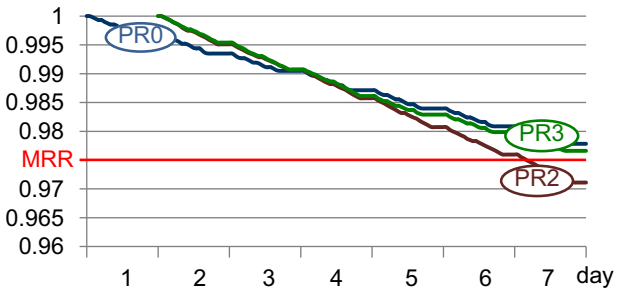

Fig. 23 Mission adjustment from PR2 to PR3

7 Related Work

To the best of our knowledge, aircraft operational reliability modeling has been seldom addressed in the literature. The studies carried out are rather concentrated on safety aspects (see [17–19] for instance), and most works about operational reliability are for design enhancement purpose [20,21]. In [22], 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 their aircraft operability.

A decision support approach to maintenance planning is presented in [23]. That is, thanks to redundancy, the aircraft can continue operating in degraded mode with some equipment inoperative; however, the risk of occurrence of an interruption is increased. A method is proposed to schedule the repairs taking into account some optimization criteria: cost, remaining useful life and operational risks. It is worth noting that this work is not aimed at assessing the operational reliability. It uses the reliability measure as input.

In [13], the operational consequences of system failures are studied using event tree analysis. The paper discusses the possible consequences of failures taking into account the flight phase during which they have occurred. A modeling approach based on the fault trees of the targeted aircraft system is presented in [20], together

with a computing algorithm to estimate the bounds of the considered probability measure. The approach considers a series of flight cycles and provides a means to evaluate the probability of occurrence 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. Note that the paper only deals with dispatch events and does not consider in-flight operational consequences. The probability of more than one component failure during a flight is also neglected.

Concerning modeling aspects, the problem is generally categorized, with regard to the system, as a Phased Mission System (PMS) problem. The PMS approach was used in [24] to analyze the performance of a fault-tolerant aircraft computer. The model developed was structured in three hierarchical levels that are connected based on dedicated compositional processes. The study was limited to a single flight, and maintenance activities were not considered. [25] analyses the PMS and present a dependability modeling approach. It is shown that, under some given conditions, the model can be processed using an analytical method. [26] addresses the problem using the concept of maintenance-free operating periods; the system evolves through a series of phases with no possible maintenance. The developed model is solved by simulation.

Of all these works, none is aimed directly at modeling aircraft operability during its missions' achievement. The closest works [13,20] are carried out for long-term operational dependability analysis and are based on event trees and fault trees. Our paper addresses aircraft operational reliability, for maintenance planning while the aircraft is in operation, using stochastic state-based models.

Concerning online dependability assessment, [27] and [28] address the use of online assessment for the selection of the most adapted solution to system operation. [28] is dedicated to online dependability assessment to support interoperability between networked systems, through adapted connectors.

8 Conclusion

In this paper we have developed a modeling approach for building a stochastic dependability model to assess the operational reliability of an aircraft. The model can be easily tuned and updated, either before or during mission achievement, to take into account the real operational conditions, after occurrence of a significant change such as a component failure. The results of model processing, in combination of the current existing dispatch conditions, support maintenance planning. It is worth to mention that the dispatch conditions are mainly related to the safety of the next flight, while our results address aircraft operability for the whole remaining mission.

A meta-model has been proposed to support the construction of the stochastic operational dependability, allowing the building of models for different systems and planes in a unified way. It has been used in this paper to build the model of an aircraft subsystem, the Control of the Movable Surfaces, and to derive examples of results.

The case study illustrates the use of the dependability model to assess aircraft operational reliability before and during its mission. We have in particular shown that according to the affected component and to the day of failure occurrence with respect to the remaining part of the mission, the mission can be continued without maintenance or maintenance should take place before a given time limit, identified by the obtained results.

In this paper, the model is constructed using the SAN formalism and solved using its supporting tool to obtain the quantitative results. The final model is based on the AltaRica language that is of common use at Airbus. A prototype tool implementing the proposed approach is currently under development.

9 References

[1] Ahmadi A, Söderholm P, Kumar U. An overview of trends in aircraft maintenance program development: Past, present, and future. Risk, Reliability and Societal Safety : Proceedings of European Safety and Reliability Conference, ESREL 2007, Norway: Terje Aven, Jan Erik Vinnem; 2007, p. 2067–76.

[2] ATA MSG. MSG-3 - Maintenance program development Document 1993.

[3] Fota N, Kaâniche M, Kanoun K. Dependability evaluation of an air traffic control computing system. Performance Evaluation 1999;35:253–73.

[4] Betous-Almeida C, Kanoun K. Dependability modelling of instrumentation and control systems: A comparison of competing architectures. Safety Science 2004;42:457–80.

[5] Rugina A-E, Kanoun K, Kaâniche M. The ADAPT Tool: From AADL Architectural Models to Stochastic Petri Nets through Model Transformation. Proceedings of the 2008 Seventh European Dependable Computing Conference, Washington, DC, USA: IEEE Computer Society; 2008, p. 85–90.

[6] Rugina AE, Kanoun K, Kaâniche M. Software Dependability Modeling Using AADL (Architecture Analysis and Design Language). International Journal of Performability Engineering 2011;7:313–25.

[7] Arnold A, Point G, Griffault A, Rauzy A. The AltaRica formalism for describing concurrent systems. Fundamenta Informaticae 1999;40:109–24.

[8] Boiteau M, Dutuit Y, Rauzy A, Signoret J-P. The AltaRica Data-Flow Language in Use: Assessment of Production Availability of a MultiStates System. Reliability Engineering and System Safety 2006;91:747–55.

[9] Daly D, Deavours DD, Doyle JM, Webster PG, Sanders WH. Möbius: An Extensible Tool for Performance and Dependability Modeling. Proceedings of the 11th International Conference on Computer Performance Evaluation: Modelling Techniques and Tools, London, UK: Springer-Verlag; 2000, p. 332–6.

[10] Tiassou K. Aircraft operational reliability — A Model-based approach and case studies. Ph.D. Univ. Toulouse, INSA, 2013.

[11] Tiassou K, Kanoun K, Kaâniche M, Seguin C, Papadopoulos C. Modeling aircraft operational reliability. Proceedings of the 30th international conference on Computer safety, reliability, and security, Naples, Italy: Springer-Verlag; 2011, p. 157–70.
[12] Tiassou K, Kanoun K, Kaâniche M, Seguin C, Papadopoulos C. Impact of Operational Reliability re-Assessment during Aircraft Missions. 2012 31st International Symposium on Reliable Distributed Systems, Irvine, California, USA: IEEE; 2012, p. 219–24.
[13] Ahmadi A, Soderholm P. Assessment of Operational Consequences of Aircraft Failures: Using Event Tree Analysis. Proc. 2008 IEEE Aerospace Conf., Big Sky, MT, USA: 2008, p. 1–14.
[14] Bernard R, Aubert J, Bieber P, Merlini C, Metge S. Experiments in model-based safety analysis: flight controls, Cachan, FRANCE: 2007.
[15] MMEL. Master Minimum Equipment List - AIRBUS A-340-200/300 2008.
[16] Tiassou K, Kanoun K, Kaâniche M, Seguin C, Papadopoulos C. Online model adaptation for aircraft operational reliability assessment. ERTS2 2012, Toulouse: 2012.
[17] Prescott DR, Andrews JD. Aircraft safety modeling for time-limited dispatch. Proceedings of the Annual Reliability and Maintainability Symposium, Alexandria, VA, USA: 2005, p. 139–45.
[18] Kehren C, Seguin C, Bieber P, Castel C, Bougnol C, Heckmann J-P, et al. Advanced simulation capabilities for Multi-systems with Altarica. Proceedings of the 22nd International System Safety Conference, International System Safety Society; 2004, p. 489–98.
[19] Ramesh A, Twigg D, Sharma T. Advanced methodologies for average probability calculation for aerospace systems. Proc. 26th international congress of the aeronautical sciences, Anchorage - Alaska, USA: 2008, p. 1–9.
[20] Saintis L, Hugues E, Bes C, Mongeau M. Computing in-service aircraft reliability. Int J Rel Qual Saf Eng 2009;16:91–116.
[21] Bineid M, Fielding JP. Development of an aircraft systems dispatch reliability design methodology. The Aeronautical Journal 2006;110:345–52.
[22] Sachon M, Paté-Cornell E. Delays and safety in airline maintenance. Reliability Engineering & System Safety 2000;67:301–9.
[23] Papakostas N, Papachatzakis P, Xanthakis V, Mourtzis D, Chryssolouris G. An approach to operational aircraft maintenance planning. Decision Support Systems 2010;48:604–12.
[24] Meyer JF, Furchtgott DG, Wu LT. Performability Evaluation of the SIFT Computer. IEEE Transactions on Computers 1980;C-29:501–9.
[25] Mura I, Bondavalli A. Markov regenerative stochastic petri nets to model and evaluate phased mission systems dependability. IEEE Trans Comput 2001;50:1337–51.
[26] Chew SP, Dunnett SJ, Andrews JD. Phased mission modelling of systems with maintenance-free operating periods using simulated Petri nets. Reliability Engineering & System Safety 2008;93:980 – 994.

[27] Malek M. Online Dependability Assessment through Runtime Monitoring and Prediction. Proceedings of the 2008 Seventh European Dependable Computing Conference, Kaunas, Lithuania: IEEE Computer Society; 2008, p. 181–181.

[28] Masci P, Martinucci M, Di Giandomenico F. Towards Automated Dependability Analysis of Dynamically Connected Systems. Proceedings of the 2011 Tenth International Symposium on Autonomous Decentralized Systems, Washington, DC, USA: IEEE Computer Society; 2011, p. 139–46.