A model-based method for system reliability analysis

Conference Paper · March 2012

CITATIONS
17

READS
326

2 authors:

Alfredo Garro
Università della Calabria
103 PUBLICATIONS **967** CITATIONS
SEE PROFILE

Andrea Tundis
Technische Universität Darmstadt
43 PUBLICATIONS **185** CITATIONS
SEE PROFILE

A Model-Based method for System Reliability Analysis

Alfredo Garro, Andrea Tundis
Department of Electronics, Computer and System Sciences (DEIS), University of Calabria,
Via P. Bucci 41C, 87036, Rende (CS), Italy
{garro, atundis}@deis.unical.it

Keywords: Reliability Analysis, Model-Based Systems Engineering, SysML, Integrated Avionics Systems.

Abstract

System reliability is an important non-functional requirement whose satisfaction is even crucial for mission critical systems. However, the increase in both system complexity and accuracy required in the reliability analyses often makes inadequate traditional techniques which are mainly based on statistical and probabilistic tools and on the hierarchical decomposition of the system in terms of its components. Moreover, the integration of classical techniques in typical system development processes, and especially in the design phases, is quite difficult and thus their use is often postponed to the later development stages (e.g. system verification) with the possible risk of having to revise even basic design choices and with a consequent increase in both completion time and development cost. To address these issues, the paper proposes a Model-Based method for system reliability analysis which combines in a unified framework the benefits of popular OMG modeling languages (UML, SysML) with the wide adopted Mathworks simulation and analysis environments (Matlab, Simulink). The flexibility and scalability of the proposal, as well as its effectiveness in evaluating through simulation the system reliability performances, is exemplified through a case study concerning the reliability analysis of a Flight Management System.

1. INTRODUCTION

In several application domains ranging from automotive to aerospace, a great variety of systems are currently designed and developed by organizing and integrating existing components which in turn can be regarded as systems. This design approach potentially offers many advantages in terms of time and cost reductions as promote the reusability of existing components and enable a natural parallel work organization in the system realization; in fact, system components can be selected/customized/realized separately and then integrated so to obtain the overall system. However, the integration of system components is a challenging task whose criticality increases as the heterogeneity and complexity of the components increase. Thus, suitable engineering methods, tools and techniques need to be exploited to prevent and manage the risks arising from the integration of system components and, mainly, to avoid their occurrence in the late phases of the system development process which may result in a significant increase in the development cost.

To overcome these issues the adoption of the Systems Engineering approach represents a viable solution as it provides a wide set of methods and practices which allow the definition of the system architecture and behavior at different abstraction level in terms of its components and their interactions [16]. Moreover, systems requirements are constantly traced during the different system development phases so to clearly specify how a system component concurs to the satisfaction of the requirements [25]. However, in the Systems Engineering field, even though great attention has been devoted to functional requirements analysis and traceability, there is still a lack of methods which specifically address these issues for non-functional requirements. As a consequence, the analysis concerning if and how non-functional requirements are met by the system under development is not typically executed contextually to the design of the system but still postponed to the last stages of the development process (e.g. system verification) with a high risk of having to revise even basic design choices and with a consequent increase in both completion time and development cost [16].

In this context, the paper aims to promote the use of flexible methods for the analysis of non-functional requirements by proposing a model-based method for system reliability analysis which is centered on a popular UML-based language for systems modeling (SysML) [7, 23] and on a *de facto* standard platform for the simulation of multi-domain dynamic and embedded systems (Mathworks Simulink) [13]. Indeed, as system reliability is an important non-functional requirement, especially for mission critical systems, there is a strong demand for new and more powerful analysis tools and techniques able not only to verify the reliability indices of a system but also to flexibly evaluate the system reliability performances and compare different design choices.

The proposed method (hereafter, RAMSAS) is based on a classical iterative process which consists of four main phases: *Reliability Requirements Analysis*, *System Modeling*, *System Simulation*, and *Results Assessment*. In

the first phase, the system reliability requirements are analyzed and the objectives of the reliability analysis are specified; then, in the *System Modeling* phase, the structure and behavior of the System are modeled by using a *SysML* based notation; moreover, specific behaviors, which model the onset, propagation and management of failures, are introduced (a wide set of basic failure behavior patterns have been defined). In the *System Simulation* phase, the previously obtained *Models* of the system are represented in terms of the constructs offered by Mathworks Simulink. Then, different simulation scenarios are set, simulations executed and simulation results analyzed with respect to the system reliability requirements elaborated in the first phase. As the process is iterative, if necessary, new partial or complete process iterations can be executed.

The proposed method is exemplified in the reliability analysis of a modern integrated avionics system as it represent a typical mission critical system. In fact, although integrated avionics architectures allow saving both costs and weight compared to federated architectures, their reliability analysis is a challenging task due to the difficulty in identifying the occurrences and propagation of faults and then the consequent system failures [14].

The rest of the paper is structured as follow: Section 2 introduces the reliability analysis discipline along with a brief survey on the related techniques; the proposed model-based method for system reliability analysis is presented in Section 3; in Section 4, the proposed method is applied to the reliability analysis of an integrated avionics system; finally, conclusions are drawn and future work delineated.

2. SYSTEM RELIABILITY ANALYSIS: FROM CLASSICAL TO MODEL-BASED TECHNIQUES

Reliability represents the ability of a system to perform its required functions under stated conditions, identified during its design, for a specified period of time [12]. This property, which is crucial especially for mission-critical systems, is strongly related to the following ones: *Availability*, which is the proportion of time a system is in a functioning condition defined at design time; *Maintainability*, which represents the ease with which maintenance of a system can be performed in accordance with prescribed requirements; *Safety*, which takes into account the effects of the system on its surrounding environment to prevent, eliminate and control hazards.

The engineering discipline which aims at providing an integrated and methodological approach to deal with the above mentioned system properties is commonly indicated by the acronym RAMS (*Reliability*, *Availability*, *Maintainability* and *Safety*) and *RAMS analysis* indicates the engineering task whose main objective is to identify causes and consequences of system failures [12]. Several techniques for performing quantitative and qualitative RAMS analyses are currently available [4, 18] and the most adopted are briefly described in the following (see Table 1).

Table 1. Reliability Analysis techniques

	Quantitative Analysis	Qualitative Analysis	Suitable for Software Intensive Systems
Series-Parallel (RBD)	x	-	-
Markov Chains	x	-	-
FMEA/FMECA	-	x	x *(S-FMEA/S-FMECA)*
FTA	-	x	x *(S-FTA)*
HAZOP	-	x	x
HSIA	-	x	x
SCCFA	-	x	x
PSH	-	x	x

Quantitative analysis techniques (such as *Series-Parallel system reliability analysis* and *Markov Chains*) are based on the identification and modeling of physical and logical connections among system components and on the analysis of their reliability through statistical methods and techniques.

Qualitative analysis techniques aim to identify the possible system failures, their rate of occurrence and (local and global) effects on the system so to individuate corrective actions; two main techniques are currently exploited: FMECA (Failure Modes Effects and Critical Analysis) and FTA (Fault Tree Analysis).

The above described techniques were originally conceived mainly for electromechanical systems thus, with the increasing adoption of software components in many modern systems, some extensions for embedded and software intensive systems have been proposed (e.g. S-FMECA, S-FTA) [5] along with specific software-oriented techniques (e.g. HSIA, SCCFA, PSH).

Nevertheless, the increase in both system complexity and accuracy required in the reliability analysis often makes inadequate the so far mentioned techniques which are mainly based on statistical and probabilistic tools and on the hierarchical decomposition of the system in terms of its components. Moreover, their integration in typical system development processes, and especially in the design phases, is quite difficult and then their use is often postponed to the later development stages (e.g. system verification). As a consequence, new techniques are emerging which are centered on model-based approaches [17, 20] so to benefit from the available modeling practices and which incorporate the use of simulation to flexibly evaluate the system reliability indices and compare different design choices [3, 6, 11, 19]. However, despite a general consensus on the advantages that could derive from the exploitation of model-based approaches for system reliability analysis, the use of these techniques has been traditionally unusual and has not been recommended by international standards until recently

[10]. This delay in the adoption is mainly due to the lack of methods able to integrate available modeling languages, tools and techniques in a consistent modeling framework. The proposal presented in Section 3 aims at contributing to fill this lack. Specifically, the proposal is strongly related to the approaches presented in [3], however, as it strongly relies on the Method Engineering paradigm [8] its main ambition is to provide a *self-consistent* method fragment for system reliability analysis which can be easy pluggable in various phases of a typical system development process.

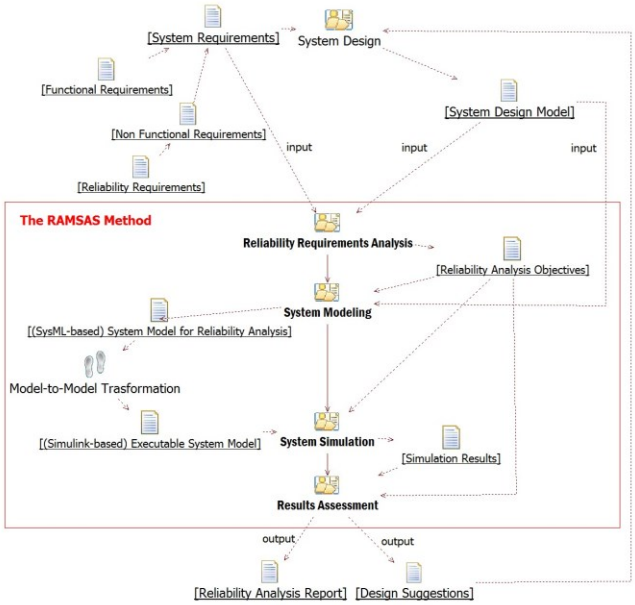

Figure 1. The RAMSAS Method (process view)

3. A MODEL-BASED METHOD FOR SYSTEM RELIABILITY ANALYSIS

RAMSAS, the proposed model-based method, which aims to support both quantitative and qualitative system reliability analyses, relies on a classical iterative process which consists of the following four main phases: *Reliability Requirements Analysis*, *System Modeling*, *System Simulation*, and *Results Assessment*. The phases of the process along with their input and output work products are shown in Figure 1 and discussed in details in the following.

In the *Reliability Requirements Analysis* phase, the following work-products should be provided as input: (i) a *System Design Model* (*SDM*), typically coming from a System Design phase; (ii) the *System Requirements* (SR), both *Functional* (FR) and *Non-Functional* (NFR). In particular, among the NFR, the *Reliability Requirements* (*RR*) specify the ability required for the system in performing the functions specified in FR under specific stated conditions and for a specified period of time. In addition, a Failure modes and effects analysis (FMEA) [4] can be also provide to highlight the potential failure modes of the system along with their severity and likelihood. Starting from these input work-products a *Reliability Analysis Objectives* (*RAO*) document is drawn in which the reliability functions and indicators, to be derived from the analysis of the simulation results, are identified along with the main analysis techniques to be applied to the data gathered from simulation.

In the *System Modeling* phase, which takes as input work products the *RAO* document and the *SDM*, the System structure and behavior are modeled by using a *SysML* based notation [7, 23] according to the identified reliability analysis objectives. More specifically, the System is decomposed in component entities by applying *in-out zooming mechanisms* [15]. A possible hierarchy decomposition levels, could be the following: *system*, *subsystems*, *equipment*, and *components*; however, different and deeper hierarchies can be also introduced. Each entity is modeled as a SysML *Block* whose structure is defined by both a *Block Definition Diagram* (BDD) and an *Internal Block Diagram* (IBD) and whose behavior is defined by a *SysML Activity Diagram* and some *Sequence Diagrams* [23]. Specifically, the BDD describes the Block with its port interfaces, internal attributes, operations, constraints, parts and relationships with other blocks; whereas, the IBD provides a description of the its internal structure, the organization of its component blocks, the type of composition and the topology of internal connections. The behavior of a *Block* is specified through a set of *Tasks* whose execution is characterized by *pre* and *post* conditions and can be periodically scheduled or triggered by events [21]. Each *Task* is modeled using two types of SysML Diagrams: (i) an *Activity Diagram* which allows completely modeling the behavior of a particular *Block* as a flow of *actions*; (ii) a set of *Sequence Diagrams* which allows modeling specific scenarios, each of which corresponding to a given sequence of actions in the Block *Activity Diagram*. Moreover, special Tasks, which model the onset, propagation and management of Block *failures*, are specified (a wide set of basic failure behavior patterns have been defined). In addition, SysML *Parametric Diagrams* can be introduced for supporting specific analysis by defining constraint blocks which express mathematical equations and their parameters that may correspond to block properties. The above described diagrams constitute the SysML-based Model for the analysis of system reliability (SMRA) which is the output work product of the *System Modeling* phase. It is worth noting that the modeling effort of this phase can be significantly reduced if the input *SDM* is also based on SysML/UML.

In the *System Simulation* phase, the previously obtained models of the System need to be represented in terms of the constructs offered by the framework exploited for the simulation so to become *executable models*. This model-to-model transformation is the first step of the *System Simulation* phase and strongly depends on the target simulation platform [2]. In the current version of RAMSAS,

the choice has fallen on *Mathworks Simulink* which represents a *de facto* standard platform for the simulation of multi-domain dynamic and embedded systems. The model-to-model transformation from the SMRA to the *Simulink-based Executable System Model* (ESM) is enabled by *IBM Rational Rhapsody* [9], a model-driven development environment based on UML and SysML which provides an integrated modeling environment for SysML and allows to generate Simulink models starting from SysML ones. Specifically, the transformation is based on a mapping between the basic SysML and Simulink constructs [22, 24] (see Table 2). Moreover, the *Mealy Machines* which model the behavior of a Simulink Block is obtained by the corresponding *SysML Activity* and *Sequence Diagrams*.

Table 2. Mapping among SysML and Simulink constructs

Entity	SysML	Simulink
System/Subsystem/ Equipment/Component	Block, Part	Block, Subsystem Block
Behavior/Constraint	Activity diagram, Sequence diagram, Parametric diagram	S-Function, State Flow diagram
Input/Output Interface	Flow Port	Input/Output Simulink Block
Association/Binding	Connection	Line

The resulting ESM is a network of blocks which is executed according to a synchronous reactive model of computation [1]: at each step, Simulink computes, for each block, the set of outputs as a function of the current inputs and the block state, then it updates the block state. Before starting the simulation, several system and configuration parameters (e.g. generation and propagation rates of failures) can be set so to evaluate system reliability performance in different simulation scenarios. The data gathered from the simulations are elaborated and reported in the *Simulation Results* (SR) work-product.

In the *Results Assessment* phase, simulation results are analyzed with respect to the system reliability requirements elaborated in the first phase and documented in the RAO. Specifically, several analyses can be directly performed in Simulink by using useful add-on like *SIMLOG*, whereas more advanced analysis can be performed by external analysis tools by exporting the obtained results though the Matlab Workspace. These analyses should give information about the reliability performances of the system under consideration; moreover, they should provide suggestions to improve reliability of the system proposing alternative design solutions.

Finally, as for any iterative process, new (partial or complete) iterations can be executed for achieving new or missed analysis objectives.

The model-based approach exploited in RAMSAS along with the adoption of a standard modeling notation makes RAMSAS easy to *integrate* in various phases of a typical System Development Process; as an example, with reference to the V-Model process shown in Figure 2, RAMSAS can be used: (i) in the *testing* phases to support the evaluation of unit and system reliability performances; (ii) in the *design* phases to support the validation and evaluation of configuration scenarios and settings of system parameters so to guide and suggest different design choices.

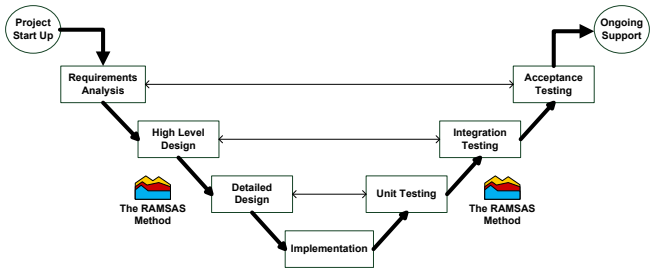

Figure 2. The RAMSAS method *plugged* into a V-Model

4. EXPLOITING RAMSAS: RELIABILITY ANALYSIS OF A FLIGHT MANAGEMENT SYSTEM

In this Section, the exploitation of RAMSAS is exemplified through a case-study concerning the reliability analysis of an Integrated Avionics System and, specifically, of the Flight Management System (FMS) of a modern aircraft. In particular, after a synthetic description of the FMS, the subsequent subsections illustrate the different phases of the proposed process.

Modern Avionics architectures consist of a set of computing modules capable of supporting numerous applications of differing criticality levels [14]. Such architectures, denoted by the acronym IMA (Integrated Modular Avionics) and in which numerous separate processors and line-replaceable units (LRU) are substituted with fewer, more centralized processing units, promise significant weight reduction and maintenance savings. However, the high level of integration of an IMA makes its reliability analysis a challenging task due to the difficulty in identifying the occurrences and propagation of faults and then the consequent system failures. A typical avionics architecture, whose specific structural organization depends on the type of aircraft, is sketched in Figure 3. Two key components are the Flight Management System (FMS) and the Flight Control System (FCS). The FCS is composed of a Flight Guidance System (FGS) that generates roll and pitch guidance commands, and an Auto-Pilot (AP) that executes them. Flight Management System (FMS) is the *brain* of the aircraft navigation system, which assists pilots in navigation and flight preparation, computes the most efficient flight in fuel and time savings, and automatically navigates the aircraft. The main components of the FMS under consideration are the two Flight Management Computers (FMCs) and two Control Display Units (CDUs). The two FMC are the *heart* of the FMS and normally while one FMC accomplishes the FMS tasks the other monitors the first one and it is ready to replace it if system faults occur. Each CDU is used to manage and provide information on its

own FMC and also to give notifications about the status of the other FMC. Moreover, in order to improve reliability of the FMS, CDUs provide alternate navigation capability if both FMCs fail.

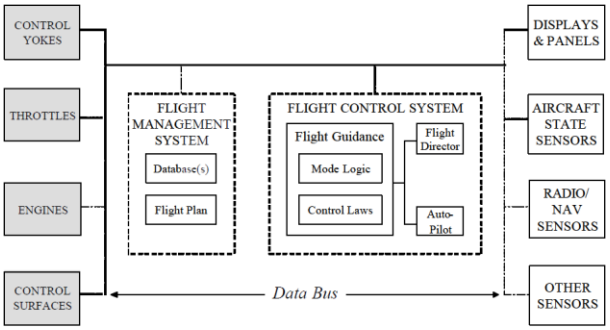

Figure 3. High Level Architecture of an Avionics System

4.1. Reliability Requirements Analysis

The reliability analysis of the Flight Management System is a challenging task in the design of an aircraft. In particular, FMS calculates performance data and the most fuel efficient route to be flown based on specific aircraft parameters such as weight, cruise altitude and actual aircraft position. FMS is responsible for a wide set of functions requiring complex logic, as an example: (i) planning of the flight path, (ii) generating a proper sequence of flight modes, (iii) calculating a feasible trajectory and regulating the aircraft along it; (iv) switching among different modes of operation to handle situations like conflict resolution among aircrafts; (v) obstacle avoidance and flight envelope protection, (vi) monitor and manage power sources to ensure the operation of individual components and, consequently, the operation of the global system (see Figure 4).

As the main aircraft system objective is to complete its missions, in this context, reliability is strongly related to availability and often defined in terms of *Mission Capable Rate (MCR),* which represents the probability that, even if failures occur, their consequences do not affect the global operation of the system so to prevent the accomplishment of the mission. *MCR*, which is strictly related to the reliability of the system and of its parts, is defined as 1 minus unavailability and thus given by *MTBCF/(MTBCF+MTTR)* where *MTBCF* (*Mean Time Between Critical Failures*) is a variations of *MTBF* (*Mean Time Between Failures*) used when is necessary to differentiate among types of failures, such as non-critical and critical failures, whereas *MTTR* represent the *Mean Time To Repair*. By introducing the *MTTCF* (*Mean Time To Critical Failure*), *MTBCF* = *MTTCF+MTTR* and *MCR* = (*MTTCF+MTTR*)/(*MTTCF+2MTTR*). As a consequence the reliability analysis of the system requires evaluating both the *MTTCF* and *MTTR* indicators (as represented in Figure 4 in blue). By combining qualitative and quantitative results a clear picture of the reliability performances of the system can be obtained and alternative design solutions to effectively address the design and maintenance of the system can be compared.

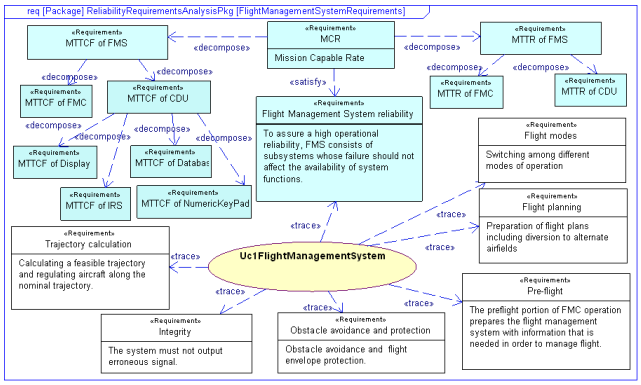

Figure 4. System Requirements of the FMS

4.2. System Modeling

In this phase structural and behavioral views of the above described System are modeled using SysML diagrams.

In particular, the System is decomposed following a hierarchical layered approach (i.e. *system*, *subsystems*, *equipment*, *components*), and the reliability requirements (derived in the previous phase) are associated to the system entities of each decomposition layer to allow requirements traceability.

In Figure 5, the *Block Definition Diagram* (BDD) of the Flight Management System (FMS) is reported by showing its *subsystems* along with their in/out port interfaces and cardinalities. The internal structure of the FMS is described by the *Internal Block Diagram* (IBD) in Figure 6 which shows the component subsystems, their connections and interaction paths along with their operations and attributes.

By applying the well-known *zooming-in* mechanisms, the internal structure of each *subsystem* highlighted in Figure 6 is represented in details by an IBD diagram which shows its constituting *equipment*; then, if necessary, each *equipment* is further specified in terms of its *components* by a related IDB diagram and so on, until a desired layer of system decomposition is reached. As an example, the IBD of the *Control Display Unit* (CDU) *subsystem* is shown in Figure 7. Each CDU is composed by the following *equipment*: (i) a *CDUManager* which elaborates the commands addressed to the CDU; (ii) a *Display* to visualize all the required information; (iii) a *Numeric Keypad* to enter commands; (iv) an *Inertial Reference System* (IRS) which calculates airplane position, acceleration, track, vertical speed, ground speed, wind speed and direction; (v) a *Database* containing specific information exploited for the management of the flight.

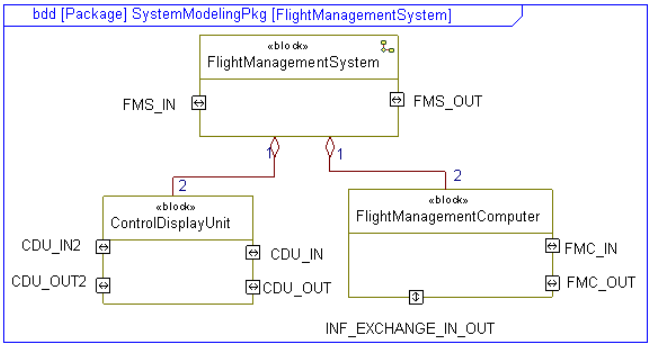

Figure 5. BDD of the Flight Management System

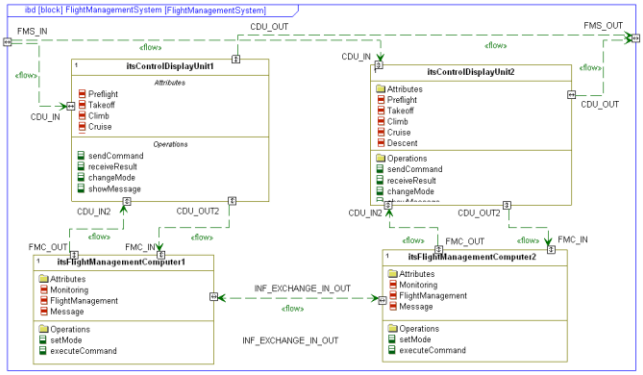

Figure 6. IBD of the Flight Management System

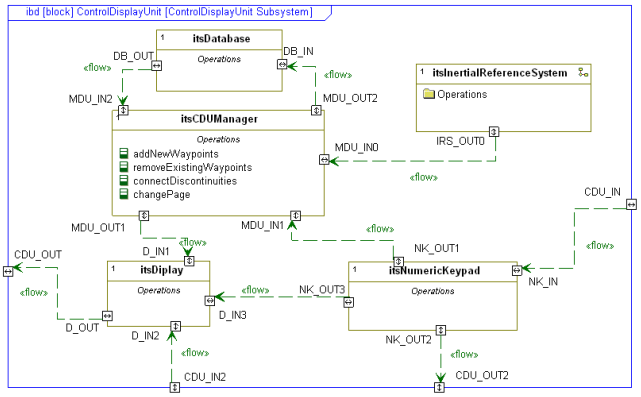

Figure 7. IBD of the Control Display Unit subsystem

Beside the System structure, the behavior of the System is also specified following the same layered approach but in a *bottom-up* fashion: from the lowest decomposition level to the top system level (e.g. from the *component* level to the *equipment*, *subsystem*, and *system* level).

Specifically, the behavior of each component is specified as a set of *Tasks* represented by SysML Activity Diagrams and, if a Task envisages different scenarios, by a related set of Sequence Diagrams (see Section 3). As an example, the specification of the behavior of the *Inertial Reference Unit*, which is a key *component* of the *IRS equipment* (see Fig. 7), is presented.

Table 3 summarizes some *Tasks* of the *Inertial Reference Unit* as well as their pre and post-conditions and the execution schedule associated with them according to the UML/SysML specifications (see Section 3).

Table 3. Tasks of the Inertial Reference Unit component

Task	Pre conditions	Post conditions	Execution Schedule
Alignment	Operation Mode and Initial Position values available	Component state changed	Triggered
Attitude Calculation	flight status and parameters	Attitude parameters calculated	Triggered
…	…	…	…
Failure Management	Component failure	Safety state reached	Triggered
Fault Generation	Component is working	(Possible) fault generation	Periodical
Fault Evaluation	Fault generated	(Possible) component failure	Triggered
Failure Propagation	Component failure	(Possible) failure propagation	Triggered

As for each component, beside the *Tasks* which define the specific component behavior (two are reported in Table 3), a set of other *Tasks* are introduced: *Fault Generation*, *Fault Evaluation*, *Failure Propagation*, *Failure Management*. When the model is executed, these *Tasks* are crucial for simulating the generation and evaluation of *faults* and the possible consequent propagation of *failures* as well as the feasible actions for their management.

The SysML Activity Diagrams of two *Tasks* of the *Inertial Reference Unit* component are reported in Figures 8 and 9.

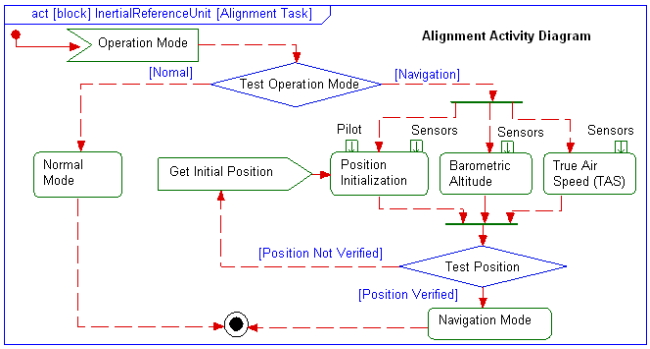

Figure 8. The Alignment Task of the Inertial Reference Unit component

In the *Alignment Task* (see Fig. 8), the *Inertial Reference Unit* switches the IRS internal state on the basis of the selected *Operation Mode*. In particular, if the selected *Operation Mode* is *Navigation,* before switching the *Operation Mode*, the Unit acquires and evaluates the *Initial Position*, the *Barometric Altitude* and the *True Air Speed*.

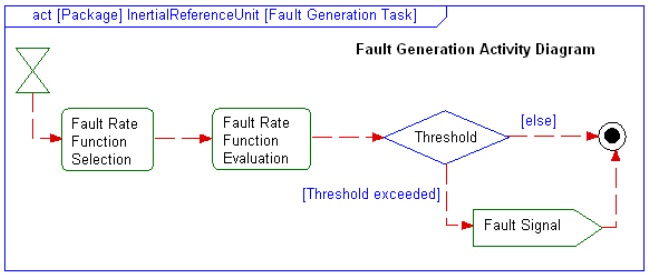

Figure 9. The Fault Generation Task of the Inertial Reference Unit component

The *Fault Generation Task* (see Fig. 9), which models a fault generation process, is activated with a given time period; upon activation, the *Fault Rate Function* specified for the component is evaluated and possibly a component fault generated.

According to the adopted layered approach, the behavior of an *equipment* is given by the composition of the behaviors of its *components*; the behavior of a *subsystem* derives from that of its constituting *equipment* and so on until the *system* level is reached. After the structure and behavior of the System under consideration has been specified through the described layered approach, the Modeling phase can be concluded. It is worth noting that, as the process is iterative, the resulting models can be refined during consecutive process iterations.

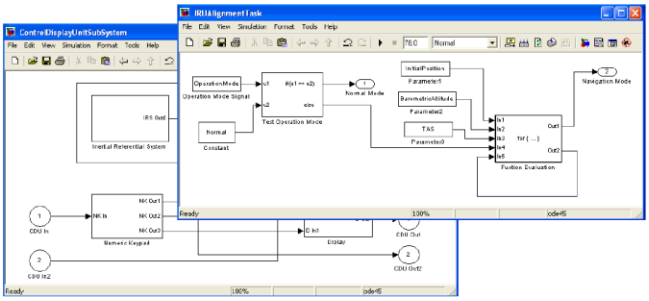

Figure 10. Simulink-based Executable System Model: some screenshots

4.3. System Simulation

In the Simulation phase the SysML based model of the System is simulated in the Mathworks Simulink environment. This phase starts by transforming the previously obtained SysML-based Model for the analysis of system reliability (SMRA) in an Executable System Model (ESM). In particular, the SysML diagrams defined through the *IBM Rational Rhapsody* tool are transformed, by applying model-to-model transformation rules, and exported for loading them in the Simulink environment. As an example, Figure 10 shows a view of the *Simulink Model* of the *Flight Management System* as derived from the SysML diagrams reported in Figures 5 and 6, and a *Simulation Model* for the *Alignment* Task of the *Inertial Reference Unit* which has been generated from the Activity Diagram of Figure 8.

After obtaining the *Simulation Model*, the setting of the simulation parameters is performed according to the *RAO*, then simulation are executed and data gathered for the analysis phase.

4.4. Results Assessment

Several simulations have been executed for analyzing the behavior of the System in different scenarios and evaluating its *reliability functions* and indicators as MTTCF, MTTR and MCR (see Section 4.1). Usually, an analytic definition of these reliability characteristics is very difficult to obtain due to the complexity of the System in terms both of its structure and behavior. The scalability and flexibility of the proposed modeling and simulation approach, instead, allowed the evaluation of the main reliability functions and indices in an inductive fashion as well as the observation of *macro-level phenomena* which are hardly captured by classical analytical/deductive models.

As an example, the analysis of the simulation data of different and extreme operative scenarios have shown that the MCR of the considered System significantly varies (from 63,9% to 75,6% in the carried out experiments) on the basis not only of the system organization and behavior but also of its configuration and parameters setting.

Finally, the analysis of the simulation data provided useful indications which allowed obtaining a more descriptive and predictive reliability system model and suggested some design choices which could improve the system reliability indicators.

5. CONCLUSIONS AND FUTURE WORK

The paper has proposed RAMSAS, a Model-Based method for the Reliability Analysis of Systems which combines in a unified framework: (i) the strengths of powerful visual modeling languages (such as OMG SysML), suitable to flexibly model the architectural and behavioral aspects of complex, dynamics, and heterogeneous systems; (ii) mature and popular tools (such as Mathworks Simulink), suitable for the simulation and analysis of multi-domain systems.

The proposed method is not intended to be an alternative to other traditional RAMS techniques (e. g. FMECA, FTA, RBD) but rather a complement to them able to provide additional analysis capabilities due to the jointly exploitation of Systems Engineering modeling and simulation languages, tools and techniques. Moreover, these distinctive features make the proposed method particularly suitable to be integrated in various phases of a typical System Development Process and especially in the design phases. This allows supporting the satisfaction and traceability of an important non-functional requirement, such as reliability, in the early stages of a development

process with considerable time and cost reductions respect to more traditional reliability analyses techniques which are often carried on in the last stages of the development with the risk of having to revise even basic design choices.

The concrete exploitation of RAMSAS in the reliability analysis of a Flight Management System (FMS) has allowed appreciating both its flexibility and scalability in complex systems modeling and its effectiveness in evaluating through simulation the system reliability performances.

Ongoing research efforts are devoted to enrich and improve the proposed method and extensively experiment it in the analysis of mission-critical systems in different application domains. Finally, another research line, concerns the integration of RAMSAS in the *Rational Harmony for Systems Engineering* process.

ACKNOWLEDGEMENTS

Andrea Tundis was supported by a grant funded in the framework of the "POR Calabria FSE 2007/2013". The authors wish also to thank Z Lab Engineering S.r.l. for its valuable support during the experimentation phase.

REFERENCES

[1] Benveniste A., Caspi P., Edwards S.A., Halbwachs N., Le Guernic P., and De Simone R., 2003. "The synchronous languages 12 years later." In Proceedings of the IEEE, Vol. 91, No. 1, pp. 64-83.

[2] D'Ambrogio A., Iazeolla G., Pieroni A., and Gianni D., 2011. "A Model Transformation approach for the development of HLA-based distributed simulation systems." In Proc. of the Int. Conf. on Simulation and Modeling Methodologies, Technologies and Applications, Noordwikerhout, The Netherlands, July 29–31.

[3] Cressent R., Idasiak V., Kratz F., David P., 2011. "Mastering safety and reliability in a model based process." In Proc. Reliability and Maintainability Symposium (RAMS), Lake Buena Vista, Florida, USA, January 24-27.

[4] Dodson B., and Nolan D., 2001. *Practical Reliability Engineering*. John Wiley & Sons Ltd.

[5] ECSS-Q80-03, 2006. *Space product assurance: Methods and techniques to support the assessment of software dependability and safety*. ESA Publications Division.

[6] Garro A., Tundis A., and Chirillo N., 2011. "System reliability analysis: a Model-Based approach and a case study in the avionics industry." In Proc. of the 3rd Air and Space Int. Conf. (CEAS), Venice, Italy, 24-28 October.

[7] Hause M., 2006. "The sysML Modeling Language." In Proc. of the 5th European Systems Engineering Conference, Edinburgh, UK, September 18-20.

[8] Henderson-Sellers B., 2003. "Method engineering for OO systems development." Communications of the ACM, Vol. 46, No. 10, pp.73–78.

[9] IBM Rational Rhapsody Web Site, http://www-01.ibm.com/software/awdtools/ rhapsody/

[10] IEC 61508, 2010. Functional safety of electrical/electronic/programmable electronic safety-related systems, Parts 1-7.

[11] Iwu F., Galloway A., McDermid J., Toyn I., 2007. "Integrating safety and formal analyses using UML and PFS." Reliability Engineering and System Safety, Vol. 92, pp. 156-170.

[12] Laprie J.C., 1992. *Dependability: Basic Concepts and Terminology*, Springer-Verlag.

[13] Mathworks Simulink Web Site: http://www.mathworks.com/products/simulink/

[14] Moir I., Seabridge A., 2008. *Aircraft Systems: Mechanical, Electrical and Avionics Subsystems Integration*, 3rd ed., Wiley.

[15] Molesini A., Omicini A., Ricci A., and Denti E., 2005. "Zooming multi-agent systems." In Proc. of the 6th Int. Workshop on Agent-Oriented Software Engineering, Utrecht, The Netherlands, July 25-29.

[16] *NASA Systems Engineering Handbook*, Revision 1, NASA/SP-2007-6105, NASA.

[17] Nicolescu G., and Mosterman P. J., 2009. *Model Based Design for Embedded Systems (Computational Analysis, Synthesis and Design of Dynamic Systems)*, CRC Press.

[18] O'Connor P., 1999. *Reliability Engineering Handbook (Quality and Reliability)*, CRC Press.

[19] Oren T. I., and Yilmaz L., 2006. "Synergy of Systems Engineering and Modeling and Simulation." In Proc. of the Int. Conf. on Modeling and Simulation Methodology, Tools, Software Applications (M&S MTSA), Calgary, Alberta, Canada, July 31-August 2.

[20] Schmidt D.C., 2006. "Model Driven Engineering." IEEE Computer, Vol. 39, No.2, pp. 25-31.

[21] Schrott G., 1993. "An experimental environment for task-level programming of robots." Lecture Notes in Control and Information Sciences, Springer-Verlag, Vol. 190, pp. 196-206.

[22] Sindico A., Di Natale M., and Panci G., 2011. "Integrating SysML with Simulink using Open-Source model transformations." In Proc. of the Int. Conf. on Simulation and Modeling Methodologies, Technologies and Applications, Noordwikerhout, The Netherlands, July 29–31.

[23] SysML: Systems Modeling Language http://www.omgsysml.org/

[24] Vanderperren Y., and Dehaene W., 2006. "From UML/SysML to Matlab/Simulink: Current State and Future Perspective." In Proc. of the Conf. on Design, Automation and Test in Europe, Munich, Germany, March 6-10.

[25] Zamparelli M., 2006. "Using a formal requirements management tool for System Engineering: First results at ESO." In Proc. of SPIE - The International Society for Optical Engineering, Vol. 6271.