EXPERIMENTING THE RAMSAS METHOD IN THE RELIABILITY ANALYSIS OF AN ATTITUDE DETERMINATION AND CONTROL SYSTEM (ADCS)

Alfredo Garro[(a)], Johannes Groß[(b)], Marius Riestenpatt gen. Richter[(b)], Andrea Tundis[(a)]

[(a)]Department of Electronics, Computer and System Sciences (DEIS), University of Calabria, Via P. Bucci 41C, 87036, Rende (CS), Italy.
[(b)]Institute for Statics and Dynamics of Aerospace Structures, University of Stuttgart, Pfaffenwaldring 27, D-70569 Stuttgart, Germany.

[(a)]{garro, atundis}@deis.unical.it, [(b)]gross@isd.uni-stuttgart.de

ABSTRACT
For the reliability analysis of modern large-scale systems new techniques centered on model-based approaches are emerging. Benefitting from the available modeling practices these techniques incorporate the use of simulation to flexibly evaluate the system reliability indices and compare different design choices. In this context, RAMSAS, a model-based method which supports the reliability analysis of systems through simulation, has been recently proposed. This paper aims at further evaluating the effectiveness and suitability of RAMSAS through a real case study concerning the reliability analysis of an Attitude Determination and Control System (ADCS) of a satellite.

Keywords: System Reliability Analysis, Model-Based Systems Engineering, Simulation, Satellite Systems.

1. INTRODUCTION

Reliability, which represents the ability of a system to perform its required functions under stated conditions, for a specified period of time, is a key requirement to satisfy especially for mission critical systems where system failures could cause even human losses (Dodson and Nolan 2001). Moreover, Reliability is strongly related to other main properties such as: 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.

Several techniques for performing quantitative and qualitative Reliability Analyses are currently available (Dodson and Nolan 2001). Specifically, 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/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). Moreover, with the increasing adoption of software components in many modern systems, some extensions of the above mentioned techniques which were originally conceived mainly for electromechanical systems are provided for embedded and software intensive systems (e.g. S-FMECA, S-FTA) along with specific software-oriented techniques (e.g. HSIA, SCCFA, PSH) (ECSS-Q80-03 2006). Nevertheless, the increase in both system complexity and accuracy required in the reliability analysis often goes beyond the capabilities of 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 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 (Cressent et Al. 2011; Iwu et Al. 2007; Oren and Yilmaz 2006). 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 (IEC 61508, 2010). 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.

To contribute to fill this lack, RAMSAS, a model-based method for the Reliability Analysis of Systems through simulation has been recently proposed (Garro and Tundis 2012a-b-c). In particular, RAMSAS aims at combining in a unified framework the benefits of

popular OMG modeling languages (UML, SysML) with the wide adopted Mathworks simulation and analysis environments (Matlab, Simulink).

RAMSAS has been experimented in the avionics domain for the reliability analysis both of a Landing Gear System (Garro, Tundis and Chirillo 2011) and of a Flight Management System (Garro and Tundis 2012b); and in the automotive domain for the reliability analysis of an Electronic Stability Control (ESC) system (Garro and Tundis 2012a). This paper aims at further evaluating the effectiveness and suitability of RAMSAS through the reliability analysis of an Attitude Determination and Control System (ADCS) (Wertz 1978) of a satellite.

The rest of the paper is structured as follows: in Section 2 the RAMSAS method is briefly described whereas in Section 3 its exploitation for the reliability analysis of an Attitude Determination and Control System (ADCS) of a satellite is reported; finally, a discussion about the lessons learned and future research directions concludes the paper.

Table 1: Phases of the RAMSAS method and related work-products

Phases	Input work-products	Output work-products
Reliability Requirement Analysis	System Design Model (SDM), System Requirements (SR)	Reliability Analysis Objectives (RAO)
System Modeling	System Design Model (SDM), Reliability Analysis Objectives (RAO)	System Model for Reliability Analysis (SMRA)
System Simulation	System Model for Reliability Analysis (SMRA), Reliability Analysis Objectives (RAO)	Simulation Results (SIRE)
Results Assessment	Simulation Results (SIRE), Reliability Analysis Objectives (RAO)	Design Suggestions Report (DSR), Reliability Analysis Report (RAR)

2. THE RAMSAS METHOD

RAMSAS is a model-based method which supports the reliability analysis of systems through simulation by providing a classical iterative process consisting of four main phases: *Reliability Requirements Analysis*, *System Modeling*, *System Simulation*, *and Results Assessment*. These phases are reported in Table 1 along with their input and output work-products. Specifically, in the *Reliability Requirements Analysis* phase the objectives of the reliability analysis are specified and the reliability functions and indicators to evaluate during the simulation are defined. In the *System Modeling* phase, the structure and behavior of the system are modeled in SysML (OMG Systems Modeling Language) by using *zooming* in-out mechanisms [9]; moreover, beside the *intended* system behaviors, specific *dysfunctional* behaviors and related tasks, which model the onset, propagation and management of faults and failures, are introduced. In the *System Simulation* phase, the previously obtained models of the system are represented in terms of the constructs offered by the target simulation platform, then simulations are executed so to evaluate the reliability performance of the system also on the basis of different operating conditions, failure modes and design choices. Finally, simulation results are analyzed with respect to the objectives of the reliability analysis; if necessary, new partial or complete process iterations are executed.

RAMSAS is strongly related to the proposal presented in (Cressent et al., 2011), however, as RAMSAS strongly relies on the Method Engineering paradigm (Henderson-Sellers 2003) it provides a self-consistent method fragment for system reliability analysis which can be easy pluggable in various phases of a typical system development process ranging from the design to the testing phases so to complement other well-known and wide adopted techniques for system reliability analysis (e.g. FMECA, FTA, RBD) by providing additional analysis capabilities.

A more complete description of RAMSAS can be found in (Garro and Tundis 2012b); in the following Section its exploitation for the reliability analysis of the ADCS of a satellite is showed and discussed in details.

3. RELIABILITY ANALYSIS OF AN ATTITUDE DETERMINATION AND CONTROL SYSTEM (ADCS)

3.1. System Description

The satellite under survey is the hypothetical FireSat mission from literature based on (Wertz 1999) with a refined system design from (Gross 2012a and b). The mission objectives of FireSat are to detect, analyze and monitor forest fires. Therefore the satellite´s Attitude Determination and Control System (ADCS) has to provide (among other modes) the ability for the satellite to scan the area below the satellite on the earth surface to detect fires. The corresponding mode is called nadir-pointing mode, which means that the satellite is pointing towards the earth's center. The satellite is orbiting the earth at an altitude of ~700 km over ground, which is called a low-earth orbit (LEO). Resulting from its altitude, the satellite has to turn with a constant angular velocity once it is aligned to nadir pointing. In Figure 1 the activation of the nadir-pointing mode is shown. After the acquisition of the attitude, the payload camera of the satellite surveys a swath of a certain width on the ground below the satellite.

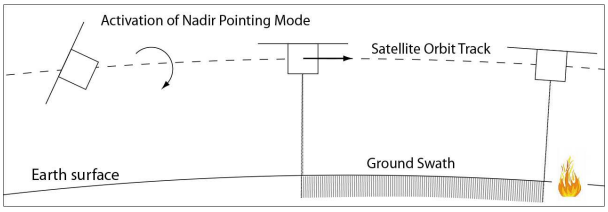

Figure 1: Sketch of the satellite flying over a fire in target pointing mode

The necessary adjustment of the satellite's attitude and its angular velocity is attained by applying torques on the satellite. The Attitude Determination and Control System (ADCS) contains therefore thrusters which imply a torque on the satellite in orbit. By firing different thrusters at the corners of the box-shaped satellite, the satellite can be turned around all axes.

3.2. Reliability Requirements Analysis

The ADCS of the satellite has to fulfill the functional requirements for the alignment of the satellite. The system consists of sensors, actuators and the on-board computer controlling the system. The thrusters are used as the actuators in the system. Due to their mission-critical role there are redundant thruster packs. The sensors determine the angular velocity and the attitude angles of the satellite. The on-board computer has a navigation unit, which calculates with the sensor data the target alignment of the satellite. Afterwards, the commands for the actuators are calculated, based on the resulting data of the navigation.

This chain of activities (determine attitude, calculate action, execute commands) has to be fulfilled over the whole lifetime of the satellite by the ADCS. Since the functional requirement for attitude control can be reached by several combinations of the redundant thruster packs, the evaluation of non-functional requirements for system reliability has to be coupled with the functional analysis of the system.

3.3. System Modeling

In the System Modeling phase the structure and both the intended and dysfunctional behavior of the system under consideration are represented in SysML by executing four modeling activities (see Garro and Tundis 2012b): *System Structure Modeling*, *Intended Behavior Modeling*, *Dysfunctional Behavior Modeling* and *Behavior Integration*. Each of these activities will be described in the following sub-sections with reference to the ADCS.

3.3.1. System Structure Modeling

In the *System Structure Modeling* activity, the system structure is modeled by using SysML *Blocks* following a *top-down* approach so to obtain a hierarchical decomposition of the system (e.g. system, subsystems, equipment, and components). Specifically, each system entity is represented by a SysML *Block* and modeled by both a *Block Definition Diagram* (BDD) and an *Internal Block Diagram* (IBD). As an example, the BDD of the ADCS system of Figure 2 shows that the ADCS consists of the following subsystems: *FlightSoftware*, *Actuators*, *Sensors*, *VehicleDynamics*, and the *PointingMode*. For each system block, its input and output interfaces are specified according to the following template: <*SourceBlock_DestinationBlock_PortName_InputOrOutputPortType*>.

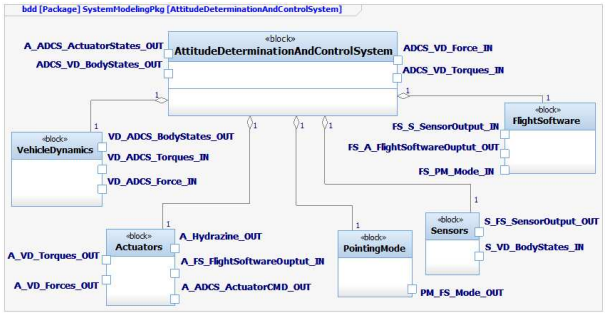

Figure 2: Block Definition Diagram of the ADCS

For providing a description of the internal structure of a block in terms of the organization of its component blocks an IBD is introduced. As an example, the internal structure of the ADCS is reported in Figure 3 in which the component subsystems, their connections and interaction paths along with their operations and attributes, are represented.

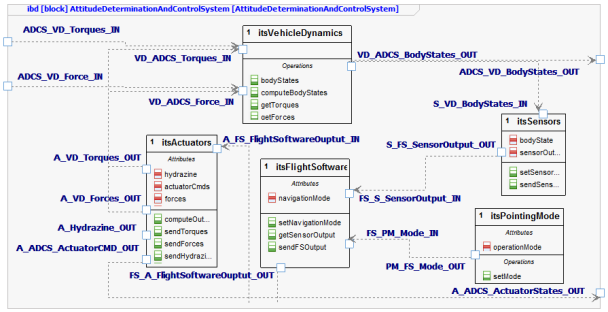

Figure 3: Internal Block Diagram of the ADCS

By applying *zooming-in mechanisms* the system block identified after the first decomposition (see Figure 2) can be further decomposed so to reach a deeper level of decomposition. As an example, the structure of the Actuator subsystem in terms of its components (*ThrustersControl* and *ComputeBodyForces*) is shown by the BDD diagram in Figure 4 whereas the connections among them are highlighted in the IBD diagram in Figure 5.

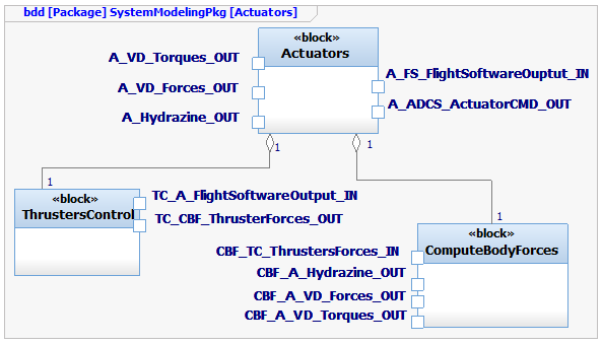

Figure 4: Block Definition Diagram of the Actuators subsystem

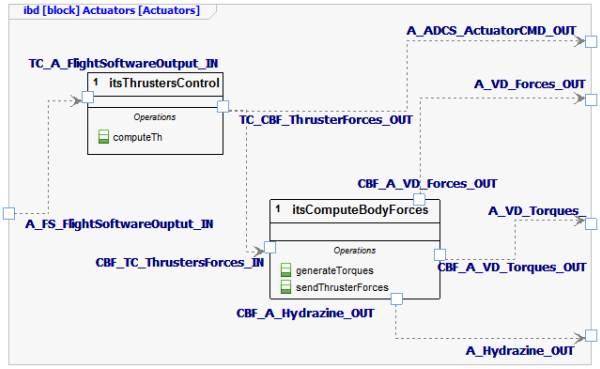

Figure 5: Internal Block diagram of the Actuators subsystem

In Figure 6 the structure of the FlightSoftware subsystem (see Figure 2) in terms of its components (*Navigation* and *AttitudeControl*) is reported exploiting a BDD, whereas its internal structure is highlighted in Figure 7 through an IBD.

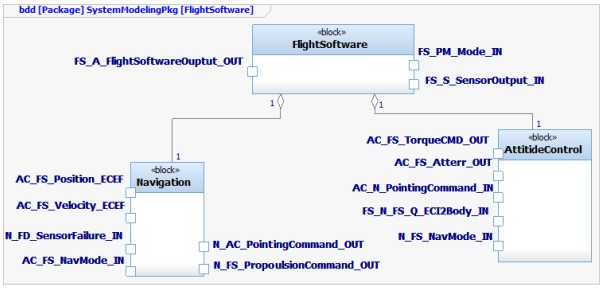

Figure 6: Block Definition Diagram of the FlightSoftware subsystem

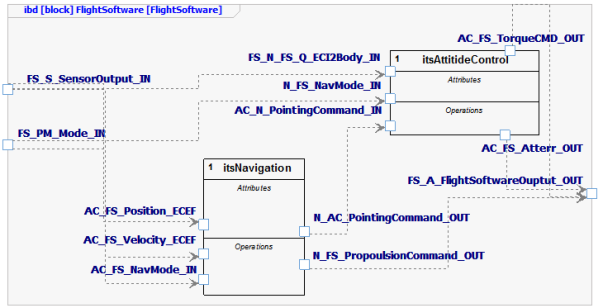

Figure 7: Internal Block Diagram of the FlightSoftware subsystem

3.3.2. Intended Behavior Modeling

The *Intended Behavior Modeling* activity takes as input the hierarchical structure of the system as obtained during the *System Structure Modeling* activity (See Section 3.3.1) and specifies the intended behavior of the system by following a *bottom-up* approach. Specifically, the behavior of the system entities at the lowest level in the hierarchy, or *leaf level* (e.g. component level), are first specified; then the behavior of the entities at higher levels of abstraction, or *non-leaf levels* (e.g. subsystem and system level), are modeled by specifying how the enclosed entities participate and determine the behavior of each considered enclosing entity.

Depending on both, the characteristics of the behavior and the abstraction level to represent, different type of SysML diagrams can be exploited to model the behavior of a given entity: *Activity*, *Sequence*, *Parametric*, and *Statechart Diagrams* (see Garro and Tundis 2012b).

With reference to the ADCS, its behavior depends on the behavior of its subsystems (*FlightSoftware*, *Actuators*, *Sensors*, *PointingMode*, *VehicleDynamics*) and their interactions. In particular, the *FlightSoftware* subsystem is the brain of the system as it takes the decisions to control the satellite system; whereas, the *Actuators subsystem* is used to apply a torque on the satellite. In turn, the behavior of the *FlightSoftware* depends on the behavior of both the *Navigation* and *AttitudeControl* components; whereas the behavior of an *Actuators* subsystem depends on the behavior of both the *ThrustersControl* and *ComputeBodyForces* components, and so on.

In Figure 8 the intended behavior of the *ThrustersControl* component is shown using a SysML Activity diagram. In particular starting from the *torque_cmds* command (or signal), which is received from the *FlightSoftware*, if this command is in one or more axes over a *torque_thresh* threshold, then the appropriate thrusters are set on. If the command falls below the *torque_thresh* threshold, then the thrusters are set off; in particular, if the thruster is set off, then the valve is closed. If the thruster is set on, then the appropriate valve is open. Because the satellite has at 4 of its 8 corners one thruster pack consisting of 3 thrusters (*x*,*y*,*z*), the cycle is executed 12 times. At the end of this task, a signal of *thruster_forces*, which is composed by the single four *packs* is produced and sent in output.

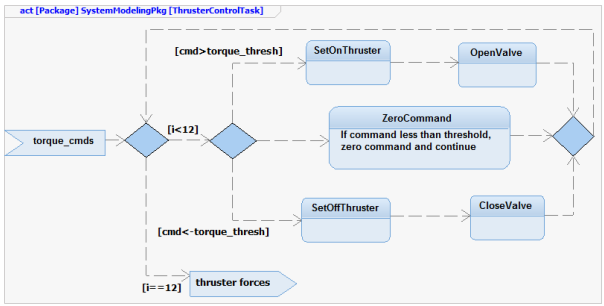

Figure 8: Intended Behavior of the ThrustersControl component

In the following, the behavior of the *ComputeBodyForces* component is described; moreover, one part of it is also represented through the exploitation of a *Parametric Diagram* (see Figure 9). Such behavior is defined as a set of equations, which take as input the *thruster_forces* signal in terms of their single packs: *xyz_pack*_i, for i=1,..,4 (where *xyz_pack_i* is the vector which shows which thruster of the thruster pack is on/off), and *thruster_position_i for i=1,..,4*. All those signals are used to compute *F_Pack1*, *F_Pack2*, *F_Pack3*, *F_Pack4* (where *F_pack* is a vector of forces

of the thruster pack) which in turn are exploited to produce in output a part of the *Forces* signals.

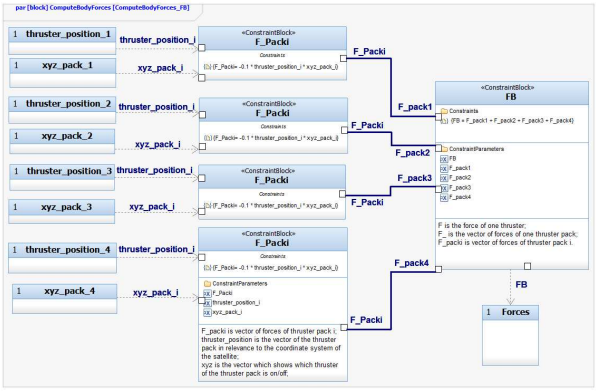

Figure 9: Intended Behavior of the ComputeBodyForces component

As described above, after defining the intended behavior of the entities at the *leaf level* (e.g. the *ThrustersControl* and the *ComputeBodyForces* component for the *FlightSoftware* subsystem), the behavior of the entities at the *non-leaf levels* is specified. As an example, the intended behavior of the *FlightSoftware* subsystem can be derived and represented through a Sequence diagram (see Figure 10) which highlights the iterations among the involved entities; the behavior specified in Figure 8 and in Figure 9 is invoked by the *computeTh()* and *computeForce()* messages respectively.

By applying a similar approach the intended behavior of the whole system can be derived.

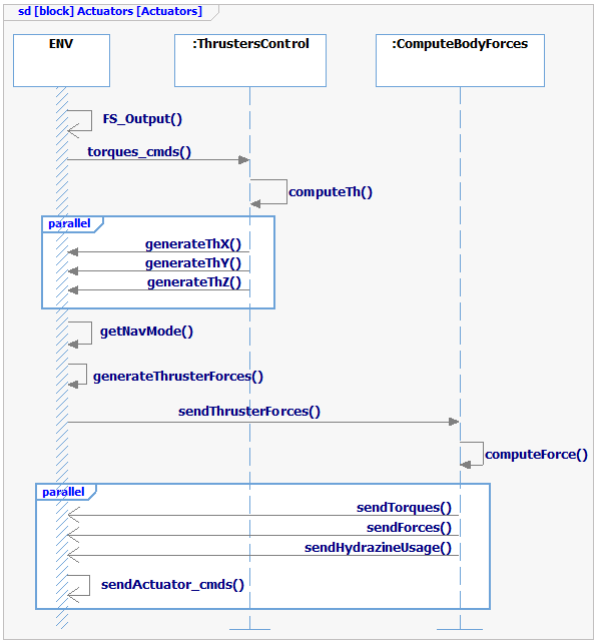

Figure 10: Intended Behavior of the Actuators subsystem

3.3.3. Dysfunctional Behavior Modeling

In the *Dysfunctional Behavior Modeling* activity, the focus is on the modeling of faults and failures, which are key concepts of the system reliability analysis. Specifically, for each entity represented by a SysML Block (see Section 3.3.1), beside the intended behavior, the behavior concerning faults and failures (i.e. the dysfunctional behavior) is specified as a set of dysfunctional tasks (see Figure 11). These tasks could receive as input a set of *failure events* (e.g. due to the failures of other blocks) and could, in turn, produce as output other *failure events* due to the *failure* of the block; moreover, internal *faults* (represented as *fault events*) can be generated and treated inside the block possibly producing block failures (and thus output *failure events*). For specifying these *dysfunctional tasks* six templates have been individuated (see Figure 11): *Fault Generation*, *Failure Generation*, *Failure Management*, *Fault Management Failure Propagation*, and *Failure Transformation*. Moreover, five fault/failure types could be considered (Grunske and Kaiser 2005): (i) *reaction too late*; (ii) *reaction too early*; (iii) *value failure*; (iv) *commission*; and (v) *omission*. By combining the individuated six dysfunctional task types with these five fault/failure types, thirty different basic fault/failure behavioral patterns can be derived (Garro and Tundis 2012b).

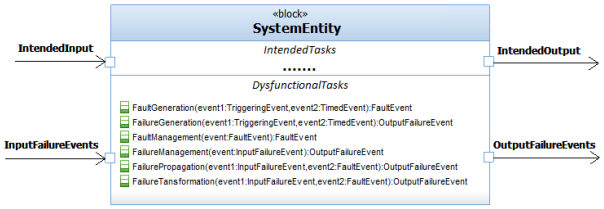

Figure 11: The reference Behavioral Model of an entity

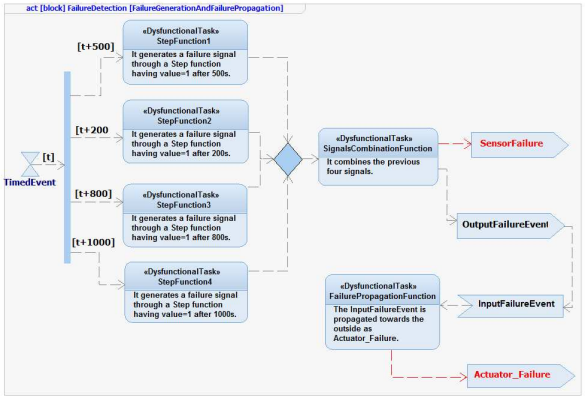

Figure 12: Dysfunctional Behavior of the FlightSoftware Subsystem

As an example, with reference to the system under consideration, both the *FailureGeneration* and *FailurePropagation* templates have been exploited to model the failure generation and failure propagation events of the *FlightSoftware* subsystem. In particular, a *FailureGeneration* task is activated by a *TimedEvent* (manually or by a clock) according to a set of *StepFunctions* having specific *function values* and *delay times* (see Figure 12). Then, two *OutputFailureSignals* are produced: (i) a *SensorFailure* signal which is

directly sent to the *Navigation* component, (ii) a *Actuator_Failure* signal, which is propagated outside towards the *Actuators* subsystem, by applying a *FailurePropagation* task.

When the *Actuator_Failure* signal reaches the Actuators subsystem, it is propagated towards the *ThrusterControl* component. Beside the intended behavior of the *TrusterControl* component, a *FailureManagement* task has been also implemented which, starting from the *Actuator_Failure* signal in input, is able to handle such *InputFailureSignal* or produces an *OutputFailureSignal* which in turn simulates the crash of a whole thruster pack after a specific time.

3.3.4. Behavior Integration

In the *Behavior Integration* activity, both the intended behaviors and the dysfunctional behaviors modeled in the previous modeling activities are integrated to obtain an overall behavioral model of the system and its component entities. As an example, in order to integrate both the *FailureGeneration* and *FailurePropagation* task in the intended behavior of the *FlightSoftware* subsystem, a new software component, called *FailureDetection*, has been introduced (see Figure 13) which implements the dysfunctional behavior represented in Figure 12.

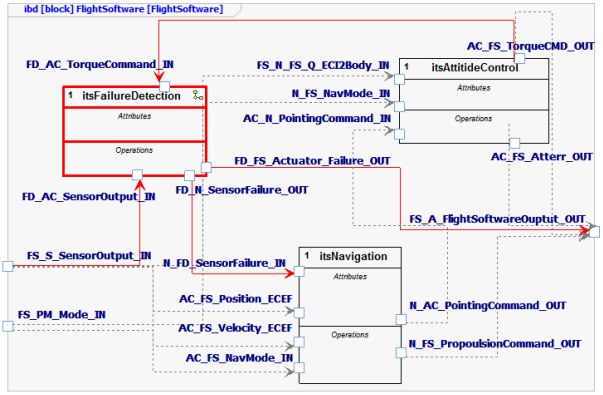

Figure 13: Behavior Integration into the FlighSoftware subsystem

In particular, the *FailureDetection* component takes as input two signals: (i) *sensor_outputs* coming from the Sensors subsystem and (ii) *torque_commands* which is in a feedback. Then, the *FailureDetection* component produces as output (through the *FailureGeneration* task) two signals: (i) *sensor_failure* which is sent to the *Navigation* component and (ii) an *Actuator_Failure* which is sent as output (through the *FailurePropagation* task) to the *FlightSoftware* subsystem.

A similar model has been derived for the *Actuators* subsystem.

This *Behavior Integration* activity closes the System Modeling phase by delivering the *System Model for Reliability Analysis* (*SMRA*) work-product.

3.4. System Simulation

The objective of the *System Simulation* phase is to evaluate through simulation the reliability performance of the system and, possibly, compare different design alternatives and parameters settings. In particular, the following three main activities are performed: *Model Transformation*, *Parameters Setting*, and *Simulation Execution*. Each of these activities is described in the following sub-sections.

3.4.1. Model Transformation

In the *Model Transformation* activity a skeleton of an *Executable System Model (ESM)* is derived from the the *System Models for Reliability Analysis* (*SMRA*) obtained in the previous phase. In particular, in the current version of the RAMSAS method the *ESM* is generated for the Mathworks Simulink platform which represents a de facto standard for the simulation of multi-domain dynamic and embedded systems. This model transformation is based on a mapping between the basic SysML and Simulink constructs; in particular: (i) a (simple) SysML Block is transformed into a Simulink Block; (ii) a (composite) SysML Block, consisting of other blocks (its parts), is transformed into a Simulink Subsystem Block; (iii) SysML FlowPorts are transformed into Input and Output Simulink Blocks; (iv) SysML Flow Specifications, used to type FlowPorts, are transformed into Simulink Bus Objects. Moreover, the SysML behavioral diagrams which model the intended and the dysfunctional system behavior are transformed in Simulink functions and/or Stateflows, according to specific transformation rules. As an example, Figure 14 shows an *ESM* model which has been derived from the ADCS system represented, through a SysML notation, in Figure 2 and Figure 3.

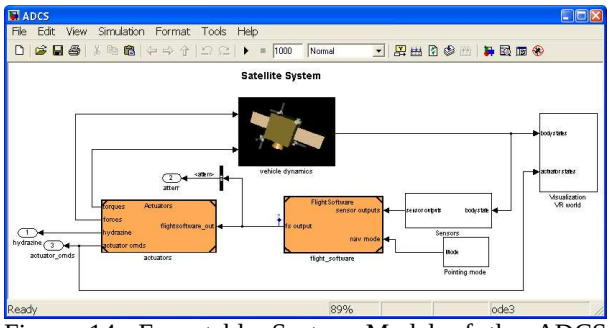

Figure 14: Executable System Model of the ADCS system

Figure 15 represents the full *ESM* model for the Intended Behavior of the *ComputeBodyForces* component, which has been derived from the *Parametric diagram* in Figure 9.

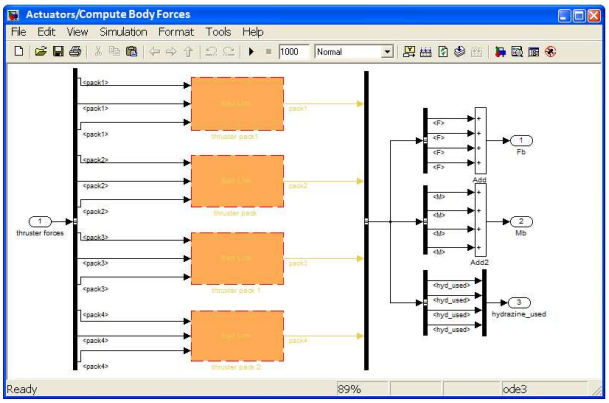

Figure 15: Executable System Model for the Intended Behavior of the ComputeBodyForces component

Figure 16 and Figure 17 show the behavior of the thruster packs through Stateflows after the Behavior Integration activities. In particular, the Stateflow of the *ThrusterLogic* (Figure 16) simulates the behavior of the thruster packs. The function, shown in Figure 17, is derived from the behavior of a thruster pack, shown in Figure 8. This function includes a Failure Management task (in combination with the Stateflow data, see Figure 16), beside the intended and dysfunctional behavior of the thruster packs.

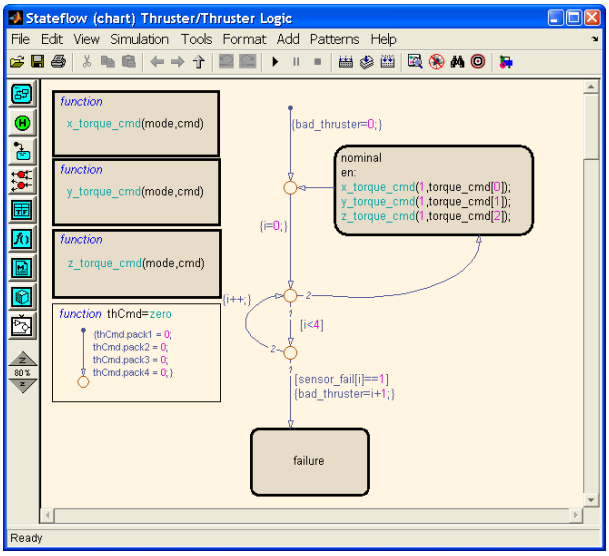

Figure 16: Stateflow of the Behavior of the Thruster Pack

In Figure 18 the Stateflow for the simulation of the behavior of a valve is shown. The valve is for the fuel connection of a single thruster. The Stateflow simulates also the intended behavior and dysfunctional behavior when a failure occurs.

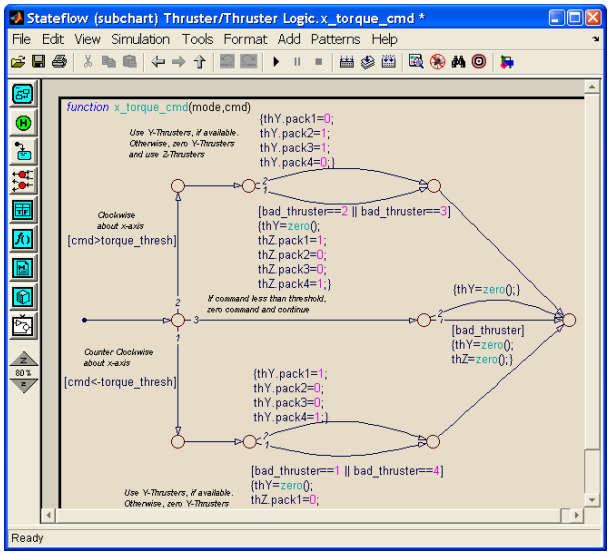

Figure 17: Thruster Pack failure management for x-torque commands

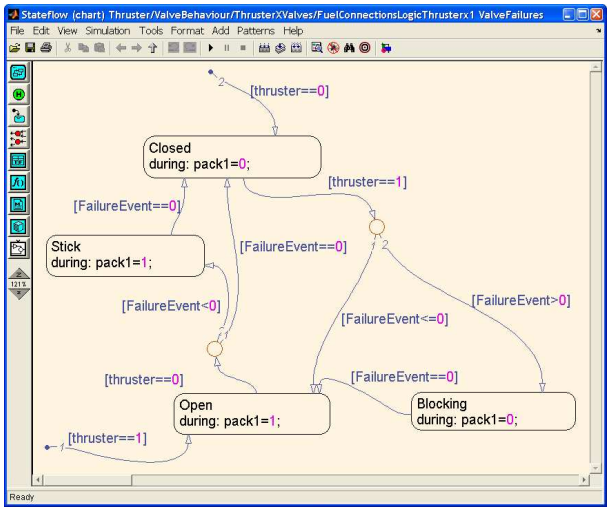

Figure 18: Stateflow of the Behavior of a Valve

3.4.2. Parameters Setting

Before starting the simulation, several system and configuration parameters can be set to evaluate system reliability performance in different simulation scenarios. In the *Parameters Setting* activity, the *ESM* is refined so to allow the flexible setting of system configuration and simulation parameters which can be tuned according to both, the characteristics of the operative scenario to simulate and the failure modes to analyze (by acting on the settings of the faults and failures generation, propagation and management tasks).

For the ADCS different parameters can be set. One parameter is the *torque_thresh* of which variation determines the operating range of the thrusters. Further, the specific impulse (Isp) of the thrusters can be changed, which represents the variation of the thruster and/or fuel. The position of the thruster packs can also be changed. This variation changes the lever for the torque calculation. These are only a few examples, explicit for the actuators, which show the flexibility range of the system.

As an example, Figure 19 shows the Model Explorer panel of Simulink by which the main parameters of the ADCS system can be tuned opportunely.

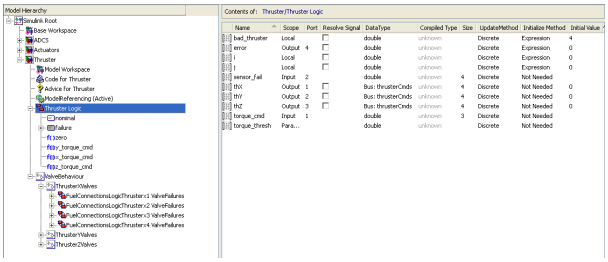

Figure 19: A Screenshot of the Parameters Setting activity

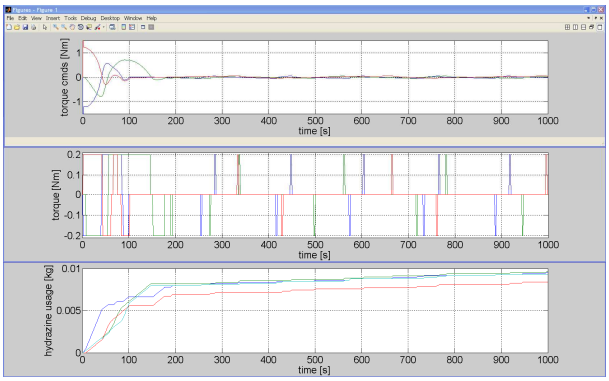

Figure 20: Diagrams for the Intended Behavior

3.4.3. Simulation Execution

In this activity the resulting *ESM*, which is a complete executable Simulink model, is represented as a network of blocks. This model is executed according to a synchronous reactive model of computation: 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. During the simulation *faults* and *failures* are injected (by *TimedEvent* or *TriggeringEvent*) and/or caused to stress and analyze the behavior of the ADCS *system*. At the end of this activity, the data generated from the simulations are reported in the *Simulation Results* (*SIRE)* work-product to be analyzed in the next phase.

Executing the system allows, beside parameter variation, the simulation of the system behavior, during the failure of some components. The simulated failure types can be distinguished in two main types. First, the failure of a thruster pack with all three thrusters can be simulated. All three thrusters of the thruster pack are out of order. The failure is compensated through the use of other thrusters (see Figure 17). The intended choice for raising the torque is with the thrusters in y-axis. However, if one of the used thrusters is defective, then the failure management tries to use the thrusters aligned in z-axis to fulfill the command. Second, the dysfunctional behavior of a valve, between the fuel connection and the thruster, is shown. As reported in Figure 18, while opening, the valve could block and stay closed. Further, while closing, the valve could stick and stay open. In the following sections two examples of simulation executions are described.

In Figure 20 the intended behavior of a simulation is illustrated. The simulation begins with a start alignment of the satellite. The results of the simulation are illustrated within the three diagrams plotted over the simulation time. The topmost diagram shows the torque command, calculated by the FlightSoftware subsystem. The torque threshold is 0.2 Nm. This means the maximum difference between the required torque for a target alignment and the actual alignment in an axis, which is allowed, is 0.2 Nm. If the threshold is exceeded, then the thrusters should create a torque to reach the intended alignment. The three different curves are the separation in the three different directions (x, y, and z). The second diagram in Figure 20 shows the torque and the direction of the torque, which acts on the satellite. It is the sum of the torques of the different thrusters. The different curves show the different directions. Therefore, if the torque command in one direction exceeds the threshold, then the actuators counteract (the same colors in both diagrams indicate the same direction). The third diagram shows the summarized hydrazine usage of the four thruster packs.

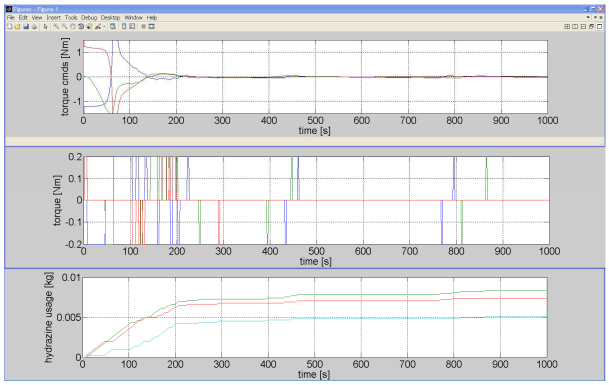

Figure 21: Diagrams for the Dysfunctional Behavior

In Figure 21 the diagrams for the dysfunctional behavior of thruster pack 1 are shown. The simulation was executed with the same parameters as in the simulation of the intended behavior (Figure 20). The diagrams show that the failure of thruster pack 1 can be compensated and the system is still fulfilling its task. Furthermore, it is visible that the compensation of the start alignment takes longer and, at the beginning, is not as exact as in the fully functional case. However, the diagrams show also that the hydrazine usage is lower than with all thruster packs (curve for thruster pack 1 hydrazine usage is x = 0). The lower hydrazine consumption results from the lower angular velocities used to align the satellite in this case.

3.5. Results Assessment

In the *Results Assessment* phase, the simulation results (*SIRE*) are elaborated with reference to the objectives of the reliability analysis identified in the initial phase of the process so to obtain important information on the reliability properties of the system under consideration.

In particular, the analysis of the resulting graphs or of the obtained data can be conducted. The use of domain experts should not be underestimated to obtain a good analysis of the results and their evaluation, since an effective re-design of the system is also an outcome of a deep knowledge of the domain. These analyses are able to give information about the reliability performances of the ADCS system under consideration as reported in the *Reliability Analysis Report* (*RAR) document*; moreover, they also provide suggestions to improve the reliability of the system proposing alternative design solutions as reported in the *Design Suggestions Report* (*DSR) document*.

A great part of these analyses are directly performed in Simulink, whereas more advanced analyses are also performed by external tools after exporting the results obtained through the Matlab environment.

The ADCS with its intended and dysfunctional behavior, which is the system under consideration, can be improved by analyzing the results of the simulation execution. A variation of the parameters expands the understanding of the reliability of the system. Some failures, which were created, have a deep impact on the system; therefore a failure management must be created to solve these problems, while other failures have almost no impact on the reliability of the system. The following simple example makes this clearer. The failure mode *blocking* of a valve causes that the effected thruster cannot be used anymore. Due to the redundant thrusters the only impact is an increased maneuver duration, because only half of the thrust for creating the torque will be available. The system itself will however not fail. An opposite failure is the failure mode *sticking* for a valve. This failure mode causes a constant fuel flow and thruster use. Due to this fault a different failure will happen: the system will counteract and therefore, the opposite thrusters will be activated. This holds the satellite in position, but the whole time fuel will be required, until the tank is empty. This failure leads to a fail of the whole system. Looking on these failures, a system design with an extra valve in front of the whole thruster could be developed. To enable the failure detection, a sensor surveying the fuel flow is also required.

Furthermore the reliability simulations lead to other aspects which could be considered. For example, the whole system fails when the fuel tank is empty. On the other hand the system has a longer lifetime, if less fuel is used. The simulation of the dysfunctional behavior showed, that with longer acquisition times allowed for the system, the fuel consumption sinks. The satellite will also reach its target, but with more time required for the target acquisition and with larger deviations from the target attitude. The attitude accuracy is required by the payload camera looking towards the earth surface under a specific angle. The availability of the camera drops if the acquisition time increases. This results lead to new requirements, which could lead to a change in the mission specification.

4. CONCLUSIONS

In the paper RAMSAS, a recently proposed model-based method which supports the reliability analysis of systems through simulation, has been exploited for the reliability analysis of an Attitude Determination and Control System (ADCS) of a satellite. Specifically, according to the RAMSAS proposal, the derivation of a simulation model from a SysML system model for both, the functional and dysfunctional system behavior, has been shown. The concrete exploitation of RAMSAS has allowed appreciating its effectiveness and suitability both in the system structural and behavioral modeling and in the evaluation through simulation of the system reliability performances. Moreover, as SysML is one of the standard modeling languages for Systems Engineering, the integration of such simulations into the system design enables a seamless development process. Indeed, the design can be developed in one SysML model from the modeling of functional and non-functional requirements up to the designed behavior simulation. Thus the impact of topological and parametrical system changes on both, the functional and non-functional system requirements, can be observed in the same model.

With reference to the presented case study, by the simulation of both, the functional and the dysfunctional behavior in one model, it was found out that the ADCS system is more fuel efficient in one of the fault modes. The combined simulations can thus lead to interesting insights about the system behavior.

For the future, it is desirable to build up the SysML model in a more automated manner so to allow for a more flexible analysis of the impact of different design choices, such as topological system changes, on the reliability performances of the system.

ACKNOWLEDGMENTS

Andrea Tundis was supported by a grant funded in the framework of the “POR Calabria FSE 2007/2013”.

REFERENCES

Cressent R., Idasiak V., Kratz F., David P., 2011. Mastering safety and reliability in a model based process. *Proceedings of Reliability and Maintainability Symposium (RAMS)*, January 24-27, Lake Buena Vista (Florida, USA).

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

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

Garro A., Tundis A., 2012a. Enhancing the RAMSAS method for System Reliability Analysis: an exploitation in the automotive domain. To appear in the *Proceedings of the 2nd Int. Conf. on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH)*, 28-31 July, Rome (Italy).

Garro A., Tundis A., 2012b. Modeling and Simulation for System Reliability Analysis: The RAMSAS Method. *Proceedings of the 7th IEEE International Conference on System of Systems Engineering (IEEE SoSE)*, 16-19 July, Genova (Italy).

Garro, A., Tundis, A., 2012c. A model-based method for system reliability analysis. *Proceedings of the Symposium on Theory of Modeling and Simulation (TMS)*, 26-29 March, Orlando (Florida, USA).

Garro, A., Tundis, A., Chirillo, N., 2011. System reliability analysis: a model-based approach and a case study in the avionics industry. *Proceedings of the 3rd Air and Space Int. Conf (CEAS)*. 24-28 October, Venice (Italy).

Groß, J. Rudolph, R. 2012a. Generating Simulation Models from UML – A FireSat Example. *Proceedings of the Symposium on Theory of Modeling and Simulation (TMS)*, 26-29 March, Orlando (Florida, USA).

Groß, J., Rudolph, R., 2012b. Dependency Analysis in Complex System Design using the FireSat Example. *Proceedings of the INCOSE Symposium*, Rome (Italy).

Groß, J., Rudolph, R., 2011. Hierarchie von Entwurfsentscheidungen im modellbasierten Entwurf komplexer Systeme. Tag des Systems Engineering der GfSE, Hamburg (Germany).

Grunske, L., Kaiser, B., 2005. Automatic Generation of Analyzable Failure Propagation Models from Component-Level Failure Annotations. *Proc. of the 5th Int. Conf. on Quality Software (QSIC)*, Melbourne (Australia).

Harland, D.M., Lorenz, R.D. 2005. *Space Systems Failures, Disasters and Rescues of Satellites, Rockets and Space Probes*. Springer Berlin, Germany.

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

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

Iwu, F., Galloway, A., McDermid, J., Toyn, I., 2007. Integrating safety and formal analyses using UML and PFS. *Reliability Engineering and System Safety*, 92, 156–170.

Larson, W. J., Wertz, J. R., eds. 1999. *Space mission analysis and design.* 3rd ed., Microcosm Press, El Segundo (California, USA).

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

Wertz, J., Ed. 1978. *Spacecraft Attitude Determination and Control*. D. Reidel Publishing Company, Dordrecht, Netherland.