SySTEMA

SYstem & Safety Tool for Executing Model-based Analyses

Alessio Costantini, Fancesco Inglima, Rodolfo Mazzei, Sergio Di Ponzio
System Engineering Local Expertise Center
ALTRAN ITALY
alessio.costantini@altran.com, francesco.inglima@altran.com,
rodolfo.mazzei@altran.com, sergio.diponzio@altran.com

Andrea Chiellini, Cristina Biagi
RAM and Safety Local Expertise Center
ALTRAN ITALY
andrea.chiellini@altran.com, cristina.biagi@altran.com

***Abstract* — This paper presents SySTEMA, an innovative approach to perform Safety Analyses on complex systems, based on the modelling of their functionalities, behaviors and architecture. System safety analysis techniques are well known and are extensively used during the design of safety-critical systems. Since these analyses are highly subjective and dependent on the skill of the practitioner, it is unlikely that they will be complete, consistent and error free. In fact, the safety engineers devote much of their effort to find undocumented details of the system behavior and to embed this information in the safety artifacts such as the fault trees. Most of the review effort is focused on uncovering and resolving misunderstandings and missing information in the system design or the informal fault model. Model-Based Systems Engineering (MBSE) is a methodology aiming to design and develop complex and/or critical systems, increasing productivity by promoting communication among different teams working on the same project. Model-Based Safety Analysis (MBSA) is an emerging discipline that extends MBSE performing safety analyses in a 'model-based' context, by building system models (both for nominal and fault behavior), reducing the effort and increasing the quality of the final results. MBSA follows a failure analysis approach, starting from the major state-of-the-art techniques such as Failure Mode Effect and Criticality Analysis, Functional Analysis, Functional Hazard Analysis and Fault Tree Analysis. This paper describes a theoretical approach to implement MBSA using one common SysML model of the system. This allows the systems engineers to perform automated safety analyses to receive quick feedback on their design decisions during the system design phase.**

Keywords—MBSE; MBSA; SysML; FMEA; FTA.

I. INTRODUCTION

SySTEMA ("System & Safety Tool for Executing Model-based Analyses") is an innovative approach that will allow to analyze the Safety of complex systems, which typically are used in the field of aerospace, defense, rail and automotive industries.

Safety analyses are currently carried out to support the design of "safety critical" systems, according to a traditional approach, i.e. starting from the documentation usually drawn downstream of the detailed design of the system.

As a consequence, this approach often leads to analyze the system safety late and in ineffective way.

Moreover, being based on the interpretation of the documents describing functionalities and architecture, these analyses can present problems of completeness, consistency and subjectivity, because highly dependent on the experience of the Safety engineer performing the activities.

The proposed project aims to overcome the above cited problems by using an approach based on the realization, through the use of a graphical language standard, called SysML, of a unique model, which is representative both of the functional/architectural characteristics and of the behavior of the system when a failure occurs.

Specifically, the idea is to define the most appropriate methodology to model the fault types and the effects that they may have about the different features of the system.

The main objective of the proposed project is to realize an innovative tool (SySTEMA) to perform Safety Analyses on complex systems, by using Systems Modelling Language (SysML).

MBSA modelling approach has been evaluated by comparing FMEA and FTA analyses carried out with traditional methods with the ones generated by SySTEMA (MBSA tool).

II. Background

This section briefly describes the steps in the SysML modelling and safety analysis process considered in this study.

A. SysML Modeling

The SysML approach used in this study considers the operational analysis, functional analysis and architectural analysis [1].

The operational analysis will define the main use cases (UC) of the system, with a particular focus on the UC significant from a safety viewpoint. The primary goals of this analysis shall be:

- The definition of the system environment from the user perspective, by identifying all the entities that directly interact with the system (actors).
- The identification of the most significant UCs from a safety standpoint, to be detailed by the functional analysis.

The functional analysis will allow to summarize the key functions of the system when it is in a specific UC. Each main function will be decomposed in lower hierarchical level functions that can be furthermore decomposed following an iterative process, in order to obtain a functional breakdown. The lowest functional level will be mapped to an architectural subsystem or component (allocation).

The functional analysis will also allow to represent the system behavior, which can be modelled by means of different behavioral diagrams:

- **State Machine Diagrams** (SMD), to represent the states, the transitions and the actions that the system will perform in response of well-defined events.
- **Sequence Diagrams** (SD), to represent the event-based behavior, representing flow of control and describing interactions among system parts.

The architectural analysis will allow to describe the system in terms of internal components decomposition (subsystems or components) and functional interfaces description. Typical diagrams used in this phase are:

- **Block Definition Diagrams** (BDD), to define the system, by means of associations and composition relationships.
- **Internal Block Diagrams** (IBD) to describe the structural aspect of the model, defining how the different items collaborate and exchange information to realize the behavior of the entire system

Functional and architectural analyses are typically performed in parallel, through an iterative process which will converge towards the final system architecture. Once the two analyses will be complete, each function should be mapped to only one architectural element.

B. Classical Safety Analysis

The classical safety analyses considered in this study are the Failure Mode Effects Analysis (FMEA) and the Fault Tree Analysis (FTA).

The FMEA is a "bottom-up" analysis based on a single-failure approach and executed on each system item or functional block, according to the following main steps:

- Identification of credible failure modes;
- Evaluation of each single failure mode effects at different levels up to system one;
- Evaluation of severity of the failure effects consequences;
- Identification of failures detection method;
- Assignment of the failure mode rate based on item reliability and apportionment criteria.

A FTA is a model that graphically and logically represents the combinations of failures occurring in a system that lead to an hazardous condition.

FTA uses a "top-down" approach, in order to identify all potential causes of a particular undesired top event.

Starting from the Top Event, the analysis systematically determines all possible causes, both single fault and combination of faults, at the subsequent lower levels until a Basic Event is encountered.

A Basic Event is defined as an event that is no further developed into a lower level of detail. If a basic event is attributed to items failures, it can be extracted from item failure modes analyzed in FMEA.

As shown in Fig. 1, the fault tree development includes two types of symbols: event and logic gate. An event symbol is used to describe an existing condition or a physical event. A logic gate is used to tie together the events and to show the logical relationship among them.

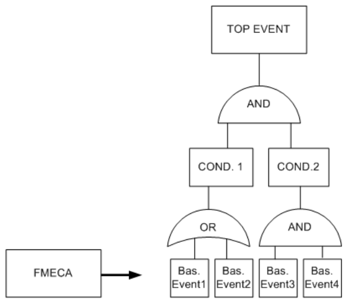


Fig. 1. FTA Events and Gates

This FTA method produces Boolean logic models representing the logic relationships between events leading to a final condition. The Boolean model is suitable to produce both qualitative and quantitative results. The qualitative results of the Fault Tree model is represented by the possibility to have a clear understanding of all the combinations of failure events, which can cause the 'top event' (Minimal Cut Sets). As for the quantitative aspect, the logical model of FTA is moreover suitable to evaluate the probability of occurrence of the top event based on the failure rates of basic events, exposure times, and the Boolean Model that is at the base of the Fault tree development.

III. SySTEMA Approach

SySTEMA approach consists of a framework for System Engineer and Safety Engineer containing:

- **Theory and Fundamentals**: definition of the theoretical approach representing the foundations of SySTEMA
- **Operative Guidelines**: definition of modelling rules, constraints and step-by-step descriptions of actions to be performed with a tool.
- **A tool**: Commercial Tool + Altran SW Application.
- **A proof-of-concept (demonstrator)**: a tangible model of a case study to see how FMEA and FTA analyses can be automatically performed by the SySTEMA tool.

The theoretical approach develops a process that leads the engineers to develop a model of the system on which the SY.S.T.E.M.A tool can automate the generation of FMEA and FTA analyses.

A. Process

The process of SySTEMA is shown in Fig. 2. It is divided into three main phases and consists of seven steps that engineers need to follow in order to automatically generate the FTA and FMEA analysis.

The process includes a phase in which to update and enrich the SysML model according to the criteria described in the following chapters. If the model does not exist, the same criteria can be used to define the entire architectural model of the system. The goal is to extend the model with stereotypes allowing to the SySTEMA tool the interpretation of the system information and produce the analysis. Furthermore, the approach describes the way to define the operations and their allocation in the same model. Finally, not least, this first phase defines the way for the allocation of the failure modes in the model. The definition of failures remains an activity related to the best practice of safety domain.

In the second phase of the process, the engineers shall model the system in nominal behavior and in presence of failures. The modeling of the behavior takes place through an abstract way, very close to a logic level description of a system, but which can be used both to describe the high-level functional behavior and the lowest level physical one. The description of behavior is a sequence of messages with abstract parameters. This permits to describe any system as an algorithm, using three fundamental constructs (Böhm-Iacopini, [2]): the sequence, the selection, the selection and the cycle (iteration). This approach can work thanks to abstraction of the logic of VALUE TAGs (described in par. V.B), that allows to express more situations (functional, logical and physical) with a language easily integrated in the FMEA and FTA formal output.

Finally, in the last phase will be defined a scenario trough a Use Case of the system in order to obtain the FMEA and FTA automatically. In the next paragraph the three macro steps of the process are described in detail.

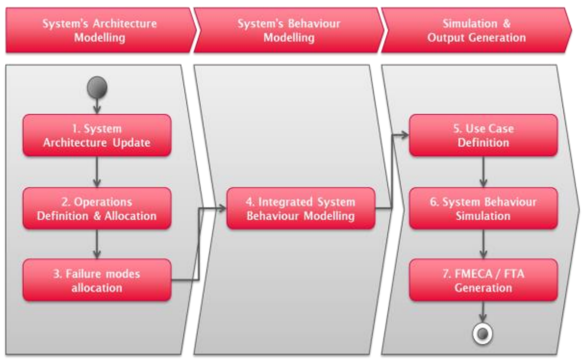

System's Architecture Modelling
System's Behaviour Modelling
Simulation & Output Generation
1. System Architecture Update
2. Operations Definition & Allocation
3. Failure modes allocation
4. Integrated System Behaviour Modelling
5. Use Case Definition
6. System Behaviour Simulation
7. FMECA / FTA Generation

Fig. 2. SySTEMA Process

B. Tool

SySTEMA tool is a general purpose tool that can operate with different MBSE commercial tools in order to obtain the automatic generation of safety analyses. For this study it has been developed only an interface for Artisan Studio. The output of the tool is an Excel Sheet.

The tool can simulate the behavior of the model thanks to the SySTEMA profile defined in the MBSE tool with custom stereotypy.

The graphical output of FTA, in classical view, is made by a commercial tool.

The entire tool-chain is described in figure 3, the MBSE tool is the base of information of the model. Through a Visual Basic (VB) software, the static model is simulated in order to obtain the FMEA and FTA analysis. The VB software provides a tabular output suitable to be manipulated to obtain a typical output of safety analysis.

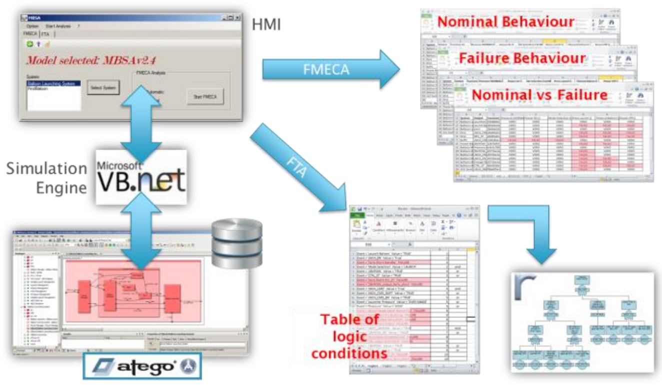


Fig. 3. SySTEMA Tool-Chain

IV. System's Architecture Modelling

The system architecture has to be adapted according to the guidelines & fundamentals and enhanced with dedicated stereotypes.

A. System Architecture Update

Starting from an existing (and maybe complex) model of the system architecture, System and Safety Engineers will have to identify the levels of interest on which SySTEMA will be able to perform the Safety Analyses.

The architectural level of each block will be "classified" through a specific stereotype (see Fig. 4) in order to identify three indenture levels of interest, required to perform the FMEA analysis according to MIL-STD-1629:

- **≪Local Level≫**: level of the specific item being analyzed.
- **≪Next Higher Level≫**: next higher indenture level above the indenture level of the specific item being analyzed.
- **≪End Level≫**: the highest (root) indenture level.

Typically, the ≪Local Level≫ is associated to "elementary" blocks of system architecture (items not further developed).

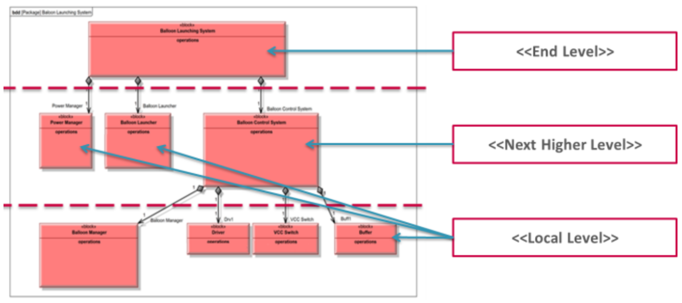


Fig. 4. Architectural Update

SySTEMA will use the system structure, described through an Internal Block Diagram (IBD) (see Fig. 5), to extract the interconnections between the elements of the system.

All the connections must be modelled by means of:

- Connectors between elements (SysML Block Properties)
- Connectors between ports

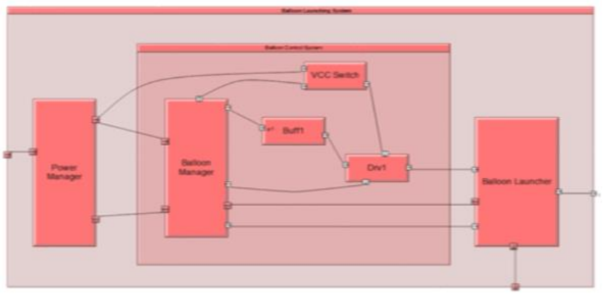


Fig. 5. Interconnections between elements

SySTEMA can manage different levels of the system architecture and the corresponding levels of abstraction:

- **Functional architecture**, which defines a solution-independent representation of the design; it is composed by pure functions.
- **Logical architecture**, which represents an intermediate abstraction between functional and physical architecture. Blocks of a logical architecture represent abstractions of physical solutions.
- **Physical architecture**, which gives the physical resources to perform the system functions.

For each level of abstraction, the architecture will contain several blocks implementing high-level or low-level functions. In SysML, when a function is defined in a block, it is called Operation. An Operation is the specification of a behavioral feature of a block, at any level

B. *Operations definition and allocation*

Functional analysis, already performed by Systems Engineers on existing systems, will be used by SySTEMA as a reference for the "Operation Definition". No particular constraint is imposed to functional analysis, thus it can be performed according to the identified level of abstraction. In functional architecture the operation of a block is strictly connected to the system functions. In logical/physical architecture each block can have one or more operations, according to design choices; as an example, a System Engineer can split a function into more than one operation of a component, or can group more functions in an operation of a component (see Fig. 6).

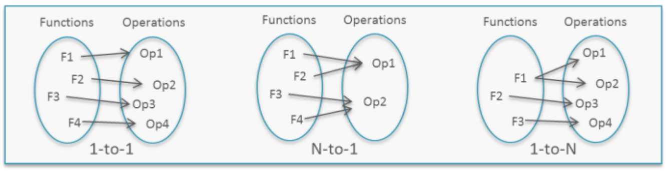


Fig. 6. Functions vs Operations

The operations must be hierarchically traced among the levels of a system, in order to propagate the effect of the operation from local level to system level (see an example in Fig. 7). The hierarchy depends on the Operation Definition and it will be defined by means of a specific stereotype. In this way it will be possible to identify for each operation its Father Operation.

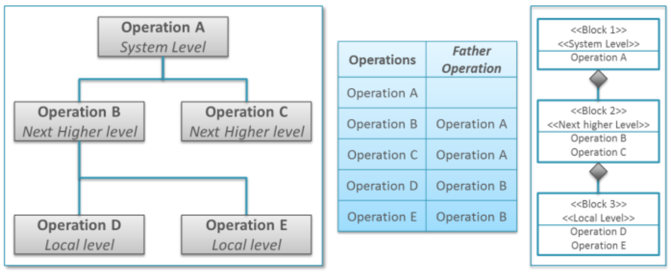


Fig. 7. Hierarchy of Operations

C. *Failure modes allocation*

Failure modes identification is a safety activity that consists in generating a list of failure modes for each architectural block of the system, classified in the BDD as ≪Local Level≫.

Failure Modes are characterized by the following features:

- **Failure Description**: description about how a failure occurs (free text).
- **Failing Entity**: type of failure.
- **Failure Behavior**: input for Integrated System Behavior Modelling.

SySTEMA allows to manage any type of failure modes, depending on the level of abstraction assigned to the ≪Local Level≫ blocks. This goal is achieved by considering the following failure modes types:

- **OUTPUT**: failure mode affecting the out-coming signal of a ≪Local Level≫ block.
- **FUNCTION**: failure mode affecting the function of a ≪Local Level≫ block.
- **COMPONENT**: failure mode affecting physical components.

For physical architectures the failed output will be a physical signal, for functional architectures the failed output will be a function.

For physical architectures, the failure will affect one or more physical signals of the failed block, for functional architectures the failure will affect one or more functions of the failed block (see Fig. 8).

Component failures are applicable only to physical architectures, including for example (not exhaustive list): Mechanical parts, Switching devices, Capacitors, Connectors, Integrated Circuits, Optical, Lamps, Resistors, Semiconductors, Microprocessors.

The output failure modes currently managed by SySTEMA are:

- Analog Signals (threshold, single value):
  - Loss
  - Above/Below Threshold
- Analog Signals (boundaries, in between):
  - Loss
  - Out of Range
  - Stuck in Range
- Digital Signals (boolean):
  - Stuck at logic HIGH
  - Stuck at logic LOW
- Digital Signals (enumerated):
  - Stuck at specific value

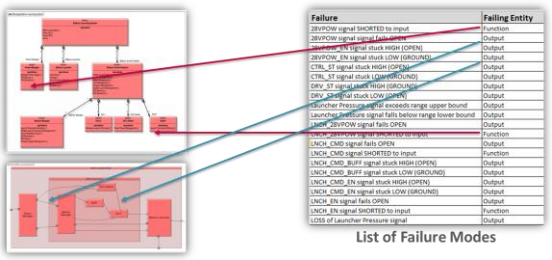
The function failure modes currently managed by SySTEMA are:

- Analog Signals:
  - Double set of analog signals
  - Short from In and Out
  - Short From In and Out with set of a third signal (Status signal)

- Digital Signals:
  - Double Stuck at logic level HIGH or LOW
  - Short from In and Out
  - Short From In and Out with set of a third signal (Status signal)

For each local level block, System Engineers will have to allocate failure modes to model through stereotypes.

Function/Component failing entities will be associated to operations and Output failing entities will be associated to signals.



Failure	Failing Entity
28VPOW signal SHORTED to input	Function
28VPOW signal signal fails OPEN	Output
28VPOW_EN signal stuck HIGH (OPEN)	Output
28VPOW_EN signal stuck LOW (GROUND)	Output
CTRL_ST signal stuck HIGH (OPEN)	Output
CTRL_ST signal stuck LOW (GROUND)	Output
DRV_ST signal stuck HIGH (GROUND)	Output
DRV_ST signal stuck LOW (OPEN)	Output
Launcher Pressure signal exceeds range upper bound	Output
Launcher Pressure signal falls below range lower bound	Output
LNCH_28VPOW signal fails OPEN	Output
LNCH_28VPOW signal SHORTED to input	Function
LNCH_CMD signal fails OPEN	Output
LNCH_CMD signal SHORTED to input	Function
LNCH_CMD_BUFF signal stuck HIGH (OPEN)	Output
LNCH_CMD_BUFF signal stuck LOW (GROUND)	Output
LNCH_CMD_EN signal stuck HIGH (OPEN)	Output
LNCH_CMD_EN signal stuck LOW (GROUND)	Output
LNCH_EN signal fails OPEN	Output
LNCH_EN signal SHORTED to input	Function
LOSS of Launcher Pressure signal	Output

Fig. 8. Failure modes allocation to model

V. System Behaviour Modelling

The system behavior follows two scopes: definition of the behavior and its modeling. The definition (see Fig. 9) is an activity that has to be performed by both system and safety engineers, in order to obtain the information useful to model the behavior. Based on the definition, the system engineers shall model the behavior. This model use an approach based on the Sequence Diagram described in the next paragraph.

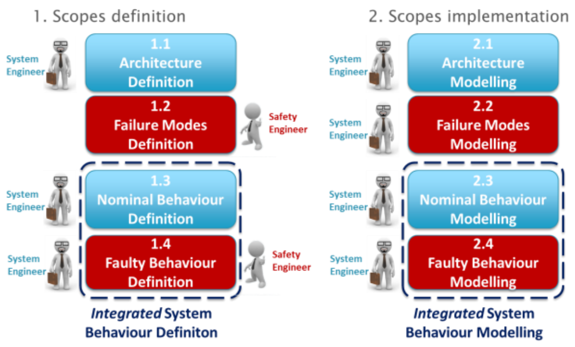

1. Scopes definition
2. Scopes implementation
System Engineer
1.1 Architecture Definition
1.2 Failure Modes Definition
Safety Engineer
1.3 Nominal Behaviour Definition
1.4 Faulty Behaviour Definition
Integrated System Behaviour Definiton
2.1 Architecture Modelling
2.2 Failure Modes Modelling
2.3 Nominal Behaviour Modelling
2.4 Faulty Behaviour Modelling
Integrated System Behaviour Modelling

Fig. 9. Behaviour definition and modeling

A. *Message-Based Approach*

System behavior is described by means of Operations described with the Message-Based Approach (MBA).

MBA consists in the usage of sequence diagrams (OSD) to represent the interaction among elements/components/items of a system as a sequence of message exchanges.

Messages can be external events, signals or failure events. The interaction can be:

- between the system and its environment (external event)
- between the elements/components/items of a system (signals)
- by means of a failure mode (failure events).

Messages can have parameters representing the information content.

Parameters must assume specific tag values, corresponding to the state of the signals (i.e. input or output of the associated operation).

B. *Operations and Tag Value*

An operation describes how the input affects the output according to a cause and effect principle (see Fig. 10).

Tag values represent the foundations of MBA, because they are used to describe different states of input and output signals (see Fig. 11).

The approach requires to treat a component as a black box and the associated operation, which "transforms" inputs into outputs, is assumed to be a cause-effect.

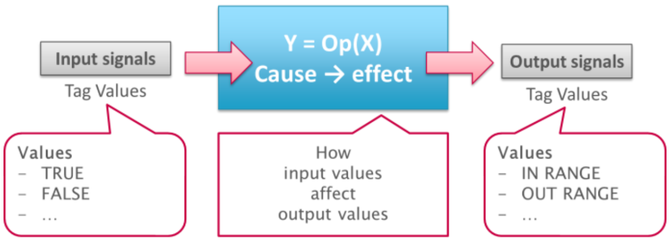


Fig. 10. Cause-Effect

The state of inputs and outputs will depend on the context of the considered system. It can be a classification or a grouping of the real values of a signal; e.g. Analog Values in a Range or out of a range, digital signal in a specific discrete values (see next figure). Otherwise, it can be a specific condition of a signal or a function (i.e. signal loss, missed function and active function). Finally, it can be a condition which may be considered valid for different values over time in the specific analysis. In this way also the temporary effect can be handled in a single state.

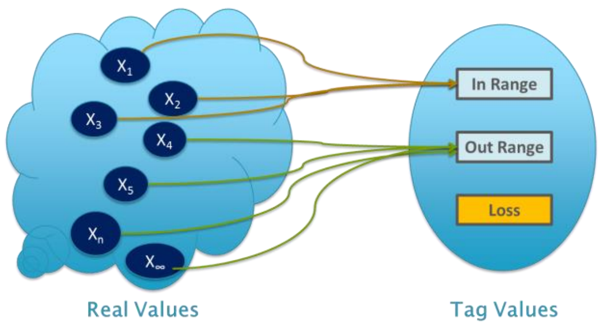


Fig. 11. Tag Value

The Operations can describe different levels of abstraction by using different classifications of tag values (corresponding to different states of inputs and outputs).

C. *How to define Tag Values*

Tag values will depend on the level of abstraction. In the Fig. 12 is reported an example. The example of the light in a room allows to understand how the Tag Value, using different meaning of the tags, can help the engineer to work at three level of abstraction. In functional description, the operation has the functionality to light the room. This functionality, in a logical view, can be considered as an ON/OFF signal, able to switch on/off a light. In the physical level, after some design choices (use the 220V voltage and physical contact), the operation describes the physical behavior of the switch.

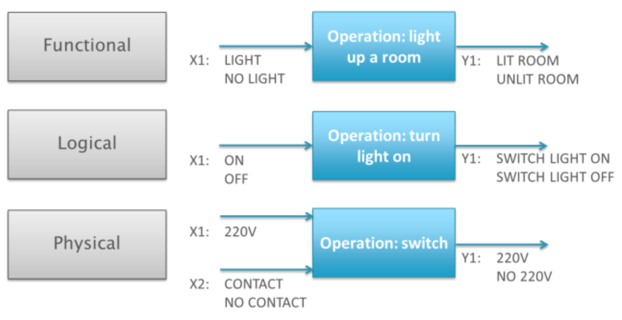


Fig. 12. Tag Value at different levels

D. Failure mode injection

Failures will affect operations depending on the failure mode type:

- **Output failure** will impose the tag value to the associated signal, see Fig. 13
- **Function or component failure** will impose the behavior to the associated component: see Fig. 14

Both failures can generate new tag values for the output of the operation.

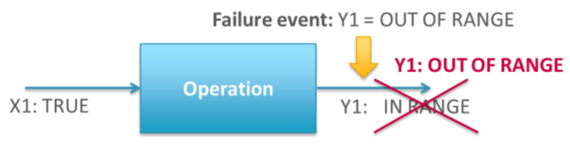


Fig. 13. Output Failure

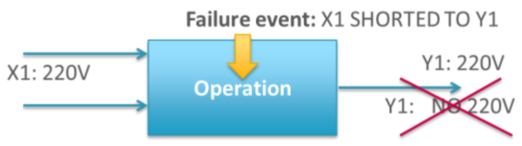


Fig. 14. • Function or component failure

A failure mode could require the definition of new tag values (w.r.t. the nominal behavior) in order to manage the following behaviors:

- Behavior associated to an internal failure of the block itself
- Behavior associated to input signals affected by a failure which has not been previously managed

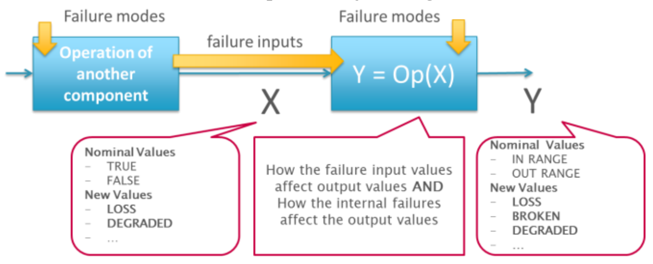


Fig. 15. Failure impact beetwen components

Tag values will depend on the level of abstraction. The failure mode could use new tag values (**bold**), see Fig. 16.

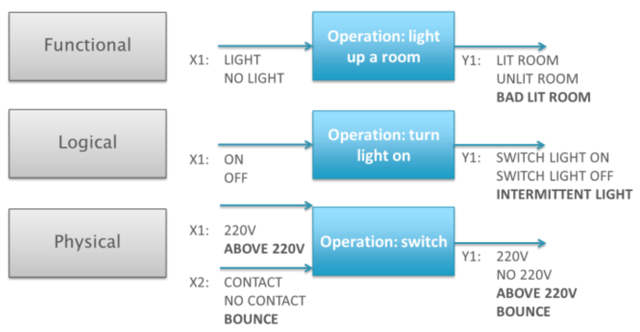


Fig. 16. Failure Tag Value in different levels

E. Cause and effect

The fundamental assumption in MBA is «If **Causes** Then **Effects**». MBA will use the constructs defined as follows:

TABLE I. CONSTRUCTS

ID	Construct
1	If Causes then Effects
2	If Causes1 OR Causes2 then Effects
3	If Causes1 AND Causes2 then Effects

- **Causes**: Start, Input Messages, Sequences.
- **Effects**: End, Output Messages, Sequences.
- **Sequences**: combination or loop of multiple constructs (ID1, ID2, ID3).

Similarly to Böhm-Jacopini theorem, it is assumed that any operation can be implemented by using only the following elementary structures (to be applied also recursively): the sequence, the selection and the cycle (iteration).

All the above cited structures are part of the OSD semantic, as reported into the following table:

TABLE II. TABLE STYLES

Structure Element	SysML OSD
the sequence	SEQ
the selection	SEQ , ALT and PAR
the cycle (iteration)	LOOP

For this reason, SySTEMA makes use of OSDs to model an operation as in the example in the Fig. 17.

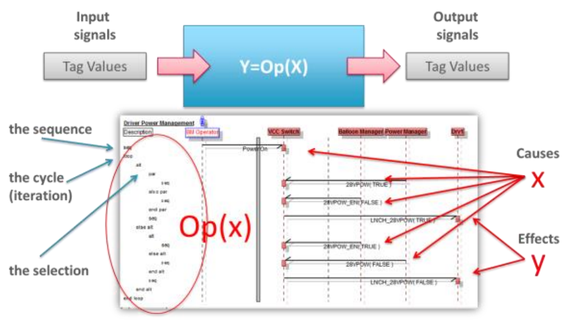


Fig. 17. Sequence Diagram for Cause-Effect behavior

The failure can be represented by a single message (failure in output) or by a sequence of multiple messages (failure in function or component).

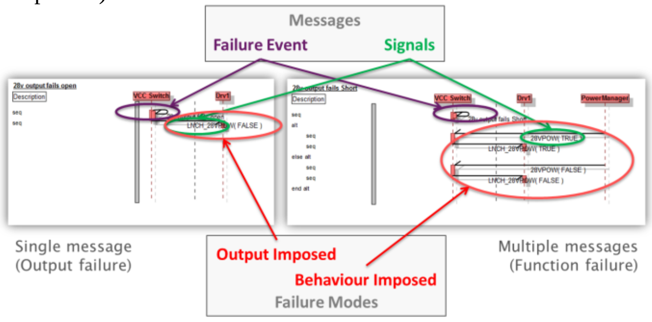


Fig. 18. Sequence Diagram for failure

State Diagrams (STD) and Activity Diagrams (AD) can be used in SysML to describe sequences, selections and iterations, but they are not the right diagrams to implement the MBA because:

- OSD is a more intuitive diagram to implement a cause-effect logic. The typical OSD steps (seq, alt, par, loop) can be easily mapped to the MBA construct (Sequence, Selection, Cycle);
- Description of messages exchange between blocks is more complex with STD or AD, than with OSD;
- In STD most of the MBA constructs have to be described in text mode and not in graphical mode;
- It is easier to implement a Reverse navigation for FTA (to perform a top-down approach), by using OSD than STD or AD;
- STD and AD are less integrated with IBD than OSD;
- OSD can be added as a new layer to an existing model behavior, with no impact to the already defined states of a block.

VI. Simulation & Output Generation

SySTEMA will generate the FMEA and FTA output, through simulations of system behavior.

A. *Preliminary operations*

The simulation is configured by means of Use Cases diagrams. UC represents the operative scenario inside which the system will operate. Use cases will consist of the following entities:

- Actors (operator, user, environment, …)
- System under analysis
- External events

System behavior is strictly dependent on external events. In SySTEMA Use Cases will be described through OSD, as in the following figure. The actor, through external signals, stimulates the system in a specific scenario. Different uses of the system will be described by different use cases.

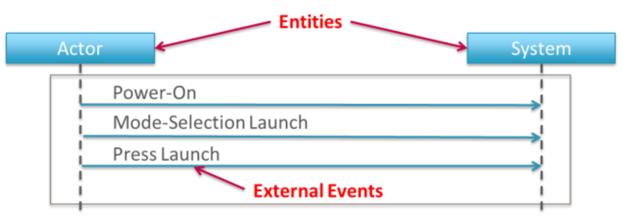


Fig. 19. Simulation Scenario

In FTA simulation, for each use case (in a specific state of the system), Safety Engineers will have to identify the undesired top events, on the basis of the hazard analysis performed with the classical methods.

B. *Simulation of System's Behaviours*

The core of Safety analyses generation is represented by the simulation of System behaviors:

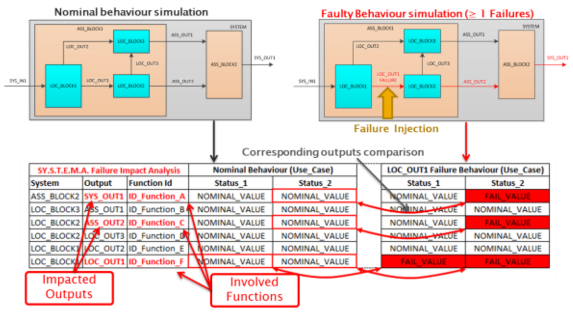
- **Nominal behavior simulation**: values assumed by all the output signals (one value for each system status in a specific Use Case) without failure injection.
- **Failure behavior simulation**: values assumed by all the output signals (one value for each system status in a specific Use Case) upon injection of one or any combination of failures.

Safety analyses generation relies upon the organization in different databases (tables) of the integrated information relevant to the system definition:

- **List of functions**: table reporting the functions breakdown structure (hierarchy and dependency).
- **List of system**: table reporting the system breakdown structure (hierarchy and dependency).
- **List of outputs**: table listing all outputs and providing also the reference to the relevant block within List of System.
- **List of failures**: table listing all the failure modes and providing also the reference to the related failing entity within List of outputs (output failure) or within List of functions/System (function/component failure).

C. *FMEA Algorithm*

The FMEA algorithm consists in the comparison of the nominal behavior with faulty ones (i.e. corresponding output values) in order to identify impacted outputs and involved functions, as shown in the following figure.



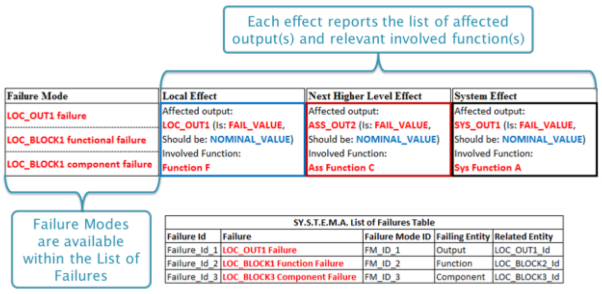
SY.S.T.E.M.A. Failure Impact Analysis			Nominal Behaviour (Use_Case)		LOC_OUT1 Failure Behaviour (Use_Case)	
System	Output	Function Id	Status_1	Status_2	Status_1	Status_2
ASS_BLOCK2	SYS_OUT1	ID_Function_A	NOMINAL_VALUE	NOMINAL_VALUE	NOMINAL_VALUE	FAIL_VALUE
LOC_BLOCK3	ASS_OUT1	ID_Function_B	NOMINAL_VALUE	NOMINAL_VALUE	NOMINAL_VALUE	NOMINAL_VALUE
LOC_BLOCK2	ASS_OUT2	ID_Function_C	NOMINAL_VALUE	NOMINAL_VALUE	NOMINAL_VALUE	FAIL_VALUE
LOC_BLOCK2	LOC_OUT3	ID_Function_D	NOMINAL_VALUE	NOMINAL_VALUE	NOMINAL_VALUE	NOMINAL_VALUE
LOC_BLOCK1	LOC_OUT2	ID_Function_E	NOMINAL_VALUE	NOMINAL_VALUE	NOMINAL_VALUE	NOMINAL_VALUE
LOC_BLOCK1	LOC_OUT1	ID_Function_F	NOMINAL_VALUE	NOMINAL_VALUE	FAIL_VALUE	FAIL_VALUE

Fig. 20. FMEA algorithm

Since SySTEMA compares output values to define failure effects, the goal to generate a FMEA, according to MIL-STD-1629, is obtained by classifying system outputs on 3 different levels:

- **LOCAL LEVEL OUTPUT**: output signal from Local (Component) level block.
- **NEXT HIGHER LEVEL OUTPUT**: output signal from any intermediate level block.
- **SYSTEM LEVEL OUTPUT**: output signal from System block.

According to MIL-STD-1629, FMEA output is provided in tabular format as the table in Fig. 21.


Each effect reports the list of affected output(s) and relevant involved function(s)
Failure Modes are available within the List of Failures

Failure Mode	Local Effect	Next Higher Level Effect	System Effect
LOC_OUT1 failure LOC_BLOCK1 functional failure LOC_BLOCK1 component failure	Affected output: LOC_OUT1 (Is: FAIL_VALUE, Should be: NOMINAL_VALUE) Involved Function: Function F	Affected output: ASS_OUT2 (Is: FAIL_VALUE, Should be: NOMINAL_VALUE) Involved Function: Ass Function C	Affected output: SYS_OUT1 (Is: FAIL_VALUE, Should be: NOMINAL_VALUE) Involved Function: Sys Function A

SY.S.T.E.M.A. List of Failures Table

Failure Id	Failure	Failure Mode ID	Failing Entity	Related Entity
Failure_Id_1	LOC_OUT1 Failure	FM_ID_1	Output	LOC_OUT1_Id
Failure_Id_2	LOC_BLOCK1 Function Failure	FM_ID_2	Function	LOC_BLOCK2_Id
Failure_Id_3	LOC_BLOCK3 Component Failure	FM_ID_3	Component	LOC_BLOCK3_Id

Fig. 21. FMEA: Example of Output from SySTEMA

D. *FTA Algorithm*

The FTA is a "Top-Down Navigation" (in reverse direction) of the OSDs in Non-Nominal behavior, starting from top event (see Fig. 22).

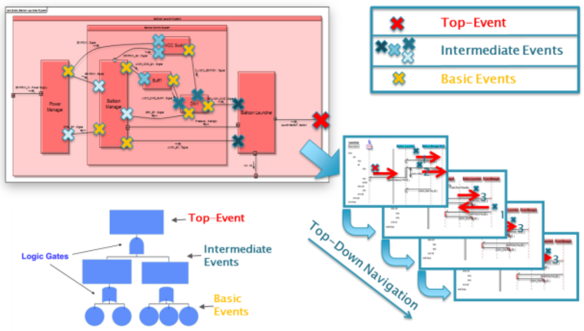
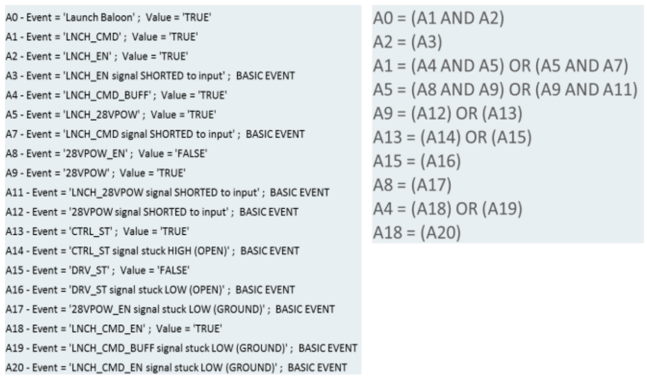


Fig. 22. Top-Down Navigation

SySTEMA will generate an EXCEL file with a table as Fig. 23, containing the list of all the logic conditions which are the basis to build the FTA graphical tree.



```
A0 - Event = 'Launch Baloon' ;  Value = 'TRUE'
A1 - Event = 'LNCH_CMD' ;  Value = 'TRUE'
A2 - Event = 'LNCH_EN' ;  Value = 'TRUE'
A3 - Event = 'LNCH_EN signal SHORTED to input' ;  BASIC EVENT
A4 - Event = 'LNCH_CMD_BUFF' ;  Value = 'TRUE'
A5 - Event = 'LNCH_28VPOW' ;  Value = 'TRUE'
A7 - Event = 'LNCH_CMD signal SHORTED to input' ;  BASIC EVENT
A8 - Event = '28VPOW_EN' ;  Value = 'FALSE'
A9 - Event = '28VPOW' ;  Value = 'TRUE'
A11 - Event = 'LNCH_28VPOW signal SHORTED to input' ;  BASIC EVENT
A12 - Event = '28VPOW signal SHORTED to input' ;  BASIC EVENT
A13 - Event = 'CTRL_ST' ;  Value = 'TRUE'
A14 - Event = 'CTRL_ST signal stuck HIGH (OPEN)' ;  BASIC EVENT
A15 - Event = 'DRV_ST' ;  Value = 'FALSE'
A16 - Event = 'DRV_ST signal stuck LOW (OPEN)' ;  BASIC EVENT
A17 - Event = '28VPOW_EN signal stuck LOW (GROUND)' ;  BASIC EVENT
A18 - Event = 'LNCH_CMD_EN' ;  Value = 'TRUE'
A19 - Event = 'LNCH_CMD_BUFF signal stuck LOW (GROUND)' ;  BASIC EVENT
A20 - Event = 'LNCH_CMD_EN signal stuck LOW (GROUND)' ;  BASIC EVENT
```

```
A0 = (A1 AND A2)
A2 = (A3)
A1 = (A4 AND A5) OR (A5 AND A7)
A5 = (A8 AND A9) OR (A9 AND A11)
A9 = (A12) OR (A13)
A13 = (A14) OR (A15)
A15 = (A16)
A8 = (A17)
A4 = (A18) OR (A19)
A18 = (A20)
```

Fig. 23. FTA: Example of Output from SySTEMA

VII. Conclusions

The SySTEMA approach developed in this work was motivated by the idea of an automated, integrated, operational-oriented and model based safety analysis of architectures modeled in SysML in a single tool. In doing so, this SySTEMA approach aims to enhance the classical safety analysis working on a unique model suitable for both safety and design purpose.

The creation of such a model will allow to:

- Think about safety since the starting phases of functional and architectural design;
- Identify in a timely manner any critical issues related to the impact of failures on the functionality of the system;
- Facilitate and make unique understanding of the logic of the system;
- Perform main safety analyses (FMEA, FTA) in an automated way thus receiving a rapid feedback, resulting in immediate impact on design choices.

From that it follows that an integration have to be applied between the systems engineering domain and the safety domain. For this, SysML was extended to include safety-related information. Since these extensions are realized by stereotypes, applying it to existing SysML system models requires almost no additional modeling effort.

At present, SySTEMA has been tested only on a limited type of systems (i.e. limited types of nominal and faulty behaviors). A reference library has been created starting from the analyzed systems, with the purpose to support systems and safety engineers in the description of system behavior through Sequence Diagrams.

Next steps will be to enrich the reference library with new systems behaviors or failure modes which are not currently taken into account.

References

[1] http://www.omg.org/spec/SysML/1.3/

[2] Bohm, Corrado; Giuseppe Jacopini (May 1966). "Flow Diagrams, Turing Machines and Languages with Only Two Formation Rules". Communications of the ACM. 9 (5): 366–371. doi:10.1145/355592.365646.