POLITECNICO DI TORINO
Repository ISTITUZIONALE

Comparative Analysis of Machine Learning Algorithms for Heat Exchangers Diagnosis in Electrified Aircraft

Original
Comparative Analysis of Machine Learning Algorithms for Heat Exchangers Diagnosis in Electrified Aircraft / Migliore, D.F., D'Alessio, G., Caggese, S., De Martin, A., Acerra, F., Sorli, M., Fioriti, M.. - (2024). (34th Congress of the International Council of the Aeronautical Sciences, ICAS 2024 Firenze (ITA) 9-13 September 2024).

Availability:
This version is available at: 11583/2994930 since: 2024-12-02T14:18:01Z

Publisher:
International Council of the Aeronautical Sciences

Published
DOI:

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

30 October 2026



AUTOMATIC SYSTEM REQUIREMENTS VERIFICATION FOR THE MBSE-ORIENTED AIRCRAFT DESIGN PROCESS

Alberto Dagna[1], Stefano Centomo[2], Eugenio Brusa[1], Cristiana Delprete[1] & Rocco Gentile[3]

[1]Department of Mechanical and Aerospace Engineering, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129, Torino,Italy
[2]Leonardo LABS, Leonardo S.p.A., Corso Castelfidardo 22, 10138, Torino, Italy
[3]Leonardo S.p.A., Corso Francia 426, 10146, Torino, Italy

Abstract

The requirement analysis, within the deployment of the Systems Engineering methodology, is one of the most important steps of the design process of a new product. It is paramount to guarantee robust requirements and to carefully check that each of them is allocated and fulfilled. To reach that goal, the Model-Based Systems Engineering provides some formal methods for the elicitation and trace of requirements, and drives their verification and validation. However, textual requirements can be very complex objects to be integrated in that process, due to their typical abstraction and the lack of that interpretability, which is assured, for instance, into a logic dependent environment. This paper introduces a new approach to robustly write requirements and easily integrate them into the models commonly used to describe the system to be designed. To make clearer the rationale exposed, the case of an aircraft landing gear is exploited as a test. Thus, the physical models of that system are presented, followed by the requirements elicitation and their allocation to the system functional models. Finally, the verification process is discussed. The final goal of this activity is providing the designer with a useful and versatile workflow, enabling the requirements verification since the early steps of the design process, to avoid the risk and cost associated with an unforeseen deviation of product characteristics, in later stages of the design process.

Keywords: Model-Based Systems Engineering, Requirement Analysis, Requirement Management, Verification & Validation, Product Life Cycle Development

1. Introduction

The product lifecycle development is key for the design activity. It includes several steps, being often composed by some tasks, which allow allocating completely the requirements to the product parts assembled, while complying with directives, standards, norms, and making the product suitable to be certified, according to the legislations constraints, for a given market [1] . To speed up such process, one of the most popular methodologies applied in several fields of industry, as in aerospace engineering, is the Model-Based Systems Engineering (MBSE) [2, 3, 4]. It implements the methodology of Systems Engineering (SE), defined by the INCOSE [5] as "an interdisciplinary approach focusing on defining customer needs [...] documenting requirements and then proceeding with design synthesis and system validation". The Model-Based Systems Engineering applies concepts of SE by modelling the system, from both the functional and physical view point. This allows improving the overall product development, the system quality and the interoperation between working teams. The Verification & Validation (V&V) of requirements [6, 7] plays a key role in that process. The verification of requirements allows checking that they are completely fulfilled by the designed system, while the validation demonstrates that the product satisfies the customer needs [8]. Verification deals with targets defined by the manufacturer, inside the company, and according to its policies. Validation focuses on some external inputs, as they are imposed by customer, environment and laws. The V&V activity has a paramount importance in the whole process. Conformity and product liability are

defined as those activities have been performed [9]. However, for a straight application, the V&V must be tightly integrated in the overall design process. This is foreseen by the SE, as is shown, for instance, by the widely used V-diagram, in Fig. 1 . A step-by-step process of verification starts from the basic elements of a system (components) and goes up to the whole product (in some cases, a system of systems [10]). When the V&V is carried out in a document-based fashion, various crit-

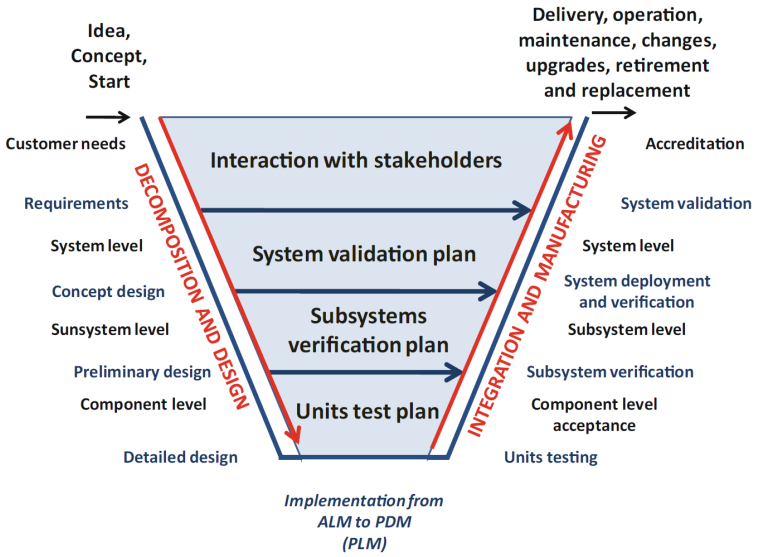


Figure 1 – Example of V-diagram, which clearly shows the V&V process on the right branch as a *bottom-up* approach.

icalities appear, especially in terms of traceability and inter-communication. Therefore, the MBSE drives towards the requirements verification, by overcoming those issues and providing a consistent and robust approach to accomplish that task. The current state-of-the-art of the MBSE is still refining the tools to perform an effective requirements verification, to make this process as automatic as is possible [11]. This paper aims at contributing to this trend, by formalising an approach for the verification and validation process based on modelling of requirements through a machine interpretable language. This effort should allow the automatic and dynamic verification of requirements, along the product development. Particularly, in the test case herein proposed, a set of requirements will be listed. They are based on some existing norms, for the aeronautical system, and are here applied to the landing gear. Those requirements are used to develop a behavioural model, simulating the attitude of an aircraft during landing. Then, that set will be reinterpreted by the FRET open-source tool, developed by NASA [12, 13] , to make the listed requirements understandable by a digital environment. In particular, this tool is intended as an environment for the definition of temporal requirements through an ad-hoc language which is capable of bridging the gap between the easily understandable but semantically loose natural language and the diametrically opposite machine language, as shown in Fig. 2. An example of temporal requirement can be summarized with the following sentence: "when a *component/system* is in a *particular state* it shall satisfy a *specific condition* within a *certain amount of time*". Such transformation allows integrating the requirements in the landing gear physical models, to monitor, along the entire simulation, the behaviour of the system with respect to the requirements. Therefore that integration may enable the automatic verification of said requirements, checking that the designed system is compliant with the set constraints. Such process is further enhanced by the use of the FMI standard [14], developed by the Modelica Association, which enables the co-simulation of models coming from various tools. The goal is to allow total versatility of the requirements verification process, making it compatible with virtually any kind of simulation environment, including varying disciplines.

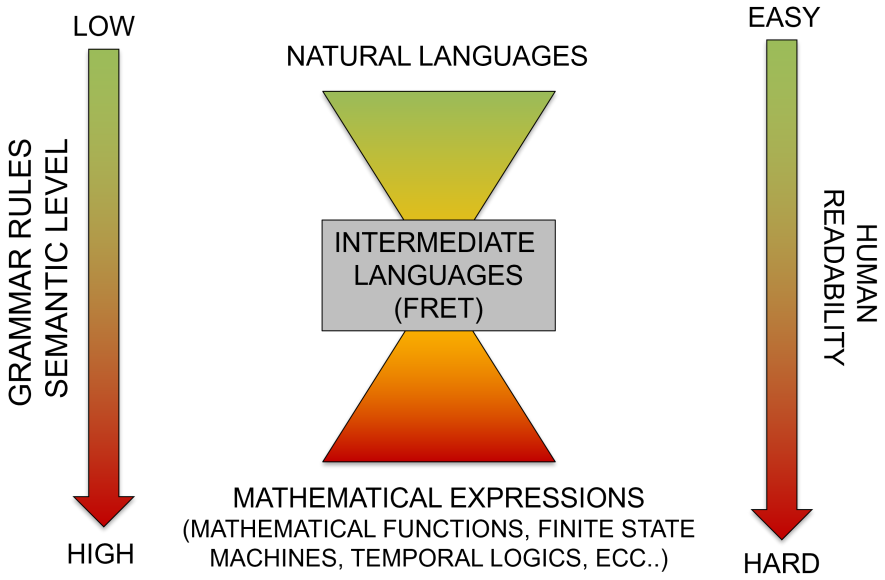


Figure 2 – Schematization of the characteristics of the languages used in a design-oriented environment.

2. Methodology

2.1 Requirements and Landing gear model

The first step of product design consists in defining and formalising requirements, starting from the customer needs. In this paper, just a concise set of requirements has been selected, to represent the complete list of system requirements, to simplify the presentation and even to cope with some non-disclosure constraints about the industrial system analysed. Those requirements have been defined by resorting to the IBM Rational Rhapsody® [15], being widely used in industry to implement the Systems Engineering and modelling systems through the SysML language [16] . Such language provides the user with a series of diagrams and objects that allow meta-modelling the system and its interaction with other systems. Among those, the Requirement Diagram allows to define hierarchically the system requirements, as in Fig. 3, which collects the requirements of the test case. Particularly, it

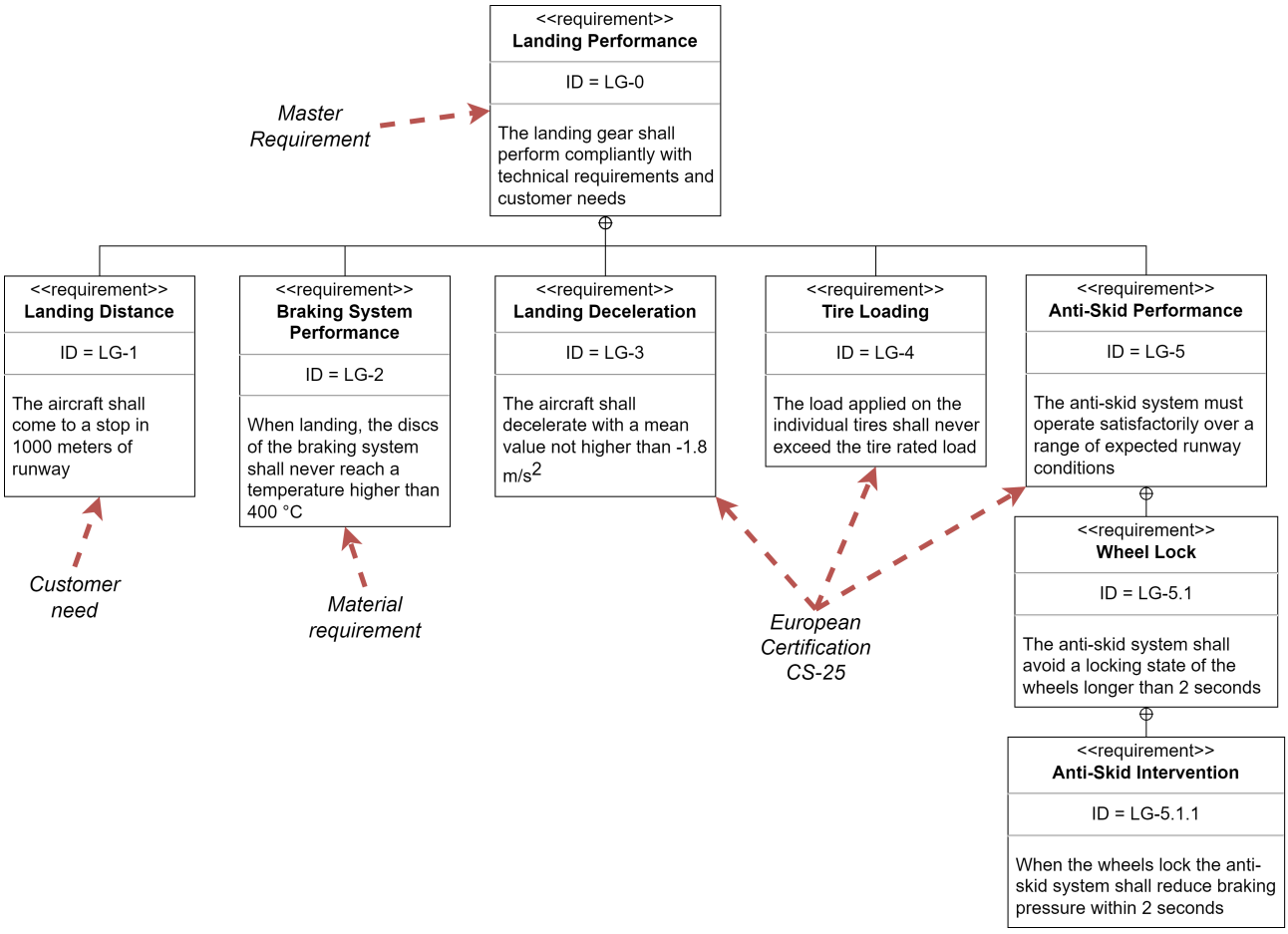


Figure 3 – Requirements diagram defined in SysML comprising of the system requirements used for the verification of the landing gear.

is remarkable that those requirements apply to several subsystems of the landing gear, such as tires, braking discs and anti-skid system. Moreover, it can be appreciated the high performance figures of the landing operation as well as the source of each requirement. The *LG-1 Landing Distance*, for instance, directly comes from a specific customer need, being expressed as the high capability of the aircraft of landing upon some kind of runways within a certain distance. The *LG-3 Landing Deceleration* corresponds to a specific certification requirement derived from the CS-25 European specification for large aircraft [17] .

Once that requirements have been defined, a parametric model of the landing gear subsystems has been set up for a preliminary sizing. It can be referred to as “sizing model”. It is coupled with a dynamic simulation, provided by a numerical model developed in the MATLAB® 2023a + Simulink® environment [18] . This is more realistically the “physical model”. So far, the “sizing model” allows designing wheels, brakes and shock strut of the main landing gear, and then, the parameters obtained

by that preliminary design are used to populate the "physical model" which, in turn, predicts the aircraft behaviour during landing, including the quantities relative to each requirement. In Fig. 4, a schematisation of the workflow implemented in the sizing model of the landing gear is shown, including the data exchanged towards Simulink®. The equations implemented in that model have been derived from a previous work of the authors [19].

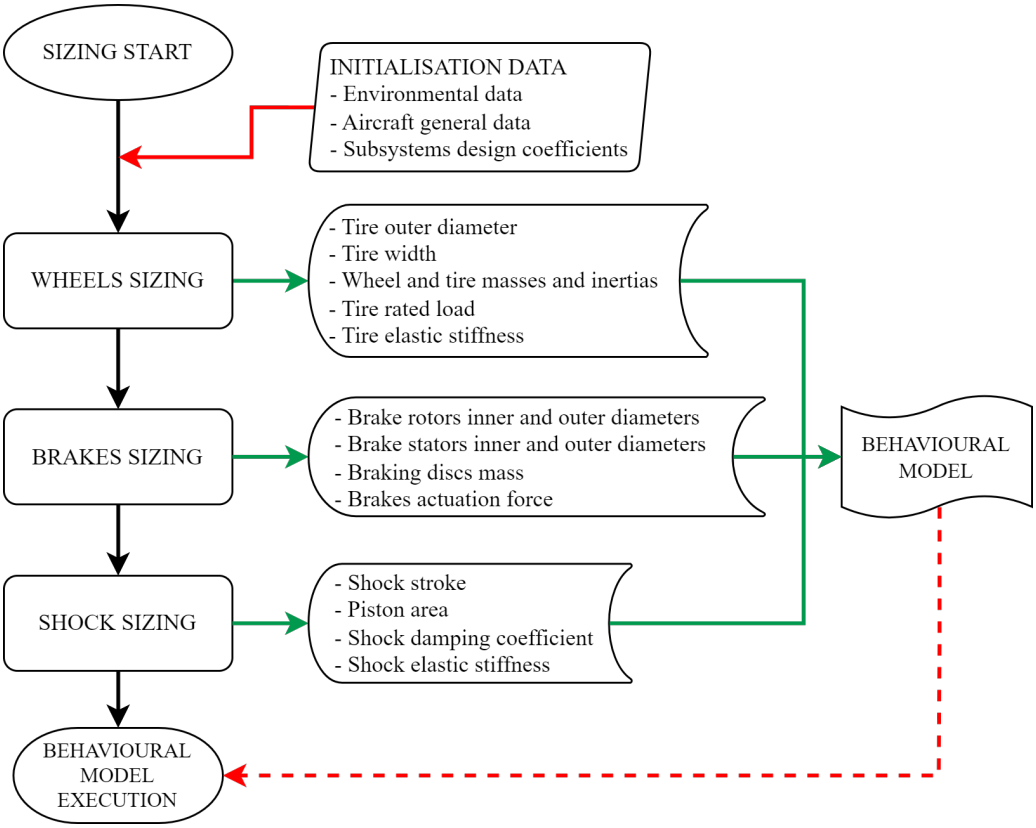


Figure 4 – Landing gear sizing workflow developed in MATLAB®.

Fig. 5 shows the main blocks of the Simulink® behavioural model. It is composed of two blocks, namely *Vertical Dynamics* and *Longitudinal Dynamics*, respectively, that predict the vertical and longitudinal behaviour of the aircraft, during landing. The system is modelled as a 1 DOF spring-damper, with initial vertical and longitudinal speeds set according to the CS-25. The third block includes the

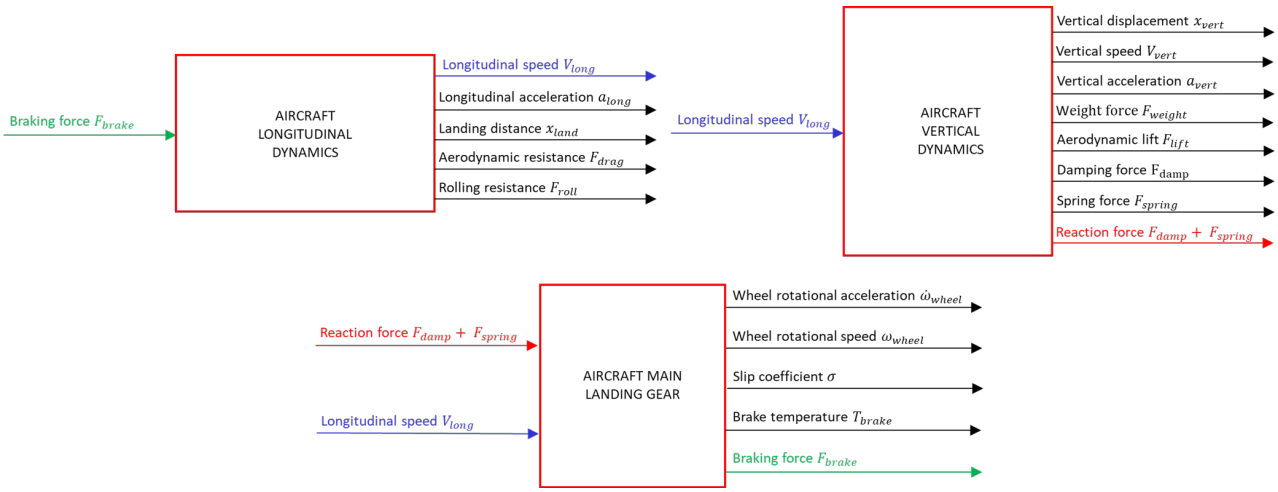


Figure 5 – Schematised landing gear behavioural model developed in Simulink®.

sub-models for the sized subsystems of the landing gear. The decomposition of the landing gear, including the data exchanged between them, is described in Fig. 6. It is noticeable the use of a

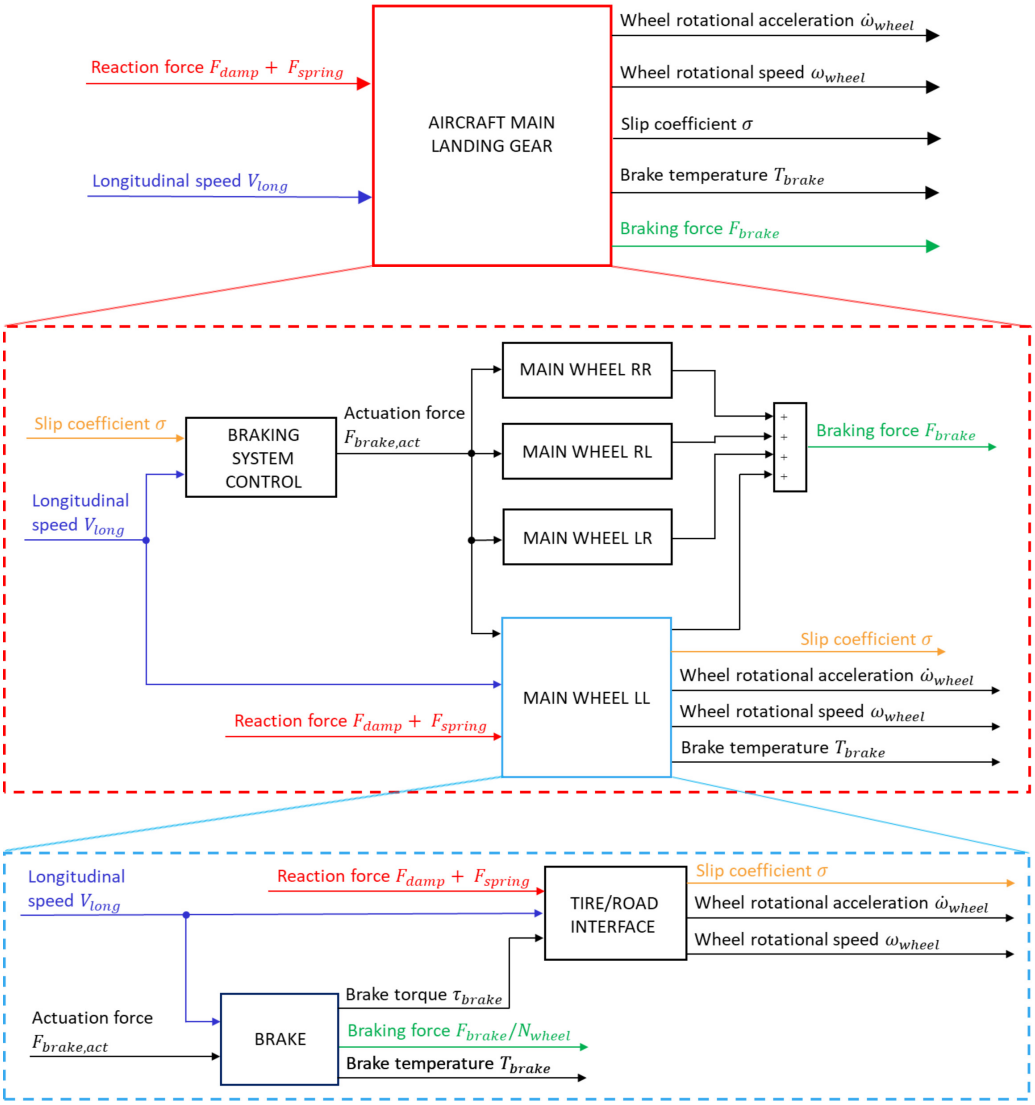


Figure 6 – Schematised decomposition of the landing gear block, comprised of the brake, wheel and braking system control blocks.

modular approach in modelling the landing gear. In fact, the data is exchanged between subsystems, such as the "Main Wheels" and the "Braking System Control", each containing the respective model. By carrying out the modelling process in such a way, it allows to better capture the behaviour of each systems and, most importantly in the context of this paper, to connect the requirements' models directly to the systems, and not just to a variable. Therefore, enabling the verification of the designed systems through the verification of its requirements.

2.2 Requirements modelling and implementation

Those requirements are expressed through a textual description and it does not allow performing the automatic verification of requirements, as the MBSE would require. Therefore, it is mandatory exploiting an intermediate language, which could be sufficiently user-friendly to be easily written, but with clearly defined semantics which allow the requirements to be read and interpreted by a digital system. The tool developed by the National Aeronautics and Space Administration (NASA), called FRET, provides a solution to such problem. That tool is based on a textual language for requirements, which resorts to some clear and understandable grammatical rules, for a logical environment as a computer is. Particularly, it has a dedicated user-interface, which allows re-elaborating the requirements, with six constructs. Each of those constructs identifies a different aspect of the context and constraints of the requirement, defined as follows:

- Scope: it specifies the mode/phase of the system in which the requirement is relevant.
- Conditions: it indicates a condition which must be true for the requirement to be applicable.
- Component: it specifies the system which must respect the requirement.
- Shall: the verbal statement, it imposes that the requirement is something that must be satisfied.
- Timing: it specifies the time-point is which the requirement must be satisfied.
- Responses: the actual condition specified by the requirement that must be satisfied.

Following those rules, the original set of requirements has been modelled as in Fig. 7. In the new

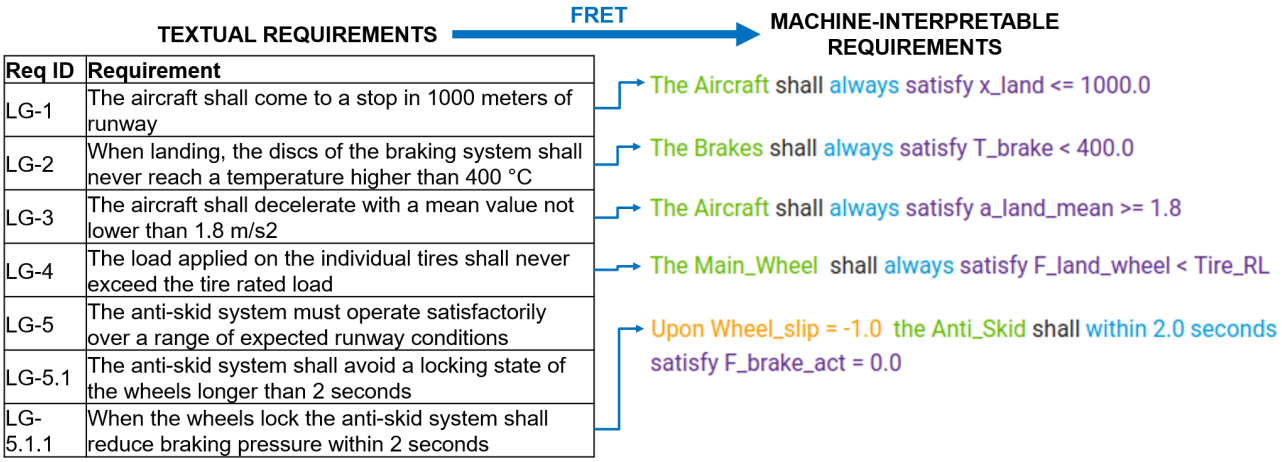


Figure 7 – Modelling and translation of landing gear textual requirement into FRET-like requirements.

set of requirements, a few characteristics can be appreciated. First, each requirement highlighted in green identifies the component or subsystem that must satisfy the condition expressed. For instance, the component called *Brakes* identifies the subsystem in the behavioural model that in numerical simulation provides a result useful to check the fulfilment of the related requirement. The colour magenta highlights in the system response the condition to be monitored during the dynamic simulation, by checking some variables specified in the FRET requirement. In blue, the timing of requirement is included. In case of requirement *LG-5.1.1*, an additional condition is indicated in orange, since the verification of the related requirement must be guaranteed only when the wheels of the landing gear are locked. Once the set of requirements has been translated, the tool is capable to convert them into a machine language, which can be easily exploited by a digital model to actively monitor the requirement metrics and to warn the user about its verification. The language used for that purpose is Lustre, a synchronous data-flow language [20] . In this language each requirement is associated to the physical model of system, embedded in the newly created requirement model, as in the “assume-guarantee contract”, being widely recognised as a formal relation for system verification [21, 22] . At this step, FRET can generate the requirement monitoring action simply as a Simulink

block, which can be imported into the system behavioural model. Such monitors embed the logical relation between the model variables, allowing to check along the entire dynamic simulation whether the requirement is verified or not. However, as helpful as this step can be to the designer in capturing the requirements verification, it also locks it to a single simulation environment (i.e. Simulink), which can be a heavy hindrance, especially in a multi-disciplinary and complex project. Therefore, to improve the versatility of the process, a widely adopted standard for interoperability between different simulation tools has been implemented, called Functional Mock-Up Interface [23], also known as FMI. The FMI standard allows to connect multiple dynamic models coming from different tools (which use different and mutually incompatible solvers) by encapsulating them into black-box-like structures called Functional Mock-Up Units (FMU) which can be imported into any FMI-compliant simulation environment. Being black-boxes, the content of the FMU cannot be directly accessed but the inputs, outputs and specific parameters of the model are exposed. So, each model will be executed based on their logic and solver, independently from the others, and then, through the exposed interface ports, the computed values will be exchanged with the other models, based on the connections made between them. In Fig. 8 an example of co-simulation FMU, which wraps a simplified model for the computation of the vertical kinematics of a 1-DOF spring-damper-mass model of the landing gear, is reported. We can see that in this case the FMU receives the longitudinal speed of the aircraft V_{long}

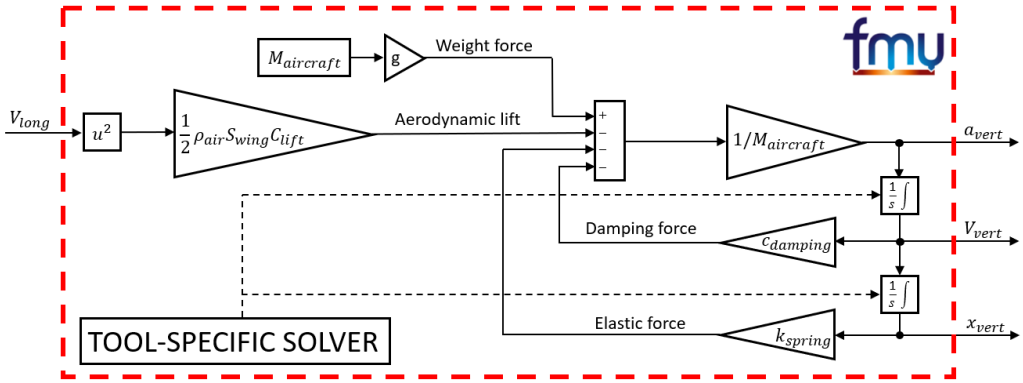


Figure 8 – Example of co-simulation FMU for a 1-DOF mass-spring-damper model.

to compute its aerodynamic lift and sums it with the other vertical force contributions to derive the vertical acceleration a_{vert}. Then, through two integrations performed by the solver, the vertical speed V_{vert} and the vertical displacement x_{vert} are found and sent outside the FMU towards other models. It can also be noticed that this kind of FMU (co-simulation) has the characteristic of including the solver of the original tool, unlike the model-exchange FMU which is not considered in this paper [24]. Therefore, this FMU can be connected to any other model that uses these kinematic quantities, without the issue of tool-compatibility. This exact process has been used to import the requirements into the landing gear model, which in this instance has been modelled in Simulink, but could derive from any other modelling tool for dynamic simulations. In Fig. 9, the schematisation of the requirements FMUs is reported, including what data a subsystem sends to which requirement. For instance the requirement LG-1 monitors the landing distance of the aircraft, thus the block for the simulation of the longitudinal dynamics provides this computed quantity to the requirement.

As a recap, a schematisation of the whole process of requirements importing into the behavioural model is described in Fig. 10. It can be noticed that it starts with the identification of the actual requirements, enabling the modelling of systems and subsystems that allow to effectively guarantee the compliance of the designed models. Then, the requirements are translated into FRET language and into LUSTRE contracts, which can be encapsulated into FMUs useful for the concurrent simulation of the requirements with the physical model. Once the setup is completed, the next step is running the model. It enables the user to extract the results related to behaviour of individual systems and to verify requirements, as the simulation finishes.

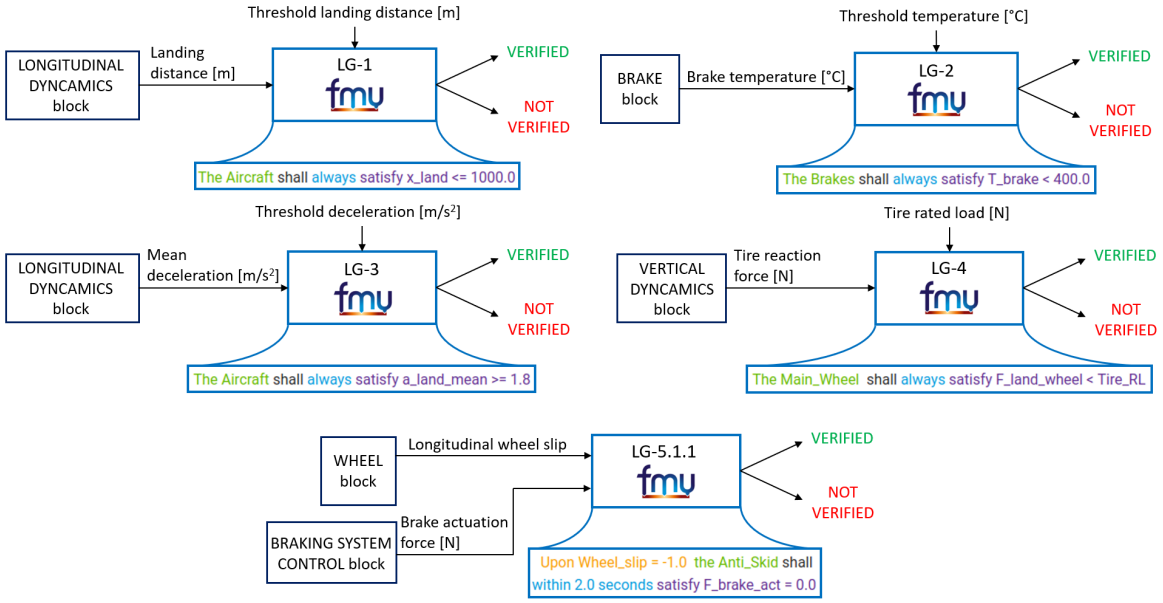


Figure 9 – Depiction of the imported requirements monitors in the behavioural model, including the signals exchanged.

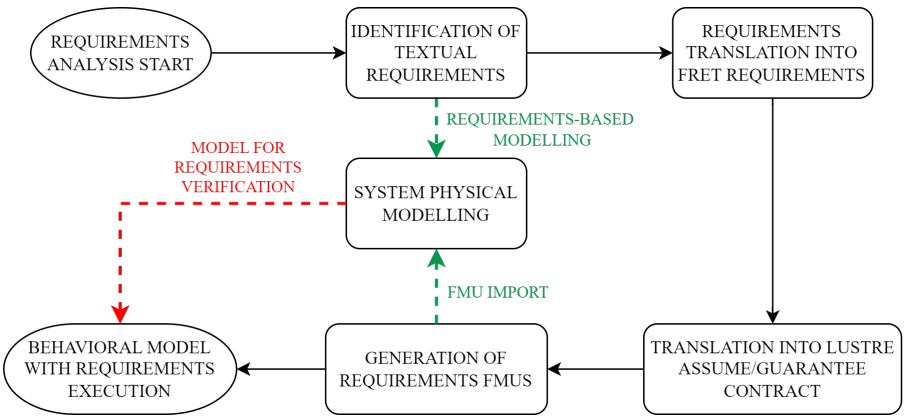


Figure 10 – Workflow for the entire process for the verification of requirements through the behavioural model of the landing gear.

3. Results and Discussion

The execution of the requirements verification, provides some important outcomes. First of all, as in Fig. 11, an immediate feedback to designer is provided, through a simple but effective visual cue, about the verification of the individual requirements. Next, a deeper analysis of results can be

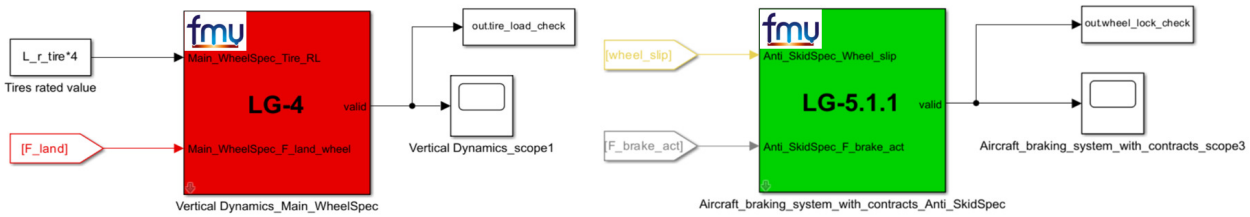
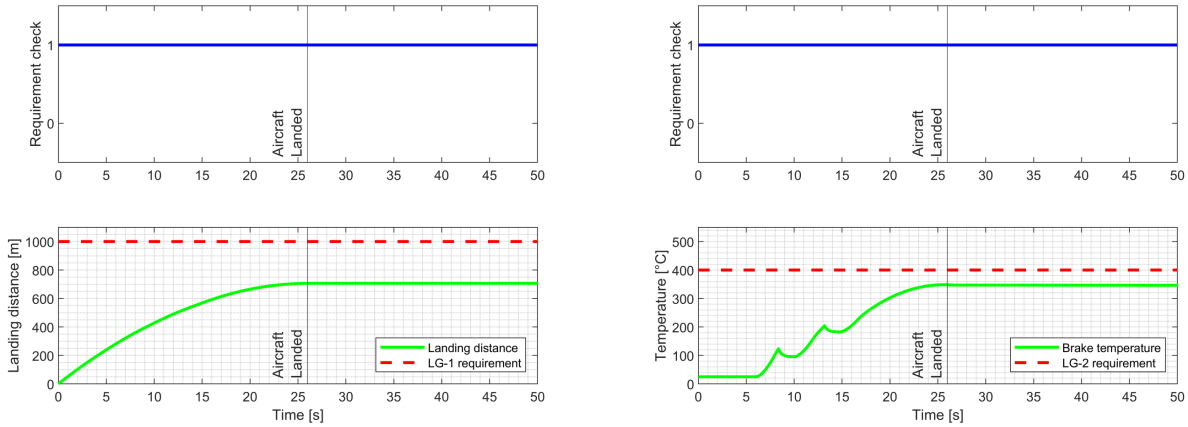


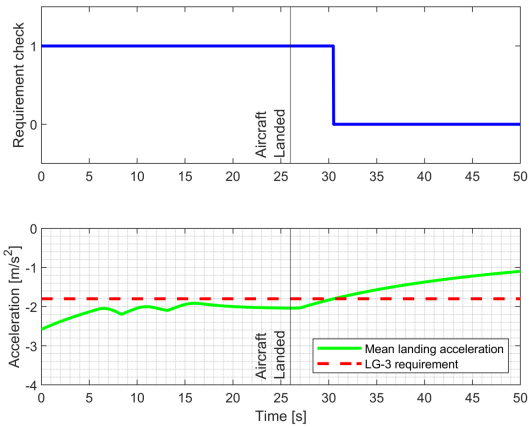
Figure 11 – Examples of visual feedback of a non-verified (left) and a verified (right) requirement.

performed, through the extraction of the output signals. Particularly, for each requirement monitor, the boolean signal *TRUE* $\rightarrow$ *1* or *FALSE* $\rightarrow$ *0* associated to the requirement verification at any time of the numerical simulation can be plotted. Moreover, monitored signals can be visualised, to immediately get either confirmation or deviation of the verified requirement, and in case of deviation, it is possible to identify when the specific condition never fulfilled. In Fig. 12, an example of that implementation is proposed. Plots apply to the first three requirements (*LG-1, LG-2* and *LG-3*). Looking at the



(a) LG-1 Landing distance

(b) LG-2 Brake temperature



(c) LG-3 Mean deceleration

Figure 12 – The resulting plots from the execution of the requirements verification of the first three requirements.

top and bottom plots of each requirement, it can be appreciated that those three requirements are

verified. More specifically, it can be noticed that landing distance is below 1000 meters. Similarly, the temperature reached in braking is under control. The mean deceleration fulfils requirement as well. Nevertheless, it can be detected at about 30 seconds that requirement does not look satisfied. This effect occurs as the aircraft comes to a halt, but it can be noticed that it is located outside the regime covered by the related requirement.
In case of the *LG-4 Tire rated load* requirement (Fig. 13), during the first part of the simulation, corresponding to the aircraft touchdown, the reaction force on tires becomes higher than the imposed limit of the tire rated load (this is a typical manufacturer's term to indicate the nominal maximum load applicable to given tire), as defined by requirement. The monitor sets at *FALSE* value during the first

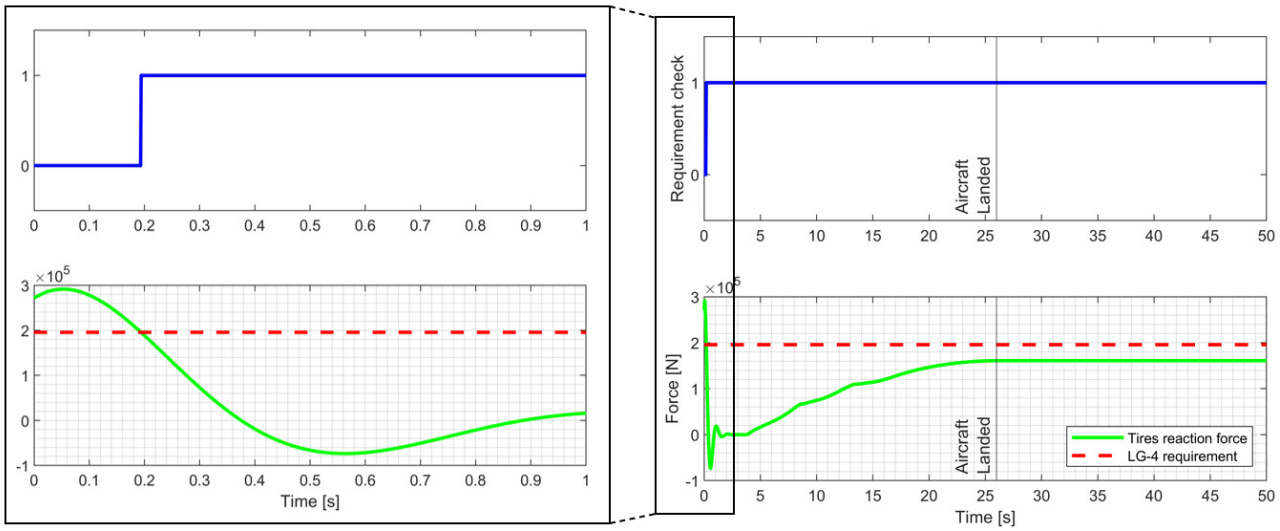


Figure 13 – Plot of the reaction force applied on the tires during landing, relative to the requirement *LG-4*.

0.2 seconds of simulation, and thus it leads to a non verified condition. That result suggests that the tire chosen would not be able to absorb the impact on a runway of the landing aircraft and motivates resorting to a different selection.
Figure 14 describes the last requirement. In this case, it is verified. The time needed by the braking

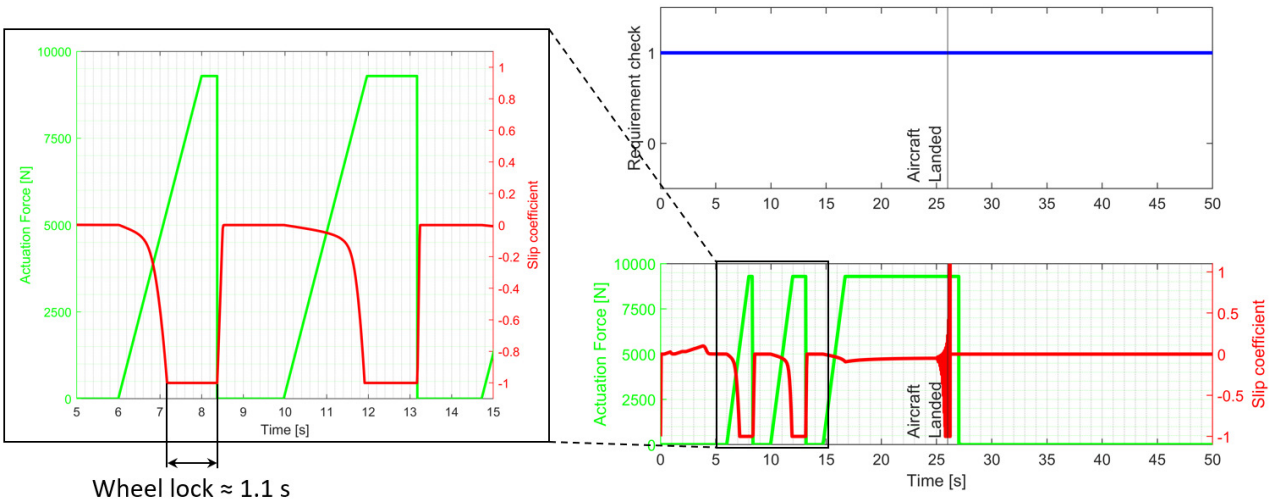


Figure 14 – Plot of the longitudinal wheel slip coefficient and the brake actuation force during landing, relative to the requirement *LG-5.1.1*.

system control to reduce the applied force to brakes from the condition of $Slip\ coefficient = -1$ (which indicates a completely locked wheel) is around 1.1 seconds, which is less than the 2 seconds that the requirements imposes. This requirement is very interesting, because it implies a rather difficult

evaluation of system behaviour. It checks the time used by the anti-skid system to react, as a wheel is locked, and therefore to release the braking action. This prediction could be very time-consuming, in terms of data extraction and analysis, since it requires to go over the entire simulation checking whenever a locking condition of the wheels has occurred and confirming that the reaction time is compliant with the requirement. A process which can be very difficult to perform visually in case of very narrow time constraints. The proposed approach checks automatically and continuously over time whether locking effect arises, and gives a useful feedback to the designer.
An additional example of this monitoring action is reported in Fig. 15. In this case, the requirement

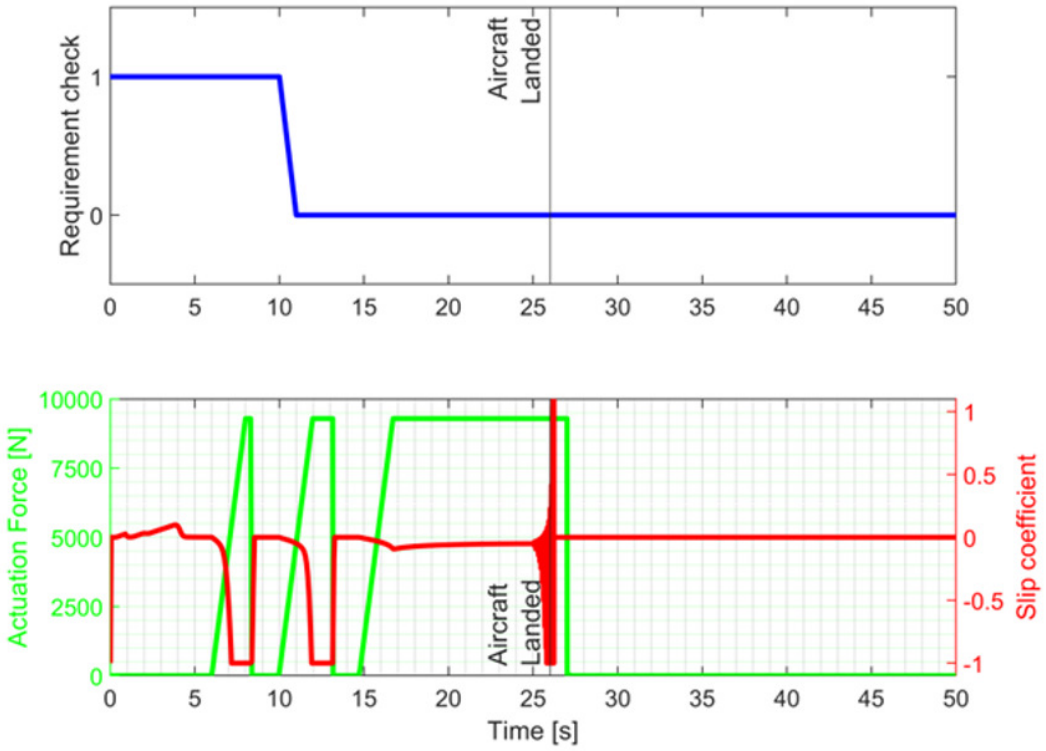


Figure 15 – Plot of the longitudinal wheel slip coefficient and the brake actuation force during landing, with a more stringent *LG-5.1.1* requirement.

LG-5.1.1 has been changed to be more stringent, and impose that the Braking System must stop its braking action within 1 second from when the wheel lock condition begins. It can be confirmed that, since the simulation of the physical system is exactly the same and the brake actuation force is released around 1.1 seconds after the wheel lock, the requirements monitor signals that the requirement is not verified during the entire landing, and precisely indicating where the attention of the designer must be focused.
A final point of discussion that should be highlighted is regarding the simulation performance. In fact, the inclusion of the requirements monitors and the use of the FMUs most definitely increases the complexity of the models. This complexity in turn may affect the time required to perform the simulation. In Fig. 16 a summary of the simulation times of different setups is showcased. In particular, three parameters have been considered: the number of FMUs for the landing gear model, the number of FMUs for the requirements and the communication step between FMUs. Regarding the communication step, it represents the time interval for the exchange of data between FMUs. As it can be seen, its reduction in magnitude causes an exponential increase in simulation time. However, as it can be noticed, the primary factor in such increase is not actually the time to perform the simulation, but in saving the results. In fact, the simulation environment saves the results of the simulation using a print interval equal to the communication step. So the number of data points for a 0.001 seconds step will be much higher than with a 0.01 seconds step, thus increasing the total simulation time. The number of requirements FMUs is relative to the use of a different number of FMUs for

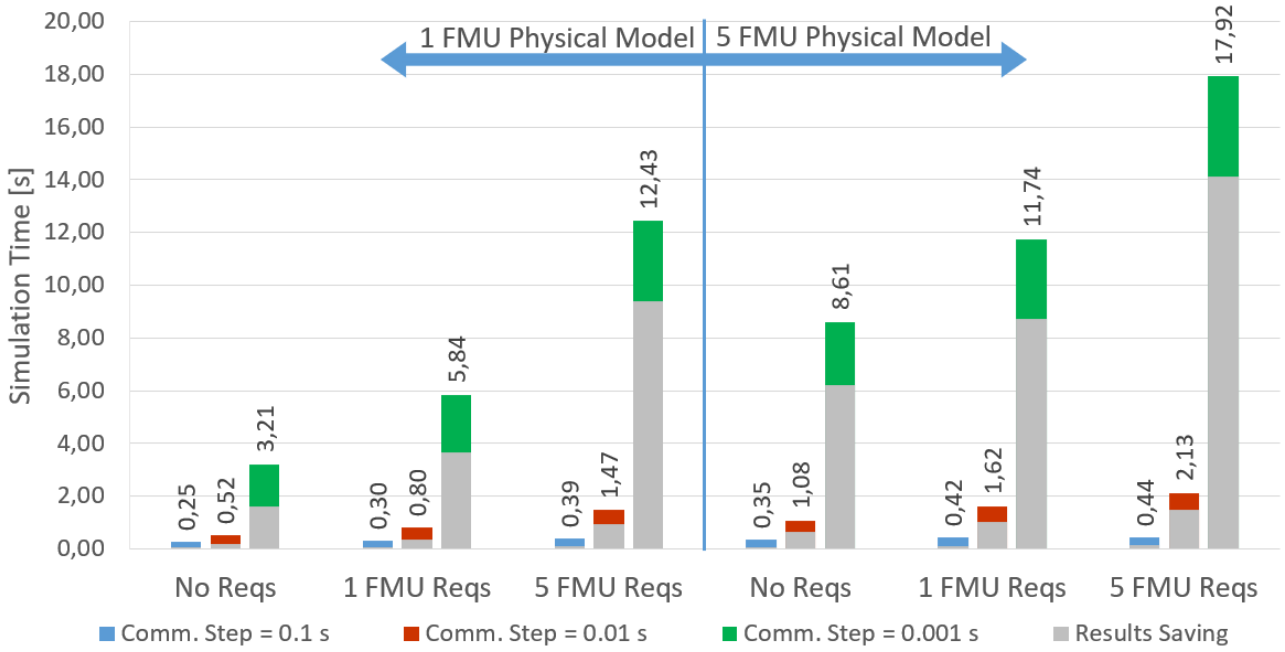


Figure 16 – Plot of simulation performance, comparing the increase in FMUs used and the reduction of communication step.

importing the requirements. More specifically, the model has been tested without requirements, with all requirements packaged in a single FMU and in five separate FMUs. It can be seen how the increasing number of FMUs tends to increase the simulation time, due to the higher numerical complexity faced by the environment in managing more elements. The same consideration can be made for the use of changing number of FMUs for the physical model: on the left side of the plot the results for the entire model enclosed in a single FMU, while on the right 1 FMU for each system is used (i.e. 5, 1 for the aircraft dynamics, 1 for the braking system, 1 for the braking system control, 1 for the wheel assembly, 1 for the shock absorber).

4. Conclusion

Current transition of industry from a document-oriented approach to a model-based systems engineering requires to be supported by several effective tools, while methods and processes are optimised. A key issue of design like the requirements analysis specifically needs some suitable supports to help designer in simplifying the verification and validation process. Making automatic those activities is rather difficult, since they include even textual contents in addition to some numerical data. Integrating qualitative and quantitative analyses along the product lifecycle development and assuring the interoperability of tools is currently a key goal of the research activity. In this context, a novel approach has been developed and herein introduced to facilitate that action. Particularly, system requirements are expressed by resorting to a structure which enables their tighter integration with the physical models than in the past.

Use of a semantics chain, starting from the purely textual requirements and generating then their digital counterpart in a machine-interpretable model is deeply exploited. That strategy allows importing requirements into the system physical models, being widely used in industry to predict the product behaviour in operation. Moreover, that integration increases the robustness of the requirements verification process. This benefit is related to the elimination of an interpretation of requirements performed by the designer when the outputs of the design synthesis are manually checked. Also, the proposed approach allows anticipating in the overall process, at an earlier step of the product development, some issues which are traditionally considered only when the design activity is complete, and they positively affect the trade-off between design solutions.

The proposed approach resorts to some technological resources which are still under development. Therefore, some issues need to be further investigated. Particularly, improving the user-friendliness of the requirements translation would be a priority, allowing the designer to integrate the requirements into the simulation model in a more streamlined manner, possibly reducing the number of steps required. Moreover, the test case is representative for the development of this approach, but the selected set of requirements is still limited to test the real performance of tools. Therefore, a future step consists in testing multiple types of requirements, including those more qualitative than quantitative, to investigate further the overall system performance.

5. Contact Author Email Address

mailto: alberto.dagna@polito.it

6. Copyright Statement

The authors confirm that they, and/or their company or organization, hold copyright on all of the original material included in this paper. The authors also confirm that they have obtained permission, from the copyright holder of any third party material included in this paper, to publish it as part of their paper. The authors confirm that they give permission, or have obtained permission from the copyright holder of this paper, for the publication and distribution of this paper as part of the ICAS proceedings or as individual off-prints from the proceedings.

References

[1] Ioan Dorian Stef, George Draghici, and Anca Draghici. Product design process model in the digital factory context. *Procedia Technology*, 9:451–462, 2013.

[2] John M. Borky and Thomas H. Bradley. *Effective Model-Based Systems Engineering*. Springer International Publishing, 2019.

[3] A. L. Ramos, J. V. Ferreira, and J. Barcelo. Model-based systems engineering: An emerging approach for modern systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(1):101–111, jan 2012.

[4] Azad Madni, Carla Madni, and Scott Lucero. Leveraging digital twin technology in model-based systems engineering. *Systems*, 7(1):7, jan 2019.

[5] David D. Walden, editor. *Systems engineering handbook*. Wiley., Hoboken, NJ, fifth edition. edition, 2023. "INCOSE-TP-2003-002-05.". - Includes index. - Online resource; title from PDF title page (EBSCO, viewed June 13, 2023).

[6] Mourad Debbabi, Fawzi Hassaïne, Yosr Jarraya, Andrei Soeanu, and Luay Alawneh. *Verification and Validation in Systems Engineering: Assessing UML/SysML Design Models*. Springer Berlin Heidelberg, 2010.

[7] Eugenio Brusa, Ambra Calà, and Davide Ferretto. *Systems Engineering and Its Application to Industrial Product Development*. Springer International Publishing, June 2019.

[8] Jhong S. Lee and Leonard E. Miller. *NASA systems engineering handbook*. National Aeronautics and Space Administration, 1995.

[9] A. Terry Bahill and Steven J. Henderson. Requirements development, verification, and validation exhibited in famous failures. *Systems Engineering*, 8(1):1–14, 2005.

[10] J. Boardman and B. Sauser. The meaning of system of systems. In *2006 IEEE/SMC International Conference on System of Systems Engineering*. IEEE, 2006.

[11] Xi Zheng, Christine Julien, Miryung Kim, and Sarfraz Khurshid. Perceptions on the state of the art in verification and validation in cyber-physical systems. *IEEE Systems Journal*, 11(4):2614–2627, dec 2017.

[12] Dimitra Giannakopoulou, Anastasia Mavridou, Julian Rhein, Thomas Pressburger Johann Schumann, and Nija Shi. Formal requirements elicitation with fret. In *REFSQ Workshops*, 2020.

[13] Andreas Katis. Capture, analyze, diagnose: Realizability checking of requirements in fret. In Dimitra Giannakopoulou, contributed to this work prior to joining AWS. This, is a U.S. government work, not under copyright protection in the U.S.; foreign copyright protection may apply, S. Shoham, and Y. Vizel, editors, *CAV 2022*, volume 13372 of *LNCS*, page 490–504. Springer, 2022.

[14] Christian Wolf, Miriam Schleipen, and Georg Frey. Secure exchange of black-box simulation models using fmi in the industrial context. In *Proceedings of the 15th International Modelica Conference 2023, Aachen, October 9-11*, Modelica 2023. Linköping University Electronic Press, December 2023.

[15] IBM. *IBM Rational Rhapsody 9.0.1 User Guide*. IBM, Armonk, New York, USA, 9.0.1 edition, 2022.

[16] Lenny Delligatti. *SysML distilled*. Addison-Wesley, Upper Saddle River, NJ, 2014.

[17] EASA. CS-25 large aeroplanes certification, October 2003.

[18] The MathWorks Inc. Matlab version: 9.14.0 (r2023a), 2023.

[19] Cristiana Delprete, Alberto Dagna, and Eugenio Brusa. Model-based design of aircraft landing gear system. *Applied Sciences*, 13(20):11465, October 2023.

[20] P. Caspi, D. Pilaud, N. Halbwachs, and J. A. Plaice. Lustre: a declarative language for real-time programming. In *Proceedings of the 14th ACM SIGACT-SIGPLAN symposium on Principles of programming languages - POPL '87*, POPL '87. ACM Press, 1987.

[21] Antonio Iannopollo, Pierluigi Nuzzo, Stavros Tripakis, and Alberto Sangiovanni-Vincentelli. Library-based scalable refinement checking for contract-based design. In *Design, Automation and Test in Europe, DATE 2014*, DATE14. IEEE Conference Publications, 2014.

[22] Sophie Quinton and Susanne Graf. Contract-based verification of hierarchical systems of components. In *2008 Sixth IEEE International Conference on Software Engineering and Formal Methods*. IEEE, 2008.

[23] T. Blochwitz, M. Otter, M. Arnold, C. Bausch, C. Clauss, H. Elmqvist, A. Junghanns, J. Mauss, M. Monteiro, T. Neidhold, D. Neumerkel, H. Olsson, J.-V. Peetz, and S. Wolf. The functional mockup interface for tool independent exchange of simulation models. In *Proceedings from the 8th International Modelica Conference, Technical Univeristy, Dresden, Germany*, Modelica. Linköping University Electronic Press, June 2011.

[24] Torsten Blockwitz, Martin Otter, Johan Akesson, Martin Arnold, Christoph Clauss, Hilding Elmqvist, Markus Friedrich, Andreas Junghanns, Jakob Mauss, Dietmar Neumerkel, Hans Olsson, and Antoine Viel. Functional mockup interface 2.0: The standard for tool independent exchange of simulation models. In *Proceedings of the 9th International MODELICA Conference, September 3-5, 2012, Munich, Germany*, MODELLICA. Linköping University Electronic Press, November 2012.