On Reference Systems Architectures

Daniel Krob, Loic Le Sauce, Adrien Roques

On Reference Systems Architectures

Daniel KROB[1], Loic LE SAUCE[2], Adrien ROQUES[3]

August 26, 2023

Keywords

Abstraction, Family of systems, Reference systems architecture, System, Systems architecture

Abstract

A reference systems architecture is a key asset, that intends both to simplify the work of the systems engineer and to guarantee the exhaustiveness of the coverage of the system analyses that one has to manage. This paper presents two formal definitions of a reference systems architecture, based on the theory of abstract interpretation that formalizes the notion of abstraction, which work respectively for large and small-scale families of systems. These definitions are illustrated by automotive, building and mining reference systems architecture cases studies.

Acknowledgements

We would like to thank the many CESAMES' collaborators who contributed to the various industrial case studies that supported our work: Sevag AINEJIAN, Cécile BEYSSAC, Augustin CURLIER, Yann DECRE, Jérôme GUI and Antoine MOUTTAPA.

[1] INCOSE Fellow – Center of Excellence on Systems Architecture, Management, Economy & Strategy (CESAMES) –France – daniel.krob@cesames.net

[2] Center of Excellence on Systems Architecture, Management, Economy & Strategy (CESAMES) – France – loic.lesauce@cesames.net

[3] Center of Excellence on Systems Architecture, Management, Economy & Strategy (CESAMES) – France – adrien.roques@cesames.net

1 – Introduction

Model-based systems engineering (MBSE) fundamentally deals with system models, positioned at the core of the systems engineering & architecting process (cf. [1], [7], [8], [11], [14], [15], [16]). Succeeding in better defining the precise contours of such system models and the generic methods that make it possible to build these models, is thus clearly both a huge scientific challenge – because the theoretical bases of system modeling still remain very largely to be developed – and a huge economic challenge since eliminating all factors of degradation of quality and performance of complex industrial systems is linked to the capacity to build efficiently exhaustive and relevant models of such systems.

In order to improve the development of these system models in a context where system complexity – in particular functional – is only increasing, having effective systems engineering tools becomes just a necessity. This leads to the question of capitalizing on predefined reference systems architectures to be able to rapidly deploy repeatable systems architecting and engineering approaches in the industry, these reference systems architectures being a brick natural basis in this matter.

A *reference systems architecture* is especially a valuable aid to the systems engineer in the initial phase of preparation of a system design work: it indeed allows to quickly identify the relevant use cases, as well as the functions and components that are to be specified, but also, above all, the set of functions and components at the interface of the perimeter of interest, which has now become a highly non-trivial problem due to the complexity of modern complex architectures. The purpose of a reference systems architecture is therefore both to simplify the work of the systems engineer and to guarantee the exhaustiveness of the coverage of the system analyses that one has to manage. However, note that a reference systems architecture is not a tool to support detailed design: such an architecture remains a macroscopic object which specifies at a quite high level of abstraction the whole of a given system from operational, functional and structural perspectives.

We should also point out that a reference systems architecture is key, due to its generic nature, to structure the business architecture of the information system that supports the development process of a complex industrial system, i.e. the PLM (Product Lifecycle Development) information systems, which are used by manufacturers to collaboratively store and exchange technical data during a system development process (see for example [12] or [13] for good overviews of this topic).

Reference systems architectures are in fact the equivalent of a well-established practice in software engineering, namely "design patterns" (cf. [2] to discover the corresponding literature), which allow to optimize the time of design and construction of a software architecture. Unfortunately, there are only very few similar references in the systems engineering literature, both in terms of systems architecture construction patterns and generic systems architectures: the references that we have been able to identify relate more to reusable system modeling principles than on systems architecture principles or reusable systems architectures (see [3], [4], [9], [19]). To be completely exhaustive, let us point out that an exhaustive search on the keyword “reference architecture” has made it possible to bring out a limited number of fields – embedded systems, medical equipment and satellite constellations – where such approaches have been implemented empirically (cf. [5], [10], [17]).

The challenge of this paper is thus to propose a general framework for reference systems architectures, validated by practice. To this aim, we followed a methodology consisting in abstracting a number of practical case studies that had a good level of genericity, based on the systems architecting framework that we developed along the last decade (see [11]), in order to obtain a robust and effective definition for a reference systems architecture.

2 – Preliminaries

2.1 – CESAM Systems Architecture Framework

As a matter of fact, each integrated system S can always be analyzed from three complementary and distinct perspectives that give rise to three complementary generic *systems architectural visions*, that is to say the operational, functional and constructional visions, each of them grouping different types of systemic models, as defined below (see [11] for more details):

- *Architectural vision 1 – Operational vision:* the operational vision of S groups the models of the environment of S – and not of S itself – which are involving S. Such operational models are thus describing the interactions of S with its environment.
- *Architectural vision 2 – Functional vision:* the functional vision of S groups all system-level models describing the input/output dynamics of S, without making reference to its concrete components. Such functional models are thus abstractly modeling the behaviors of S.
- *Architectural vision 3 – Constructional vision:* the constructional vision of S groups the system-level models of S, constructed by composition of its lower-level models associated with its components. Such constructional models are thus describing the structure of S.

This framework can be easily summarized in the so-called CESAM systems architecture pyramid, as presented in Figure 1. Note also that the operational and functional visions in this approach refer to the same classical systems engineering concepts (see for instance [8]), when the constructional vision is in fact an abstraction of the classical logical and physical visions (see again [8]).

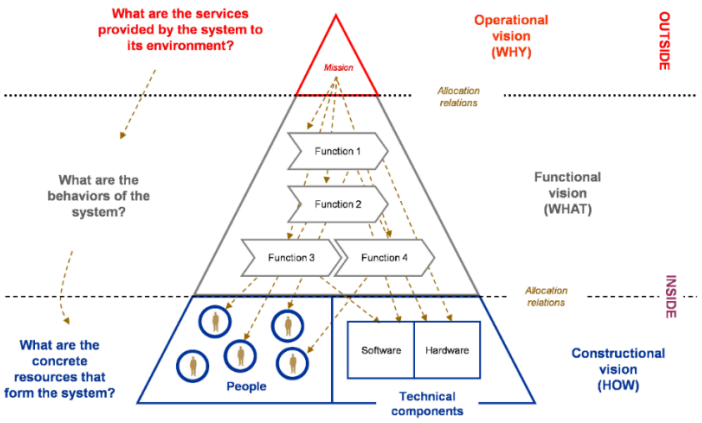

Figure 1 – The CESAM systems architecture pyramid

2.2 – Abstraction

Abstraction is a key mechanism in systems engineering & architecting. A system modeling process can for instance be seen as a permanent back-and-forth between, on a first side, an abstraction activity that constructs a formal model of a real-world system– using the methods & tools of systemic analysis – and on a second side, an experimentation activity where the structure or behavior of the real system, as given or predicted by the model, are checked against those really observed in reality. In the same way, integration of systems can be interpreted in terms of abstraction as described in [8].

It happens that the mathematical theory of abstract interpretation gives a precise definition of the concept of abstraction, introduced by Cousot & Cousot in a software engineering context (see [6]) and presented here below, that also perfectly works in the systems engineering domain.

Definition 1 – Abstraction. Let C be an ordered set, called a concrete set, and let A be another ordered set, called an abstract set. A pair (α, χ) of functions, respectively between C to A and A to C, is then a pair of *abstraction / concretization* functions if and only if the following properties are respected:

$$\forall\, c \in C,\ c < \chi(\alpha(c)) \quad \text{and} \quad \forall\, a \in A,\ \alpha(\chi(a)) < a \quad (1).$$

In other words, properties (1) do mean that if one abstracts a concrete element and then concretize the obtained abstraction, it shall contain – with respect to the concrete order on C – the initial concrete element, and if one concretizes an abstract element and then abstract the obtained concretization, it must be contained – in the sense of the abstract order on A – in the initial abstraction.

This formal notion of abstraction is for instance permanently – usually without knowing it – in systems architecting when constructing systemic hierarchies. An operational, functional or product breakdown structure is indeed a tree of use cases, functions or components where the arrows can be interpreted as abstractions in the previous meaning. In order to illustrate this last fact, let us consider the simple example of a functional breakdown structure where a function F abstracts the functions F1 and F2 as illustrated in the left hand-side of Figure 2. To connect with definition 1, one shall just introduce an abstract set A consisting of the part of the set { F } and a concrete set C consisting of the parts of the set { F1, F2 }, both naturally ordered by inclusion. The abstraction function α consists then to map the empty set in C onto the empty set in A and all other elements of C on { F }, when the concretization function χ consists in mapping the empty set in A onto the empty set in C and { F } onto { F1, F2 }. One can indeed easily check that the two characterizing properties (1) are then fulfilled.

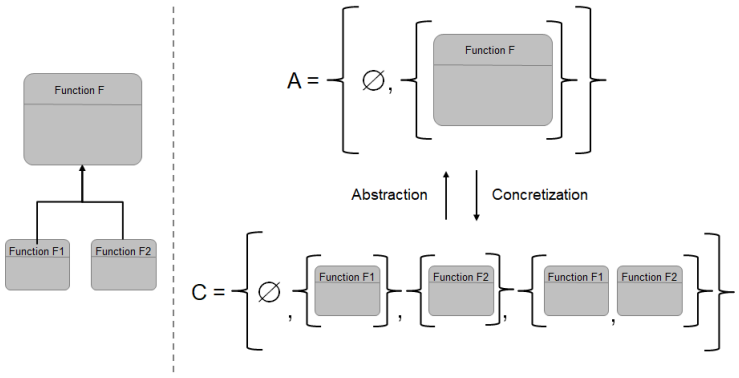

Figure 2 – Example of systems architecture abstraction & concretization

Using this simple principle, it is now an easy exercise to give a formal meaning to the notion of systems architecture abstraction that we will use widely through this paper.

3 – General Definition of a Reference Systems Architecture

3.1 – What May Be a Reference Systems Architecture?

Based on the CESAM systems architecting framework and the notion of abstraction as introduced in section 2, a reference systems architecture, or in an equivalent manner a generic systems architecture, these two terms being perfectly equivalent, can now be defined as follows.

Definition 2 – Reference systems architecture (general definition). Let F be a family of systems that abstracts a set R of real systems. A *reference systems architecture* associated with F – or equivalently with R – is then provided by the following items:

- a set A of systems architecture views that cover the operational, functional and constructional visions of the family F in full generality,
- an instantiation mechanism that allows to derive from A the concrete operational, functional or constructional architectures of any system of R.

Note that this new definition deals with abstraction in the meaning of Definition 1: the underlying abstraction function is here just the function that maps the concrete operational, functional and constructional architectures of a system of R onto A, when the dual concretization function is nothing else than the instantiation mechanism of Definition 2. The generic structure of such a reference systems architecture is illustrated in Figure 3 on an automotive example. We can see that a reference systems architecture is made of generic operational, functional and constructional views which must abstract an instantiation perimeter which aims to be as wide as possible: in our automotive case, on the one hand, many types of vehicles (low cost, thermal, family, electric, hybrid, autonomous, etc.) and, on the other hand, many car manufacturers, knowing that their vehicles are not based on the same architectural choices and assumptions at all systems architectural levels.

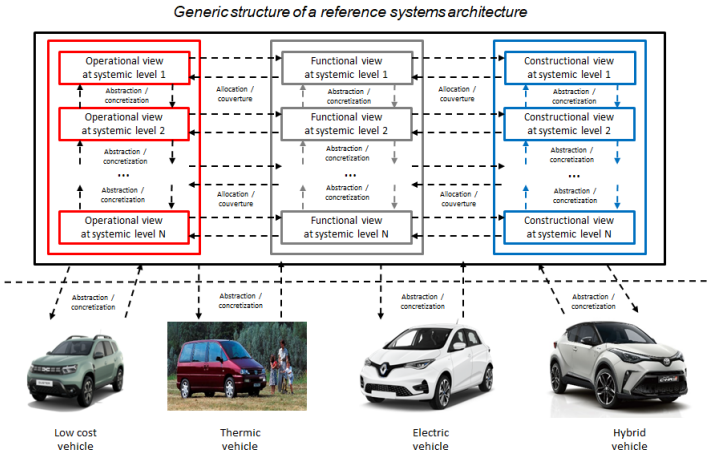

Figure 3 – Definition of a reference systems architecture illustrated in an automotive context

To be more specific, we provide in Figure 4 an example of a generic functional systems architecture, again for an automotive context, in the line of the seminal work of Ubrich et alli, who proposed in [17] a reference functional systems architecture for the software kernel of an automated vehicle (that corresponds to the "manage strategic decisions" function in the functional architecture which is presented here). This reference functional architecture, presented more in details in [8], is organized in the following functional layers, each of them grouping structurally independent functions:

- **Layer 0 – Protect during crash:** this functional layer is reduced to a single function that protects passengers during crash, which is a safety function which reflects at functional level the passive safety mechanisms that are implemented inside a vehicle.
- **Layer 1 – Manage energies:** this functional layer groups all the functions that manage the vehicle energies, i.e. fuel, electricity, vacuum, torque and movement. One can decompose it into two main functions, which respectively manage energy production (e.g. provide torque) and transformation of energy in movement (e.g. provide lateral movement).
- **Layer 2 – Control extended vehicle:** this functional layer groups all the functions that control the vehicle motion and trajectory. One can decompose it into three main functions which respectively manage the sensing of the environment (e.g. measure vehicle speed), take the

strategic decisions (e.g. decide braking) and act on the environment (e.g. brake). Note that these functions are key for autonomous driving and connected vehicle management.

- **Layer 3 – Manage user interfaces**: this functional layer groups all the functions that manage the different user interfaces that are offering value-added services to vehicle users. One can decompose it into three main functions that cover the management of respectively the mechanical user interfaces (e.g. open / close the vehicle), the visual user interfaces (e.g. provide vehicle status) and the comfort user interfaces (e.g. provide heat).

Note that the construction of this example was not easy. We indeed needed to find and guarantee the correct functional decouplings (segregation of functions by coherent functional perimeters, taking into account criticality, scalability, etc.), to identify and verify the consistency of the structuring functional chains, to guarantee traceability with the operational and constructional views of the same level, etc. while ensuring at the same time the satisfaction of the key objective of a reference view, namely its ability to abstract a very large scope of application within a given industrial domain.

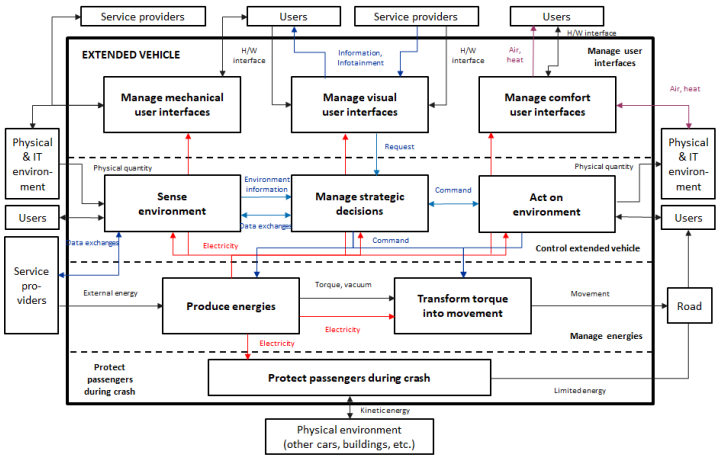

Figure 4 – Example of a reference functional systems architecture of level 1 in an automotive context

As shown by Figure 3, a reference systems architecture appears as a complex multi-dimensional object. Each of the architectural views that it contains indeed involves many elementary objects, ranging from ten for high-level views to several hundreds, or even thousands, for low-level views. Moreover, one has to connect these views between them, both within a given architectural vision (vertical axis in Figure 3), but also between architectural visions (horizontal axis in Figure 3), using abstraction / concretization relations in the meaning of Definition 1, which generates interconnection matrices that can contain tens of thousands, or even many millions for low-level architecture, of relationships that one needs to manipulate when designing a systems architecture reference in a given field.

But this is only one aspect of the problem: the reference systems architecture must also be successfully instantiated to all relevant systems within a given application context, typically low-cost, thermal, hybrid and electrical vehicles from a large number of car manufacturers in our automotive example. This ability to abstract many concrete architectures – which is the reversed facet of the problem that we just mentioned – is moreover the major criterion of success that we used to validate the relevance of a given reference systems architecture in practice.

As one can see designing a reference systems architecture is never an easy task due to the intrinsic combinatorial complexity of such an architecture and the many internal consistency issues that must be continuously overcome during its design, on which is also grafted a pragmatic problem of validation,

the reference architecture making sense only by its ability to be instantiated, which must of course also be verified, which requires confronting with a large number of concrete architectures.

Finally, it should be noted that the problem of building a reference systems architecture includes a fundamental unknown: it is indeed not possible to know a priori what is the level of systemic depth – denoted N in Figure 3 – at which one shall descend in the systemic hierarchy of the considered family of systems. One of the problems that arises when building a reference systems architecture is thus to identify the value of N, which has moreover a real scientific value since it measures the maximum level of abstraction which is covered by a systemic approach in a given application context.

Note that Definition 2 works for most of the application cases on which we could check it, such as aircrafts, automotive electrical & electronic systems, railways or smartphones (see for instance [11] for the aircraft, railway and smartphone cases; a part of a smartphone reference architecture is also provided here in Figure 5). As a matter of fact, it appeared that the maximal level of abstraction on which we could maintain a reference systems architecture for all these examples was 3, which seems to be the "magic" number in this matter, at least for medium or mass production systems, which is the common characteristic of all these examples (we will indeed see in the forthcoming section 4 that this result does not anymore work for other types of systems with more diversity).

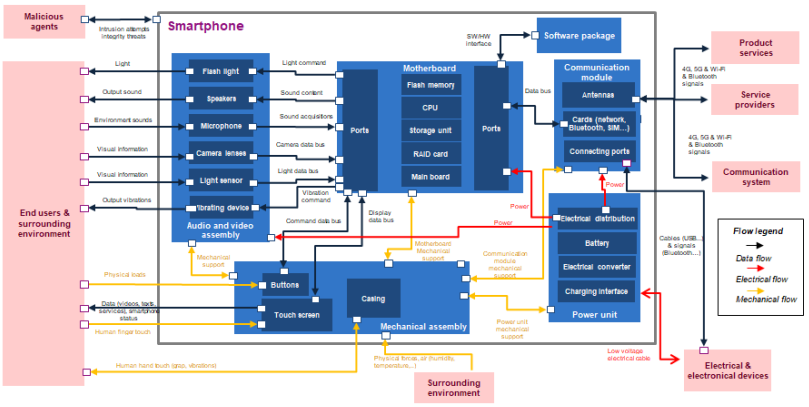

Figure 5 – Reference constructional systems architecture of level 2 for a smartphone

3.2 – Case Study: Automotive Reference Systems Architecture

In order to illustrate both what is a reference systems architecture and how to construct it, we will dig in the automotive case study that we initiated here above, by providing a complete reference systems architecture for a generic automobile vehicle family of systems. The chosen systemic perimeter is the extended vehicle, in the meaning of the ISO 20077 standard, in order to be as covering as possible. It will be recalled in this regard that this perimeter covers both the vehicle itself and all the off-board information systems with which the vehicle communicates, in particular to ensure the evolutionary and corrective maintenance of the on-board systems that a vehicle contains.

Our work is based on freely accessible public data in the literature that we first had to collect and then analyze. The data collection phase was complicated because the public information is dispersed among a large number of sources, which are generally not consolidated. Moreover, a first observation quickly appeared: many consolidated public information can serve as entry points for abstracting the use cases and the components of a vehicle, which allows to get the reference operational and constructional systems architectures, but this is not the case for functions. The only information that we had in this last matter was the high-level functional architecture of an extended vehicle, presented in Figure 4. We therefore had to deal separately with functional architecture.

For operational and constructional architecture, i.e. for use cases and components, we had to adopt different methods, depending on the nature of the captured information, as described below:

- For use cases, our starting point was the thesis of S.M. Zoepf (MIT), who already proceeded to a first consolidation of information at this level [20] and also provides many bibliographical references on this subject that we analyzed. As this work was already a bit old, we also studied the commercial documentation of several car manufacturers (i.e. Mercedes, Nissan, PSA, Renault, Tesla) to identify the commercial use cases that they are proposing to their customers, knowing that the use cases are precisely the face visible to the customer of a vehicle. By consolidating all of this information, we were able to arrive at a first "raw" list of automotive use cases that we could consider reasonably exhaustive.
- For components, the starting point was the more or less well-organized lists of components that can be found in the literature (see for example [18]). We supplemented these lists with lists from the car maintenance websites of the major car manufacturers that we mentioned above, which allowed us to obtain a first "raw" list of car components that we could again consider reasonably exhaustive.

This first phase of capture thus made it possible to obtain "raw" lists of automotive use cases and components which served as a starting point for our abstraction work. But it also broke the initial naïve plan that we imagined, due to the lack of functional information that we discovered. We will see later how we managed to overcome this initially unforeseen, but real, difficulty.

In a second step, we then carried out a meticulous work of clustering and abstraction of these data to bring out their genericity. This work was first done for the operational and constructional visions. At this stage, what became apparent, was that the reference architecture did not have to go lower than three levels of abstraction because the diversity of implementation solutions emerged at the fourth level of abstraction (e.g. we can consider that a crash protection system has its place in a generic architecture, but as soon as we want to break it down more finely, we come across objects which depend on the implementations and so no longer generic). The resulting reference operational and constructional systems architecture – obtained up to level 3 – are provided in Figure 6 and Figure 7.

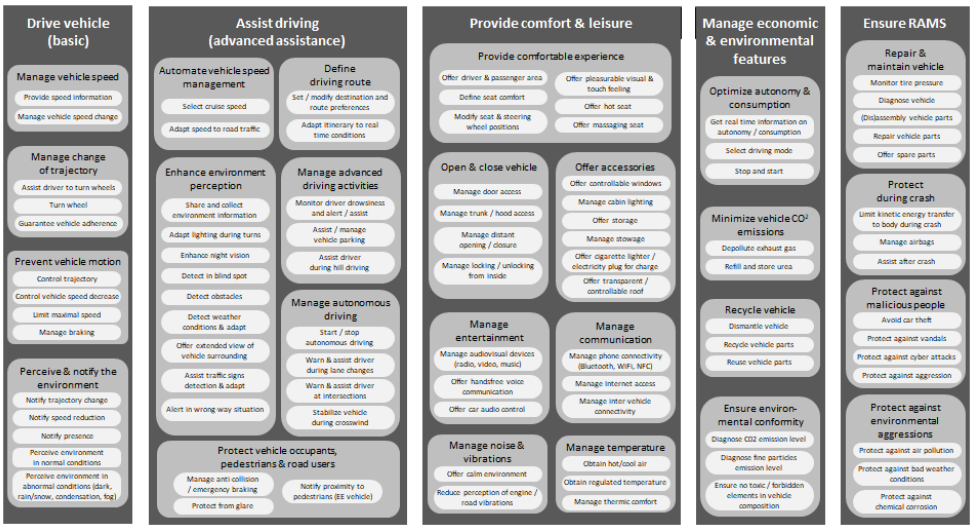

Figure 6 – Reference operational systems architecture up to level 3 for an extended vehicle

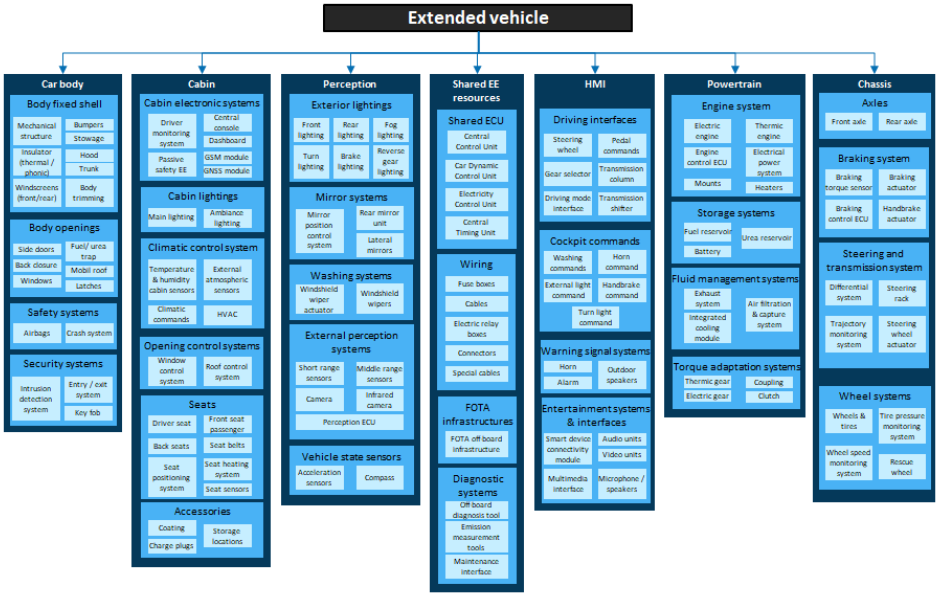

Figure 7 – Reference constructional systems architecture up to level 3 for an extended vehicle

At this stage, we could assume that the target reference systems architecture, that we were looking for, had the structure with three levels of abstraction which is described in Figure 8, where we colored the systems architecture views for which we had been able to propose a start of organization.

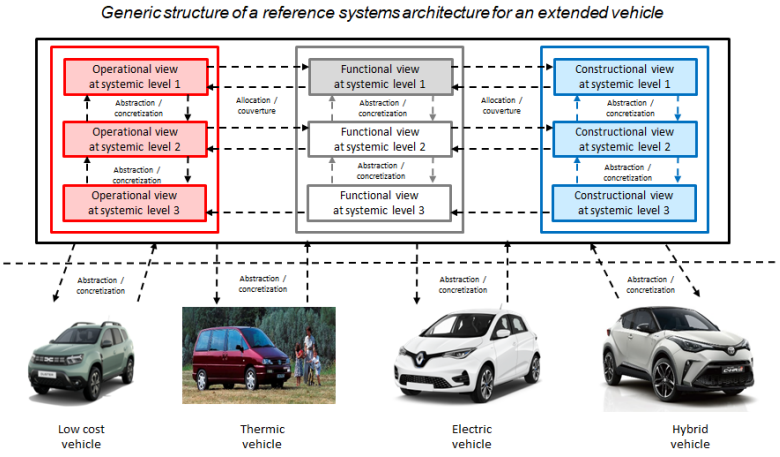

Figure 8 – Intermediate state of our construction of a reference systems architecture for an extended vehicle

The next phase consisted in "filling the holes" by identifying the level 2 (resp. level 3) functions which make it possible to implement the level 2 (resp. level 3) use cases and which are implemented by level 2 (resp. level 3) components, the key tool used here being of course functional analysis. The resulting reference functional architecture is illustrated on Figure 9. Once this work was done, we have a first version of the target reference systems architecture, but given its mode of construction, we could only be sure of one thing, namely that it is not correct and likely contained many inconsistencies. But it still has the merit of existing and was thus used as a starting point for a next stage of internal consistency checks, which consisted of identifying all inconsistency issues and all gaps in our architectural analyses, and completing them, while keeping a holistic vision, the important thing being of course the overall consistency of the reference systems architecture that emerged from this process.

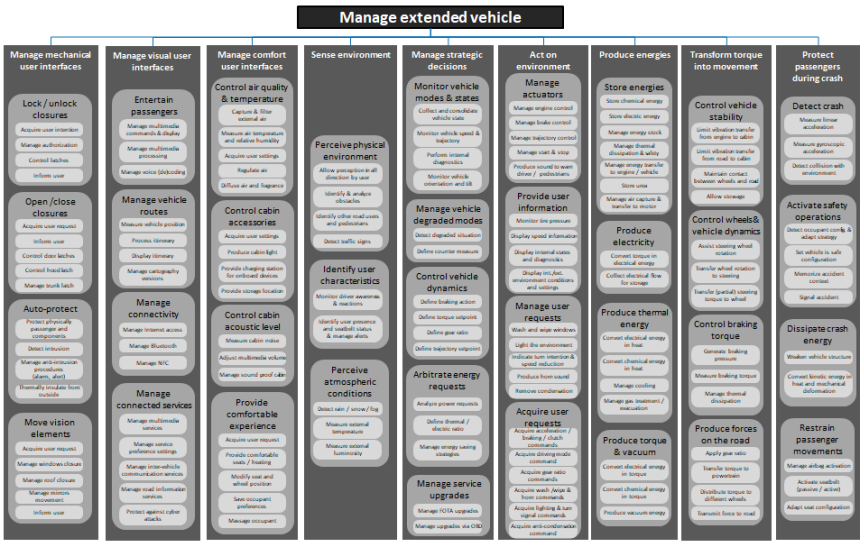

Figure 9 – Reference functional systems architecture up to level 3 for an extended vehicle

The next phase of our work consisted therefore in carrying out numerous cross-analyses to verify the solidity of the first architectural views that we brought to light and of their interrelationships. This work is difficult to formalize because there is nothing linear about it: it is made up of a multitude of small checks, where one must ensure compliance with the abstraction/concretization relationships between the different architecture views built and often going to retrieve information on this or that point and, at the end, arrive at a first version of a coherent overall reference systems architecture.

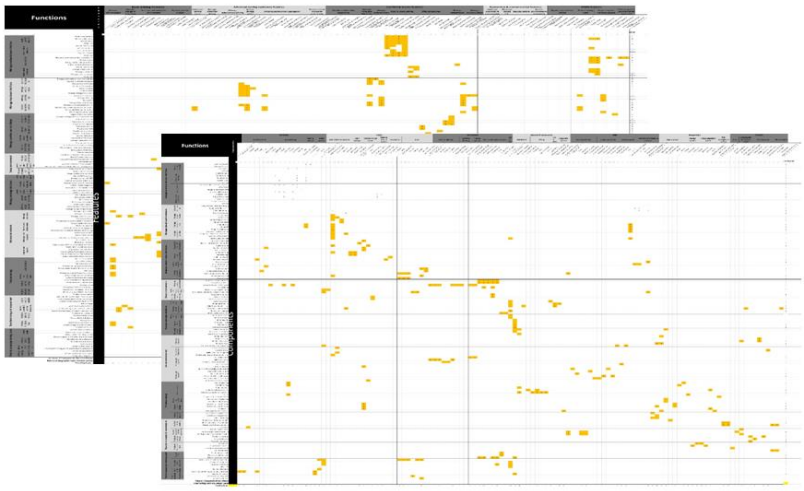

Figure 10 – Use cases to functions and functions to components traceability matrices[4]

Once such a first version of potential reference systems architecture was obtained, we then moved on to a more advanced analysis phase where the key tool is the traceability matrix: the idea is, now that we passed the where-nothing-is-stable phase, to verify one by one the relationships existing between the different objects contained in the architectural views. This requires the construction of traceability matrices between, on the one hand, all use cases and all level 1, 2 and 3 functions and, on the other hand, between all functions and all level 1, 2 components. and 3, which are both 150 x 150 arrays in

[4] The detailed corresponding material is freely available and can be communicated at demand by the authors.

terms of order of magnitude (see Figure 10). This traceability analysis raises the question of the detailed relationships existing between the different objects of our reference systems architecture and makes it possible to identify the last errors and correct them by guaranteeing the exhaustiveness of the analysis. Note finally that, since our approach was a bottom-up one, based on existing operational & constructional data, the validation criteria mentioned in section 3.1 is of course here fulfilled by construction. At the end of this stage, we therefore achieved a reasonably robust reference systems architecture for an extended vehicle, as summarized in Figure 6Figure 9 Figure 9 and Figure 7.

4 – How to Deal with Small-Scale Systems?

4.1 – Case Studies: Industrial Reference Systems Architectures

When trying to find the "right" definition of a reference systems architecture, we however faced a number of cases in which Definition 2 did not work. All these "bad" cases were dealing with families of systems of small scales, that is to say systems which only exist in small amounts, with moreover only a few available data and a lot of individual differences between each system of the considered families, which reduces the ability of achieving proper systems architectural abstractions. Despite this issue, we were however able to create reference systems architectures, as we will see, in this other situation.

Let us now present the small-scale cases that we explored. They were four of them, namely airports, buildings, nuclear plants and mines. Airports and buildings can be classified in the same category, since they both deal with civil engineering systems, when nuclear plants and mines can be analyzed as enterprises, which form another category of analysis. We will here discuss only one case study in each of these categories, since it appeared that the problem is exactly the same – with respect to defining a reference systems architecture – in each of them.

The first interesting case to discuss is the "building" case. We worked here in partnership with an international civil engineering leader which brought us a large variety of case studies, allowing us to manage an efficient abstraction work and achieve relevant reference systems architectures. The key problem that we faced here was the quite huge diversity of buildings existing in practice (e.g. individual houses, apartment buildings, factories, stadiums, etc.), relying on very different systems architectures, which prevents having a completely generic systems architecture approach for the construction sector, as is the case for example in the automotive or aeronautical industries, where all vehicle and aircraft systems can be described in the same way at a certain level of abstraction, which then opens up the possibility of constructing generic systemic models, as we saw in section 3.2 for automotive. Given this intrinsic difficulty, we choose to concentrate on a restricted perimeter for our work, i.e. collective housing, because it is already quite wide and allows to properly address the issue of reference system architectures in the field of construction.

The last difficulty with which we were confronted in the context of reference systems architectures for buildings was to take into account the great variability of the components of a building, the latter having naturally the objective of meeting as well as possible the needs of its stakeholders (architects, customers, end-users, etc.). We have therefore introduced this diversity into our reference systems architectures for buildings, by favoring the design of modular reference systems architectures, made up of modules ranging from completely fixed modules on a whole range of buildings of the same type to modules that are customized for customer needs, via intermediate standard configurable modules. An example of such a modular reference systems architecture for a retirement home room is provided in Figure 11. Note finally that on most of the case studies on which we worked on building case studies, we were not able – contrarily to the large-scale cases mentioned in section 2 – to go further than the second systemic level within the corresponding reference systems architectures.

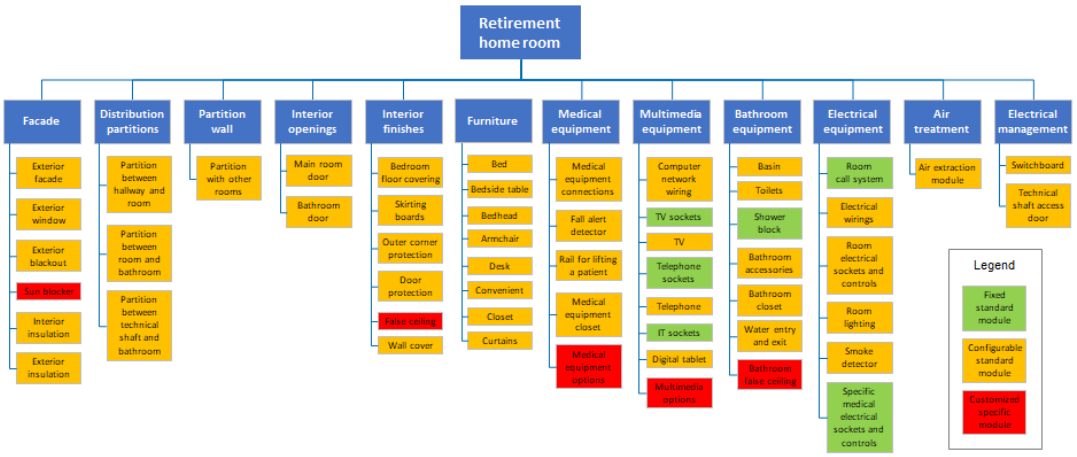

Figure 11 – Reference constructional systems architecture at level 2 for a retirement home room

Last, but not least, the importance of geometric modeling for buildings naturally led us to work on the extension of our systems architecture framework to include the geometric dimension. At this level, we had to resolve a crucial semantic problem: is geometry a new systemic vision in its own, or just a new analysis axis, transverse to the systemic visions? The second solution appeared to be the right one, since each usual systemic vision can have its own geometric representations. We can indeed geometrically represent each operational, functional or constructional view of a system, which rather gives geometry the status of a specific attribute of each systems architectural vision and not of a new systemic vision, as illustrated in Figure 12. As a consequence, it seems therefore difficult to extend the notion of reference systems architecture to geometrical architecture.

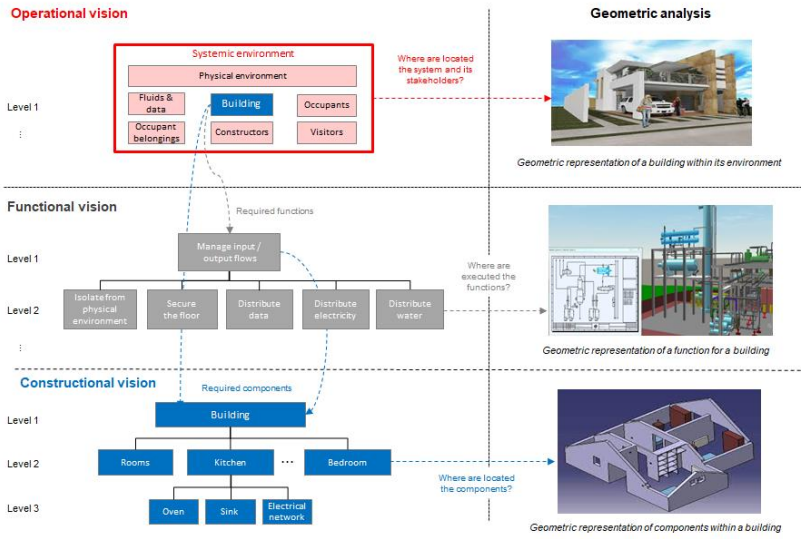

Figure 12 – How to connect geometry within a systems architecture approach

The second case that we shall discuss is the "mine" case. This new case study made it possible to open up a new dimension, namely that of enterprise architecture. A mine – or more precisely a mining company – has the particularity of being above all an organizational system coupled with a technical system, where enterprise architecture replaces systems architecture.

We worked here within a partnership with two international mining companies – a diamond mine and an iron mine on two different continents – through regular workshops with each of them, consisting in analyzing the scientific & technical literature that we captured in order to release relevant systems architectural views of their respective mines on various perspectives. The two mining examples that we had the chance to deal with, helped us to abstract the information we collected and achieve a good reference systems architecture for a mine, as partially illustrated in Figure 13. Note that again, it was not possible to go further than the second systemic level if one wants to maintain genericity.

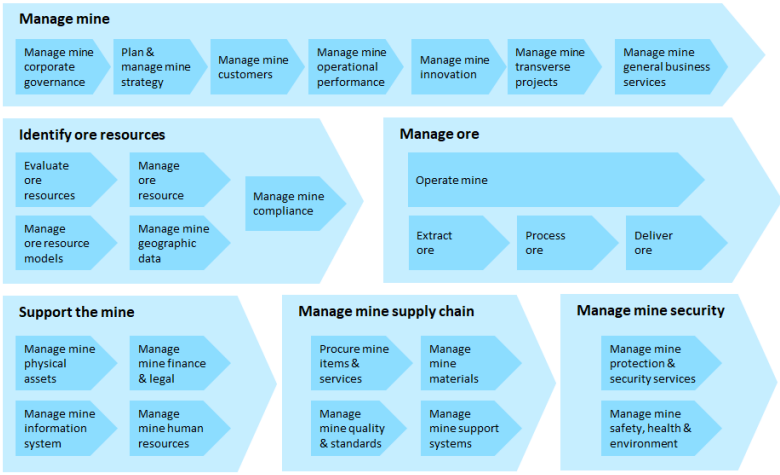

Figure 13 – Reference business systems architecture at level 2 for a mine

In this matter, we were also led to extend our systems architecture framework to take into account the two classic dimensions of an enterprise architecture, i.e. the organizational and IT dimensions on which rely any modern organization. Figure 14 summarizes how our generic system analysis framework should be instantiated to take into account the specific issues of enterprise architecture.

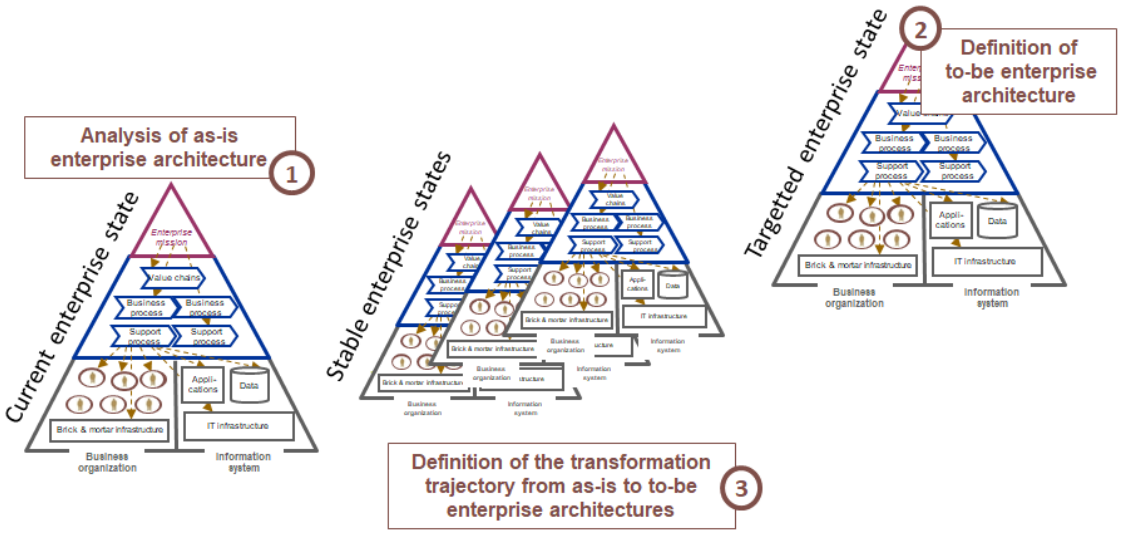

Figure 14 – Overview of our enterprise architecture framework

The main evolutions of our systems architecture framework consisted in segregating the enterprise systems architecture views into two coupled sets, one dedicated to the organizational subsystem of an enterprise and another one dedicated to its information subsystem, which leads to segregate, in the enterprise case, all system concepts – use cases, functions and components – into organizational

and information system concepts: business processes are then for instance nothing else than the functions of the organization, which explains why Figure 13, which presents the reference business architecture of a mine, can be seen as a reference systems architecture, here of functional nature for the organizational subsystem of a mine. Another key point that we shall mention is the introduction of the notion of transformation trajectory to take into account the deformable nature of organizational systems, which naturally leads to introduce the concepts of transformation state and transformation requirement. The overview of our enterprise architecture framework is illustrated on Figure 14. As explained above, this new framework is nothing else than an instantiation & adaptation of our general systems architecture framework, as presented in section 2.1, to an enterprise system.

4.2 – Another Concept of Reference Systems Architecture

The analysis of the various case studies mentioned in section 4.1 showed that Definition 2 did not work in these contexts and obliged us to propose a new conceptualization, as presented in Definition 3. The main issues that we indeed faced here were, either pragmatic considerations (lack of data to construct relevant generic systems architectures), or much deeper reasons (difficulties of abstraction and/or concretization). This new restricted definition for a reference systems architecture is based on the notion of logical architecture, which shall be seen as a synthesis between functional and constructional architectures. Using this new definition allowed then us to take into account all the small-scale case studies that we managed, which validates it in practice.

Definition 3 – Reference systems architecture (restricted definition for small-scale families). Let F be a family of systems that abstracts a set R of real systems. A *reference systems architecture* associated with F – or equivalently with R – is then provided by the following items:

- a set L of logical systems architecture views of the family F in full generality,
- an instantiation mechanism that allows to derive from L the concrete logical architectures of any system of R.

Figure 15 illustrates this new framework with do-it-yourself systems, for which reference systems architectures in the meaning of Definition 3 can actually be built. Note that the maximal level N of abstractions involved in such a reference systems architecture, as mentioned in Figure 15, appears in practice to be equal to 2 for small-scale and 3 for large-scale systems, as discussed previously.

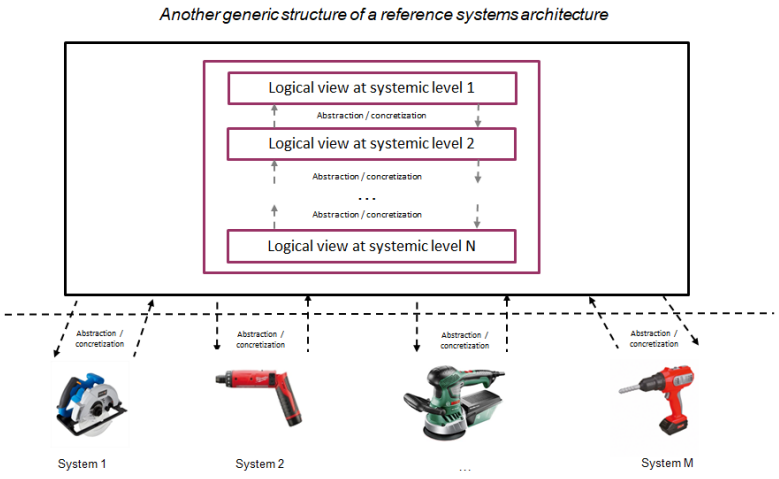

Figure 15 – Another definition of a reference systems architecture illustrated in a do-it-yourself tool context

5 – Conclusion

In this paper, we presented proposals for clear semantics of a reference systems architecture, since such systems architectures are precious assets for any systems engineer and architect. The proposed general definition appeared to work on a quite important number of concrete large-scale examples, so we do believe that it is reasonably sound. However, only experience can give us a certain level of confidence in such a definition, which is the main lesson of the different case studies that are discussed in this paper. Our work shall therefore be seen as a contribution that has still to be checked.

Another key byproduct is the emergence of the "magic" number 3 in the context of reference systems architectures. In all examples of large-scale products that we were able to address, it indeed seems that 3 is the maximal level of abstraction where one can establish a reference systems architecture, which would be consistent with the famous "7x7x7" cognitive rule, i.e. a system model can only be understood by an human being if it contains at most 3 systemic levels and 7 elements per level (see [11] for more details). For the small-scale systems, diversity reduces this limit to 2 in practice. This information is of course of pure pragmatic nature, but it may be helpful in practice. Moreover, one may ask whether it is not a deep fundamental property of systems which is highlighted in this way.

6 – References

[1] Blanchard B.S., Fabrycky W.J., *Systems engineering and analysis*, Prentice Hall, 1998.

[2] Buschmann F., Henney K., Schmidt D.C., *Pattern-oriented software architecture: On patterns and pattern languages*, John Wiley and Sons, 2013.

[3] Bussemaker, J. H., Ciampa, P. D., *MBSE in architecture design space exploration,* In Handbook of model-based systems engineering, 1-41. Springer, 2022.

[4] Cook, D., Schindel, W. D., *Utilizing MBSE patterns to accelerate system verification*. Insight, 20 (1), 32-41, 2017.

[5] Corns, S., Gibson, C., *A Model-based Reference Architecture for Medical Device Development*. In INCOSE International Symposium, 22, (1), 2066-2075, 2012.

[6] Cousot P., Cousot R., *Abstract Interpretation,* Symposium on Models of Programming Languages and Computation, ACM Computing Surveys, 28 (2), 324-328, 1996.

[7] de Weck O., *Strategic Engineering – Designing systems for an uncertain future*, MIT, 2006.

[8] INCOSE, *Systems Engineering Handbook, A guide for system life cycle processes and activities*, INCOSE, 2011.

[9] Jackson, M., Wilkerson, M., Castet, J. F., *Exposing hidden parts of the SE process: MBSE patterns and tools for tracking and traceability*, In 2016 IEEE Aerospace Conference, 1-12, IEEE, 2016.

[10] Kelly, M. S., Jacques, D., Ayres, B., Cobb, R., Ford, T. *Using a CubeSat Reference Architecture for Accelerated Model Development and Analysis*, J. of Small Satellites, 10 (3), 1097-1108, 2021.

[11] Krob D., *Model-Based Systems Architecting – Using CESAM to Architect Complex Systems*, ISTE, Wiley, 2022.

[12] Mahut F., *Intégration des approches PLM et SLM pour le développement et la gestion des Systèmes Produit-Service en contexte automobile : proposition méthodologique*, Thèse de doctorat, Université de Compiègne, 2019.

[13] Mueen Mulla F., Kulkarni V.N., Gaitonde V. N., Kottursheta B. B., *PLM as a tool for collaboration in aerospace industries – A review*, AIP Conf. Proc., 2316 (1), 2021 – doi:10.1063/5.0036546.

[14] Maier M.W., Rechtin E., *The art of systems architecting*, CRC Press, 2002.

[15] Sage A.P., Armstrong J.E. Jr., *Introduction to systems engineering*, John Wiley, 2000.

[16] Sillitto H., *Architecting systems – Concepts, principles and practice*, College Publications, 2014.

[17] Ubrich S., Reschka A., Rieken J., Ernst S., Bagschik G., Gierkes F., Nolte M., Maurer M., *Towards a Functional System Architecture for Automated Vehicles,* ArXiv, 2017 – arXiv:1703.08557.

[18] Wikipedia, *List of auto parts*, https://en.wikipedia.org/wiki/List_of_auto_parts.

[19] Wu, Q., Gouyon, D., Levrat, E., Boudau, S., *A review of know-how reuse with patterns in model-based systems engineering*, In Proceedings of the Ninth International Conference on Complex Systems Design & Management (CSD&M Paris 2018), E. Bonjour, D. Krob, L. Palladino, F. Stephan Eds., 219-229, Springer, 2018.

[20] Zoepf S.M., *Automotive features: mass impact and deployment characterization*, MIT Libraries, 2011.