# Capitalization and reuse with patterns in a Model-Based Systems Engineering (MBSE) framework

Quentin Wu, David Gouyon, Sophie Boudau, Eric Levrat

# Capitalization and reuse with patterns in a Model-Based Systems Engineering (MBSE) framework

Quentin Wu
Zodiac Aero Electric
Montreuil, France
quentin.wu@zodiacaerospace.com

David Gouyon
Université de Lorraine, CNRS, CRAN
Nancy, France
david.gouyon@univ-lorraine.fr

Sophie Boudau
Zodiac Aero Electric
Montreuil, France
sophie.boudau@zodiacaerospace.com

Éric Levrat
Université de Lorraine, CNRS, CRAN
Nancy, France
eric.levrat@univ-lorraine.fr

***Abstract*— In order to promote capitalization and reuse within a Model-Based System Engineering (MBSE) framework, this paper proposes a methodological approach that relies on the concept of pattern in order to encapsulate the know-how to be capitalized and reused. Indeed, formalizing and maintaining know-how within a company is essential in order to have a common base of "good practices" available to all engineering teams. To do this, it is necessary to undertake a capitalization process in order to encapsulate these practices. However, it is equally important to make this know-how available and to facilitate its reuse so that engineers can adapt it to their needs. The flexibility of patterns during reuse is an advantage that will contribute to the efficiency of MBSE and where engineering teams are able to rely on the company's know-how.**

***Keywords—Systems modelling, Systems analysis and design, Systems architecture, Pattern recognition***

## I. INTRODUCTION

It is essential for each organization to capitalize and maintain its know-how in order to create a common base of key know-how to be shared among its engineering teams. This allows a better understanding of the system to be developed as well as the company's own processes and methods. It improves engineering efficiency and presents many advantages when tackling risks such as: departure of experienced employees [1] (who possess know-how), "white page" syndrome, copy and paste [2], deviation from needs and requirements [3], repetition of the same mistakes, "reinventing the wheel"... Moreover, key know-how should be “dynamic” and not “static”. Instead of being stuck inside individual, it must be shared amongst everyone to promote a common database of developed Systems Of Interest and System Engineering Activities [4]. However, the implementation of a know-how reuse approach, in particular within a Model-Based System Engineering (MBSE) framework, must on the one hand meet needs of capitalization, selection, reuse and update, and on the other hand, answer the following questions:

- What should be capitalized?
- In what form should this key know-how be capitalized?
- Which reuse approach should be undertaken within a MBSE framework?

This article, which context is Model-Based System Engineering, takes advantages of patterns [5]–[7] to facilitate the reuse of models, and proposes a methodological contribution in that sense. After a state of the art on patterns in Systems Engineering and patterns in MBSE in section II, section III proposes to characterize the levels of abstractions that can be achieved upon capitalization, and introduces the MMI approach which objective is to be an efficient methodological guide for capitalization and reuse with patterns. This approach is then applied on an electrical distribution system in section IV. Advantages and current limitations of the approach are then discussed in section V.

## II. STATE OF THE ART

### A. Patterns in Systems Engineering

Reuse know-how gathered by engineers from their past experiences is an important challenge to tackle for companies, as those "archives" are stuck in engineer’s mind, making it difficult to share them to someone else [8], [9]. Research works have already been done for reusing knowledge and know-how in Systems Engineering [6], [10]–[15] and one way that looks particularly promising is achieved through the adoption of patterns to systematize complex systems engineering [16].

It appears that some recurrent characteristics of patterns seem to be fruitful for Systems Engineering. First of all, it is necessary to acknowledge the fact that similar designs can emerge from independent engineering teams [17]. Those similarities implies one of the first characteristic of patterns: they “are not created from a blank page; they are mined” [18]. The “mining” of pattern appears to be a scientific issues that is essential to resolve as Systems Engineering patterns are embedded in existing designs [4]. Some research works have tried to classify mining’s processes such as [19] with three categories of contributions: individual, second-hand and workshops/meeting. Other works have tried to guide the writing of patterns during mining’s process. In this way, some works have extended the observation made that the core meaning of a pattern is composed only of a “Minimal Triangle” : {Context, Problem, Solution} [17], by creating a specific format to describe a pattern for Systems Engineering containing all the necessary information such as [13], [20].

The use of such patterns have therefore been studied, and their value have also been studied. Works conducted in a software community have shown that the larger a team size was, the more patterns were used [21]. Indeed, it appears that pattern has allowed to deliver at each level of the development the correct information and thus has eased the creation of a common lexicon between users fostering a

common understanding of the context, problems, and solutions.

### B. *Patterns in Model-Based Systems Engineering*

Research works have been made to investigate about the concept of pattern in Systems Engineering. However, as the interest for MBSE is increasing, the value of pattern in a MBSE framework has not been fully explored. Yet, it appears that introducing or reinforcing reuse capacity in MBSE methodologies allows the design of a new project with much less human effort, benefiting from the reuse of the already existing system models [22], since no physical limitations get in the way.

It is therefore interesting to examine current limitations that slow the adoption of pattern as an introduction of reuse capacity in MBSE. First, on the one hand, as [14] stated, the "biggest problem is to transfer and manage the knowledge [of] what is actually available for re-use". On the other hand, the adoption of MBSE due to the steep learning curve induced for organizations is an obstacle that prevent them to "quickly identify not only valid architectural solutions, but optimal value solutions for the mission need" [23]. As underlined in those previous works, a reuse approach in MBSE needs to efficiently identify, locate but also to allow to search, update and especially reuse know-how. Thus, the creation of patterns library is not sufficient, as the purpose is to allow engineers to seamlessly reuse those patterns in their ongoing projects [14]. In that sense, [24] do not focus on the creation of library but proposes to semi-automatically create an activity diagram from existing activity diagrams according to the input use case diagram. The approach is adapting promising reusable elements during a model reuse process thanks to know-how that have been capitalized, with the aim of simplifying know-how reuse.

In order to be efficiently used, it is necessary to clearly define how to interlock a pattern-based approach to the different phases of a design cycle. That is why, works have been done to introduce patterns during various phases of the engineering cycles. [3] described behavioural construct patterns (Figure 1) to facilitate and systematize the modelling of system behaviour. It helps engineers, by elevating the abstraction level at which they can think from an atomic graphical elements to a more structured elements called behavioural constructs. This approach allows a higher modelling level that will permit to work in an algorithmic way of thinking: more design, less aesthetics.

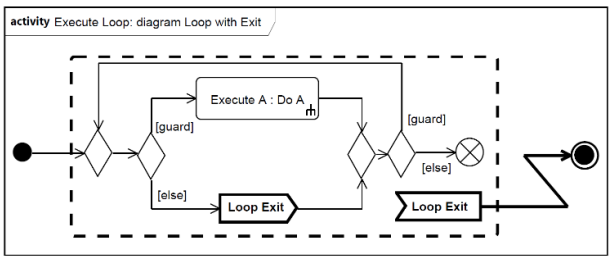


*Figure 1. Loop Exit Construct, extracted from [3]*

Once a pattern is designed to be used at a certain phase of a design cycle, it should help engineers to focus on what is important. In that way, some research works assume that patterns should guide the development to avoid deviation. Indeed, adequate guidelines for modelling benefits the development process by resulting on more mature models and a certain modelling homogeneity. This is why [25] proposed a process for the development of mechatronic systems based on a SysML design pattern. By allowing an efficient traceability of all information within the system model to trace change influences more easily, this approach proves to be particularly helpful for facilitating the impact analysis in later lifecycle phases and for the reuse for future projects.

In the same spirit of guiding modelling, Pattern-Based Systems Engineering (PBSE) [26] is an engineering paradigm where patterns are re-usable models, which can be configured or specialized into product lines or into product systems. In this context, [27] apply patterns to requirements and design. At a high-level, they constitute a generic system pattern model that can be customized according to enterprise needs, configuration, uses, so that engineers can benefit from the concepts of MBSE without being an expert of modelling methodologies.

As a main issue for system engineers is to shorten engineering cycle period, MBSE and pattern appears to be a great combination to face this challenge. However, this state of the art has highlighted the strong methodological need to capitalize on previous projects to reuse know-how. It is only once this step has been completed that it becomes possible to focus on sharing know-how and foster its reuse for future MBSE projects. This is why the concept of patterns appears to be an answer to tackle this challenge, as it offers the possibility to make information dynamic between stakeholders during the development of complex systems

## III. METHODOLOGICAL PROPOSITION

This section proposes first to characterize the levels of abstractions that can be achieved upon capitalization. Based on these levels, it then introduces the MMI approach which objective is to be an efficient methodological guide for capitalization and reuse through the concept of patterns.

### A. *Levels of abstraction*

During the design of a complex system, it is important to take a step back in order to see the big picture and ensure high-level consistency, as promoted by Systems Thinking principles [28]. This can be achieved by increasing the level of abstraction of the modelling objects on which engineers usually work. In that way, the formalization of patterns is possible on several levels of abstraction. This paper proposes a classification in four levels (Figure 2), ranging from "models" developed by the engineer at the lowest level of abstraction to "abstract patterns" at the highest level of abstraction.

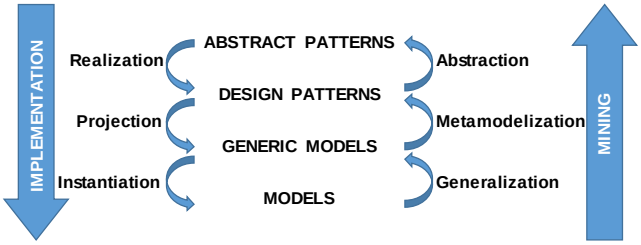


*Figure 2. Levels of abstraction (Model - Pattern)*

This makes it possible to describe a top-down and a bottom-up flows, corresponding respectively to a mining

process ([18], [19], [29]) and an implementation process, presented in section III.B.

*1) Models*

Models are the result of engineering processes and rely on engineering know-how. Each project has its own models, expressed in specific engineering languages. These sets of models, specific to each project, can therefore be isolated into "silos". Their characteristics are that they potentially have different syntaxes and semantics if there are no common modelling rules. Similarly, naming rules can be specific to each project, as can be the project structure and interactions with the engineering environment.

*2) Generic models*

At this level, generic models are also expressed in a modelling language specific to the engineering domain. However, unlike the lower level models, generic models are intended to highlight the similarities and recurrent design in systems and projects, which can be of different types: requirements, design, interfaces, functions... The idea is to generalize models with a higher level of abstraction to systematize design, and improve the readability and understanding of models. Generic models can be instantiated to build models.

*3) Design patterns*

At this level, the description of the pattern is done independently of the engineering modelling language. The purpose of this pattern is to describe the elements to be reused in a modelling process. The objective is therefore to capture know-how, in a formalism that is independent of the context of each engineering team (metamodelization), in order to facilitate its promotion and reuse. However, these patterns remain described and adapted to a specific field. Design patterns can be projected on specific engineering languages to build generic models.

*4) Abstract patterns*

Abstract patterns are domain independent abstractions of design patterns. This conceptual level, which is not mandatory, makes it possible to explain system at a very high level of abstraction such as general structuring principles.

Abstract patterns can be realized into design patterns.

*5) Transitions between levels of abstraction*

Every transition needs to focus on one engineering artefact at the same time such as: use case, function, component… It is also necessary to focus on one point of view of the system model to ensure consistency at the higher level of abstraction.

For the transition "generalization", it is important to check repetition of modelling artefact or group of them. Sometimes, it can be a sequence (functional chain for example) that needs to be highlighted. However, it is also necessary to pay attention to informal "signature" of modelling. For instance, engineers can create diagrams in which separations between elements are modeled but not formalized.

For the transition "metamodelization", it is necessary to isolate the concept of the modelling language that needs to be capitalized. For instance, inside a functional architecture it is possible to capitalize the structure (functional breakdown), but also function (in and out flows, interfaces) or a specific functional chain… It may be necessary to capitalize those three aspects, however, it will be necessary to create one pattern for each concept.

For the transition "abstraction", the objective is to free oneself from strict syntax and semantic rules in order to create a medium that makes it easier to convey the chosen concept. Creativity is the key here.

## B. MMI Methodological approach

The objective being to provide a method for the formalization and effective reuse of patterns, this paper proposes a methodological approach to meet this need, that consists in the search for patterns ("Mining" process), the maturation of these patterns for reuse ("Maturation" process), and finally the concrete reuse of these capitalized patterns at different levels of abstraction for modeling ("Implementation" process). This approach is therefore called Mining-Maturation-Implementation (MMI).

*1) High-level view of MMI*

At a high level, the MMI approach aims at producing a system model compliant to the customer needs and requirements by reusing existing models. First, the implementation of a capitalization approach is not trivial and requires a significant investment that is sometimes difficult to reconcile with the life of a project and its multiple constraints (deadlines, costs, staff, etc.). Indeed, the decision to initiate the process of capitalization is not self-evident and is an important step in continuing the approach.

As represented in Figure 3, after the decision of capitalization, two processes must take place, they are complementary and iterative. Indeed, one concerns the operational side of the approach ("MMI_OP"), while the

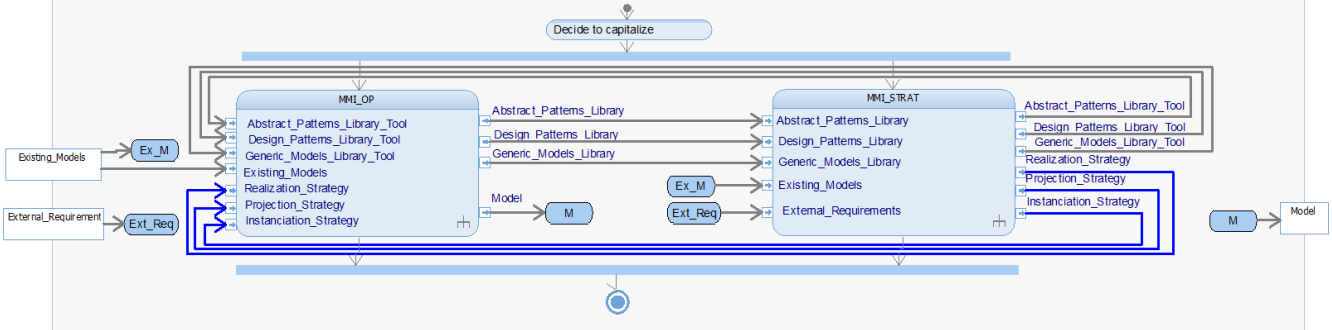


*Figure 3. High-level MMI*

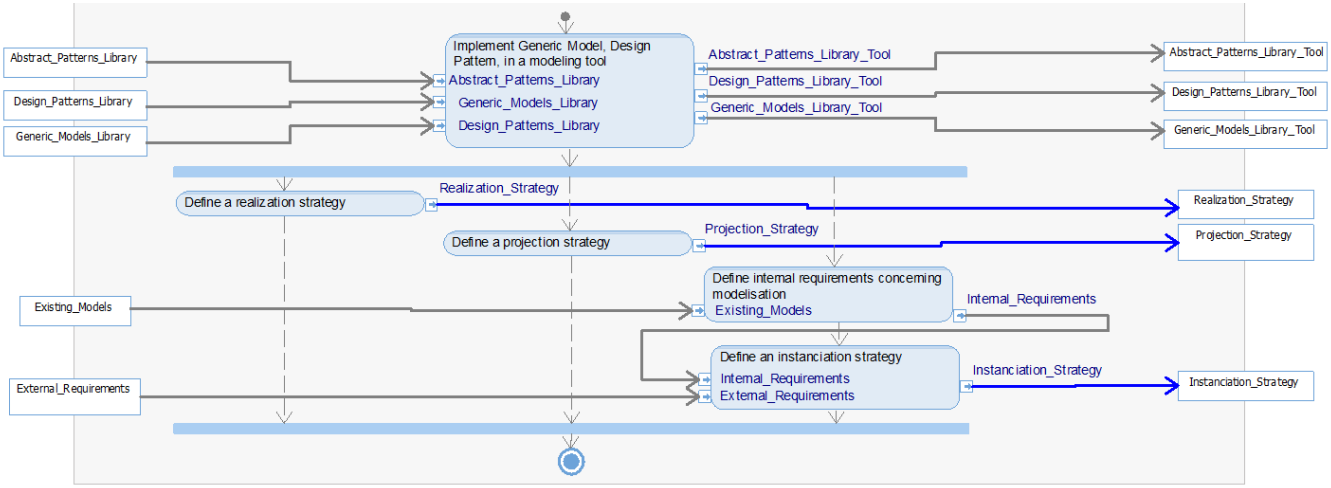


*Figure 4. Strategic MMI process*

other one is dedicated to defining the strategic rules governing the operational aspect ("MMI_STRAT").

*2) Strategic MMI ("MMI_STRAT")*

From a library of abstract patterns, design patterns, and generic models, the "Strategic MMI" (Figure 4) process aims at two main objectives.

The first one concerns the implementation of those libraries in a modelling tool, which means to comply with a specific modelling language. Indeed, it is necessary to implement those libraries in a tool in order to allow a seamless reuse of know-how by engineers. Once implemented in a tool, those libraries feed the "Operational MMI" process.

The second objective is dedicated to the definition of the transition from one level of abstraction to another, in order to stay consistent during the application of the approach. Due to the possibility of divergence between different engineers, it is necessary to agree on a common strategy. Once again those strategies will feed the "Operational MMI" process.

*3) Operational MMI ("MMI_OP")*

The operational MMI Process (Figure 5**Erreur ! Source du renvoi introuvable.**) aims in the search for patterns ("Mining" process), the maturation of these patterns for reuse ("Maturation" process), and finally the concrete reuse of these capitalized patterns at different levels of abstraction for modelling ("Implementation" process).

*a) Mining*

The mining process requires an analysis of previous projects and, if possible, ongoing projects. The analysis of these models will make it possible to start the mining process

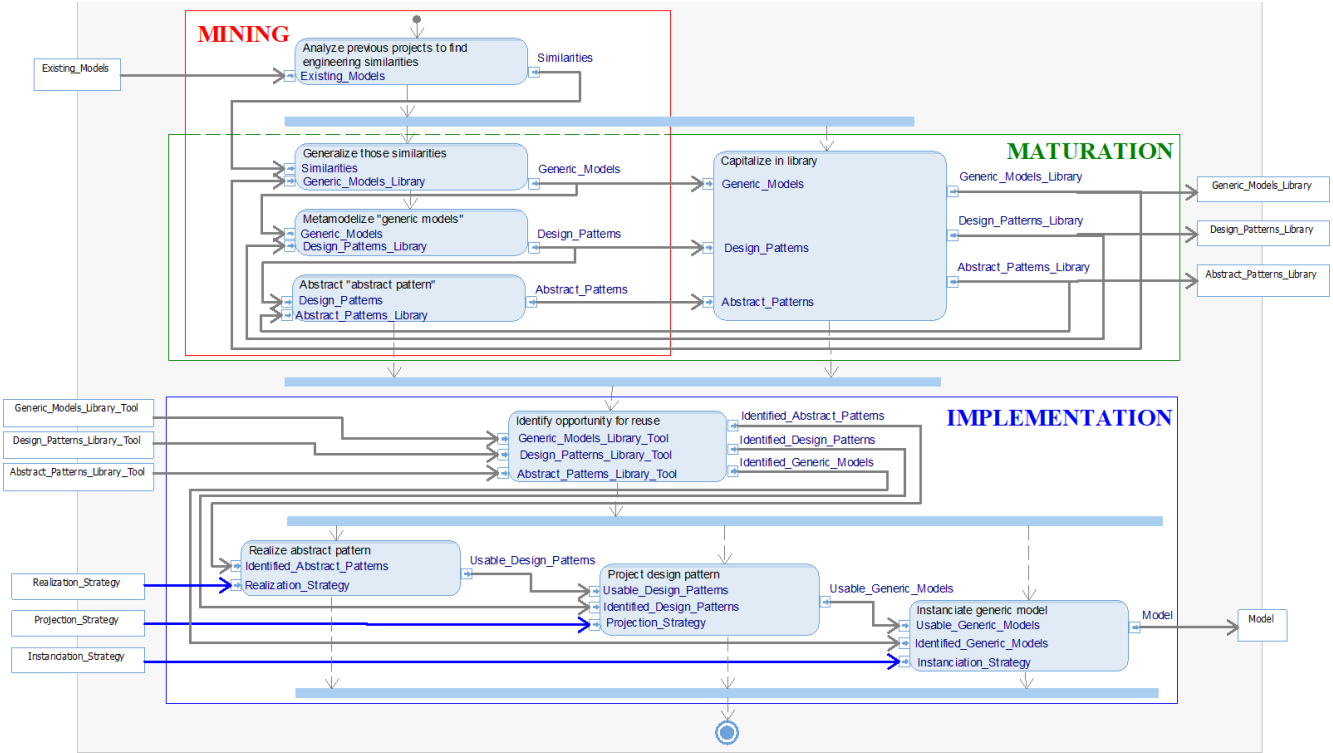


*Figure 5. Operational MMI process*

by identifying, locating, isolating similarities of System Engineering that are reused in several places in the same project or in different projects, in order to propose generic models. These generic models are then metamodelized into design patterns, which can be abstracted into abstract patterns.

These elements will then be used in the “Maturation” process.

*b) Maturation*

Maturation is a crucial process of the methodological approach because it has a very strong impact on the "implementation" process. Indeed, it will be necessary to evaluate the identified generic models, design and abstract patterns so that their level of maturity (level of confidence) corresponds to a level that allows them to be reused on new projects. Once they have reached a sufficient level of maturity, they can be stored and classified into a library. The goal of this library is to ease the update, search and reuse of patterns, by fostering capitalized know-how towards engineers.

These libraries are then used in the “Strategic MMI” process.

*c) Implementation*

During this process, when a reuse opportunity is identified, strategies identified in the “Strategic MMI” process allow either the realization of an abstract or the projections of a design pattern or the instantiation of a generic model to model the system. The ‘Implementation’ process leaves an active part to the user who will integrate reusable elements into his model, depending on the requirements to be met. This integration is not automatic in the sense that the engineer must be able to modify his model according to the operational, functional, logical and organic groupings he wishes to make.

## IV. CASE STUDY

In the aeronautics field, an electrical power distribution system is a subsystem which purpose is to generate, regulate and distribute electrical power throughout the airplane. Design evolutions of these systems have allowed to benefit from significant advantages such as the routing of power around localized faults to maintain airworthiness. However, this has resulted in a significant increase in the complexity of these systems in such a way that it becomes difficult for one person to understand the entire system.

This article conducts a case study on such a complex system in order to explore the capacity of the proposed methodology. However, due to the large perimeter covered by the MMI approach, only the “MMI_OP” process (Figure 5) will be illustrated. Moreover, for confidentiality reasons, figures will be blurred in order to protect the data inside. This will not prevent the example from being understood in the sense that the focus will be on the approach implemented rather than on the processed models.

As stated in Figure 3, the first step towards reuse, is to take the decision of capitalizing know-how. This implies being able to recognize a situation where the reuse of know-how can take place. Concerning the case study of this article, the situation appeared when analyzing the functional breakdown structure of different systems. As represented in the left of Figure 6**Erreur ! Source du renvoi introuvable.**, it appears that some functions and their sub-functions were identical in various locations inside a system functional breakdown structure, or at the same location inside the various systems at stake.

*Figure 6. Example of criteria to start a capitalization process*

Engineering iterations on this breakdown structure have made it possible to factorize functions. The breakdown structure then became simpler to read (right of Figure 6**Erreur ! Source du renvoi introuvable.**). Indeed, from a first draft of 700+ functions divided on 8 hierarchical levels, it appears that the core functions represents around 120 functions on 6 hierarchical levels. This means that the ratio of functions being reused is around 6, and therefore it appears relevant and valuable to start a capitalization process.

The functional breakdown structure is linked to a description of a functional architecture as presented in Figure 7 with IBM Rhapsody in SysML language. The analysis of this model puts in evidence similarities which are repeated at various locations, as highlighted on the Figure 7. The mining of previous projects also shown that those similarities are being used at the same location on other models.

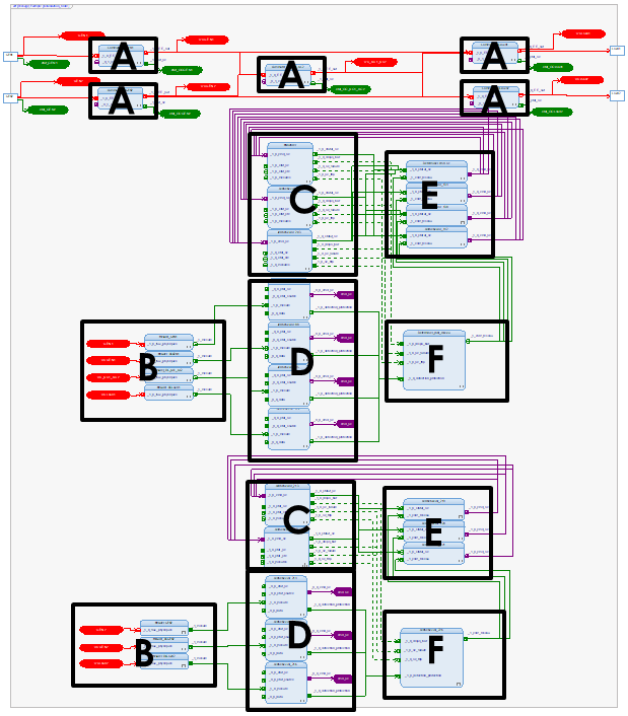


*Figure 7. Similarities at the “Models” level (SysML)*

The functional architecture of Figure 7 is currently at the first level of abstraction defined in Figure 2. However, after the analysis has been performed, it is possible to generalize the model into a generic functional architecture (Figure 8)

that corresponds to the second level of abstraction of Figure 2.

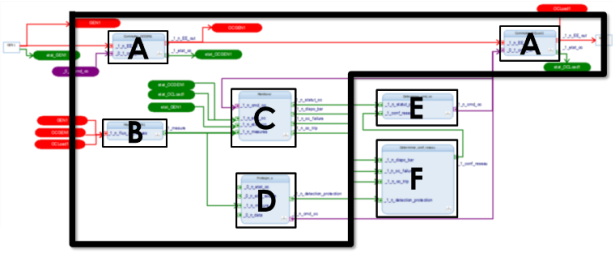


*Figure 8. Similarity at the "Generic Models" level (SysML)*

This generic model shows a generic architecture of generic functions characterized by their inputs and outputs, in terms of numbers (use of cardinality) and flow types (physical, information, command…). Those generic elements can be capitalized into a library (Figure 9). It is also possible to characterize those generic functions with other engineering artefacts like requirements, functional modes… This choice is not exclusive and depends on the desired methodology to be implemented.

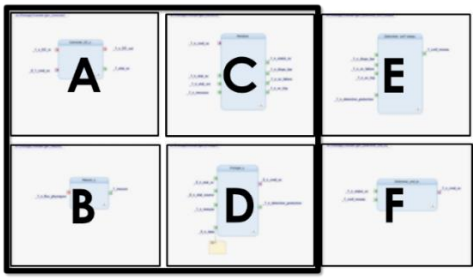


*Figure 9. Library of generic functions models in IBM Rhapsody*

The big black frame on Figure 8 and Figure 9 corresponds to the perimeter of a function F1, one of the main function of the system. It is possible to metamodelize this function to reach the third level of capitalization defined in Figure 2. As represented in Figure 10, it is possible to describe, for example, the functional breakdown design pattern of the function F1 in UML, which is independent from the engineering modelling language SysML used at "Models" and "Generic Models" level.

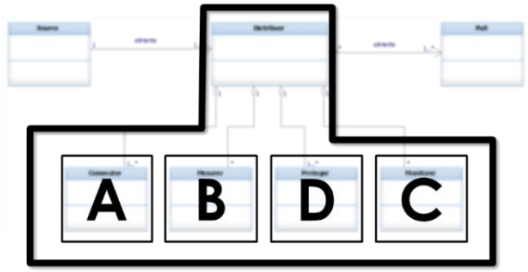


*Figure 10. Description of the functional breakdown structure of function F1 at the "Design Patterns" level (UML)*

After mining and capitalizing in library the "MMI_OP" process aims at facilitating the implementation of the know-how for future projects. It means help the user to identify reuse opportunities and to search for capitalized know-how from library.

During the design of another project, if the function F1 is identified to be reused, then the implementation process of the operational MMI process (Figure 5) can start. From the design patterns library constructed before, it is possible to project those patterns into a tool using another engineering language than SysML, for example, the tool Capella. This illustrates one of the interest of the approach, which is that various modelling tools and engineering languages can be used for the mining and implementation processes. The result for the library of generic functions models is shown in Figure 11.

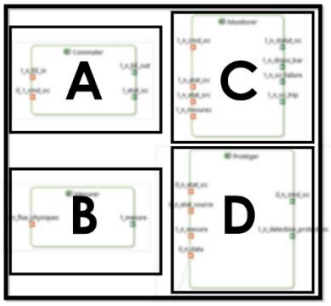


*Figure 11. Library of generic functions models in Capella*

From those generic functions models, it is possible to construct a generic functional architecture (Figure 12).

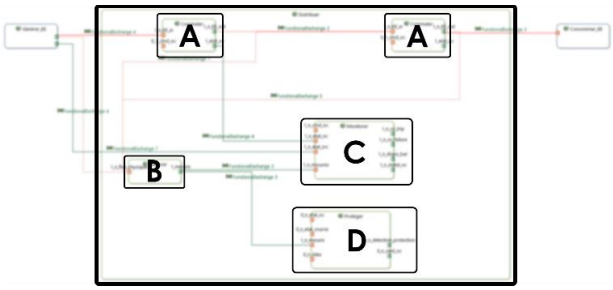


*Figure 12. Generic model constructed from the generic functions*

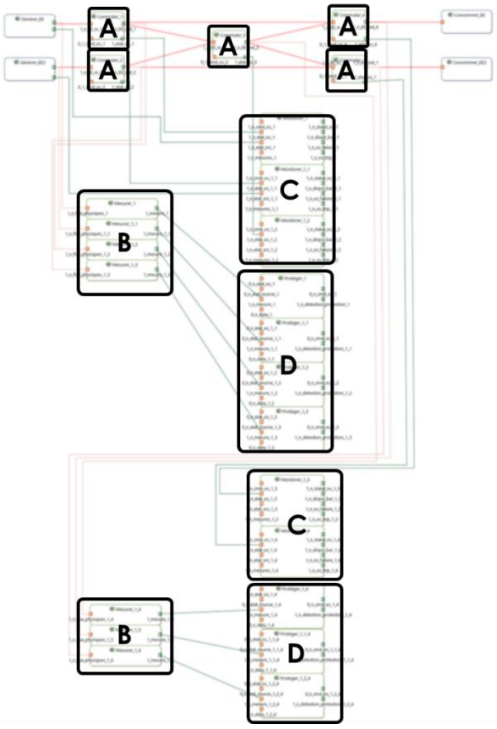


*Figure 13. Models constructed from generic models (in Capella)*

The generic functional architecture is then instantiated into a project-specific architecture model (Figure 13).

## V. Discussion

The use of an object oriented approach implies that system engineers can easily loose the level of abstraction at which they are modelling and therefore spend time improving the aesthetics of their diagrams instead of the expected design [3]. The purpose of the MMI approach is to guide users to optimize the capitalization and reuse processes and to allow them to focus on their design and not on aesthetic. Indeed, on one hand, definitions of levels of abstraction and on the other hand definitions of transitions between these levels are providing consistency. These are not clear cut categories but the MMI approach is formalizing a methodological pattern to supervise each step of the process.

By formalizing "Design Patterns" in a modelling language independent from engineering modelling language, the MMI approach is also answering a recurrent problem faced by current tools on the market, which is, that no standard exchange format for model exist at the moment. It means that models developed in one tools cannot be transferred in another one except by manually reconstructing it. Therefore, by describing at an abstraction level higher than tools on the market, it is possible to project capitalized know-how contained in design patterns in any tools (Figure 14), on the condition that the necessary concepts are defined in the tool.

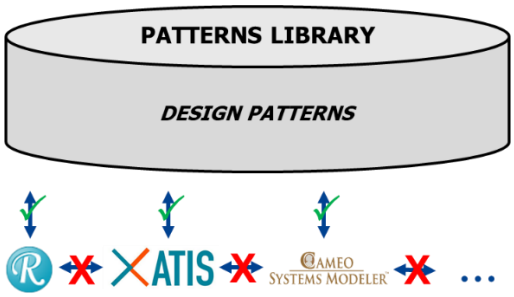


*Figure 14. "Design Patterns" are tools agnostic*

Current limitations of the approach are linked to the definition of concepts inside tools and the maturity of patterns.

On the latter, as *n* architects can provide *p* different patterns, the difficulty is to succeed at rapidly converging toward a mature pattern, accepted by all the stakeholders. Moreover, costs are high at the start of the MMI approach as the libraries have to be constructed. Therefore, it is necessary to set up an agile framework during which it is necessary to focus on short cycles in order to quickly converge on patterns (defined in a tool and not only in a document). This will allow their effective use in order to assess their maturity in a real project.

However, the capacity to formalize libraries inside tools may be a lock. It implies that the concept of library exists, otherwise, it will be necessary to make a detour to reach the goal but this will be at the expense of the tool's ergonomics or the user's autonomy.

## VI. Conclusion & Perspectives

In the context of Model-Based Systems Engineering, this paper aims at proposing a methodological contribution based on the use of patterns to facilitate the reuse of models. The proposed MMI approach can be used iteratively to enrich a know-how repository made of System Of Interest patterns.

However, pattern reuse requires the ability to identify, select, and apply patterns in a fluid manner so that the user can focus on the needs during development. An agile approach must be considered in order to initiate short mining, maturation and implementation cycles.

Thus, in future works on pattern reuse, a scale will be developed to identify maturity levels of a company's reuse process and target efforts. This scale will be multiaxial in order to cover the different aspects of the process (models, capitalization...). This will allow quantifying, on the one hand, degrees of maturity that will be specific to certain activities, and on the other hand, an overall level of maturity at the level of the process that will depend on the level on each axis.

Other ongoing works are focusing on the integration of the MMI approach in the V cycle by proposing an Y cycle, as shown in Figure 15. The objective is to define how patterns libraries elements will support Systems Engineering activities during the design phase.

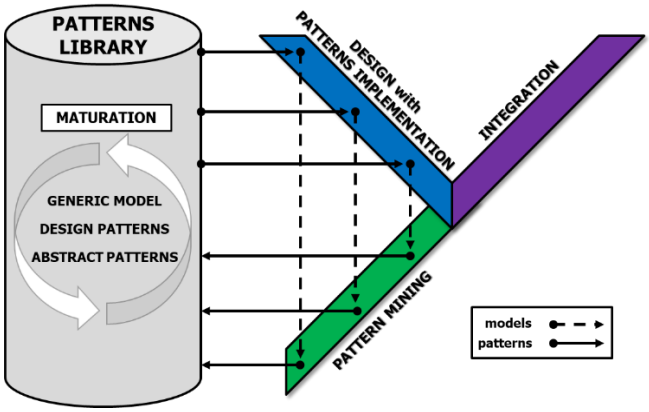


*Figure 15. Integration of patterns in the V Cycle*

Currently, all the processes of the MMI approach can be done by hand on modelling tools. A last key step to enable seamless capitalization and reuse of patterns is the implementation, in a software tool, of artificial intelligence algorithms to automate MMI processes such as patterns mining or reuse opportunities identification.

## References

[1] R. J. Cloutier, "Toward the Application of Patterns to Systems Engineering," *Syst. Eng.*, no. January 2005, pp. 73–80, 2005.

[2] R. Darimont, W. Zhao, C. Ponsard, and A. Michot, "Deploying a Template and Pattern Library for Improved Reuse of Requirements Across Projects," *Proc. - 2017 IEEE 25th Int. Requir. Eng. Conf. RE 2017*, pp. 456–457, 2017.

[3] L. Gasser, "Structuring activity diagrams," in *14th IFAC Symposium on Information Control Problems in Manufacturing, Bucharest, Romania*, 2012.

[4] F. Pfister, V. Chapurlat, M. Huchard, C. Nebut, and

J.-L. Wippler, “A Proposed Meta-Model for Formalizing Systems Engineering Knowledge, Based on Functional Architecture Patterns,” *Syst. Eng.*, vol. 15, no. 3, 2012.

[5] Q. Wu, D. Gouyon, É. Levrat, and S. Boudau, “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*, 2018, pp. 219–229.

[6] D. Cook and W. Schindel, “Utilizing Mbse Patterns To Accelerate System Verification,” *Insight*, vol. 20, no. 1, pp. 32–41, 2017.

[7] R. S. Kalawsky, D. Joannou, Y. Tian, and A. Fayoumi, “Using architecture patterns to architect and analyze systems of systems,” *Procedia Comput. Sci.*, vol. 16, no. March 2015, pp. 283–292, 2013.

[8] D. Mourtzis, M. Doukas, and C. Giannoulis, “An Inference-based Knowledge Reuse Framework for Historical Product and Production Information Retrieval,” *Procedia CIRP*, vol. 41, pp. 472–477, 2016.

[9] P. Demian and R. Fruchter, “An ethnographic study of design knowledge reuse in the architecture, engineering, and construction industry,” *Res. Eng. Des.*, vol. 16, no. 4, pp. 184–195, 2006.

[10] L. Gzara, D. Rieu, and M. Tollenaere, “Product information systems engineering: An approach for building product models by reuse of patterns,” *Robot. Comput. Integr. Manuf.*, vol. 19, no. 3, pp. 239–261, 2003.

[11] R. H. Barter, “A Systems Engineering Pattern Language,” in *INCOSE International Symposium*, 1998, pp. 350–353.

[12] R. J. Cloutier, “Model Driven Architecture for Systems Engineering,” *INCOSE Int. Work.*, no. September, 2008.

[13] C. Haskins, “Application of patterns and pattern languages to systems engineering,” in *INCOSE International Symposium*, 2005, pp. 1619–1627.

[14] A. Korff, “Re-using sysml system architectures,” in *Proceedings of the 4th International Conference on Complex Systems Design and Management*, 2013, pp. 257–266.

[15] G. Wang, R. Valerdi, and J. Fortune, “Reuse in systems engineering,” *IEEE Syst. J.*, vol. 4, no. 3, pp. 376–384, 2010.

[16] T. Cochard, “Contribution à la génération de séquences pour la conduite de systèmes complexes critiques,” Université de Lorraine, 2017.

[17] A. Gaffar and N. Moha, “Semantics of a Pattern System,” *Proc. STEP Int. Work. Des. Pattern Theory Pract. IWDPTP05*, 2005.

[18] R. S. Hanmer and K. F. Kocan, “Documenting architectures with patterns,” *Bell Labs Tech. J.*, vol. 9, no. 1, pp. 143–163, 2004.

[19] D. E. DeLano, “Patterns mining,” in *The Pattern Handbook: Techniques, Strategies, and Applications*, 1998, pp. 87–96.

[20] R. J. Cloutier and D. Verma, “Applying the concept of patterns to systems architecture,” *Syst. Eng.*, vol. 10, no. 2, pp. 138–154, 2007.

[21] M. Hahsler, “A quantitative study of the adoption of design patterns by open source software developers,” in *Free/Open Source Software Development*, Igi Global, 2005, pp. 103–124.

[22] U. Shani and H. Broodney, “Reuse in model-based systems engineering,” *9th Annu. IEEE Int. Syst. Conf. SysCon 2015 - Proc.*, pp. 77–83, 2015.

[23] C. Oster, M. Kaiser, J. Kruse, J. Wade, and R. Cloutier, “Applying Composable Architectures to the Design and Development of a Product Line of Complex Systems,” *Syst. Eng.*, vol. 19, no. 6, pp. 522–534, Nov. 2016.

[24] S. Paydar and M. Kahani, “A semi-automated approach to adapt activity diagrams for new use cases,” *Inf. Softw. Technol.*, vol. 57, no. 1, pp. 543–570, 2015.

[25] G. Barbieri, K. Kernschmidt, C. Fantuzzi, and B. Vogel-Heuser, “A SysML based design pattern for the high-level development of mechatronic systems to enhance re-usability,” in *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 2014, vol. 19, no. 3, pp. 3431–3437.

[26] W. Schindel, “Requirements Statements Are Transfer Functions: An Insight from Model-Based Systems Engineering,” in *INCOSE International Symposium*, 2005, pp. 1604–1618.

[27] W. Schindel and T. Peterson, “Introduction to Pattern-Based Systems Engineering (PBSE): Leveraging MBSE Techniques,” *INCOSE Int. Symp.*, vol. 23, no. 1, p. 1639, 2013.

[28] J. Boardman, B. Sauser, L. John, and R. Edson, “The Conceptagon A Framework for Systems Thinking and Systems Practice,” pp. 3299–3304, 2009.

[29] C. Haskins, “Using Patterns to Transition Systems Engineering from a Technological to Social Context,” *Syst. Eng.*, 2007.