A Review of Know-How Reuse with Patterns in Model-Based Systems Engineering

Quentin Wu, David Gouyon, Eric Levrat, Sophie Boudau

A review of Know-How reuse with patterns in Model-Based Systems Engineering

Quentin Wu, David Gouyon, Éric Levrat, Sophie Boudau

Abstract The increasing complexity of systems to be developed requires engineers to review their practices in order to improve the efficiency of engineering and meet the needs of a competitive market. That is why models supported by formal or semi-formal languages are preferred to avoid the understanding variability of natural languages. In this context, Model-Based Systems Engineering (MBSE) made it possible to change the engineering paradigm by putting forward a unique, shared system model. To promote its adoption, a solution would be to allow reuse of knowledge and know-how, to encourage engineers seizing and adapting MBSE to their needs. This paper aims to review and evaluate the concept of patterns towards reuse in engineering, especially in a MBSE approach.

1 Introduction

The design of increasingly complex systems is implicating longer engineering phases and greater costs during the design lifecycle of a project. Those negative impacts are accentuated by the current document-centred application of Systems Engineering (SE) processes inside companies. Indeed, system development teams are working on standalone models, communicating with other teams through documents written in natural language. This implies a huge work concerning consistency and comprehension, as information shared through those documents has to be comprehensive and unique, to avoid rework and non-conformity to customer expectations. So, there is a challenge concerning the engineering efficiency (how to enhance productivity, quality, communications, and reduce risk) needed in a highly competitive environment, where the need is to shorten engineering cycle period.

Quentin Wu, Sophie Boudau
Zodiac Aero Electric
7, Rue des Longs Quartiers, 93100 Montreuil, France
quentin.wu@zodiacaerospace.com; sophie.boudau@zodiacaerospace.com

David Gouyon, Eric Levrat
Université de Lorraine, CNRS, CRAN
Faculté des Sciences et Technologies, BP 70239, Vandoeuvre-lès-Nancy, France
david.gouyon@univ-lorraine.fr; eric.levrat@univ-lorraine.fr

In order to manage complexity, maintain consistency, and ensure traceability during systems engineering, the SE community has turned to the Model-Based Systems Engineering (MBSE) (Estefan 2008). Popularized by the International Council on Systems Engineering (INCOSE) with the MBSE Initiative[1], MBSE is defined as "the formalized application of modelling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases". However, adoption of MBSE takes time, as many inhibitors remain such as cultural and general resistance to change, lack of related Knowledge and Know-How (K&KH), or the need to higher degree of guidance and reuse.

For a wider MBSE adoption, several advances seem to be necessary concerning organizational, methodological, and tools perspectives. In particular, from a methodological point of view, reuse seems to be promising. Reusing engineer's Knowledge and Know-How is an act of capitalization on previous experiences or projects, whether it is on the System Of Interest (SOI) or on the Systems Engineering Activities (SEA). But, often, those data are kept in their mind, and works have to be done to formalize them, with the goal of sharing them so they can be reused. The expected benefits make the assumption that reused modelling artefacts satisfy some maturity criteria to grant that they have reached a level of quality compatible with reuse objectives.

This article reviews engineering practices which intent to capitalize on K&KH, and to facilitate information sharing and reuse. A focus is made on reusing K&KH through the concept of "pattern". In this way, the second section presents reuse challenges in engineering and related works, the third a short history on patterns, the fourth a literature review of pattern for SE, and the last section discusses on the interest of using patterns in MBSE.

2 Challenges and related works

The fundamental difference between knowledge and know-how is that knowledge provides only solutions and answers to problems and questions, whereas know-how provides solutions but also a manner to construct these solutions (Gzara, Rieu, and Tollenaere 2003). Thereby, engineer's know-how is built from their experience, allowing them to reuse information gathered in order to be more efficient in their tasks. However, those "archives" are stuck in engineers' mind, making it difficult to share them to someone else to foster reuse (Mourtzis, Doukas, and Giannoulis 2016; Demian and Fruchter 2006). Yet, dynamic information flowing among engineering teams is a critical challenge for many companies who need to manage complex systems as information must be shared,

[1] http://www.omgwiki.org/MBSE/doku.php (visited on 31/05/2018)

comprehensive, and coherent among the project (Miled 2014). This aspect is very important as it allows a better comprehension of the SOI and SEA. For example in requirement engineering, (Darimont et al. 2017) presents the results of a survey where 55-60% of engineers use "copy & paste" and "duplication" techniques, and only 13% use "requirements patterns catalogue". As the complexity, the quantity of K&KH and also engineering artefacts are exploding, those practices are no longer sufficient to answer challenges of nowadays complex system development. That is why there is a need to promote efficient way to transfer K&KH, in order to facilitate their circulation and reuse, and this is why current expectations are to promote models over natural language and its variability of understanding.

Research works have already been done for reuse K&KH in SE (Bollinger and Evins 2015; Barter 1998; Cloutier 2008; Cook and Schindel 2017; Gautam, Chinnam, and Singh 2007; Gzara, Rieu, and Tollenaere 2003; Haskins 2005; Korff 2013; Wang, Valerdi, and Fortune 2010), but as there are many different ways to capitalize K&KH, it is important to define the targeted perimeter or the engineering artefacts before considering reusing. Indeed, "reuse" activities in SE can be distinguished in three different approaches: *Opportunistic reuse*: when the first project was not developed with reusable capacity; *Planned reuse*: when the first project was developed with reusable capacity; *Variance*: on a product line, common core model but different options. Those approaches belong to the process of "knowledge transfer" which consists of two sub-processes defined by (Majchrzak, Cooper, and Neece 2004). On the one hand, the process by which an entity's K&KH is captured, called "knowledge sharing", and on the other hand, the process by which an entity is able to locate and to use K&KH, called "knowledge reuse". It is important to ensure that the engineering artefact on which a reuse solution is applied may be the SOI or SEA (Pfister et al. 2012).

Within existing reuse approaches for the SOI, the use of Components Off The Shelves (COTS) consists in breaking down a problem into solvable sub-problems by already existing components. However, the advantages of COTS are accompanied by integration issues, early identified by (Boehm and Abts 1999) which are: functionality and performance (what it is expected to do), interoperability (no standard exists), product evolution (risk of no longer meeting the need) and vendor behaviour (false promises). For the reuse in SEA which aim to produce the SOI, (Darimont et al. 2017) deployed a local reuse library for each engineers and a shared reuse library for improving reuse of requirements across projects, and (Majchrzak, Cooper, and Neece 2004) identifies a six-stage reuse-for-innovation process, with the capacity to quickly capture and present information on potentially reusable ideas. Other works both addressed the SOI and SEA, such as the extension of the Constructive Systems Engineering Cost Model (COSYSMO) by (Wang, Valerdi, and Fortune 2010) that consists in defining reuse categories and weights for each of the category. While COSYSMO considers the whole, the PABRE approach focuses on requirements management (Palomares, Quer, and Franch 2014) and is based on a metamodel of Software

Requirements Patterns (SRP), a method of reuse, a catalogue of 111 SRP, and a software tool that supports the management and the use of the catalogue.

As shown, many research works are looking to reuse K&KH to improve engineering efficiency. One way that looks particularly promising is achieved through the adoption of patterns, for both SEA and SOI, to systematize complex systems engineering (Cochard 2017). As they can be used in all stages of the development cycle (Gzara, Rieu, and Tollenaere 2003), reuse of patterns seems to be a very suitable form of reuse (Schindel 2005).

3 A little history of patterns

Most people in the pattern community attribute the first promoter of the value of "pattern" to (Alexander, Ishikawa, and Silverstein 1977) in a book on architecture, urban design and community liveability. They formalized a "pattern language", made of a myriad of patterns that helped them to express design in terms of relationships between the parts of a house, and the rules to transform those relationships (Coplien 1997). They began to identify patterns with the idea that “Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice” (Alexander, Ishikawa, and Silverstein 1977). The same way engineers reuse their knowledge based on their previous experience, (Cloutier 2006) point out that Alexander and his co-authors "did not invent these patterns, they came from observation and testing; and only then were they documented as patterns".

Since these pioneer works, the pattern approach has been introduced in various engineering fields such as Software, Requirements, Telecommunications and Control Systems Engineering (Cloutier 2006). (Beck and Cunningham 1987) were the first to propose object-oriented patterns in the Software community. The goal was to improve quality and to facilitate code writing by adopting good practices. (Gamma et al. 1995), also known as the "Gang of Four", wrote an authoritative book describing 23 Software Design Patterns such as Composite, Iterator, Command… A Design Pattern is a general, reusable solution to a recurring problem in the design of object-oriented applications; it describes a proven solution for solving software architecture problems. As Design Patterns are not a finished design (concrete algorithm), but a structured description of computer programming, it means they are independent from programming languages. Design Patterns have been widely accepted, and encouraged other domains to write patterns to capture their experience.

In the field of SE, the value of patterns appears towards the growing complexity of systems and the difficulty to capture large body of knowledge. That is why (Barter 1998) proposes the creation of a Systems Engineering Pattern Language, which is a collection of patterns that, when combined, address problems larger than the problems that an individual pattern can address. In the same way, (Haskins 2003) proposes the use of SE patterns to capture the information in the Systems Engineering Body of Knowledge (SEBOK). Other works have used the concept of pattern in SE, especially in the Product Information System field, where (Cauvet et al. 1998; Gzara 2000) propose a methodological framework based on the reuse of patterns during all the lifecycle, or (Conte et al. 2001) who proposed patterns libraries to support a methodological framework for the conception of product information system.

After this short history of patterns, the next section aims at improving the comprehension of what is a pattern in SE.

4 Patterns for Systems Engineering

It happens that similar designs are made independently by different engineers (Gaffar and Moha 2005). This phenomenon acknowledges the fact that the same design elements exist in multiple designs, and the study and documentation of such designs foster reuse among projects. Indeed, it prevents from "reinventing the wheel" and provides a vocabulary for the design concepts that projects can share. This is consistent with the notion that patterns "are not created from a blank page; they are *mined*" (Hanmer and Kocan 2004). It appears that SE patterns are embedded in existing designs, and that it is necessary to find a mechanism to identify them. Those patterns are called "buried patterns" by (Pfister et al. 2012) and represent a scientific issue. As the process of "Mining" appears to be essential for creating Pattern Languages, various approaches have been identified to write patterns from the element extracted from pattern mining. According to (DeLano 1998) classification, it is possible to classify mining's processes into three categories: *Individual contributions* where writers of the pattern used their own experiences or ones from their colleagues; *Second-hand contributions* where patterns are written based on interviews with experts or by guiding another person in the writing of patterns, it can also come from borrowing patterns from the literature or from companies in the same domain; *Workshops/Meeting contributions* that consists of groups of around ten people who brainstorm the elements of a patterns, along with a moderator and a facilitator.

When mining a pattern, depending on the language used (textual or modelling), it appears that a minimal set of information is always provided, as a pattern seems to possess an inherent triptych composed of {Context, Problem, Solution}. (Gaffar and Moha 2005) define a "Minimal Triangle" that defines the core meaning of a

pattern (Fig. 1). It summarizes the idea that a pattern provides a solution to a recurring problem in a particular context. However, a general consensus enlarges the minimal elements needed in a pattern, (Barter 1998) describe a generic pattern with the minimal elements needed to be written (Fig. 2). (Cloutier and Verma 2007) conduct a survey that allow them to list a recommended Systems Pattern Form. They also underline the fact that concepts used in Systems Engineering represent higher levels of complexity and abstraction that the prevailing notions of Alexander in architecture. For instance, the architecture of the underlying concepts of control-command requires a more complex notation than the sketch used in (Alexander, Ishikawa, and Silverstein 1977), thus (Pfister et al. 2012) used the Enhanced Functional Flow Block Diagram (eFFBD) to represent the model of their control-command and rely on formal conceptual foundations in the form of a meta-model.

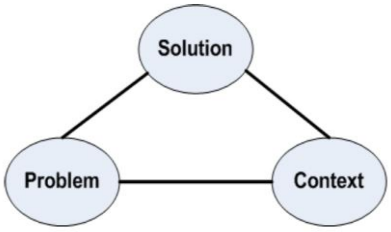

Fig. 1 The Minimal Triangle, extracted from (Gaffar and Moha 2005)

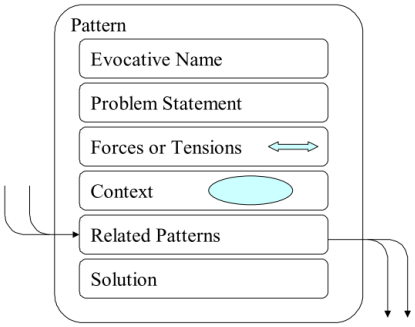

Fig. 2 Generic Pattern, extracted from (Barter 1998)

Like models, patterns are abstractions or a set of abstractions of the reality and not a magical solution. They allow people to solve complex problems by leveraging experience, K&KH from their predecessors. The results of a study conducted, in the Open Source Software community, by (Hahsler 2005) show that the larger the team size was, the greater the use of patterns was for documenting changes: from 11.4% for a unique developer to 82.2% in a team of ten or more developer. The capacity of patterns to deliver at each level of the development the correct amount of information for the stage it is applied, allow its quick adoption and most importantly its active use as Hahsler concludes in his study: "design patterns are adopted for documenting changes and thus for communication in practice by many of the most active open source developers". Patterns offer the possibility to create a common lexicon between systems architects that foster a common understanding of systems architecture, validated by experts. In this way, the experience acquired by the software community on pattern will be valuable, and help systems engineers to walk in their footsteps in order to develop patterns that will foster reuse, as well as helping control the complexity of a system.

As the interest for MBSE increases, it is important to also examine the work done for integrating the concept of pattern in this framework. The integration of the OMG System Modelling Language (OMG SysML) and its consequences on how to define problems and describe solutions are particularly interesting and will be examined in the next section.

5 Patterns for Model-Based Systems Engineering (MBSE)

Although research works have been made to assess whether the concept of pattern can be applied in the Systems Engineering field such as (Pfister et al. 2012; Cloutier 2006; Haskins 2005), the value of patterns in a MBSE framework has not been fully explored. Yet, it appears crucial to consider all the different needs, requirements and constraints of the different stakeholders in the early design stages. Perceived by many companies as a time loss, 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 (Shani and Broodney 2015). In this way, the capitalization and reuse of system models through the concept of pattern can be implemented in MBSE, and thus, favour its adoption at a larger scale.

Models are abstraction or a set of abstractions of the reality (i.e. the reality can be represented under different consistent views), which means that it can be easy to reuse a model in a new project since no physical limitations get in the way. However, depending on the type of reuse to do, the complexity of the system under design, and also the heterogeneity of methodologies and tools, it appears that the adoption of MBSE is penalized. Indeed, reusing existing modelling artefacts (even if their designs have been made to be reusable) is harder than expected. As (Korff 2013) stated, the "biggest problem is to transfer and manage the knowledge [of] what is actually available for re-use". He emphasizes on the fact that it is necessary for system engineers to be aware of system assets that can be defined and propagated among teams designing complex systems. However, the creation of assets library is not sufficient, as the purpose is to allow engineers to reuse those assets in their ongoing projects. Korff underlines the fact that users should have the possibility to search, publish, and reuse assets in defined libraries and catalogues, without any specific technical pre-requisite. Contrary to (Korff 2013), (Paydar and Kahani 2015) do not focus on the creation of assets but propose an approach concerning the adaptation of promising reusable assets during a model reuse process, especially on the adaptation of OMG Unified Modeling Language (OMG UML) activity diagrams to new use cases, in the context of web engineering. This work proposes to semi-automatically create an activity diagram from existing activity diagrams according to the input use case diagram. Even though this approach is not presented in a MBSE framework, the fact that between OMG UML and OMG SysML, use case diagrams are identical

and that activity diagrams presents the same use, allows considering a transposition in the SE field.

In the case of variant modelling in MBSE, (Oster et al. 2016) propose an approach for building and exploiting composable architectures to the design and development of a product line of complex systems in the aerospace and defence market. They choose OMG SysML as the core language to define descriptive models of the composable system reference architecture and extended it to define parametric models. This methodology allowed the product line to evolve more readily as the impact of information propagation of adding, updating or modifying new components was automatic. As their works consider physical layer, (Di Maio et al. 2016) focus their attention on the development of a functional architectures that can accommodate to change due to decisions made in the logical layer for System of Systems (SoS). The results of their study are a MBSE process that consists in the integration of a system model before the consideration of the variants. It requires that the system model should contain both the original configuration and the variant one. This separation is important in case of a new technology is introduced but the older one are not abandoned yet. They also investigate the aspects of including variant modelling into the OMG SysML, with a focus on extending an existing and operating model to support a new variant in the case where a similar technology is used.

The introduction of a reuse capacity in MBSE frameworks has proven to improve engineering efficiency in engineers work. However, the steep learning curve induced for organizations to adopt MBSE methodologies, results in the need of helping the engineers to "quickly identify not only valid architectural solutions, but optimal value solutions for the mission need" (Oster et al. 2016). Thus, it appears that the concept of patterns could be an answer to this challenge. Indeed, works have been done to introduce patterns during various phases of the engineering cycles. (Gasser 2012) described behavioural construct patterns (Fig. 3) to facilitate and systematize the modelling of system behaviour. Instead of thinking at the level of atomic graphical elements, he defined a structured way to represent elementary behavioural constructs. In this way, he advocates the use of an "insert policy", like in the construction of Functional Flow Block Diagram (FFBD) where the resizing of the diagram is automatic when new elements are inserted. The proposed behavioural construct patterns will allow engineers to work in an algorithmic way of thinking, which implies a higher modelling level that will permit to focus more on the expected behaviour than on the aesthetics of the diagrams.

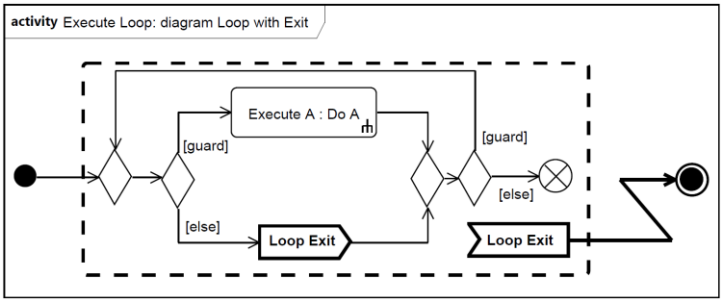

Fig. 3 Loop Exit Construct, extracted from (Gasser 2012)

In order to help engineers to focus on what is important, patterns should guide the development to avoid deviation. For example, (Barbieri et al. 2014) proposed a process for the development of mechatronic systems based on a SysML design pattern. Their intent is to demonstrate that adequate guidelines for modelling can benefits the development process 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.

Pursuing the work of (Haskins 2005) on patterns, (Schindel 2005) proposed an engineering paradigm where patterns are re-usable models, that enables what he calls Pattern-Based Systems Engineering (PBSE), where patterns can be configured or specialized into product lines or into product systems. With the advent of MBSE, this modelling framework has led to the creation of an INCOSE working group called MBSE Patterns[2]. In this context, (Schindel and Peterson 2013) developed their approach, they see "patterns as re-usable models" and apply them 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. (Cook and Schindel 2017) applies it for the Verification and Validation processes, and (Bradley, Hughes, and Schindel 2010) in the pharmaceutical market.

6 Conclusion

As presented in the introduction, a main issue for system engineers is to shorten engineering cycle period, and MBSE appears to be a great candidate to face this challenge. For a wider MBSE adoption, this paper highlights the strong

[2] http://www.omgwiki.org/MBSE/doku.php?id=mbse:patterns:patterns (visited on 31/05/2018)

methodological need to capitalize on previous projects to reuse K&KH, and focuses on the concept of pattern, which offers the possibility to make information dynamic between stakeholders during the development of complex systems, in order to share it and foster its reuse for future MBSE projects.

From a methodological perspective, improvements from processes, methods and tools should be made. It appears that the act of capitalization is not self-evident, as patterns need to be mined, and imply the ability to detect and bring out K&KH. A first step is to evaluate the maturity of such capitalized patterns, as done in the automated production systems domain by (Vogel-Heuser et al. 2018) on the maturity on control modules in libraries. A second step is to improve the general maturity of reuse approaches as done in the software domain by (Manzoni and Price 2003), using for example metrics inspired by Capability Maturity Model. A next step to improve engineering effectiveness concerns the development and the adoption of MBSE software tools that integrate patterns libraries supporting their capitalization, selection, reuse, and update.

References

Alexander, C, S Ishikawa, and M Silverstein. 1977. *A Pattern Language*. *Ch. Alexander*. doi:10.2307/1574526.

Barbieri, G., Konstantin Kernschmidt, C. Fantuzzi, and Birgit Vogel-Heuser. 2014. "A SysML Based Design Pattern for the High-Level Development of Mechatronic Systems to Enhance Re-Usability." In *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 19:3431–37. IFAC. doi:10.3182/20140824-6-ZA-1003.00615.

Barter, Robert H. 1998. "A Systems Engineering Pattern Language." *Incose*, 350–53.

Beck, Kent, and Ward Cunningham. 1987. "Using Pattern Languages for Object-Oriented Programs." *OOPSLA-87 Workshop on the Specification and Design for Object-Oriented Programming*.

Boehm, B., and C. Abts. 1999. "COTS Integration: Plug and Pray?" *Computer* 32 (1): 135–38. doi:10.1109/2.738311.

Bollinger, L. Andrew, and Ralph Evins. 2015. "Facilitating Model Reuse and Integration in an Urban Energy Simulation Platform." *Procedia Computer Science* 51 (1). Elsevier Masson SAS: 2127–36. doi:10.1016/j.procs.2015.05.484.

Bradley, Jeffrey L., Mark T. Hughes, and William Schindel. 2010. "Optimizing Delivery of Global Pharmaceutical Packaging Solutions, Using Systems Engineering Patterns." *20th Annual International Symposium of the International Council on Systems Engineering, INCOSE 2010* 3: 2441–47. doi:10.1002/j.2334-5837.2010.tb01175.x.

Cauvet, Corine, Dominique Rieu, Bernard Espinasse, Jean-Pierre Giraudin, and Michel Tollenaere. 1998. "Ingénierie Des Systèmes d'information Produit : Une Approche Méthodologique Centrée Réutilisation de Patrons." *Inforsid*, 71–90. http://dblp.uni-trier.de/db/conf/inforsid/inforsid1998.html#CauvetREGT98.

Cloutier, Robert J. 2006. "Applicability of Patterns to Architecting Complex Systems," no. May 2006: 466.

———. 2008. "Model Driven Architecture for Systems Engineering." *Language*, no. September. http://personal.stevens.edu/~pkorfiat/CONOPS/Research/1_018.pdf.

Cloutier, Robert J., and Dinesh Verma. 2007. "Applying the Concept of Patterns to Systems Architecture." *Systems Engineering* 10 (2): 138–54. doi:10.1002/sys.20066.

Cochard, Thomas. 2017. "Contribution à La Génération de Séquences Pour La Conduite de Systèmes Complexes Critiques."

Conte, Agnès, Mounia Fredj, Jean-Pierre Giraudin, and Dominique Rieu. 2001. "P-Sigma : Un Formalisme Pour Une Représentation Unifiée de Patrons." *XIXème Congrès INFORSID*, no. January: 67–86. http://liris.cnrs.fr/inforsid/sites/default/files/a366c1YfHw5cvgN2I.pdf.

Cook, David, and William Schindel. 2017. "Utilizing Mbse Patterns To Accelerate System Verification." *Insight* 20 (1): 32–41. doi:10.1002/inst.12142.

Coplien, James O. 1997. "Idioms and Patterns as Architectural Literature." *IEEE Software* 14 (1): 36–42. doi:10.1109/52.566426.

Darimont, Robert, Wei Zhao, Christophe Ponsard, and Arnaud Michot. 2017. "Deploying a Template and Pattern Library for Improved Reuse of Requirements Across Projects." *Proceedings - 2017 IEEE 25th International Requirements Engineering Conference, RE 2017*, 456–57. doi:10.1109/RE.2017.44.

DeLano, David E. 1998. "Patterns Mining." In *The Pattern Handbook: Techniques, Strategies, and Applications*, 87–96.

Demian, Peter, and Renate Fruchter. 2006. "An Ethnographic Study of Design Knowledge Reuse in the Architecture, Engineering, and Construction Industry." *Research in Engineering Design* 16 (4): 184–95. doi:10.1007/s00163-006-0010-x.

Estefan, Jeff A. 2008. "Survey of Model-Based Systems Engineering (MBSE) Methodologies." *INCOSE MBSE Initiative*. doi:10.1109/35.295942.

Gaffar, Ashraf, and Naouel Moha. 2005. "Semantics of a Pattern System." *Proceedings of the STEP International Workshop on Design Pattern Theory and Practice IWDPTP05*.

Gamma, Erich, Richard Helm, Ralph Johnson, and John Vlissides. 1995. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Longman Publishing Co., Inc.

Gasser, L. 2012. "Structuring Activity Diagrams." In *14th IFAC Symposium on Information Control Problems in Manufacturing, Bucharest, Romania*. IFAC. doi:10.3182/20120523-3-RO-2023.00153.

Gautam, Naveen, Ratna Babu Chinnam, and Nanua Singh. 2007. "Design Reuse Framework : A Perspective for Lean Development." *International Journal of Product Development* 4 (5): 485–507. doi:10.1504/IJPD.2007.013044.

Gzara, Lilia. 2000. "Résumé - Les Patterns Pour l'Ingénierie Des Systèmes d'Informations Produit."

Gzara, Lilia, Dominique Rieu, and Michel Tollenaere. 2003. "Product Information Systems Engineering: An Approach for Building Product Models by Reuse of Patterns." *Robotics and Computer-Integrated Manufacturing* 19 (3): 239–61. doi:10.1016/S0736-5845(03)00028-0.

Hahsler, Michael. 2005. "A Quantitative Study of the Adoption of Design Patterns by Open Source Software Developers." In *Free/Open Source Software Development*, 103–24. Igi Global.

Hanmer, Robert S., and Kristin F. Kocan. 2004. "Documenting Architectures with Patterns." *Bell Labs Technical Journal* 9 (1): 143–63. doi:10.1002/bltj.20010.

Haskins, Cecilia. 2003. "1.1.2 Using Patterns to Share Best Results - A Proposal to Codify the SEBOK." *INCOSE International Symposium* 13 (1): 15–23. doi:10.1002/j.2334-5837.2003.tb02596.x.

———. 2005. "Application of Patterns and Pattern Languages to Systems Engineering." *15th Annual International Symposium of the International Council on Systems Engineering, INCOSE 2005* 2: 1619–27.

http://www.scopus.com/inward/record.url?eid=2-s2.0-84883318751&partnerID=tZOtx3y1.

Korff, Andreas. 2013. “Re-Using Sysml System Architectures.” In *Complex Systems Design and Management - Proceedings of the 4th International Conference on Complex Systems Design and Management*, 257–66. Springer. doi:10.1007/978-3-319-02812-5-19.

Maio, M. Di, G. D. Kapos, N. Klusmann, and C. Allen. 2016. “Challenges in the Modelling of SoS Design Alternatives with MBSE.” *2016 11th Systems of Systems Engineering Conference, SoSE 2016*. doi:10.1109/SYSOSE.2016.7542937.

Majchrzak, Ann, Lynne P. Cooper, and Olivia E. Neece. 2004. “Knowledge Reuse for Innovation.” *Management Science* 50 (2): 174–88. doi:10.1287/mnsc.1030.0116.

Manzoni, Lisandra V., and Roberto T. Price. 2003. “Identifying Extensions Required by RUP (Rational Unified Process) to Comply with CMM (Capability Maturity Model) Levels 2 and 3.” *IEEE Transactions on Software Engineering* 29 (2): 181–92. doi:10.1109/TSE.2003.1178058.

Miled, Achraf Ben. 2014. “Reusing Knowledge Based on Ontology and Organizational Model.” *Procedia Computer Science* 35 (C). Elsevier Masson SAS: 766–75. doi:10.1016/j.procs.2014.08.159.

Mourtzis, Dimitris, Michael Doukas, and Christos Giannoulis. 2016. “An Inference-Based Knowledge Reuse Framework for Historical Product and Production Information Retrieval.” *Procedia CIRP* 41. Elsevier B.V.: 472–77. doi:10.1016/j.procir.2015.12.026.

Oster, Christopher, Michael Kaiser, Jonathan Kruse, Jon Wade, and Rob Cloutier. 2016. “Applying Composable Architectures to the Design and Development of a Product Line of Complex Systems.” *Systems Engineering* 19 (6): 522–34. doi:10.1002/sys.21373.

Palomares, Cristina, Carme Quer, and Xavier Franch. 2014. “Requirements Reuse with the PABRE Framework.” *Requirements Engineering Magazine*.

Paydar, Samad, and Mohsen Kahani. 2015. “A Semi-Automated Approach to Adapt Activity Diagrams for New Use Cases.” *Information and Software Technology* 57 (1). Elsevier B.V.: 543–70. doi:10.1016/j.infsof.2014.06.007.

Pfister, F, Vincent Chapurlat, M Huchard, C Nebut, and J.-L. Wippler. 2012. “A Proposed Meta-Model for Formalizing Systems Engineering Knowledge, Based on Functional Architecture Patterns.” *Systems Engineering* 15 (3). doi:10.1002/sys.21204.

Schindel, William. 2005. “Requirements Statements Are Transfer Functions: An Insight from Model-Based Systems Engineering.” *INCOSE International Symposium* 15 (1): 1604–18. doi:10.1002/j.2334-5837.2005.tb00775.x.

Schindel, William, and Troy Peterson. 2013. “Introduction to Pattern-Based Systems Engineering (PBSE): Leveraging MBSE Techniques.” *INCOSE International Symposium* 23 (1): 1639. doi:10.1002/j.2334-5837.2013.tb03127.x.

Shani, Uri, and Henry Broodney. 2015. “Reuse in Model-Based Systems Engineering.” *9th Annual IEEE International Systems Conference, SysCon 2015 - Proceedings*, 77–83. doi:10.1109/SYSCON.2015.7116732.

Vogel-Heuser, Birgit, Juliane Fischer, Eva-Maria Neumann, and Sebastian Diehm. 2018. “Key Maturity Indicators for Module Libraries for PLC-Based Control Software in the Domain of Automated Production Systems.” In *16th IFAC Symposium on Information Control Problems in Manufacturing*.

Wang, Gan, Ricardo Valerdi, and Jared Fortune. 2010. “Reuse in Systems Engineering.” *IEEE Systems Journal* 4 (3): 376–84. doi:10.1109/JSYST.2010.2051748.