Systems-of-Systems Design Patterns: A Systematic Literature Review and Synthesis

Jakob Axelsson
Mälardalen University and RISE Research Institutes of Sweden
Email: jakob.axelsson@mdu.se

***Abstract*—Design patterns are an established approach for reusing knowledge about good solutions to recurring problems. Patterns can be seen as a way of describing the best practices, and have been used in many different fields, ranging from building architecture and city planning to software development. There are also scattered results relating to patterns for systems-of-systems. The purpose of this paper is to summarize and review the literature on patterns for systems-of-systems and make a synthesis of a recommended approach for the field. Specifically, the novel contributions of the paper are to propose a consolidated structure for describing individual patterns and suggest the dimensions along which a pattern catalog can be organized. The paper also summarizes the concrete patterns suggested in the existing literature and classifies them according to the recommended structure.**

***Index Terms*—System-of-systems, patterns, styles, architecture, literature review.**

I. Introduction

Systems engineering is a purposeful activity, aiming at providing improvements in a certain context through informed decisions. Efficiency is always an issue in engineering, where the effort must be balanced by the value created. One way of reducing cost is to reuse previously created assets, such as mass-produced or standardized material artifacts and immaterial assets including software libraries. In both these cases, the benefit is that the development cost is shared by many users, and the remaining task of the systems engineer becomes to integrate those assets in the best way to solve the problem at hand.

As systems become more complex, there is also a potential for reusing best practices for the integration of elements to achieve particular effects. Such reusable best practices are commonly referred to as *patterns*, and this concept was made popular by Christopher Alexander's book *A pattern language* [1]. It describes 253 different patterns for city planners and building architects, where each pattern can be seen as an abstraction that highlights certain parts of a solution and leaves other parts open for tailoring to the specific usage context.

The idea of design patterns was later picked up in software engineering, with the highly influential "Gang of four" book (so-called since it has four authors) [2]. It describes 24 patterns from object-oriented programming at a level close to the source code. At around the same time, the field of software architecture was starting to develop, and this included the idea of *architectural styles* [3].

Although the literature is not entirely consistent in definitions, a style appears to be a pattern capturing certain aspects of the overall structure of a system, whereas a pattern denotes a limited part of the architecture or a detailed design solution. The idea of capturing and reusing knowledge and experience is however the same, and the difference in terminology is mainly due to the level of abstraction on which the idea of patterns is applied. Therefore, in this paper, we will primarily use the term pattern for the general concept, regardless of the level of application.

A. Contribution and Research Questions

In this paper, we will look into how patterns can be used in the context of *systems-of-systems (SoS)*, where independent constituent systems (CS) are combined to achieve a capability that a single CS cannot provide on its own. Combining and reusing existing elements is at the heart of SoS engineering (SoSE), since many CS pre-exist. The integration activity becomes an ever-ongoing endeavor as CS come and go.

The design space of SoSE is dominated by architectural design, which includes finding appropriate structures as well as dynamics that foster the desired CS collaboration. Due to the integration focus of SoSE, it seems natural to apply patterns as a way to structure best practices in the discipline.

The literature contains many contributions related to SoS patterns. However, the knowledge is scattered, and the contribution of this paper is to give a consolidated view of SoS patterns through a systematic literature review (SLR). The following research questions are addressed:

1) How are SoS patterns described in the literature?
2) What categories of SoS patterns are used in the literature?
3) Which SoS patterns are proposed in the literature?

These questions together provide insights into how a pattern catalog can be structured so that a user could navigate it efficiently to find solutions in a given situation. It also indicates what the community considers the SoS design space to be, and what knowledge is reusable. A concrete contribution of the paper is to make a synthesis for each question and thereby provide a consolidated way of describing patterns and organizing a pattern catalog for SoS.

This research was funded by Vinnova, Sweden's Innovation Agency, under grants nos. 2019-05100 and 2021-02499.

B. Overview of Paper

The remainder of this paper is structured as follows: In the next section, the methodology for SLRs is introduced together with details of how it was applied in this study. Section III presents the findings of the research, structured according to the three research questions. Section IV discusses these findings and put them in a broader perspective. Finally, in Section V, the paper is summarized together with proposals for future research.

II. Methodology

In this section, the process of an SLR will be briefly explained, following the established guidelines for this type of research [4]. In each step, the focus is on how the research was concretely done in this study.

A. Database Search

The initial step is to search a database for literature relevant to the stated research questions. The database used was the Scopus abstract and citation database, which is provided by Elsevier, and is claimed to be the largest such database in the world. Both literature [5] and prior experience of the researcher [6] indicated that this database was likely to provide a large set of relevant papers for the topic of this study.

After some experiments and tuning, the following final search string was used:

```
TITLE-ABS-KEY(
    "systems-of-systems" AND
    ((pattern OR style) W/5
     (design OR architectur*)))
```

This search will look for terms in the title, abstract, or keywords of the publications. The first part of the search string requires that the term "systems-of-systems" is explicitly mentioned. Note that Scopus automatically extends terms so that this also covers other common ways of writing, e.g., "system-of-systems", "systems of systems", and "system of systems". The second part requires that either "pattern" or "style" appears close to (more precisely, within five words, as indicated by the W/5 operator) either "design" or "architecture" (or similar terms, such as "architectural", which is captured by the asterisk wildcard character).

In total, this search rendered 87 documents, which were imported into the Covidence tool for SLRs to support the further steps of screening and data extraction.

B. Screening and Full Paper Review

A database search broad enough to capture most of the relevant studies will also yield some irrelevant papers. The next phase was therefore to manually select those papers from the search results that are really relevant to the study. First, a screening was performed based on title and abstract, to remove papers that do not contribute to the research questions. In total, 48 papers were excluded, and typical reasons were: the paper uses patterns or style in a common language sense, rather than denoting design or architectural patterns; the paper addresses a development process where patterns come into play, but does not discuss any concrete patterns; and non-papers, such as conference proceedings volumes.

The next step was to perform a full-text review of the remaining 39 papers. In total, another 25 papers were excluded in this step, and typical reasons were: the paper is not really about SoS; it does not present any patterns; or full text was not accessible.

The remaining 14 papers were included in the study [7]–[20]. In addition, a further three relevant papers were already known by the author [21]–[23], bringing the grand total to 17.

C. Data Extraction

To extract the relevant information, a data collection form was defined, with one text field for each research question. Then, relevant parts of each paper concerning each of the questions were summarized. The data was subsequently exported into a spreadsheet, thereby providing an overview of which papers address each question, and allowing a comparison of what the papers propose. The analysis and subsequent synthesis differ somewhat for each research question, and the method used for each of them will therefore be outlined in the next section, together with a description of the results.

III. Analysis and Results

In this section, each research question is discussed in more detail, including how the extracted data was analyzed and how that analysis was synthesized into a holistic account.

A. Pattern Description

The first research question was related to how patterns are described. It was analyzed by comparing pattern descriptions provided in the literature, and clustering similar approaches. In general, the descriptions consist of informal text, structured text, and figures. Informal text, i.e., a free text description, is used by all papers in one form or the other, whereas structured text and figures are used in different ways and to various extent. The structured text and figures are discussed in more detail in this subsection, ending with a synthesis of a consolidated template for the description of SoS patterns.

1) Structured text: In addition to the informal text, some papers also introduce a structured text format, consisting of a set of subheadings that is reused in all pattern descriptions [10], [12], [15], [16], [21], [22]. This is also the presentation used for software design patterns in [2]. There is no agreement between the papers on what structure to use, but still many similarities. Some use fewer categories, and some give a much more detailed structure. Many authors roughly agree on categories but disagree on the subheading to use for them.

2) Diagrams: The informal or structured text is often combined with diagrams. Some use informal drawings without well-defined semantics [11], [20]. Others apply a formally defined modeling language, and the two options found in the literature are the general-purpose systems engineering modeling language SysML [7], [8], [14], [18], and the more specific SoS architecture description language SoSADL [10], [17]. Typically, the diagrams are used to describe the solution

TABLE I
STRUCTURE OF PATTERN DESCRIPTIONS

Subheading	Interrogative	Description	References
Name		Name of pattern	all
Description	What?	Informal overview of the pattern	all
Purpose	Why?	What can be achieved by using the pattern, e.g., providing a certain capability	[12], [15], [16], [21], [22]
Applicability	Where?	Conditions under which the pattern can be used, e.g., SoS, constellation, or CS level	[15], [16], [22]
Participants	Who?	Types of CS involved, and distribution of responsibilities	[10], [15]
Solutions	How?	*Structure:* Kind of elements and relations involved, with attributes and other details *Dynamics:* Interaction between elements, changes in structure *Rationale:* Explanation of how the solution supports the purpose *Supporting illustrations:* Ad hoc or formal diagrams	[10], [12], [15], [16], [22]
Effects	How much?	Positive and negative effects on quality attributes	[16], [22]
Examples		Illustrations of usage	[12], [15], [16], [21]

offered by the pattern, in particular representing structure and to some extent dynamics.

Using a formal modeling language has its pros and cons. For users acquainted with the language in question, it provides clarity and precision, but for users without that prior knowledge, it significantly raises the threshold for understanding the patterns. In the examples provided in the literature, the diagrams are quite rudimentary and use very little of the power of formal notations. Hence, it is questionable if this level of formality adds much value. In the end, the use of patterns is to communicate an idea of a solution that has worked before, not to provide a detailed design specification.

3) Consolidated template: In Table I, we propose a consolidated structure for pattern descriptions, based on the most common categories found in the literature. A novelty in our structure is the introduction of interrogative terms, to provide a simple and easily remembered key for pattern developers and users.

To provide clarity in the descriptions, an SoS ontology is recommended, which clearly defines key concepts used in pattern descriptions and the relations between the concepts. One proposed ontology can be found in, e.g., [24]. As an example, the participants in a pattern can be made clearer by using standardized terms for different kinds of CS and different roles. This ontology, or meta-model, should be used in both the structured text and illustrative diagrams.

B. Pattern Categorization

A catalog of patterns can potentially become extensive, as is illustrated in Alexander's work with over 250 named patterns described in a book of 1,200 pages [1]. It is thus evident that a user needs support in navigating such a resource. Several attempts at providing structures for pattern categorization exist in the literature, and the various principles behind these are discussed and evaluated in this section. Since there were relatively few papers that discuss the categorization, the analysis consisted of a simple clustering of a few recurring principles suggested. In this subsection, we will first summarize how pattern catalogs in other fields were structured, and then discuss various alternative categories distilled from the SoSE literature. Finally, a consolidated categorization is proposed.

1) Alexander and Gang of four: Alexander organizes the patterns hierarchically into three broad categories: Towns (94 patterns), Buildings (110), and Construction (49). This is further subdivided into groups of 5-10 patterns each, where the connections within a group are explained in an introductory text of the corresponding book section.

The software patterns [2] are divided into a two-dimensional structure, where the dimensions are Scope and Purpose. Since the focus is on object-oriented programming, the Scope is divided into Class and Object, and the Purpose into Creational, Structural, and Behavioral. In total, there are thus six categories and each pattern is placed into one of them.

2) Relation: In [22], a two-dimensional structure is proposed for integration-oriented patterns. One dimension is Data, subdivided into Shared and Independent, and the other is Control, which can be Hierarchical or None. The principle behind this classification is thus to focus on the dominant relation type in the pattern, which can also be seen as the purpose of integration. However, it is hard to see how this particular categorization generalizes to a broader set of SoS patterns that go beyond integration issues.

Some authors propose patterns related to self-organization, self-adaptiveness, and reconfiguration [16], [17], [20], which can be seen as focusing on structural dynamics. However, they do not propose this as a general principle of organizing the patterns catalog, and when looking in detail at the patterns proposed, they are a lot about control, and could hence fit within a relations-oriented categorization.

3) Element: Instead of focusing on dominant relations, one could focus on dominant elements. This is the principle followed in the work on mediator patterns [10], but there is no literature providing other examples of dominant elements than mediators, so it is not evident how this dimension can be extended to include more categories.

4) Capability: The mediator patterns in [10] are subdivided into the categories Communication, Conversion, and Control, which appear to be an extension of the relational categories Data and Control described above [22]. However, these can also be seen as generic capabilities, which would then constitute another possible dimension, where further categories than these three can also be imagined. (The data fusion patterns suggested by [19] could be seen as a kind of conversion.)

5) Level of abstraction: Alexander's three main categories are to some extent based on the level of abstraction, which also implies a level of scope and extent. A similar idea is proposed in [13], which uses the categories Architectural (focusing on operational models), Interaction (systems models), and Design (component models). However, the paper does not provide any concrete examples of these categories, which makes it hard to evaluate the appropriateness of the categorization.

A related structure is proposed by [8], which distinguishes between Architectural patterns (describing specific, constrained, system architectures in terms of structure and behavior) and Enabling patterns (specific constructs of modeling elements that can be combined into many applications). Architectural patterns are exemplified by service-oriented architecture, centralized, and publish-subscribe. Enabling patterns include interface and traceability. It thus seems that the architectural patterns refer to the overall structure of the SoS, and the enabling patterns are generic solutions that apply to parts of the SoS.

6) Quality attribute: Some other papers implicitly suggest possible dimensions for categorization by discussing a particular pattern. This includes patterns related to dominant quality attributes, such as testability [18] or security [15]. However, this is difficult to use as a general characterization, since a pattern may very well affect many attributes, some positively and some negatively. Therefore, we find it better to include this in the description of effects (see Table I) than as a dimension of categorization.

7) Consolidated categorization: In summary, the main dimensions proposed in the literature are: relation type; element type; capability; level of abstraction; and quality attributes. As discussed above, many of the proposed dimensions have drawbacks, such as being hard to generalize. We find that the capabilities and levels of abstraction seem to be most promising, and could serve as the axes of a two-dimensional categorization. Emphasizing capability is also in line with contemporary accounts of SoS, such as the recent standards which define SoS as a "set of systems [...] that interact to provide a unique *capability* that none of the CS can accomplish on its own" [25].

For the capabilities dimension, the categories Communication, Conversion, and Control suggested by [10] appear to be a good starting point. However, one should be aware that due to the independent nature of CS, control is not absolute. It could instead be seen as a kind of communication, where one actor explains what it would like another actor to do, and the power relation between the actors determines the likelihood of the action taking place. Related to this is also the question of incentives, which express values (monetary or other) and could be an important part of negotiations within an SoS.

For the dimension related to levels of abstraction, the proposals in the literature are only described briefly, and without concrete examples of patterns at the different levels [8], [13]. A proposal that is more in line with contemporary SoS terminology is the following three levels:

- *System-of-systems.* Patterns that describe aspects of the overall SoS architecture, such as general communication, governance, coordination, etc. This resembles the Architectural level in [8], [13].
- *Constellation.* A constellation is a temporary assembly of a subset of CS, which is formed to realize a particular SoS capability [26]. This category thus covers patterns involving several CS, but not the entire SoS. Relevant patterns would describe the dynamics of constellation formation and dissolution, as well as communication, coordination, and control within the constellation. This resembles the Interaction level in [13].
- *Constituent system.* Many aspects of CS internal design are irrelevant to the SoS, but examples of useful patterns could be methods for preparing an existing system to become a CS in a particular SoS. This resembles the Design level in [13].

Some patterns are generic and applicable at several levels of abstraction or for several purposes. This resembles the enabling patterns in [8].

It should also be noted that the two dimensions of capabilities and levels of abstraction connect naturally to the proposed structure for describing patterns shown in Table I, where the provided capability can be explained as part of the purpose (Why?), and the level of abstraction as part of applicability (Where?).

C. Concrete Patterns

Many papers propose concrete patterns, and in total 38 individual patterns were extracted from the literature. Particularly rich sets of patterns were found in [10], [21]. Some of the patterns resemble each other at least partly, but an attempt to cluster the recurring ones was inconclusive since only a few patterns were the same in multiple sources. Therefore, the analysis proceeded by trying to map the proposed patterns to the categories described in the previous subsection. The result of this is shown in Table II.

It should be stressed that this is a preliminary analysis based on how the patterns are described in the sources. Some patterns map very well to the suggested categories, but in some cases, there are elements in the pattern that map to several categories (such as both communication and control, or generic concepts useful on several levels). In those cases, a somewhat subjective decision on the most suitable category was made, based on which category along each dimension appeared to be emphasized.

There is also a broad spectrum of scopes for the patterns, where some are generic and involve relatively few elements, such as the mediator-focused patterns [10]. Others are very specifically written with a certain application area in mind, such as command and control [19] or automotive [15]. To make a fully consistent pattern categorization, the descriptions of many patterns need to be aligned in the same presentation style, and some suggested patterns would need to be reformulated, possibly merged, or in other ways refined. The structure proposed in Table I would be a good starting point for such a clarification.

TABLE II
CONCRETE PATTERNS FROM LITERATURE ORGANIZED ACCORDING TO THE CONSOLIDATED CATEGORIZATION

	Capabilities		
Level of abstraction	**Communication**	**Conversion**	**Control**
System-of-systems	Publish-subscribe [12], [21] Blackboard [12], [21] Collaborator [10] Distributor [10] Router [10] Trickle-up [19] Crowd-sourced incident [16]	Service-oriented [12], [21] Pipe and filter [10], [12], [21] Supply chain [21] Adaptor / Translator [10] Aggregator / Data fusion [10] Sandwich [11]	Centralized [12], [14], [21] Infrastructure grid [21] Monitor / Collecter [10] Analyzer [10] Planner / Decider [10] Executer / Actuator [10] Local adaptations [20] Regional monitoring with local adaptations [20] Collaborative adaptations [20] Multi-scale feedback [9]
Constellation	Collaborative learning [16]		Reconfiguration control [21] Contract monitor [21] C2-zone [19] Swarm discovery and cooperation [16] Testability aiding mediator [18]
Constituent system	Separation of networks [15]	Wrapper [10] Singleton [11] Disconnected [23] Embedded [23] Remote [23] Split [23]	Contract [8] Interface [7]

IV. DISCUSSION

We will now take a step back, and discuss the usage of patterns in SoSE from different perspectives. First, we will reflect upon the quality of pattern descriptions, and on how an engineer can find the most appropriate patterns efficiently. We will also highlight some areas in which there seems to be room for more patterns than have been described so far in the literature, and evaluate the validity of this study's results.

A. Pattern Quality

The patterns proposed in the literature are of quite different nature, some being specific to a certain type of application, and others being generic or very simple. This raises questions about what constitutes a good pattern.

The essence of a pattern is to convey an understanding of a design idea, so the descriptions must be clear and accessible, which makes the motivation for formal modeling notations weak. The patterns need to be non-trivial, otherwise, the engineer could easily reinvent them for the concrete problem at hand. At the same time, they should not be so detailed that they become hard to understand. It must also be possible to recognize and apply them in a concrete setting, which means that they should not be too abstract, but if they are too concrete, they become less general. Hence, there are delicate trade-offs to be made when designing a set of patterns.

B. Navigating the Patterns Catalog

In the SoS literature, almost 40 patterns have already been proposed, and it can be expected that this number would increase drastically if more systematic efforts were made in collecting patterns. Therefore, finding good support for engineers to navigate the patterns catalog is essential. In this paper, we have therefore proposed a two-dimensional structure, which is the approach taken in previous work [2].

However, one could also consider the pattern catalog to be an information system, rather than a booklet, and this opens up new possibilities for navigation. It would be possible to use database searches along several dimensions and include support for trade-offs between different quality attributes.

Eventually, integrating the patterns database with tools for model-based systems engineering would make the search for and application of patterns even more efficient. This further increases the demand for well-described patterns.

C. Missing Patterns

Table II reveals that the literature has focused on certain types of patterns, whereas other categories have received far less attention. In particular, there are more patterns on the SoS level than on the lower levels, and particularly few related to communication and conversion in constellations. However, one should be aware that depending on the nature of the SoS and its constellations, some of the patterns on the SoS level may very well be applicable also on the constellation level, but on a smaller scale, so this picture is a bit deceptive. There is also only one pattern for communication within a CS, and this is not too surprising since this is mostly an internal matter for CS designers and with marginal interest to SoS researchers.

The proposed patterns also reveal that some aspects of the SoSE design space have received much more attention than others. The proposed patterns are mainly technical and information oriented. Areas that are completely missing include "soft" aspects such as how to organize SoS governance, or how to provide incentives for CS to act in the interest of the overall SoS to create a viable SoS business model. Other lifecycles than the operational phase would also deserve further study.

D. Validity

The validity of research addresses to what extent the results can be trusted. In SLRs, one question is always whether all the

relevant literature has been found. In this case, we choose to use Scopus, as it is the broadest available literature database, but it still does not cover all publications, and we did have to add a few papers that were known beforehand. The validity of the search could be improved by either looking at additional databases or doing snowballing (i.e., trace references to and from the identified papers). However, it should be noted that a similar search as ours was performed in [10], which only resulted in four papers, so our study with 17 papers is significantly broader than previous research.

Apart from missing papers, there is also a risk of misinterpretations of the proposed patterns, which could lead to incorrect classification. This risk is in some cases quite large since many patterns are described informally and leave a lot of room for subjective interpretation.

Another aspect of validity is the quality of the primary sources. In this case, it should be noted that the included papers are primarily theoretical, and there is very little evidence from practical usage of the proposed patterns.

V. Conclusions and Future Work

This paper has presented a systematic literature review on SoS patterns, focusing on how to describe patterns; how catalogs of patterns should be structured; and what concrete patterns have been presented. The findings concerning each research question can be summarized as follows:

1) *Pattern description.* We recommend that SoS patterns are described using structured text following the template shown in Table I combined with informal figures to illustrate the solutions.
2) *Pattern categories.* We recommend a two-dimensional structure. One dimension is the capability, with the initial categories Communication, Conversion, and Control. The other dimension is the level of abstraction, with categories System-of-systems, Constellation, and Constituent system.
3) *Concrete patterns.* We were able to identify almost 40 different patterns, covering most combinations of capability and level of abstraction. However, the majority of the patterns cover the SoS level, and there is less research on specific patterns for constellations and for adapting CS.

This work could continue in several directions, some of which were indicated in Section IV above. Improving the quality of the available pattern descriptions through the application of a common format as proposed in the paper would be a big step forward. This, combined with a consolidated and well-organized catalog of patterns, would make the knowledge more available and increase the chances of the patterns being tried in practice. In that way, the validity will also be improved. An important asset for writing clearer patterns would be a common ontology of key SoS concepts.

References

[1] C. Alexander, *A pattern language: towns, buildings, construction*. Oxford university press, 1977.

[2] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley, 1995.

[3] D. E. Perry and A. L. Wolf, "Foundations for the Study of Software Architecture," *ACM SIGSOFT Software Engineering Notes*, vol. 17, no. 4, pp. 40–52, 1992.

[4] B. Kitchenham and S. Charters, "Guidelines for performing Systematic Literature reviews in Software Engineering Version 2.3," Keele University, Tech. Rep. 4ve, 2007.

[5] M. E. Falagas, E. I. Pitsouni, G. A. Malietzis, and G. Pappas, "Comparison of PubMed, Scopus, Web of Science, and Google Scholar: strengths and weaknesses." *FASEB journal*, vol. 22, no. 2, pp. 338–42, feb 2008.

[6] J. Axelsson, "A Systematic Mapping of the Research Literature on System-of-Systems Engineering," in *IEEE 10th System of Systems Engineering Conference*, 2015.

[7] J. Bryans, R. Payne, J. Holt, and S. Perry, "Semi-formal and formal interface specification for system of systems architecture," in *Annu. IEEE Int. Syst. Conf., Proc.*, 2013, pp. 612–619.

[8] J. Bryans, J. Fitzgerald, R. Payne, A. Miyazawa, and K. Kristensen, "SysML contracts for systems of systems," in *Proc. Int. Conf. Syst. Syst. Eng.*, 2014, pp. 73–78.

[9] A. Diaconescu, L. J. DI Felice, and P. Mellodge, "Multi-Scale Feedbacks for Large-Scale Coordination in Self-Systems," in *Int. Conf. Self-Adaptive and Self-Organizing Syst.*, 2019, pp. 137–142.

[10] L. Garces, F. Oquendo, and E. Y. Nakagawa, "Towards a taxonomy of software mediators for systems-of-systems," in *ACM Int. Conf. Proc. Ser.*, 2018, pp. 53–62.

[11] R. H. Hodges, M. A. Bone, R. J. Cloutier, and P. Korfiatis, "Singleton to sandwich chunking into buslets for better system development," in *Proc. Int. Conf. Syst. Syst. Eng.*, 2011, pp. 125–130.

[12] C. Ingram, R. Payne, S. Perry, J. Holt, F. O. Hansen, and L. D. Couto, "Modelling patterns for systems of systems architectures," in *Annu. IEEE Int. Syst. Conf.*, 2014, pp. 146–153.

[13] R. S. Kalawsky, D. Joannou, Y. Tian, and A. Fayoumi, "Using architecture patterns to architect and analyze systems of systems," in *Procedia Comput. Sci.*, vol. 16, 2013, pp. 283–292.

[14] R. S. Kalawsky, Y. Tian, D. Joannou, I. Sanduka, and M. Masin, "Incorporating architecture patterns in a SoS optimization framework," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 2013, pp. 1726–1731.

[15] N. Marko, A. Vasenev, and C. Striecks, "Collecting and Classifying Security and Privacy Design Patterns for Connected Vehicles: SECREDAS Approach," pp. 36–53, 2020.

[16] C. Nichols and R. Dove, "Architectural Patterns for Self-Organizing Systems-of-Systems," in *INCOSE Intl. Symp.*, 2011, pp. 851–862.

[17] F. Petitdemange, I. Borne, and J. Buisson, "Approach Based Patterns for System-of-Systems Reconfiguration," in *Proc. Int. Workshop Softw. Eng. Syst.-Syst., SESoS*, 2015, pp. 19–22.

[18] A. Riboni, L. Guglielmo, M. Orru, P. Braione, and G. Denaro, "Design for testability of ERMTS applications," in *Proc. IEEE Int. Symp. Softw. Reliab. Eng. Workshops, ISSREW*, 2019, pp. 128–136.

[19] K. J. Rothenhaus, J. B. Michael, and M.-T. Shing, "Architectural patterns and auto-fusion process for automated multisensor fusion in SOA system-of-systems," *IEEE Syst. J.*, vol. 3, pp. 304–316, 2009.

[20] D. Weyns and J. Andersson, "On the challenges of self-Adaptation in systems of systems," in *Proc. Int. Workshop Softw. Eng. Syst.-Syst., SESoS Proc.*, 2013, pp. 47–51.

[21] C. Ingram, R. Payne, and J. Fitzgerald, "Architectural Modelling Patterns for Systems of Systems," in *INCOSE Intl. Symp.*, 2015, pp. 1177–1192.

[22] R. Kazman, K. Schmid, C. B. Nielsen, and J. Klein, "Understanding patterns for system of systems integration," in *Proc. 8th Intl. Conf. on System of Systems Engineering*, 2013, pp. 141–146.

[23] J. Axelsson, J. Fröberg, and P. Eriksson, "Architecting systems-of-systems and their constituents: A case study applying Industry 4.0 in the construction domain," *Systems Engineering*, vol. 22, no. 6, pp. 455–470, nov 2019.

[24] Y. Baek, J. Song, Y. Shin, S. Park, and D. Bae, "A Meta-Model for Representing System-of-Systems Ontologies," in *6th Intl. Workshop on Software Eng. for Systems-of-Systems*, 2018, pp. 1–7.

[25] ISO/IEC/IEEE, "Standard 21841 Systems and software engineering - Taxonomies of systems-of-systems," ISO/IEC/IEEE, Tech. Rep., 2019.

[26] J. Axelsson, "A Refined Terminology on System-of-Systems Substructure and Constituent System States," in *IEEE Systems of Systems Conference*, Anchorage, Alaska, 2019, pp. 31–36.