2023 Conference on Systems Engineering Research

The Emphasis of Design Patterns in Expressing Expert Knowledge from a Technical Solution – A Framework for Continued Research

S. Russell[a]*, B. Kruse[b], R. Cloutier[c], D. Verma[d]

[a]*NASA Johnson Space Center, Houston, TX 77058, USA*
[b]*e:fs TechHub GmbH, Gaimersheim, D-85080, Germany*
[c]*University of South Alabama, Mobile AL, 36688, USA*
[d]*Stevens Institute of Technology/SERC, Hoboken NJ, 07030, USA*

Abstract

Digital Engineering is a transformative strategy that leverages an integrated model-based approach to improve communication, decision making, design understanding, and acquisition efficiency of system development. As modern systems are derived from pre-existing systems, harvesting expert knowledge from proven systems in a useful, model-based way will reduce the experiential learning and cognition required for new system development, contributing to a Digital Engineering transformation. Motivated by performance gains observed during a multi-year, sequential development activity, this survey reviews knowledge, architecture, and pattern literature to establish a framework for research of architectural methods for expert knowledge identification and description using Model Based System Engineering. The multi-year sequential development activity is offered as the experimental system of interest for this research. This work aims to enable a digital engineering strategy that improves concept phase decision making, accelerates knowledge acquisition from lessons learned repositories, and eases the burden of generational knowledge loss.

Keywords: Design Patterns; Architecture; Architectural Theory; Expert Knowledge; Energy Storage; Lithium Ion

1. Introduction

This study establishes a conceptual framework for a digital engineering enabling methodology aimed at accelerating experiential learning and improving concept development phase decision making by expressing and describing expert knowledge from an existing system. Digital engineering is strategy for improving acquisition

* Corresponding author. Tel.: +1-281-483-8721
E-mail address: samuel.p.russell@nasa.gov

integrity (1) by capitalizing on recent trends in globalization (2). Despite accelerating growth in technology and information transfer (2), the development of large, complex systems remains a skill honed by experience (3) (4) (5). By depending solely on experience, organizations face a costly burden preparing the workforce, and this burden is increased when the business supplies a dynamic marketplace or suffers generational knowledge loss. This study establishes the groundwork for research that aims to reduce the organizational learning burden and improve concept phase decision making by making expert knowledge embedded in proven systems accessible to the modern engineer.

The concept development phase of the project lifecycle is a critical stage for long term project performance. During the concept development phase, stakeholder requirements are translated into design solutions (4) and errors in this process represent 86% of reported cost impacts (6) and are unrecoverable with more than 95% confidence (7). Thus, concept phase decision making is paramount to project success. Interestingly, architectural theory (8) (9) suggests that systems are derived from debate on existing systems. Therefore, a methodology suitable for harvesting foundational decisions made in the formulation of existing systems may offer utility in framing and informing decisions during new system development activities.

The architectural theory of patterns offers such a methodology (10). Demonstrated in the field of software engineering (11), the architectural theory has been validated in the systems engineering domain as a means of describing expert knowledge (12). Operating as a heuristic, a pattern describes an observed solution in terms of the balance of relationships necessary for achieving the solution (10). As these relationships may not exist wholly within the system of interest, a complete pattern must include relationships to other patterns (10). A reference architecture has been proposed as one such description by expressing relationships between technical and business architectural descriptions in a customer context (13). In the context of a reference architecture, a technical architecture describes the solution to a problem in technology using design patterns (13). A design pattern represents the expert knowledge of design and is the subject of this study.

This study is motivated by the demonstrated performance of a design team responsible for the sequential, multi-year development of three novel energy storage solutions. Differing in shape, dimension, and energy density, the three designs were approached in order of increasing complexity and mission criticality. Each design adhered to a common approach, enabling previously proven solutions to be refined in increasingly complex solutions. The observed reductions in budget and schedule need and gains in design performance suggest the development of something more than material and procedural knowledge reuse. This work aims to identify and describe the problem-solving heuristics developed over the course of the sequential development activity and express that information in a reusable and archivable way. The architectural theory of patterns and Model Based Systems Engineering are proposed as enabling techniques for achieving this objective.

The goal of this writing is to establish a conceptual basis for research in architectural methods as a means of identifying and describing expert knowledge from existing systems for capture in model based tools. In this paper, the concepts of knowledge, architecture, and architectural theory are defined, and a framework and methodology for applying architectural theory proposed. The system of interest is introduced, and physical and anecdotal evidence are provided to communicate the development of expert knowledge and highlight the utility of existing knowledge transfer methodologies. This work concludes with research questions that guide further research.

2. Concept and Term Development, A Literature Review

2.1. Knowledge and the Role of Knowledge Management Systems

Human learning, the acquisition of knowledge, occurs in three domains: cognitive, affective, and psychomotor (14) (15). The cognitive domain incorporates knowledge recall and recognition, and intellectual ability and skill development (16) and is critically important in knowledge management (17). Rooted in Greek philosophy (18) (19) (20), the modern study of knowledge management is derived from the writings of Polanyi (17) (21) (22) and the more recent work of Nonaka (19), and Nonaka and Takeuchi (17) (20). Polanyi postulated that knowledge includes a hidden, or tacit, component that cannot be easily communicated (23) (21). Nonaka surmised that tacit information includes both cognitive and technical elements, and when combined with explicit knowledge, represents the entirety of

knowledge contained within an organization (18) (19). Applying this definition in a study of innovation organizations (20) laid the groundwork for modern organizational knowledge management systems (18) (23). Coincident with Nonaka's work, research in education and cognitive psychology acknowledged that knowledge exists at both a tacit and explicit level (24) and that knowledge is the combination of domain (declarative, procedural, and conditional (14) (25)) and specific (cognitive awareness and strategy) knowledge (24) (26) (27). Although the two definitions differ in rigor, both agree that knowledge contains both a tacit and an explicit dimension.

The concept of tacit knowledge remains a topic of debate in the field of knowledge management (17) (23). Tacit knowledge is expert knowledge that is known but difficult to describe by specification or standard (28) (29). According to Grant (17), Polanyi defined knowledge as a continuum ranging from explicit to ineffable (tacit). Adhering to Polanyi's original work, Kingston (23) proposes four categories for characterizing tacit knowledge: explicit information shared in writing or figure, symbolic experiential knowledge shared between experts, non-symbolic experiential knowledge shared by demonstration, and true tacit knowledge unknowingly possessed by the owner. Kingston's description of expert knowledge echoes that of studies in cognitive psychology by incorporating both learned 'best practice' knowledge and 'wisdom' or strategic meta-knowledge (23) (24) and suggests three principle formats describe symbolic experiential knowledge: heuristics, classification hierarchies, and pattern recognition (23). Heuristics are "rule of thumb" simplifications representing mental shortcuts (23) (30) which are used to make decisions when the problem feels familiar enough that no additional data or information is required (30). Classification hierarchies describe how the problem solver organizes the information presented within the problem statement (23). Cognitive psychology research has shown that "perceptual chunking" of information presented in a problem statement is organized according to literal aspects (declarative knowledge) by the novice and with fundamental principles (procedural knowledge with application) by the expert (23) (31) (32). Pattern recognition refers to the identification and recall of perceptual chunks of information based on familiarity with the relationships between elements within the perceived configuration (23) (32). For example, a short term memory experiment showed that expert chess players were able to recreate a layout, or configuration, of chess pieces, or elements, on a game board after brief observation when the chess pieces were arranged according to the rules of origin and movement, or relationships, established for the game of chess (32) (33). Based on these findings, expert knowledge is defined as tacit knowledge that represents best practices and strategic metaknowledge that enables application of fundamental principles and is deducible by observation when the observer is familiar with the rules and relationships operating within a system.

2.2. Architecture and the Role of Architectural Theory and Ontologies

Architecture is an abstract concept that is difficult to describe. The dictionary provides a five-part definition including: the art or science of building, the formation or construction resulting from or as if from a conscious act, a product, a method or style of building, and the way a system is organized (28). Analyzing this definition leaves the reader wondering if architecture is a system of thought, the act of conscious construction, or a structural diagram. Historical writings provide additional insight by defining architectural theory as "…any written system of architecture, whether comprehensive or partial, that is based on aesthetic categories." (8) Aesthetic pertains to a sense of beauty, or concerns emotion or sensation as compared to intellectuality (28). Aesthetic categories include the abstract notions of proportion, symmetry, balance, rhythm, and unity (34). Therefore, architecture is a study of aesthetics, and architectural theory is a system of categorizing observed aesthetics.

Applying the notion of aesthetics to system design reinforces Griffin's (5) call for elegant system design and affirms the balance of art and science described by Ryschkewitsch et.al (3). However, as systems engineering is concerned with the logical aspects of a system, a practical definition is required. Logical pertains to a study of logic, or the principles governing correct and reliable inference such as the relationship between facts (28). The NASA Systems Engineering handbook describes several categories of logical aesthetic including functional, behavioral, and temporal expressions of the observed system (4). The categorical allocation of system aesthetics aligns with the definition of architectural theory and the example provides a practical example of logical architectural expressions for a system.

However, the relationship between architecture and solution remains unclear. While the NASA handbook describes how the user progresses from architectural expression to a solution (4), architectural theory provides a more practical

example in a discussion of industry and architecture. In translated excerpts from writings on *Industriebau*, Gropius (8) noted that improvements in product quality are no longer sufficient to achieve market success, instead aesthetic values must be considered from the outset resulting in a technically excellent product permeated with intellectual content, with form, to secure a presence among a mass of similar products. In this writing, Gropius defined the relationship between technology, design, and architecture given the equivalence between technology and product, design and form rich with intellectual content, and architecture and aesthetic value. This relationship between architecture, design, and solution is preserved in architectural ontologies such as Enterprise Architecture (35), Department of Defense Architectural Framework (DoDAF) (36), NATO Architectural Framework (NAF) (37), and ISO 42010 (38). Although differing in name and convention, each ontology provides a conceptual framework for describing a system architecture. For example, Enterprise Architecture uses interrogatives and perspectives to provide a comprehensive view of the system (39). Interrogatives represent the fundamentals of communication, namely What, How, When, Who, Where, and Why (39). Perspectives represent the reification of a system from the abstract to the concrete and are organized according to business stakeholder (40). It is important to note that the different perspectives in this example do not refine adjacent perspectives with additional detail as each stakeholder holds a complete view of the system (39). The Enterprise Architecture ontology considers the business perspectives of the enterprise architect, designer, and technician and therefore enables the sought-after relationship between system, design, and solution. Additionally, the What interrogative represents a logical view of the system, enabling a logical perspective for each stakeholder, thereby satisfying the architectural framework objectives for this study.

2.3. The Architectural Theory of Patterns

In his seminal work, Alexander used diagrams to provide an abstract view of physical relationships that resolve small systems of interacting and conflicting forces that are independent of all other forces and all other possible diagrams (41) (42). Retrospectively defined as a Pattern (42), the contemporary definition of a pattern is a solution to a problem in a context (10) (12) (42). Alexander applied this concept in the holistic description of a complex system by relating individual patterns observed within a system to form of network diagram he called a Pattern Language (10). In so doing, a framework for deducing and describing system architecture was developed (12).

The notion of patterns has deep roots in studies of human problem solving. Studies on cognition use the term pattern to describe a type of heuristic, an efficient mental process, or shortcut, that reduces complex problem solving to a simpler judgmental operation (43) (44) (45) (28) by ignoring part of the information (46). The study of heuristics began a shift from mathematically rigorous to the general with the work of Rene Descartes in the 17th century and Bernard Bolzano in the 18th century to reduce the rigor of considering all conceivable solutions to a problem (45). Gaining acclaim in mathematics, heuristics were applied in the study of economics and decision making (45). Psychologists adopted heuristics to describe human mental processes of learning (47) and perception (48), and were the first to consider certain regularities in the environment, such as proximity between objects, as part of the observation and problem solving process (45). Acquiring and applying patterns has formed the basis of an educational taxonomy (14) and has been attributed to engineering problem solving (49).

Alexander applied the work of cognitive psychologists to develop Pattern Language as a means of describing the perceived organization and environmental adaptation of human dwellings, villages, towns, and cities(12) (50) (51) (52). Pattern Language combines structural hierarchy with Gestalt Theory (53) to describe the invisible parts of a system (54) by expressing design in terms of relationships and rules to transform those relationships (50) (51) (52). Bauer (53) illustrates this point through an example of a kiosk located on a street corner by noting that Pattern Language does not structurally decompose a city into discrete elements for replacement or improvement, but instead attempts to differentiate the integrated urban street corner from other city complexes based on features invisible in kiosk design such as pedestrians waiting to cross the roadway, bus schedules, magazine subscriptions, and traffic light phases, all of which make the newspaper kiosk a functional element of the city (53) (55) (56). In so doing, Pattern Language provides a practical means of describing the “wholeness” of an observed system. From the perspective of a systems engineer observing an engineered system, wholeness represents the application of expert knowledge.

Although Alexander's Pattern Language (10) enables the identification and acquisition of expert knowledge from an observed system, the review of knowledge literature identified two critical aspects that must be considered when applying architectural theory. First, barring expert knowledge, the user of a procedure or method is likely to pursue literal aspects of the methodology, as has been observed with the proliferation of pattern books in software and systems engineering. Therefore, any effort aimed at methodizing the application of architectural theory is counterproductive. Secondly, the success of the short-term memory experiment depended on pre-existing familiarity with the origin and rules of motion for the chess pieces, suggesting the success of an architectural theory methodology may be dependent on pre-existing expert knowledge. Therefore, for a pattern to be valid, it must be authored or reviewed and approved by someone possessing relevant expert knowledge.

2.4. Adoption and Use in Engineering Domains, Establishing Precedent

A review of available literature suggests engineering communities have not yet realized the complete benefit of the architectural theory of patterns. For example, Gamma et.al (11) introduced Alexander's Pattern Language to software engineering and the community embraced the notion as a means of improving code reusability and reliability, documenting and communicating efficient program solutions, and improving programming productivity (60) (61) (62). In so doing, a pattern was understood to be an important and reoccurring system construct, and pattern language as the structure guiding pattern application (52). However, it has been noted that adoption has fallen short of architectural theory as the popular pattern texts are merely collections of isolated patterns (52). Using the development of the Macintosh computer as exemplar text, Kerth (52) postulates the utility of architectural theory in the management of disperse project teams and recommends focus areas for improving the realization of architectural theory including encouraging result evaluation and feedback, expanding artifactual use to include description of the guiding philosophy, and enhancing investigative interviewing skills. Coplien (51) echoes these recommendations when reiterating that patterns and architecture are linked, and that patterns transcend the definition of objects and logical architecture to include such issues as sociology and psychology that exist far from the tangible physical structure of a system. In a writing on lean systems architecture, Coplien (63) refines the role of architectural theory in the software domain to a logical expression of system functionality and cautions the reader that imposing structure beyond abstract base class slows productivity and creates rework in software development. In the described writings by Coplien (51) and Kerth (52), architectural theory is credited with providing benefit beyond object definition, however both authors suggest the concept requires renewed promulgation if these benefits are to be realized.

The systems engineering community is at risk of following the software community to a similar end. Introduced as a system architecture methodology, emerging modeling technologies were seen as a potential accelerant for system architectural methods (12). Unfortunately, the methodology itself appears to be in question. For example, the terminology has been used in both abstract and artifactual descriptions ranging from a metamodel for a generic system (64) to the description of objects (65) including reoccurring blocks of written text (66) (67). However, the introduction of Reference Architecture offers the community a renewed focus on theoretical principles. A Reference Architecture identifies design patterns as the foundation of a technical architecture description and relates this description to the business perspectives of the enterprise (mission, vision, strategy) in a customer context (13). However, no work has been found illustrating the implementation of design patterns in developing a technical architecture description.

2.5. Implementing Design Patterns, A Methodology

A fundamental tenant of Pattern Language is that patterns are mined, not created (12) (57). Pattern mining is the process of identifying expert knowledge used in problem solving. Leitner [11] and Iba [12] proposed frameworks for the pattern-mining process. Both use a three-step process of candidate identification, candidate consolidation and refinement, and pattern documentation. The principal difference between the two methods is pre-existing knowledge of the system. Iba's methodology depends on a conversational interview process (29) while Leitner's framework requires prerequisite knowledge of the system (58). The interview method assumes the interviewer has no prior knowledge of the solution and must deduce problem solving logic through a series of conversational interviews (59).

Conversely, Leitner's pre-requisite knowledge methodology assumes pattern mining is performed by the solution designer (58). Each framework is dependent on expert knowledge, but neither describes the refinement and consolidation process.

Alexander provides pattern mining guidance beginning with observation to identify a feature worth abstracting, identifying, and refining interacting forces, and describing the interacting forces in a context that limits relevance of the observation (10). Acknowledging that patterns are without scale (10), feature identification guidance is left to the pattern miner. Once identified, the observation is refined by functional assessment (10). The refining process translates observation into a pattern that describes a dense system of internal forces having only a weak external interaction (10) (42). The pattern is not complete until the context is defined. The context serves to limit applicability of the pattern to instances where the interacting forces exist, and the solution maintains the observed force balance (10). Alexander provides an example of a Danish home that appears simultaneously "spacious" and "cozy" because the common room has alcoves on either side large enough to hold one to two family members, allowing members to partake in separate hobbies without cluttering the common area nor isolating themselves from one another (10). This example translates a general observation into a pattern refined by functional assessment, defines the interacting forces balanced by the pattern, and provides the context where the solution remains relevant. The process of pattern mining refines observation by functional assessment to describe a system of dense internal and weak external force interaction within a limiting context.

Adopting communities may be tempted to script the process of creating patterns. However, Alexander (42) deeply opposes this, suggesting "…methods result in the creation of diagrams, not on the diagrams themselves…". Interestingly, the software community is one such example of scripting pattern expression. Doing so has enabled code reuse (11) and system design and inspection methodologies (68), but a focus on object and procedure has limited the architectural benefit (51) (52) (69). Therefore, works on pattern use should refrain from adopting a procedural focus and instead rely on an existing methodology.

2.6. Synopsis and Research Hypothesis

Systems Engineering is the art and science of developing an operable system that meets requirements within imposed constraints, and as a discipline, is a core competency in technical organizations (3). Responsible for translating stakeholder need to an operable system, the systems engineering discipline has been held accountable for a lack of elegance in design (5) and costly rework caused by incorrect design decisions made during the concept development phase (6). The community has responded by developing experiential training programs to accelerate the growth of expert knowledge (3) and producing comprehensive texts for guiding systems development (4). However, the individual focus of workforce development programs, the 5-fold cost impact of concept phase decision error (6), and the observation that cost overruns exceeding 15% at preliminary design complete are unrecoverable (7) warrant additional corrections.

Based on the observation of operable systems, architectural theory is a system of writing that categorizes the aesthetics of a system (8), including the logical arguments of interest to systems engineering including the functional, temporal, and behavioral expressions of a system (4) (8). The architectural theory of patterns is one such theory. A pattern describes the observable aesthetics of a system based on the functional assessment of forces balanced by the refined solution within a context where the forces exist, and the force balance is preserved (41). Documented as an object-based diagram (10) (12), the force balances (solutions) are shown as objects, and relationships between objects shown as lines (29) (58). By structuring the diagram according to perspectives of the architectural framework, a hierarchical network diagram can be produced. The object-based pattern is suited for capture in an object-oriented modeling tool such as SysML (70), a Model Based Systems Engineering language. Therefore, by demonstrating the utility of architectural theory in harvesting expert knowledge from existing systems and communicating this knowledge in an object-oriented system modeling language, the Systems Engineering domain gains the ability of describing and capturing expert knowledge from systems with demonstrated operability and performance. By archiving, sharing, and accessing this information repository during new development activities, future system

engineers will incorporate the combined benefit of modern technology and historically proven design decisions, thereby reducing costly rework due to system concept phase error, and implementing the digital engineering strategy.

3. A Knowledge Development Experiment

The development of three energy storage devices represents the system of interest for a study of expert knowledge using architectural principles. Each of the three devices uses a commercial Lithium Ion technology to convert electrical to chemical energy for storage and use as part of NASA's human spaceflight program. Designed to accommodate different energy, mass, and volume constraints, the three batteries use a common electrochemical cell and cell packaging schema to enable operation over the life of the powered system. Derived from a heritage approach to battery design (71), the three devices incorporate novel advancements that prevent single cell thermal runaway from propagating to other cells in the design or the release of flame or spark to the external system. Thermal runaway occurs when an internal short circuit results in rapid heat accumulation due to uncontrolled discharge of stored energy resulting in the thermolytic decomposition of cell construction materials and the generation of both cell body heat and combustion products, or ejecta, released to the internal environment of the battery (72) (73). With no prior experience in mitigating thermal runaway, each battery development was performed sequentially using a design-build-test process to enable incremental knowledge development and procedure and material reuse. Each of the three batteries are operational in low earth orbit.

Adapting the heritage design (71) to accommodate a single cell catastrophic failure event required mechanistic discovery as field failure rarely allows diagnostic study (74). Beginning with the development of a reliable failure initiation method and a rudimentary understanding of the failure scenario, a small group of government and commercial experts used an iterative and recursive design, build, and test approach to develop a functional prototype (75). Once the prototype was seen to perform successfully, the design was refined to ensure specification and standard compliance before validation and verification testing. The battery and project performance metrics of Fig. 1 indicate knowledge generation in the form of design improvement (adjacent cell temperature reduction) and work effort (cost and schedule reductions) for increasingly complex battery design (increasing energy density). Although each battery required a unique approach to preventing failure propagation due to differences in mass, volume, and energy requirements, the sequential nature of the development activity enabled continual refinement and adaptation of geometry, materials, and energy management methods to deliver increasingly energy dense and higher performance batteries at increasing lower development cost and schedule.

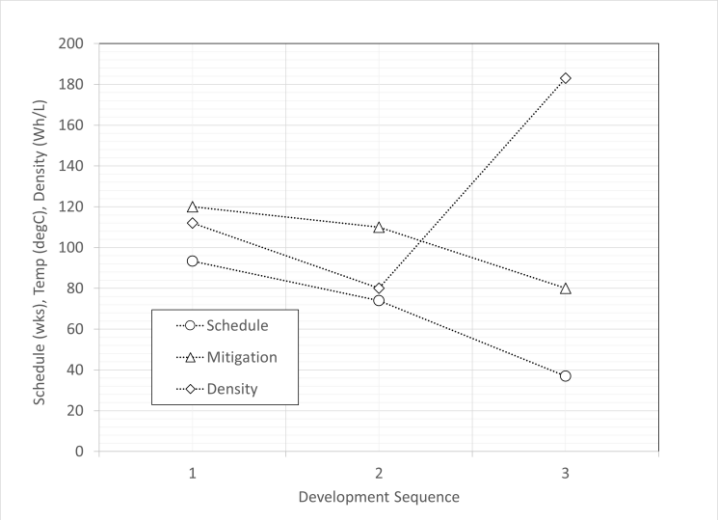

Fig. 1. Performance Trends over Three Sequential Development Activities

Lessons learned during the development activity have been described as guidelines or rules for achieving similar performance in multicell, lithium-ion batteries (76) (77). Sharing the guidelines with two partner organizations of differing design experience provided a practical opportunity for assessing the effectiveness of rule-based knowledge transfer. While one organization had pioneered the use of commercial lithium ion in multicell aerospace batteries, the other had no prior experience in battery design but a rich history in spacecraft design. With access to the same written materials and personnel, the two organizations independently developed an energy storage solution. The organization experienced in battery design delivered a functional device that achieved the failure propagation prevention with little to no rework, while the organization with less experience required an additional design cycle to correct issues not related to the failure propagation prevention guidelines. Although the evidence is anecdotal, the result highlights a potential weakness of rule-based knowledge capture and transfer systems. Without a comprehensive description or experiential knowledge-based context, rule-based knowledge transfer is incomplete. The goal of this work is to demonstrate and validate a knowledge harvesting and description technique that does not depend on physical description to enable a knowledge-based interpretation of rules, guidelines, and lessons learned.

4. A Framework for Continued Research

This writing provides a conceptual description of expert knowledge, architecture, and architectural theory; explores the precedence of adopting architectural theory in an engineering domain; and, introduces a practical knowledge generation experiment. The literature explored in this work exposed both a great need for describing expert knowledge, and a notable trap for the application of the architectural theory of patterns in an engineering domain. Motivated by a desire for legacy planning and a goal of accelerating knowledge transfer in a learning organization, continued research in expressing expert knowledge must abide by theoretical principles if the postulated benefits are to be realized. Furthermore, as exemplar descriptions of compliant knowledge capture exercises were not found, so to documentation methods for object-oriented modeling of compliant expressions were not identified. To avoid the physical expression bias of preceding domain adoption, enable the recommended (4) (78) but often overlooked (79) logical expression, and enable both archival and reuse of knowledge expressions rich in contextual description, new methods will be required. The following questions are postulated as a framework for continued research in expressing expert knowledge with architectural theory:

- How are regions of dense interacting forces described from physical composition (engineering drawing)?
- How does the pattern miner ensure the pattern is useful without the benefit of expert knowledge?
- Is the pattern inclusive of each device of the experimental system even though external constraints may differ?
- Does the Enterprise Architecture provide a suitable framework for the pattern mining process?
- Is the description extensible to archival techniques such as a pattern library or a Reference Architecture?
- Does the pattern contain metadata suitable for sorting or searching in a pattern library?
- How can the pattern be used to advance knowledge / inform future decision?

5. Conclusions

Inspired by observed performance gains in battery and team performance during the development of an increasingly complex novel energy storage system design, this work explores available literature in the areas of knowledge, architecture, and architectural theory to define concepts, identify the role of an architectural framework, and identify an expert knowledge expression methodology. The literature review identified the role of design patterns in technical architecture, but did identify practical or demonstrative examples of adoption in the systems engineering domain nor identify exemplar documentation methodologies. A framework for continued research aimed at filling the identified void is provided in the form of several research questions.

References

1. Office of the Deputy Assistant Secretary of Defense for Systems Engineering. *Department of Defense Digitial Engineering Strategy.* Washington DC : Department of Defense, June 2018.
2. *Globalization and Technology: How Will They Change Society? .* Chareonwongsak, K. 2002, Technology in Society, pp. 191-206.
3. Ryschkewitsch, Michael, Schaible, Dawn and Larson, Wiley. *The Art and Science of Systems Engineering.* Washington DC. : National Aeronauctics and Space Administration, Jan 18, 2009.
4. Office of the Chief Engineer. *Systems Engineering Handbook (NASA/SP-2016-6105 rev 2).* Washington D.C. : National Aeronautics and Space Administration, 2016. NASA/SP-2007-6105 Rev 1.
5. *How Do We Fix Systems Engineering?* Griffin, Michael D. Prague : 61st International Astronautical Congress, 2010. IAC-10.D1.5.4.
6. *A comparison of design decisions made early and late in development.* Tan, James, Otto, Kevin and Wood, Kristin. Vancouver, Canada : University of British Columbia, 2017. ICED 17: 21st Internation Conference on Engineering Design.
7. Christensen, David S. An Analysis of Cost Overruns on Defense Acquisition Contracts. [ed.] Francis T. Hoban, William M. Lawbaugh and Edward J. Hoffman. *Readings in Program Control.* Washington DC : National Aeronautics and Space Administration Science and Technical Information Office, 1994.
8. Kruft, Hanno-Walter. *A history of architectural theory : from Vitruvius to the present.* [trans.] Antony Wood, Elsie Callander and Ronald Taylor. New York : Princeton Architectural Press, 1994.
9. *Systems Engineering and the "Two Cultures" of Engineering.* Griffin, Michael D. 3, Third Quarter 2007, IEEE Engineering Management Review, Vol. 35, pp. 44-44.
10. Alexander, Christopher. *A Pattern Language.* New York : Oxford University Press, 1977.
11. Gamma, Erich, et al. *Design Patterns, Elements of Reusable Object-Oriented Software.* Boston : Addison-Wesley, 1995.
12. Cloutier, Robert J. and Verma, Dinesh. Applying the Concept of Patterns to Systems Architecture. *Systems Engineering.* 2007, Vol. 10, 2, pp. 138-154.
13. Cloutier, R., et al. The Concept of Reference Architectures. *Systems Engineering.* 2010, Vol. 13, 1, pp. 14-27.
14. Bloom, Benjamin S. (Ed). *Taxonomy of Educational Objectives, Book 1 Cognitive Domain.* White Plains : Longman New York, 1956.
15. Anderson, Lorin W. and Krathwohl, David R. *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives.* s.l. : Pearson, 2000.
16. Anderson, Patricia. Bloom's Taxonomy. *Center for Teaching.* [Online] Vanderbilt University. [Cited: 2 February 2020.] https://cft.vanderbilt.edu/guides-sub-pages/blooms-taxonomy/.
17. Grant, Kenneth A. Tacit Knowledge Revisited – We Can Still Learn from Polanyi. *The Electronic Journal of Knowledge Management.* 2007, Vol. 5, 2, pp. 173-180.
18. *Review: Knowledge Management and Knowledge Management Systems: Conceptual Foundations and Research Issues.* Alavi, Maryam and Leidner, Dorothy E. 1, March 2001, MIS Quarterly, Vol. 25, pp. 107-135.
19. Nonaka, Ikujiro. A Dynamic Theory of Organizational Knowledge Creation. *Organization Science.* 1994, Vol. 5, 1, pp. 14-37.
20. Nonaka, Ikujiro and Takeuchi, Hirotaka. *The Knowledge Creating Company, How Japanese Companies Create the Dynamics of Innovation.* Oxford : Oxford University Press, 1995.
21. Polanyi, Michael. *The Tacit Dimension.* Chicago : Univeristy of Chicago Press, 1966.
22. —. Personal Knowledge. [book auth.] Michael Polanyi and Harry (Eds) Prosch. *Meaning.* Chicago : Univeristy of Chicago Press, 1975, pp. 22-45.
23. Kingston, John K.C. Tacit Knowledge: Capture, Sharing, And Unwritten Assumptions. *Journal of Knowledge Management Practice.* 2012, Vol. 13, 3.
24. Alexander, Patricia A. and Judy, Judith E. The Interaction of Domain-Specific and Strategic Knowledge in Academic Performance. *Review of Educational Research.* 1988, Vol. 58, 4, pp. 375-404.
25. Paris, Scott G., Lipson, Marjorie Y. and Wixson, Karen K. Becoming a Strategic Reader. *Contempory Educational Psychology.* 1983, Vol. 8, 3, pp. 293-316.
26. *Knowledge Development and Memory Performance.* Chi, Michelene T. H. [ed.] M.P. Friedman, J.P. Das and N. O'Connor. Boston MA : Springer, 1981. NATO Conference Series (III Human Factors). Vol. 14.
27. Greeno, J. G. Trends in the Theory of Knowledge for Problem Solving. [book auth.] D. T. Tuma and F. Reif. *Problem Solving and Education: Issues in Teaching and Research.* Hillsdale N.J. : Erlbaum, 1980.
28. Random House Living Dictionary Project. *Webster's College Dictionary.* New York : Random House, 1991.
29. *Pattern Language 3.0 and Fundamental Behavioral Properties.* Iba, Takashi. Krems an der Donau, Austria : Creative Commons, 2016. In Pursuit of Pattern Languages for Societal Change (PURPLSOC). pp. 200-233.
30. Korteling, Johan E., Brouwer, Anne-Marie and Toet, Alexander. A Neural Network Framework for Cognitive Bias. *Frontiers in Psychology.* 2018, Vol. 9, 1561.
31. Chi, Michelene T. H. , Feltovich, Paul J. and Glaser, Robert. Categorization and Representation of Physics Problems by Experts and Novices. *Cognitive Science.* 1981, Vol. 5, pp. 121-152.
32. Chase, William G. and Simon, Herbert A. Perception in Chess. *Cognitive Psychology.* 1973, Vol. 4, pp. 55-81.
33. Groot, Adriaan D. De. *Thought and Choice in Chess.* New York : Ishi Press International, 1965.
34. Vinchu, Ar. Gourav Nandkishor, Jirge, Neela and Deshpande, Ar. Archana . Application of Aesthetics in Architecture and Design. *International Journal of Engineering Research and Technology.* 2017, Vol. 10, 1.
35. Zachman, J. A. A Framework for Information Systems Architecture. *IBM Systems Journal.* 1987, Vol. 26, 3, pp. 276-292.
36. Chief Information Office. The DoDAF Architecture Framework Version 2.02. *DODAF Home.* [Online] U.S. Department of Defense, August 2010. [Cited: 23 June 2019.] https://dodcio.defense.gov/Library/DoD-Architecture-Framework/.
37. Architecture Capability Team. *NATO Architectural Framework Version 4.* s.l. : NATO, 2018.

38. International Organization for Standardization. *Systems and software engineering - Architecture Description.* Geneva : ISO copyright office, 2011. ISO/IEC/IEEE 42011:2011.
39. Lapalme, James, et al. Exploring the future of enterprise architecture: A Zachman perspective. *Computers In Industry.* 2016, Vol. 79, June.
40. Zachman, John A. *The Concise Definition of The Zachman Framework by: John A. Zachman.* [webpage] s.l. : Zachman International Inc., 2008.
41. Alexander, C. *Notes on the Synthesis of Form.* Cambridge MA : Harvard University Press, 1964.
42. *Undiscovered Patterns.* West, D. and Quillien, J. Pittsburgh, PA : The Hillside Group, 2015. Proceedings of the 22nd Conference on Pattern Languages of Programs.
43. Todd, P.M. Heuristics for Decision and Choice. [book auth.] Neil J. Smelser and Paul B. Baltes. *International Encyclopedia of the Social & Behavioral Sciences.* s.l. : Elsevier Ltd, 2001, pp. 6676-6679.
44. Tversky, Amos and Kahneman, Daniel. Judgment under Uncertainty: Heuristics and Biases. *Science.* 1974, Vol. 185, 4197, pp. 1124-1131.
45. Hertwig, Ralph and Pachur, Thorsten. Heuristics, History of. *Psychology.* 2015.
46. Gigerenzer, Gerd and Gaissmaier, Wolfgang. Heuristic Decision Making. *Annual Review of Psychology.* 2011, Vol. 62, pp. 451-482.
47. Dunker, K. *Zur Psychologie des produktiven Denkens [The Psychology of Productive Thought].* Berlin : Springer, 1935.
48. Wertheimer, M. Untersuchungen zur Lehre von der Gestalt II [Studies on the Theory of Gestalt II]. *Psychologische Forschung.* 1923/1938, Vol. 4, pp. 301-350.
49. Pahl, G., et al. *Engineering Design, A Systematic Approach.* London : Springer-Verlag, 2007.
50. Alexander, Christopher. *The Timeless Way of Building.* New York : Oxford University Press, 1979.
51. Coplien, James O. Idioms and Patterns as Architectural Language. *IEEE Software.* 1997, Vol. 14, 1, pp. 36-42.
52. *Using Patterns to Improve our Architectural Vision.* Kerth, Norman L. and Cunningham, Ward. 1, Jan/Feb 1997, IEEE Software, Vol. 14, pp. 53-59.
53. *The Potential of Christopher Alexander's Theory and Practice of Wholeness: Clues for Developing an Educational Taxonomy.* Bauer, Reinhard and Baumgartner, Peter. Isree, Germany : EuroPLoP'10, 2010. 15th European Conference on Pattern Languages of Programs. p. Article #12.
54. Burckhardt, Lucius. *Design ist unsichtbar (Design is Invisible).* Berlin, Germany : Hatje Cantz Verlag, 1995.
55. Baker, E. Reading and the Street: An Inventory of Madrid Kiosks in 1911. [book auth.] J. Zamostny and S. Larson. *Kiosk Literature of Silver Age Spain: Modernity and Mass Culture.* United Kingdom : Intellect Books Limited, 2017.
56. *New Concept for Newspaper Kiosk Through Understanding Users' Behavior.* Kohdadadeh, Y. and Toobaie, A. Boca Raton FL : CRC Press, 2013. Advances in Affective and Pleasurable Design.
57. *Documenting Architectures with Patterns.* Hanmer, Robert S. and Kocan, Kristin F. 1, s.l. : Wiley Periodicals, 2004, Bell Labs Technical Journal, Vol. 9, pp. 143-163.
58. Leitner, H. Working with Patterns: An Introduction. [book auth.] D. Bollier and S. Helfric. *Patterns of Commoning.* Amherst : Levellers Press, 2015.
59. *A Pattern Language for Creating Pattern Languages: 364 Patterns for Pattern Mining, Writing, and Symbolizing.* Iba, Takashi and Isaku, Taichi. October 2016. Proceedings of the 2016 Conference on Pattern Languages of Programs.
60. History and Need for Design Patterns. *J2EE Reference.* [Online] j2eereference.com, 6 July 2017. [Cited: 2 February 2020.] https://j2eereference.com/history-need-design-patterns/.
61. *Patterns in Practice.* Helm, R. New York : Association for Computing Machinery, 1995.
62. Duell, M. Managing change with patterns. *IEEE Communications.* 1999, Vol. 37, 4.
63. Coplien, James O. and Bjornvig, Gertrud. *Lean Archtiecture: for Agile Software Development.* Chichester : John Wiley & Sons, 2010.
64. Schindel, William D. and Peterson, Troy. *An Overview of Pattern-Based Systems Engineering (PBSE): Leveraging MBSE Techniques.* Philadelphia : INCOSE IS2013 Tutorial, 2013.
65. *The OpenSE Cookbook: A practical, recipe based collection of patterns, procedures, and best practices for executable systems engineering for the Thirty Meter Telescope.* Karban, Robert, et al. Austin, TX : SPIE Astronomical Telescopes + Instrumentation, 2018.
66. *Using Patterns to Share Best Results - A proposal to codify the SEBOK.* Haskins, Cecilia. Washington DC : INCOSE, 2003. pp. 15-23.
67. *Application of Patterns and Pattern Languages to Systems Engineering.* Haskins, Cecilia. Rochester NY : INCOSE, July 10-14, 2005. pp. 1619-1627.
68. Object Management Group. Introduction to OMG's Unified Modeling Language (UML). *Unified Modeling Language.* [Online] July 2005. [Cited: 22 November 2021.] https://www.uml.org/what-is-uml.htm.
69. Grossman, Martin, Aronson, Jay E. and McCarthy, Richard V. Does UML make the grade? Insights from the software development community. *Information and Software Technology.* 2005, Vol. 47, pp. 383-397.
70. International Organization for Standardization. *Information Technology - Object Management Group Systems Modeling Language (OMG SysML).* Geneva : s.n., 2017. ISO/IEC 19514:2017.
71. *The Extravehicular Maneuvering Unit's New Long Life Lithium Ion Battery and Lithium Ion Battery Charger.* Russell, Samuel P, et al. Anaheim : AIAA SPACE 2010, 2010.
72. *Considerations for the Thermal Modeling of Lithium Ion Cells for Battery Analysis.* Rickman, Steven L., et al. Vienna, Austria : 46th International Conference on Environmental Systems, 2016. ICES-2016-009.
73. Finegan, Donal P, et al. In-operando high-speed tomography of lithium-ion batteries during thermal runaway. *Nature Communications.* 15 Dec 2014, pp. 1-10.
74. *A Scientific Methodology for Investigation of a Lithium Ion Battery Failure.* Mikolajczak, Celina J., Hayes, Troy and Meger, Marcus V. s.l. : 2007 IEEE International Conference on Portable Information Devices, 2007. pp. 1-6.
75. Iannello, C. J., et al. *Assessment of International Space Station (ISS)/Extravehicular Activity (EVA) Lithium Ion Battery Thermal Runaway (TR) Severity Reduction Measures.* Hampton, VA : NASA Engineering Safety Center, 2017.
76. Ruiz, Vanessa and Pfang, Andreas. JRC Exploratory Research: Safer Li-Ion Batteries by Preventing Thermal Propagation. *JRC Technical*

Reports. 2018.
77. *Design Guidelines for Safe, High Performing Li-Ion Batteries with 18650 Cells.* Darcy, Eric, et al. Petten : JRC Exploratory Research Workshop, 2018.
78. Friedenthal, Sanford, Moore, Alan and Steiner, Rick. *A Practical Guide to SysML, The Systems Modeling Language.* Amsterdam : Morgan Kaufmann, 2015.
79. *Modeling with SysML.* Friedenthal, Sanford and Wolfrom, Joseph. Chicago IL : INCOSE, 2010. INCOSE 2010.