An Architecting Book of Knowledge (BoK) to Improve Architectural Decision-Making

Bryan Mesmer
University of Alabama in Huntsville
95 PUBLICATIONS **539** CITATIONS

Alejandro Salado
University of Arizona
195 PUBLICATIONS **1,866** CITATIONS

35th Annual INCOSE international symposium
hybrid event
Ottawa, Canada
July 26 - 31, 2025

An Architecting Book of Knowledge (BoK) to Improve Architectural Decision-Making

Gordon Hunt
Skayl, LLC
Phoenix, AZ (USA)
gordon@skayl.com

Bryan Mesmer
The University of Alabama in Huntsville
Huntsville, AL (USA)
Bryan.Mesmer@uah.edu

Marcell Padilla
CRL Technologies, Inc.
Huntsville, AL (USA)
PadillaMS@crltechnologies.com

Anthony Edwards
Intrepid, LLC
Huntsville, AL (USA)
anthony.edwards@intrpid.llc

Bryan Joyner
Intrepid, LLC
Huntsville, AL (USA)
brian.joyner@intrepid.llc

Alejandro Salado
The University of Arizona
Tucson, AZ (USA)
alejandrosalado@arizona.edu

Abstract. To advance architecting from its current heuristics-based mode to a principles-based mode, this paper presents a systematic approach to architectural decision-making within the framework of a Book of Knowledge (BoK) and highlights how such decisions can shape complex systems over time. It emphasizes the need for a community-driven database of architectural knowledge, where best practices, patterns, and insights from past decisions can inform future work. Using a structured approach, the document outlines the Comprehensive Architecture Strategy (CAS), which aids architects in documenting and analyzing key elements such as patterns, ilities, and rules that govern architectural decisions. By breaking down architectural processes into well-defined components, CAS provides a standardized way to assess the impact of design choices on system ilities, supporting informed decision-making and enabling consistency across architectural projects.

Keywords. Architecture, book of knowledge, design, model-based design, system modeling, system engineering

Introduction

The system's architecture plays a crucial role in determining how well it will demonstrate various ilities (Osorio et al., 2011). Here, ilities refer to characteristics that bring value to stakeholders by enabling the system to manage high levels of uncertainty throughout its lifecycle, especially during extended operational periods (McManus & Hastings, 2006). Examples of ilities include adaptability to changing operational environments, economic fluctuations, and advancements in technology (Fricke & Schulz, 2005). In literature, other terms like quality attributes, lifecycle properties, or non-functional requirements are also used to describe ilities within the requirements context. In this paper, we use these terms interchangeably.

Identifying early how an architecture may impact ilities is widely regarded as a beneficial practice (Grunske, 2007). While architecting is considered both an art and a science (Jansma, 2012; Ryschkewitsch et al., 2009), some argue that the artistic aspect primarily shapes the ilities enabled by an architecture (Muirhead & Thomas, 2010), given the perception that "many ilities are hard to characterize in ways that are precise enough to support rigorous engineering" (Dou et al., 2015) and/or methods that rely on quantitative approaches require the whole architecture to be established, being unable to support the evaluation of individual architectural choices as they are made. Consequently, ilities are often pursued through heuristics (e.g., (Iandoli et al., 2020; Maier & Rechtin, 2009; Salado et al., 2016)), with assessments generally relying on informal expert judgment (Kazman et al., 2000). It seems sensible thus that successful architecting strongly depends on talent and experience (Griffin, 2010).

To reduce dependency on informal expertise and increase the predictive success of achieving ilities in system architecture, various methods have been proposed to formalize the relationship between architectural choices and ilities (e.g., (Baykasoğlu, 2009; Cormier et al., 2008; Fitzgerald et al., 2009; Potts et al., 2020; Ross et al., 2014)). However, the effectiveness of these methods is often limited due to the lack of accessible evidence to support them (Salado, 2022).

To advance architecting from its current heuristics-based mode to a principles-based mode, this paper presents a systematic approach to architectural decision-making within the framework of a Book of Knowledge (BoK) and highlights how such decisions can shape complex systems over time. The paper emphasizes the need for a community-driven database of architectural knowledge, where best practices, patterns, and insights from past decisions, as well as experimentations and empirical studies can inform future work. Using a structured approach, the paper outlines the Comprehensive Architecture Strategy (CAS), which aids architects in documenting and analyzing key elements such as patterns, ilities, and rules that govern architectural decisions. By breaking down architectural processes into well-defined components, CAS provides a standardized way to assess the impact of design choices on system ilities, supporting informed decision-making and enabling consistency across architectural projects.

Decisions in Architecting

The creation of an architecture is achieved through a series of decisions made by an architect, ideally heavily influenced by stakeholders. For the purpose of this paper, a normative decision theory perspective is adopted.

Preferences, beliefs (understood as predictions under uncertainty), and alternatives are the necessary components of a decision that are needed to properly analyze the decision (Howard & Abbas, 2015). As such, they are the three components of a decision that are necessary to record in order to justify a decision in the future. Many processes are associated with decision making. In this paper, we will briefly examine the common forms of representation. Incrementally improving on such forms improves the decision making in architecting and systematically improves architecting as a whole.

The representation of the three components of decision making (preferences, beliefs, and alternatives) allows for accurate communication of the components to enable effective use of the components as well as justification through repeatability. The representation process transforms the data gathered in the elicitation process into a usable and recordable form. The techniques in which the data is transformed each have benefits and drawbacks. The following subsections provide overviews of techniques that can be used to represent alternatives, beliefs, and preferences, respectively.

Representation of Alternatives

Alternatives should be represented in a form that describes what the alternative is. The outcome that results from selecting the alternative is a separate representation, handled through the decision-making component of beliefs. A summary of different forms of representation is given in Table 1.

An OK representation of an alternative would be the name of the alternative. This is the lowest ranked form as just a name leaves significant interpretation of the alternative by 1) future decision makers who are told this information as what was chosen, 2) current stakeholders as information is elicited concerning the alternative, 3) the architect as information is gathered from sources concerning the alternative (multiple sources may mean different things but use the same name), and 4) future stakeholders who investigate the justification for a decision. A good representation of an alternative would be a qualitative description. A qualitative description provides information about the alternative beyond its name. This information may relate to features of the alternative, parts, etc. A better form of an alternative would be a quantitative description. A mathematical characterization provides numerical descriptions of the alternative, graphical representations, potential structural models, etc. This quantitative description form further removes potential miscommunication between the architect and any current sources of information or future users/assessors. Ideally, a representation of an alternative would leave no room for multiple interpretations of what the alternative is.

Table 1. Alternative Representations

Accuracy	**Form**	**References**
OK	Name Only	Strandh Tholin, 2021
Good	Qualitative Description	Sormaz et al., 1999
Better	Mathematical Characterization	Scothern, 1991

Representation of Beliefs

Beliefs should be represented in a form that captures the complete understanding of the outcome of an alternative. A summary of different forms of representation is given in Table 2.

A poor form of a belief would be just stating the outcome's name. While this does provide information that the alternative impacts this aspect of an outcome, it says nothing more. As such it is unclear the direction and magnitude of that impact. A better but still poor form is to provide the believed direction of the impact. Extreme care must be taken with this form of a belief as no magnitude information is represented.

An OK representation of beliefs is to express an outcome with certainty that actually has uncertainty associated with it. Very few decisions in architecting will have alternatives with precisely known outcomes, due to those outcomes likely being impacted by later decisions and outside factors. A good representation of beliefs is to express an outcome as a range of possible values. This range is characterized by a lower and upper bound. This can later be used in probabilistic analyses by assuming distributions between that range or non-probabilistic methods that can be applied to ranges. A better representation would be to define the probability distribution directly. This gives a clearer representation of anticipated outcomes. This form provides a more thorough representation of uncertain outcomes that are extremely typical in architecting decision making.

Table 2. Belief Representations

Accuracy	**Form**	**References**
Poor	Name Only Direction	Ricci et al., 2014
OK	Certain Outcome	Collopy & Hollingsworth, 2011; Keller & Collopy, 2013
Good	Range of Outcomes	Renou & Schlag, 2010; Tuan et al., 2019
Better	Probability Distribution	Pinsky & Karlin, 2011; Malak et al., 2015

Representation of Preferences

Preferences should be represented in a form that captures the decision maker's desires concerning the outcomes of the alternatives. A summary of different forms of representation is given in Table 3.

A poor representation of preferences is through requirements. A requirement separates alternatives through their outcomes into acceptable (feasible) or unacceptable (infeasible). However, practice indicates that there is some additional ordering of the alternatives; it is not uncommon for projects to accept non-compliances of some requirements, which indicates that what initially is considered to be unacceptable is only less preferred than full compliance.

An OK representation of preferences is rank order. Rank order represents preference as a vector of outcomes ordered in such a way that the first outcome in the vector is most preferred and the last outcome in the vector is least preferred. Hence the alternatives that produce those outcomes would have the same order. A good representation of preferences is multiple objective functions. Multiple objective functions are mathematical aggregation functions that give a single numerical value for an alternative's outcome. Typically, multiple objective functions are a summation of weighted objectives. The objectives in the multiple objective functions are often functions of outcomes or the outcomes directly. A better representation of preferences that is designed to overcome the challenge of weights in multiple objective functions is value models. Value models are similar to multiple objective functions; however, they are formed to convert inputs (outcomes) into a common unit that is understood by stakeholders (such as US dollars, USD) in terms of how much they value each combination of inputs. By doing so a meaningful output of the model is created. Architects can use the value model to express a meaningful valuation of the outcome to stakeholders, architecture users, future assessors, etc. A challenge in value models is that significant effort is needed to properly form the functions that convert the outcomes to a common unit. For example, a function would need to be created to convert man-hours into dollars. For both value models and multiple objective functions, care must be taken to also capture any couplings between outcomes as to not "double count" any impacts or ignore conflicts or cancellation effects.

Table 3. Preference Representations

Accuracy	**Form**	**References**
Poor	Requirements (Shall Statements)	Robertson and Robertson, 2012, Hooks, 1994

OK	Rank Order (e.g., Prioritization)	Tsiporkova and Boeva, 2006
Good	Multiple Objective Function	Roy, 1971, Hwang and Masud, 2012
Better	Value Model/Utility Function	Clerkin and Mesmer, 2018, Lee, Binder, and Paredis, 2014, Collopy and Hollingsworth, 2011

Need for Community Knowledge to Inform Representations

The representation approaches for elements of an architecting decision necessarily needs data as input to the approaches. The questions that arise for an architect are often "what am I representing?" and "what data do I have about that thing?" For example, for preferences, perhaps the "what am I representing?" question has the answer of 'the stakeholder wants more maintainability and a little more interoperability'. For the "what data do I have about that thing?" question, perhaps the architect has access to some previous data on maintainability that can help in understanding relationships between maintainability and interoperability, and some basic constraints from the stakeholder that could be useful for a requirements approach. In this scenario, the architect is at the whim of the stakeholder's provided data and the architect's own past experiences. The architect can use some past experiences to help identify potential ility (also called quality attribute (QA) in the literature) relationships and may have experience in similar requirements formation that they can leverage. However, it is very much a heroic architect scenario. What is needed is a database of knowledge that can store the lessons learned, heuristics, relationships, etc. that are critical to the tasks performed by architects. Doing so gives architects the ability to identify and leverage community generated data that can enable them to use better approaches throughout architecting (in this example case better representation approaches could be used).

Past work has presented an experimentation and model framework designed to validate architectural properties that support desired ilities (quality attributes) in complex systems (Salado, 2022). The framework's purpose is to formalize how architectural choices (e.g., design patterns, tactics) impact ilities, providing architects with evidence rather than relying only on their individual experience. Ultimately, this framework seeks to establish a consistent approach for creating a Book of Knowledge (BoK) that will support architecture decisions in early development stages by linking mechanisms to QAs with quantifiable data.

Architecture Elaboration Structure and Decomposition

The Comprehensive Architecture Strategy (https://apps.dtic.mil/sti/pdfs/AD1103295.pdf) outlines the development of a meta-model for documenting architecture concepts. Specifically, it introduces a meta-model designed to formalize the structure and contents of a Book of Knowledge (BoK) for architecture. This BoK is organized to guide the documentation and analysis of architectural elements, mechanisms, QAs, and their interrelations, which support system architecture and engineering processes.

The CAS model also supports the BoK's use as a computational and experimental tool, aiming to quantify and validate architectural decisions through a well-defined relational model. This includes structures for business drivers, quality attributes, and mechanisms, among other elements, that influence the architecture's effectiveness and alignment with both technical and business objectives. CAS emphasizes a multi-level structure for managing architecture across military systems. Key points include:

1. **Three-Tier Architecture**:

   - **Reference Architecture (RefArch)**: Establishes foundational guidelines and structures to guide all subsequent architecture levels. It includes high-level business drivers, quality attributes, and regulatory constraints that influence system designs.
   - **Objective Architecture (ObjArch)**: A technology-independent architecture derived from the RefArch, tailored for specific product lines or families of systems. It provides more refined requirements and architectural constraints.
   - **System Architecture (SysArch)**: Specific to individual products, the SysArch is derived from ObjArch, detailing system-level performance, design, and implementation requirements.
2. **Key Business Drivers (KBDs) and Key Architecture Drivers (KADs)**:
   - CAS introduces KBDs and KADs to guide architectural decisions. KBDs address core business needs, such as affordability and interoperability, while KADs encompass quality attributes critical to meeting these business drivers.
3. **Policy and Regulatory Constraints**:
   - The strategy integrates relevant policies and mandates, providing guidance on compliance with standards, data rights, and best practices.
4. **Functional, Hardware, Software, and Data Architecture Specifications**:
   - Each architecture level defines specific requirements in these areas, from high-level function grouping in RefArch to detailed design considerations in SysArch.
5. **Governance and Guidance**:
   - CAS emphasizes governance structures for configuration management, change control, and data rights, ensuring consistency and traceability across architectural levels.

This structured, hierarchical approach allows flexibility and adaptability while ensuring alignment with enterprise objectives and regulatory compliance across the Department of Defense's system architectures.

Key Elements of the BoK Framework

Overview

The CAS meta-model and BoK framework defines a structured framework to describe and document architectural concepts systematically. At its core, this meta-model distinguishes between several key elements: Mechanisms**,** Quality Attributes (QAs)**,** Parameters**,** Rules**,** Operations, and Effects. Mechanisms are defined as outcomes derived from applying rules to sets of parameters, forming the functional foundation of architectural components. Each mechanism is governed by a rule set, determining how various parameters interact to achieve desired effects within specific operational contexts.

1. **Mechanism**: These are individual architectural choices, such as design patterns or tactics, that influence specific QAs.
2. **Property**: Observable aspects of mechanisms, like the number of nodes in a network or the density of connections in a topology.
3. **Consequential Attributes**: Variables that result from the operation of mechanisms or properties, which may not directly relate to a QA but affect it indirectly.
4. **QA Characteristics**: Conditions required for a quality attribute to be exhibited, providing more granular detail about QAs.

Quality attributes represent the desired non-functional characteristics of the system, such as maintainability, security, and scalability. These are further broken down into Quality Characteristics, measurable factors that define each QA in more detail. The relationships among mechanisms, parameters, and quality attributes are essential, as they allow architects to trace how specific design choices affect different system attributes at different levels of elaboration. Specifically, a mechanism's effectiveness is evaluated based on its effect on quality characteristics, ultimately impacting the system's ability to meet high-level quality requirements.

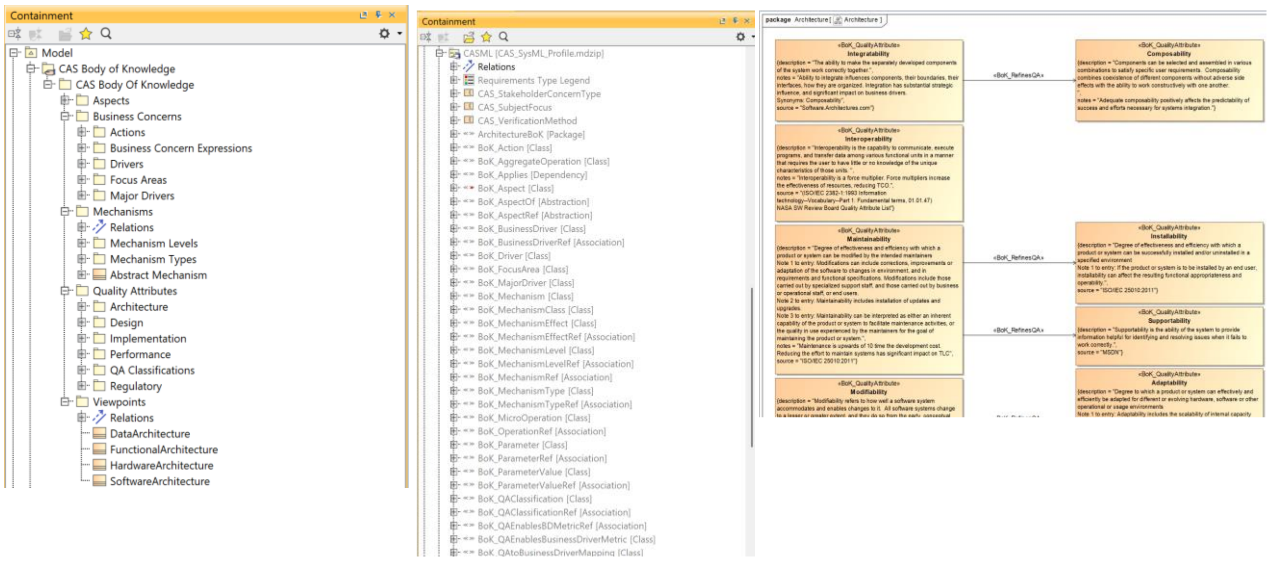
The meta-model employs a hierarchy to connect architectural mechanisms with QAs, capturing the cascading effects of operational decisions on system quality. This relational framework aids architects in balancing various quality attributes when making design choices, acknowledging that enhancements in one area (e.g., security) might introduce trade-offs in another (e.g., performance). By formalizing these relationships, the model supports consistent analysis, allowing stakeholders to predict the impact of changes and optimize architectures based on measurable quality targets.

Additionally, the model's structured approach facilitates contributions to a broader Body of Knowledge, enabling researchers and practitioners to build on existing architectural knowledge. Through experimentation and documentation, new architectural mechanisms and quality attributes can be iteratively incorporated into the BoK, enhancing the model's applicability and relevance across diverse systems.

These elements form a relationship map that connects mechanisms to QAs. This map helps architects systematically examine how mechanisms may impact system qualities through a defined experimentation process. The formal structure of these elements enables their use for architecting & system elaboration, individual QA research and decomposition, mechanism assessments and measurements.

Model of Elements and Relations

The BoK serves as a structured repository of information that supports the architecture elaboration and decision-making processes. While a finalized and normalized repository structure has not been proposed, our current efforts have led to a SysML representation of the BoK content in a popular modeling tool ecosystem (ref. Figure 1). This model serves as a repository of current elements and relationships and provides a means to incrementally, through varied approaches, develop, test, and validate independent element for the BoK.


```
Containment
Model
  CAS Body of Knowledge
    CAS Body Of Knowledge
      Aspects
      Business Concerns
        Actions
        Business Concern Expressions
        Drivers
        Focus Areas
        Major Drivers
      Mechanisms
        Relations
        Mechanism Levels
        Mechanism Types
        Abstract Mechanism
      Quality Attributes
        Architecture
        Design
        Implementation
        Performance
        QA Classifications
        Regulatory
      Viewpoints
        Relations
        DataArchitecture
        FunctionalArchitecture
        HardwareArchitecture
        SoftwareArchitecture
```

```
Containment
CASML [CAS_SysML_Profile.mdzip]
  Relations
  Requirements Type Legend
  CAS_StakeholderConcernType
  CAS_SubjectFocus
  CAS_VerificationMethod
  ArchitectureBoK [Package]
  BoK_Action [Class]
  BoK_AggregateOperation [Class]
  BoK_Applies [Dependency]
  BoK_Aspect [Class]
  BoK_AspectOf [Abstraction]
  BoK_AspectRef [Abstraction]
  BoK_BusinessDriver [Class]
  BoK_BusinessDriverRef [Association]
  BoK_Driver [Class]
  BoK_FocusArea [Class]
  BoK_MajorDriver [Class]
  BoK_Mechanism [Class]
  BoK_MechanismClass [Class]
  BoK_MechanismEffect [Class]
  BoK_MechanismEffectRef [Association]
  BoK_MechanismLevel [Class]
  BoK_MechanismLevelRef [Association]
  BoK_MechanismRef [Association]
  BoK_MechanismType [Class]
  BoK_MechanismTypeRef [Association]
  BoK_MicroOperation [Class]
  BoK_OperationRef [Association]
  BoK_Parameter [Class]
  BoK_ParameterRef [Association]
  BoK_ParameterValue [Class]
  BoK_ParameterValueRef [Association]
  BoK_QAClassification [Class]
  BoK_QAClassificationRef [Association]
  BoK_QAEnablesBDMetricRef [Association]
  BoK_QAEnablesBusinessDriverMetric [Class]
  BoK_QAtoBusinessDriverMapping [Class]
```

package Architecture [Architecture]

«BoK_QualityAttribute»
Integratability
{description = "The ability to make the separately developed components of the system work correctly together.",
notes = "Ability to integrate influences components, their boundaries, their interfaces, how they are organized. Integration has substantial strategic influence, and significant impact on business drivers.
Synonyms: Composability",
source = "Software.Architectures.com"}

«BoK_QualityAttribute»
Composability
{description = "Components can be selected and assembled in various combinations to satisfy specific user requirements. Composability combines coexistence of different components without adverse side effects with the ability to work constructively with one another.
",
notes = "Adequate composability positively affects the predictability of success and efforts necessary for systems integration."}

«BoK_QualityAttribute»
Interoperability
{description = "Interoperability is the capability to communicate, execute programs, and transfer data among various functional units in a manner that requires the user to have little or no knowledge of the unique characteristics of those units.",
notes = "Interoperability is a force multiplier. Force multipliers increase the effectiveness of resources, reducing TCO.",
source = "ISO/IEC 2382-1:1993 Information technology—Vocabulary—Part 1: Fundamental terms, 01.01.47)
NASA SW Review Board Quality Attribute List"}

«BoK_QualityAttribute»
Maintainability
{description = "Degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers
Note 1 to entry: Modifications can include corrections, improvements or adaptation of the software to changes in environment, and in requirements and functional specifications. Modifications include those carried out by specialized support staff, and those carried out by business or operational staff, or end users.
Note 2 to entry: Maintainability includes installation of updates and upgrades.
Note 3 to entry: Maintainability can be interpreted as either an inherent capability of the product or system to facilitate maintenance activities, or the quality in use experienced by the maintainers for the goal of maintaining the product or system.",
notes = "Maintenance is upwards of 10 time the development cost. Reducing the effort to maintain systems has significant impact on TLC",
source = "ISO/IEC 25010:2011"}

«BoK_QualityAttribute»
Installability
{description = "Degree of effectiveness and efficiency with which a product or system can be successfully installed and/or uninstalled in a specified environment
Note 1 to entry: If the product or system is to be installed by an end user, installability can affect the resulting functional appropriateness and operability.",
source = "ISO/IEC 25010:2011"}

«BoK_QualityAttribute»
Supportability
{description = "Supportability is the ability of the system to provide information helpful for identifying and resolving issues when it fails to work correctly.",
source = "MSDN"}

«BoK_QualityAttribute»
Modifiability
{description = "Modifiability refers to how well a software system accommodates and enables changes to it. All software systems change to a lesser or greater extent, and they do so from the early, conceptual

«BoK_QualityAttribute»
Adaptability
{description = "Degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments
Note 1 to entry: Adaptability includes the scalability of internal capacity

Figure 1. BoK Content in a SYSML Model

It is anticipated that the data in the BOK will be disparate, certainly at the beginning of being populated and likely during its lifetime. Disparity in data in the BoK will be caused by at least one or more of the following:

- Data may come from different types of evidence: experimentation, simulation, heuristics based on experience, empirical studies on real projects, etc.
- Data may cover different types of relationships. While some evidence may apply to the whole chain from a mechanism to a quality characteristic, some evidence may hide some of the intermediate steps.
- Data may measure the relationships between elements in the model differently. For example, some evidence may be formulated in statistical terms, while some evidence may be formulated qualitatively.
- Metadata associated with the evidence may differ in the accuracy and/or procession with which the context in which the evidence was collected is characterized.
- There may be different number of evidence for different mechanisms and quality characteristics.
- Data may be inconsistent since some projects may have observed the impact of one mechanism on a quality characteristic in one direction and another project may have had the opposite experience.

Because of this, the design of the BOK usage must account for aggregating or at least navigating through disparate data sources (Salado, 2025). The following sections describe possible approaches to aggregate disparate data sources in the BOK. All examples are notional and use synthetic data. The approaches have not been evaluated in detail, but just explored for feasibility.

Relationship to ISO420x0 Series

The decision-making processes, formal meta-model, and supporting BOK content outlined in this paper and the standards ISO/IEC 42010, 15288, 12207, 25010, and TR 24748 converge on a shared goal: enabling principled, traceable, and quality-focused architectural decisions. This paper's structure model and its supporting BoK framework propose a formal, evidence-driven approach to architecting that complements and operationalizes the architectural processes outlined in the ISO standards.

ISO/IEC/IEEE 42010 emphasizes the creation of structured architecture descriptions that capture stakeholder concerns through multiple views and viewpoints. The meta-model aligns with this by defining mechanisms, parameters, quality attributes (QAs), and their interactions, effectively structuring architecture knowledge in a way that supports stakeholder traceability and consistent analysis. Likewise, the BoK's emphasis on representing alternatives, beliefs, and preferences maps directly to the decision-making and evaluation processes highlighted in ISO/IEC/IEEE 15288 and 12207, which focus on integrating architecture definition and evaluation into the broader system/software life cycle.

Furthermore, the BoK's focus on quantifying impacts of mechanisms on ilities (e.g., modifiability, availability, performance) aligns with the quality model defined in ISO/IEC 25010, where non-functional requirements are key drivers of architectural fitness. By documenting how design patterns and tactics influence QAs, the BoK serves as an operational repository that supports quality attribute assessment, effectively fulfilling the quality evaluation intent of 25010.

The structured modeled approach along with the BoK content presents a practical instantiation of the architecture-related principles across ISO/IEC standards. It bridges theory and application by offering a repeatable, data-informed process for architectural decisions, precisely what the standards framework encourages for improved consistency, interoperability, and system quality.

Developing BoK Content

A possible experimental framework and methodology would involve several key stages, each tailored to analyze the impact of mechanisms on 'ilities' (quality attributes) through experimental design. This approach leverages the structured elements and relationships in the CAS meta-model so that consistency and repeatability of results can be achieved. Simulation in the context of the BoK and its content is focused on measuring and defining the effects design elements have on quality attributes and their characteristic decomposition providing one means to fully define and develop the BOK content with repeatable, testable experiments. Much akin to material data sheets with all the different properties that are separately tested (hardness, tensile strength, …) a software and system design pattern can be decomposed into properties that can be individually assessed.

1. Define Research Question and Hypothesis

The first step in designing an architecture experiment is to define a research question and a hypothesis. This focuses the experiment on specific mechanisms and QAs, determining the experiment's purpose and scope. Researchers identify which ilities are under investigation, such as Integrability or Modifiability, and select an experiment scenario that defines the system's context and challenges.

For example, suppose a researcher wants to study the effect of the Factory Method pattern (a design pattern for creating objects without specifying exact classes) on Modifiability. A sample hypothesis might be: "The Factory Method pattern will improve Modifiability by simplifying future changes to classes within a system."

2. Develop Experiment Framework Model

After defining the research question, researchers develop a model based on framework objectives, mechanisms, and context. This step involves detailing parameters, rules, operations, and defining the specific scenario for the experiment.

- Parameters: Elements that define the mechanism, such as an intermediary used in the Factory Method pattern to serve entities.
- Rules: Principles governing the use of parameters. These can be "Greenfield" (for new systems) or "Brownfield" (for modifying existing systems). For instance, in a Greenfield setup, the Factory Method might require all clients to access entities via the factory, ensuring flexibility in future system changes.

In testing the Factory Method, a researcher defines parameters and rules. For parameters, they note that entities must adhere to a common interface and that the factory handles lifecycle control for objects it serves. Rules for Brownfield (existing system) modifications specify that any new entity added to the system must follow the common interface and be instantiated through the factory.

3. Simulation Development and Execution

With the framework model established, the next step is to simulate the experiment, using methods such as Monte Carlo simulation. This simulation type allows the model to account for uncertainties by generating a wide range of contexts, each representing different conditions under which the mechanism might operate. Other simulation types may include discrete event or agent-based simulations for specific use cases, such as systems where timing and event sequences are critical.

- Define Simulation Scope: Simulation only includes rules, parameters, and mechanisms relevant to the scenario. For instance, if the experiment tests the Factory Method's effect on Modifiability, the simulation might exclude operations irrelevant to that QA.
- Select Simulation Type**:** For mechanisms affecting runtime behaviors, researchers may choose discrete event simulations to model how mechanisms perform in real-time conditions.

- Develop Conceptual Model**:** To accurately represent mechanism effects, researchers use simplified models based on foundational abstractions, such as network science for complex dependencies or queuing models for resource management.

Testing the Factory Method in a Monte Carlo simulation might involve multiple system configurations (contexts). The researcher could vary the types of clients accessing objects through the factory or introduce changes like adding new object types to see if Modifiability remains consistent. If results show that adding new object types is smooth and does not disrupt existing functionalities, the pattern can be said to support Modifiability.

4. Collect, Analyze, and Refine Data

Once the simulation is complete, researchers gather data to analyze how effectively the mechanism met the targeted QA in different contexts. They calculate QA characteristic values to measure success, allowing them to identify any statistically significant relationships between the mechanism and QA. The data analysis may also involve sensitivity analysis to determine if certain rules or parameters disproportionately influence outcomes.

If results are inconsistent or unpredicted, researchers review the experiment model for possible adjustments, such as refining the context variables or QA parameters.

BoK Content Examples

The following is a simplified example of an architecture elaboration process and decision support using a BoK. This approach utilizes the CAS approach and would be repeated across each level of elaboration. In short, the approach tries to codify the systems architect's expert opinions into a process that a system engineer can follow. The process should identify what decisions are being made, the architectural impact of those decisions, and the information used to justify and assess the impact of the decision on the desired architecture.

Scenario

A company developing a mission-critical communication platform needs to ensure high availability and security while also maintaining performance under high user load. The system will operate in a hybrid cloud environment, and thus the architecture must support both on-premises and cloud-based components.

Step 1: Define Requirements and Quality Attributes

Using the BoK, the team identifies high availability, security, and performance as primary ilities (QAs). For high availability, they define a target uptime of 99.99%, and for security, they aim to comply with industry standards (e.g., ISO 27001) and ensure robust data protection. Performance benchmarks include response times below 200ms under peak load. Note, these requirements are performance orientated but are impacted highly by the actual architectural decisions that could be made by an architect. It is these relationships and their impact on the decision space that the BoK seeks to formalize.

Step 2: Select CAS-Appropriate Mechanisms and Patterns

The CAS framework helps structure these quality requirements, translating them into specific architectural mechanisms:

- **High Availability (HA) Mechanisms**: The CAS model identifies that previous knowledge indicates distributed architecture and redundancy for HA. An architect can use the BoK to identify a **Service-Oriented Architecture (SOA)** with load balancing and failover mechanisms as appropriate patterns.
- **Security Mechanisms**: An architect can use the BoK to identify encryption in transit and at rest, role-based access control, and a zero-trust security model. A Publish-Subscribe Pattern is suggested to minimize direct dependencies between components, reducing the attack surface.
- **Performance Mechanisms**: For performance, knowledge embedded in CAS may lead an architect to caching strategies and lightweight microservices that enable rapid data access. The BoK past knowledge advises implementing a Microservices Architecture pattern with an API Gateway for routing and monitoring requests to balance load efficiently.

Step 3: Evaluate and Document Architectural Decisions

The team selects specific mechanisms from the BoK, noting their anticipated impact on each quality attribute. For instance:

- **HA with Distributed Load Balancing and Redundancy**: The selected SOA pattern is documented to achieve redundancy by distributing services across multiple cloud zones and utilizing load balancing.
- **Security with Zero-Trust Architecture**: The zero-trust model is applied to the API Gateway to enforce authentication and authorization for each request.
- **Performance with Caching and Efficient APIs**: Implementing a dedicated cache layer for frequently accessed data, along with an API Gateway for centralized request handling, will help meet the 200ms response time target.

Step 4: Trace Impact and Trade-offs

Using CAS, the team maps each selected mechanism's potential trade-offs. For instance:

- **Performance vs. Security**: Implementing encryption increases security but could impact performance. To mitigate this, only sensitive data is encrypted, while non-sensitive data flows are kept unencrypted for faster processing.
- **HA vs. Complexity**: Distributed load balancing and redundancy increase availability but add architectural complexity. The BoK advises regular testing and monitoring for failure scenarios to handle this trade-off effectively.

Step 5: Integrate with the BoK and Iterate

The decisions are documented in the BoK as entries that can be referenced in future projects or by other teams. By detailing the CAS-based choices and associated quality trade-offs, the BoK evolves to better guide future architectural decisions, making CAS an iterative, knowledge-enhancing process.

In this example, CAS and the BoK work together to guide a well-informed architecture decision, balancing high availability, security, and performance while providing a traceable, repeatable approach for future architectures.

Recommendations & Next Steps

The BoK and supporting approach offers a structured way to quantify QA-mechanism relationships, which is essential for consistent architecture decisions. There is a need to relate the data present in the BoK to potential decision-making approaches. For example, if beliefs are all recorded in the BoK with complete certainty, then decision-making approaches that would incorporate uncertainty and risk would not be appropriate. Another example would be that data is recorded in such a way that "better" decision-making approaches are possible and the BoK can inform the architect that these approaches are possible with the available data. A helpful takeaway from a community informed database for architect decision makers is the identification of dominant strategies. The researchers introduced the concept of "super patterns," which are versatile enough to mimic other mechanisms. A super pattern could cover multiple mechanisms within the same class, simplifying decision-making in complex systems. For example, a mesh topology might serve as a super pattern for network structures, as it can mimic other topologies (e.g., ring, hub, or star) through configuration changes.

There are some areas where additional research will be needed to fully support the BoK content maintenance and curation when testing and modeling mechanisms and their effects on architecture. A comprehensive literature review to catalog system and data abstractions useful in experimentation is needed. This catalog would support researchers in selecting the best abstractions for specific experiments, potentially automating mechanism parameterization to improve consistency. Also, developing standardized documentation for submitting results to the BoK managing authority, which would facilitate knowledge sharing and improve the quality and reliability of entries in the Book of Knowledge is recommended. Possible exploration of vector space models for their potential in characterizing parameters and measuring mechanism similarity, an area inspired by techniques in machine learning, could prove valuable.

Finally, collaborative exploration is needed on how the meta-model and associated BoK content would be curated, developed, moderated, and deployed with appropriate data-rights free from company restrictive intellectual property restrictions.

Conclusion

The field of architecting needs the community to come together to perform research and to provide evidence to evolve from a primarily heuristic-based domain to a principles-based domain. Research can be conducted on elements of architecting to advance the evolution of the field. Decisions are a critical element within architecting with theories to support rigorous approaches that can be applied to architecting. A primary challenge in architecting decisions is the data needed to apply the approaches. Presented in this paper was the Architecting BoK which provides a framework and location to capture critical knowledge, useful to architecting decision making, that is currently stored in the minds of heroic architects.

The BoK can be leveraged to quantify the impact of mechanisms on QAs, providing architects with reliable data for architecture decisions. Through experimentation, simulation, and formalization, the framework helps advance architecture knowledge. By adopting this methodology, organizations can make architecture choices with higher confidence, supported by a growing BoK that captures and shares validated insights on achieving desired QAs in complex systems.

Acknowledgment

Effort sponsored in whole or in part by the U.S. Army Combat Capabilities Development Command Aviation & Missile Center, under Recipient Agreement No. *W911W6-17-3-0002*. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. Army Combat Capabilities Development Command Aviation & Missile Center. U.S. Army Combat Capabilities Development Command Aviation & Missile Center provides increased responsiveness to the nation's Warfighters through aviation and missile capabilities and life cycle engineering solutions.

References

Bachmann, F.; Bass, L.; and Klein M., "Deriving Architectural Tactics: A Step Toward Methodical Architectural Design," CMU/SEI-2003-TR-004, 2003.

Baykasoğlu, A. (2009). Quantifying machine flexibility. *International Journal of Production Research*, 47(15), 4109-4123. https://doi.org/10.1080/00207540802007589

Clerkin, J. H., & Mesmer, B. (2019). *A Review of Value Modeling in the NASA Systems Engineering Research Consortium*. https://doi.org/10.1007/978-3-030-00114-8_34

Collopy, P. D., & Hollingsworth, P. M. (2011). Value-Driven Design. *Journal of Aircraft*, *48*(3), 749–759. https://doi.org/10.2514/1.C000311

Cormier, P., Olewnik, A., & Lewis, K. (2008). An Approach to Quantifying Design Flexibility for Mass Customization in Early Design Stages. *ASME 2008 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference.*

Dou, K., Wang, X., Tang, C., Ross, A., & Sullivan, K. (2015). An Evolutionary Theory-systems Approach to a Science of the Ilities. *Procedia Computer Science*, 44, 433-442. https://doi.org/https://doi.org/10.1016/j.procs.2015.03.064

Fitzgerald, G., Barad, M., Papazafeiropoulou, A., & Alaa, G. (2009). A framework for analyzing flexibility of generic objects. *International Journal of Production Economics*, 122(1), 329-339. https://doi.org/https://doi.org/10.1016/j.ijpe.2009.06.005

Fricke, E., & Schulz, A. P. (2005). Design for Changeability (DfC): Principles to enable changes in systems throughout their entire lifecycle. *Systems Engineering*, 8, 342-359.

Graves, T., "What's the Difference between Architecture and Design?" Tom Graves / Tetradian, Oct. 2009. https://weblog.tetradian.com/2009/10/09/architecture-versus-design/

Griffin, M. D. (2010). How do we fix systems engineering? *61st International Astronautical Congress*, Prague, Czech Republic.

Grunske, L. (2007). Early quality prediction of component-based systems – A generic framework. *Journal of Systems and Software*, 80(5), 678-686. https://doi.org/https://doi.org/10.1016/j.jss.2006.08.014

Harrison N. B., and Avgeriou P., "Implementing Reliability: The Interaction of Requirements, Tactics and Architecture Patterns," Architecting Dependable Systems VII, edited by A. Casimiro; R. de Lemos; and C. Gacek, Vol. 6420, Springer, Berlin, Heidelberg, 2010, pp. 97-122.

Hooks, I. (1994). Writing Good Requirements. *INCOSE International Symposium*, *4*(1), 1247–1253. https://doi.org/10.1002/j.2334-5837.1994.tb01834.x

Howard, R. A., & Abbas, A. E. (2015). *Foundations of Decision Analysis* (1st Edition ed.). Pearson.

Hwang, C.-L., & Masud, A. S. M. (2012). *Multiple Objective Decision Making — Methods and Applications: A State-of-the-Art Survey*. Springer Science & Business Media.

Iandoli, L., Salado, A., & Zollo, G. (2020). The role of aesthetic reasoning in knowledge management: the case of elegant systems architecture design. *Knowledge Management Research & Practice*, 18, 93-109. https://doi.org/10.1080/14778238.2019.1678410

Jansma, P. A. T. (2012). Exploring the art and science of systems engineering *IEEE Aerospace Conference*, Big Sky, MT, USA.

Kazman, R., Klein, M., & Clements, P. (2000). *ATAM: Method for Architecture Evaluation*.

Keller, S., & Collopy, P. (2013). Value Modeling for a Space Launch System. *Procedia Computer Science*, *16*, 1152–1160. https://doi.org/10.1016/j.procs.2013.01.121

Kellerman, C., "Cyber-Physical Systems and Internet of Things Foundations," NIST, Oct. 2019. www.nist.gov/programs-projects/cyber-physical-systems-and-internet-things-foundations

Kim, S.; Kim, D.; Lu, L.; and Park, S., "Quality-Driven Architecture Development using Architectural Tactics," Journal of Systems and Software, Vol. 82, (8), August 2009, pp. 1211-1231. https://doi.org/10.1016/j.jss.2009.03.102

Klein, J.; Cohen, S.; and Kazman, R., "Common Software Platforms in System-of-Systems Architectures: The State of the Practice," 8th International Conference on System of Systems Engineering (SoSE), Maui, HI, June 2-6, 2013.

Lee, B. D., Binder, W. R., & Paredis, C. J. (2014). A Systematic Method for Specifying Effective Value Models. *CSER*, 228–236.

Maier, M. W., & Rechtin, E. (2009). *The art of system architecting*. CRC Press.

Malak, R., Baxter, B., & Hsiao, C. (2015). A Decision-based Perspective on Assessing System Robustness. *Procedia Computer Science, 44*, 619–629. https://doi.org/10.1016/j.procs.2015.03.069

Masood, A., "System Quality Attributes for Software Architecture," LinkedIn SlideShare, June 2013. www.slideshare.net/adnanmasood/system-quality-attributes

McManus, H., & Hastings, D. (2006). A framework for understanding uncertainty and its mitigation and exploitation in complex systems. *IEEE Engineering Management Review*, 34(3), 81-81. https://doi.org/10.1109/EMR.2006.261384

Muirhead, B. K., & Thomas, D. (2010). The Art and Science of Systems Engineering Tightly Coupled Programs. *SAE Int. J. Passeng. Cars – Electron. Electr. Syst.*, 3(2), 117-130. https://doi.org/10.4271/2010-01-2321

Osorio, C. A., Dori, D., & Sussman, J. (2011). COIM: An object-process based method for analyzing architectures of complex, interconnected, large-scale socio-technical systems. *Systems Engineering*, 14(4), 364-382. https://doi.org/https://doi.org/10.1002/sys.20185

Padilla, M. O.; Davis, J. B.; and Jacobs, W. J., "Comprehensive Architecture Strategy (CAS)", January 2022. https://apps.dtic.mil/sti/pdfs/AD1185001.pdf

Padilla, M. O.; Davis, J. B.; and Jacobs, W. J., "Overview of CAS and the Role of the Future Airborne Capability Environment (FACE) Technical Standard," US Air Force FACE Technical Interchange Meeting, Dayton, OH, Sept. 17, 2019.

Pinsky, M. A., & Karlin, S. (2011). *An introduction to stochastic modeling* (4. ed). Acad. Press, Elsevier.

Potts, M. W., Sartor, P. A., Johnson, A., & Bullock, S. (2020). A network perspective on assessing system architectures: Robustness to cascading failure. *Systems Engineering*, 23(5), 597-616. https://doi.org/https://doi.org/10.1002/sys.21551

Renou, L., & Schlag, K. H. (2010). Minimax regret and strategic uncertainty. *Journal of Economic Theory, 145*(1), 264–286.

Ricci, N., Fitzgerald, M. E., Ross, A. M., & Rhodes, D. H. (2014). Architecting systems of systems with ilities: An overview of the SAI method. *Procedia Computer Science, 28*, 322–331.

Robertson, S., & Robertson, J. (2012). *Mastering the Requirements Process: Getting Requirements Right*. Addison-Wesley.

Ross, A. M., Stein, D. B., & Hastings, D. E. (2014). Multi-Attribute Tradespace Exploration for Survivability. *Journal of Spacecraft and Rockets*, 51(5), 1735-1752. https://doi.org/10.2514/1.A32789

Roy, B. (1971). Problems and methods with multiple objective functions. *Mathematical Programming, 1*(1), 239–266. https://doi.org/10.1007/BF01584088

Ryschkewitsch, M., Shaible, D., & Larson, W. J. (2009). The art and science of systems engineering. *Systems Research Forum*, 03(02), 81-100. https://doi.org/10.1142/s1793966609000080

Salado, A. (2022). An experimentation framework for validating architectural properties as proxies for the ilities. *Systems Engineering*, 25(4), 342-359. https://doi.org/https://doi.org/10.1002/sys.21618

Salado, A. (2025). A Survey of Approaches for Aggregating Disparate Evidence in Support of Architectural Decisions on Ilities. *Conference on Systems Engineering Research (CSER)*, Long Beach, CA, USA, 2025.

Salado, A., Iandoli, L., & Zollo, G. (2016). Painting Systems: From Art to Systems Architecting. *INCOSE International Symposium*, 26(1), 773-787.

Scothern, D. G. C. (1991). A Knowledge Based Support Tool For The Early Stages Of Electronic Engineering Design. https://doi.org/10.24382/4417

Scott, J., and Kazman, R., "Realizing and Refining Architectural Tactics: Availability," CMU/SEI-2009-TR-006, 2009.

Sormaz, D., & Khoshnevis, B. (1999). Intelligent Manufacturing Based on Generation of Alternative Process Plans. Proceedings of 9th Int. Conference on Flexible Automation and Intelligent Manufacturing, Tilburg, 35–49.

Strandh Tholin, O. (2021). Enriched Functional Modelling Software to Assess Aerospace System Architectures. https://hdl.handle.net/20.500.12380/302889

Tsiporkova, E., & Boeva, V. (2006). Multi-step ranking of alternatives in a multi-criteria and multi-expert decision making environment. *Information Sciences*, *176*(18), 2673–2697. https://doi.org/10.1016/j.ins.2005.11.010

Tuan, T. A., Kreinovich, V., & Nguyen, T. N. (2019). Decision Making Under Interval Uncertainty: Beyond Hurwicz Pessimism-Optimism Criterion. In V. Kreinovich, N. N. Thach, N. D. Trung, & D. Van Thanh (Eds.), *Beyond Traditional Probabilistic Methods in Economics* (Vol. 809, pp. 176–184). Springer International Publishing. https://doi.org/10.1007/978-3-030-04200-4_14

Wigginton, S. A., and Jacobs, W. J., "Strength in Architecture," Army Acquisition Logistics & Technology (AL&T), January-March 2018, pp. 101-105.

Biography

Gordon Hunt is a co-founder of Skayl. His focus is on system of systems integration and semantic data architectures which enable increased systems flexibility, interoperability, and cyber security. His technical expertise spans embedded systems, distributed real-time systems, data modeling and data science, system architecture, and robotics & controls. He is a recognized expert in industry on Open Architecture and data-centric systems and as a regular author and presenter, he speaks frequently on modern combat-system and command and control architectures. Hunt earned his BS in Aeronautical Engineering from Purdue University and his MS in Aerospace Engineering & Robotics from Stanford University.

Dr. Bryan Mesmer is an Associate Professor in the Department of Industrial and Systems Engineering and Engineering Management at The University of Alabama in Huntsville. Dr. Mesmer's research reimagines systems engineering using approaches that span traditional areas of decision theory and modeling and non-traditional areas of communication arts and psychology. He has been PI, Co-PI, or Lead Researcher on over $13M in research sponsored by NASA, Army, Navy, DoD, and others. Dr. Mesmer's latest research is exploring the differences between architecting and designing. He is also characterizing architecting through a game theoretic lens.

Marcell "OPUS" Padilla is a Senior Engineer for CRL Technologies, Inc. and is an expert on several Army, Navy, Air Force and international open architecture initiatives with extensive experience in identifying areas of overlap, synergy and recommendations for resolving incompatibility between architecture approaches. He was one of the principal authors of a Comprehensive Architecture Strategy implementing a tiered architecture approach to define enforceable architectural requirements, mechanisms and metrics to achieve an organization's the technical and business objectives. This was utilized by the Architecture Collaboration Working Group (ACWG) in development of the Future Vertical Lift Architecture Framework (FAF) Mr. Padilla part of initial cadre for the FACE initiative, principal author of the FACE Technical Standard, founding participant of the FACE Consortium and lead of several subcommittees and current Chair of the Enterprise Standing Committee. He also participated in the development of the Hardware Open Systems Technologies (HOST) standard. He has supported multiple Army and Navy acquisition projects and Army S&T projects implementing open architecture principles.

After graduating from Texas A&M University, Mr. Padilla spent twenty years in strategic development, operational and tactical planning, mission execution and evaluation as a Naval Aviator flying the F-14 and F/A-18 before joining CRL Technologies, Inc.

Anthony Edwards is a software architect at Intrepid, an SPA company, supporting the U.S. Army's Aviation and Missile Center (AvMC). He brings over 25 years of experience in software architecture, systems engineering, and software development. Mr. Edwards holds a Master's degree in Computer Science from the University of Alabama in Huntsville. He has contributed to a wide range of Department of Defense (DoD) architecture initiatives within the Army's aviation and missile defense communities.

Brian Joyner is a Software Engineer at Intrepid LLC supporting DEVCOM AvMC TDD. His research interests include software architecture & design, artificial intelligence, machine learning, and natural language processing. Brian received his Bachelor's degree in Computer Engineering from Mississippi State University in 2017 and is currently pursuing a Master's degree in Computer Science at Georgia Tech.

Alejandro Salado, Ph.D. is an associate professor of systems engineering with the Department of Systems and Industrial Engineering at the University of Arizona and the director of the systems engineering program. In addition, he provides part-time consulting in areas related to enterprise transformation, cultural change of technical teams, systems engineering, and engineering strategy. Alejandro conducts research in problem formulation, design of verification and validation strategies, model-based systems engineering, and engineering education. Before joining academia, he held positions as systems engineer, chief architect, and chief systems engineer in manned and unmanned space systems of up to $1B in development cost. He has published over 150 technical papers, and his research has received federal funding from the National Science Foundation (NSF), the Naval Surface Warfare Command (NSWC), the Naval Air System Command (NAVAIR), and the Office of Naval Research (ONR), among others. He is a recipient of the NSF CAREER Award and the International Fulbright Science and Technology Award. Dr. Salado holds a BS/MS in electrical and computer engineering from the Polytechnic University of Valencia, a MS in project management and a MS in electronics engineering from the Polytechnic University of Catalonia, the SpaceTech MEng in space systems engineering from the Technical University of Delft, and a PhD in systems engineering from the Stevens Institute of Technology.