System architecting and design space characterization

Ali K. Raz1 | C. Robert Kenley2 | Daniel A. DeLaurentis1

1Aeronautics and Astronautics, Purdue University, West Lafayette, IN, USA
2Industrial Engineering, Purdue University, West Lafayette, IN, USA

Correspondence
Ali K. Raz, Aeronautics and Astronautics, Purdue University, West Lafayette, IN, 47907, USA.
Email: akraz@purdue.edu

Abstract
This article provides a process for system architecting that incorporates a holistic approach for architecture design space characterization by integrating decision alternatives in functional, physical, and allocational design spaces and accounting for interactions. System architects are faced with numerous decisions for system form, functions, and operations when defining a system architecture. Systems designers are tasked with selecting design options which provide the necessary functionality in support of the architecture. Since modern systems, especially system-of-systems, are composed of interacting and interwoven functions and elements, it is imperative to holistically evaluate variations in the system architecture and system design, and discover interactions among and between architecture decisions and design decisions. In this article, this design space characterization is made an integral part of the system architecting process and a set-theoretic framework is developed for managing an extensive design space. The design space characterization problem is formulated as identification of the significant decisions variables and quantification of their impact on the system objectives. A Design of Experiments framework—utilizing Analysis of Variation (ANOVA) and Range Tests—is presented to holistically characterize system architecture design space including the interactions between system form, function, operations, and design decisions.

KEYWORDS
design of experiments, design space exploration, system architecting, system-of-systems architecting, trade-off analysis

1 | INTRODUCTION

System architecture realizes the operational concept of a system, embodying its components and top-level capabilities necessary for achieving the system’s mission objectives. The system architecture is constructed from a set of decisions that orchestrate the system’s structure, behavior, and performance, and directly influence its ability to create value. It is the system architecture that brings the system form and function together to achieve the system objectives and lays down the foundations for its design. Although, design definition of a system is not the primary purpose of system architecting, the system architecture can effect and in some cases dictate the design. According to Martin,1 “Architecture focuses on suitability and desirability, whereas design focuses on compatibility with technologies and other design elements and feasibility of construction and integration.” The influence of system architecture on the system design cannot be overlooked as there can be more than one possible system architecture and different system architectures make certain design options more favorable than others, which ultimately impacts the system performance. The system design cannot proceed without a system architecture, and a system architecture should not be finalized without preliminary considerations of system design. Hence, to realize the system mission and provide an effective solution, the system architects and designers need to comprehend the interactions between architectures, design decisions, and operational considerations.

Architecting begins with an operational concept and produces candidate system architectures which are then evaluated based on their fitness for purpose and to meet system objectives. Typically, once the architectures are identified (perhaps selected), the next step is to design the system, where trade-off analysis of system design decisions is performed. The problem with this approach is the disjoint treatment of system architecture and system design trade-off analysis. A common trait of today’s independent systems is that they can unequivocally be considered as complex systems and often in their operation become part of a system-of-systems (SoS).2 The dichotomy of system operations, form, and function, followed by isolated design and evaluation does not serve justice to the analysis of complex systems that by definition are comprised of interacting and interwoven functions and elements.3
The objective of this article is to provide a system architecting process where the system architecture decisions are incorporated in the system design trade-off analysis. In this article, the system architecture and design is represented as a set of decision variables in the system design space, and analysis techniques based on Design of Experiments (DoE) are proposed for characterizing the design space. Specifically, characterizing a design space means identifying the significant decision variables and quantifying the impact such decision variables have on the system performance. The article describes a framework that system architects can use to characterize implications of system architecture, high-level design decisions, and, most importantly, interactions among and between architectural decisions and design decisions. This article provides the following three contributions:

- A system architecting and design space characterization process is presented that enables decision makers to assess and comprehend implications of system operational, architectural, and design decisions on the system performance.
- A mathematical framework is developed to represent and manage the extensive design space using a set-theoretic formulation.
- A design space characterization methodology is developed based on DoE, which utilizes statistical analysis techniques such as Analysis of Variance (ANOVA) and Range Tests, to identify the significant design variables in the design space and quantify their impact on the system operational performance.

2 | LITERATURE REVIEW

Historically, system development has employed multiple methods to incorporate trade-space exploration into the process of defining the system architecture. Kennedy\(^4\) describes how the Wright brothers carried out experiments to generate data that were used to create so-called limit curves. Torenbeek\(^5\) formalized the concept of parametric trade studies that had been used by industry to design aircraft subsequent to the Wright brothers’ work. According to Torenbeek, one of the purposes of these studies is “determination of combinations of parameters, characterizing designs that satisfy specified operational requirements.” McClinton\(^6\) explicitly adopted parametric methods for application to architecture development in systems engineering. He stated that “the initial selection of a free variable constraint is usually a major architectural decision that is selected by the system architect or Chief Systems Engineer.” McClinton describes a process that synthesizes a collection of parametric trade-offs in two dimensions that are derived from applying separate analytic methods to evaluate a set of customer requirements, which may include both system performance measures and physical design constraints. Each requirements analysis result defines a subspace of a two-dimensional design space for which the requirement being analyzed is met. The intersection of all of the subspaces for which the various requirements are met defines a constrained subspace in two dimensions from which designs that comply with the requirements can be selected. McClinton provides two examples for defining satellite constellations that are parameterized using the satellite altitude and the total number of satellites as the two dimensions of the design space. He advocates using engineering judgment to select a single design point that he calls the “sweet spot” from the design space in a way that “provides some margin against analysis assumptions and accuracies.” In contrast to McClinton, set-based design\(^7–\(^9\)) advocates not selecting a single design “sweet spot” but prefers that all design options be brought forward in the design process by using set-theoretic notation to communicate the subspace in multiple dimensions from which designs that comply with the requirements can be selected.

Mavris et al.\(^10\) have made significant contributions to developing methodologies for exploring system design spaces that have culminated in the relational-oriented systems engineering and technology trade-off analysis (ROSETTA) framework. This methodology uses a quality function deployment (QFD) to identify a mapping from customer requirements in the “R” space to engineering characteristics, or metrics, in the “m” space. In their approach, it is assumed the requirements are measurable and a function of the metrics, which are in turn a function of the independent design variables “x.” A key insight in their approach is that typically the requirements and metrics can be combined into an “R-m” space that is generated as outputs of a simulation, which is run with different settings of the independent design variables as its input space. The simulation data are used to fit response surface equations that are used as surrogate models for evaluating trade-offs. The surrogates are used to generate requirements-metrics results by selecting the levels for the independent design variables for each simulation run using appropriate DoE methods for the functional form of the response surface equations.

MacCalman et al.\(^11\) use the DoE approach of ROSETTA to develop response surface equations from simulations for three metamodels:

- The operational metamodel that relates operational inputs, which are equivalent to engineering characteristics (m) in the ROSETTA QFD matrix, to output measures of effectiveness (MoEs), which are equivalent to customer requirements (R) in the ROSETTA QFD matrix.
- The physical metamodel that relates physical inputs, which are equivalent to input independent design variables (x) in the ROSETTA approach, to output design considerations, which specify the constraints on the physical design options.
- A metamodel that links the operational domain to the physical domain by relating the operational inputs to physical inputs using mathematical expressions, empirical lookup tables, or additional response surface equations derived from simulation results.

MacCalman et al.\(^11\) cross-plot the results of the metamodels to synthesize a collection of parametric trade-offs in two dimensions that are similar to the visualization technique described by Ref. 6. They separately display two different two-dimensional design spaces: the first design space is constrained to meet the customer requirements in the operational domain; the second, to meet the constraints
on physical design options in the physical domain. In contrast, the approach used by McClinton illustrates all of the trade-offs as a single two-dimensional architectural design space regardless of the source domain of the constraints generating the trade-offs.

Selva and Crawley\(^ {12}\) notes the lack of multidimensional quantitative tool sets for system architecture trade-space exploration and propose Value Assessment of System Architecture using Rules (VASSAR). VASSAR methodology builds upon the Rule-Based Expert Systems formulation to assess system architecture merit by comparing it's generated value with the stakeholder requirements. The VASSAR approach uses expert opinion to formulate capability rules, emergence rules, and performance rules, which can incorporate executable results among other forms of knowledge, to compute performance.

Kujawski\(^ {13}\) investigates various methods for implementing DoE to develop surrogate models for system architecting. He points out that the orthogonal array experiment methods such as the standard Taguchi method are inappropriate for architecting complex systems, because they severely limit the number of interactions that can be included in the surrogate model. He proposes a D-optimal DoE method that chooses the factor settings to maximize the determinant of the information matrix \(X^\top X\) of the design based on a response surface equation that uses a general linear model with main effects plus two factor interactions (MEPTFI). This is equivalent to minimizing the volume of the multivariate confidence region for the estimated regression coefficients of the response surface equation.

Research in the field of SoS have developed a number of approaches to quantify developmental, functional, and operational dependence of one or more systems on the overall SoS performance. Mane et al.\(^ {14}\) has proposed analysis of interdependencies via Markov Chains to capture delay propagation in the SoS due to developmental delays of individual systems. They do not provide a mechanism to capture system performance based on individual system design parameters. Han\(^ {15}\) has developed a Bayesian Network approach for characterizing interdependencies that quantifies capability degradation of the SoS due to individual system failures. It requires a priori information on failure probability distribution on individual systems. Guariniello and DeLaurentis\(^ {16}\) has developed a functional and developmental interdependency analysis for SoS based on Functional Dependency Network Analysis (FDNA).\(^ {17}\) It quantifies interdependencies based on a system operability measure that aggregates multiple measures of performance (MoPs) and design variables into a single measure. This lack of granularity inhibits FDNA's ability to quantify the impact of variation in individual system design options.

It is evident from the above discussion that despite the availability of a number of evaluation methods for quantifying system attributes under various assumptions to support making system design decisions, there is a lack of a holistic framework that incorporates system design space characterization into the system architecting. This article describes such a capability that connects the system operational concepts with its various architectures and high-level design decisions while enabling trade-off of architecture and design decisions on the system performance (i.e., ability to meet system objectives).

3 | System Architecture Development Process

In this article, we build upon a well-known process for synthesizing system architectures that was originally articulated by Levis and Wagenhals,\(^ {18}\) enhanced by Buede,\(^ {19}\) and further improved upon for complex system evaluation by Kenley et al.\(^ {20}\) This article proposes a holistic architecture design and evaluation methodology that has evolved from these approaches and integrates a design space characterization capability to systematically identify significant decision variables and quantify their impact on the system performance.

Figure 1 illustrates the historical evolution of the system architecting process that concludes with the process proposed in this article. The system architecting process as formulated by Levis and Wagenhals\(^ {18}\) included the following key constituent elements: Operational Concept, Functional Architecture View, Physical Architecture View, Technical Architecture View, Dynamics Model, and Executable model. Incorporating this process as part of the "Engineering Design of Systems" textbook, Buede\(^ {19}\) proposed a bifurcation of the Operational Concept into functional architecture and a physical architecture, followed by their integration to form an allocated architecture. Kenley et al.\(^ {20}\) developed an approach to use agent-based modeling, derived from the allocated architecture, to create an executable discrete-event simulation for independently operating physical systems that interact and perform different functions. Building upon Buede's and Kenley's contributions, Raz et al.\(^ {21}\) proposed a decomposition of Dynamics model into Functional Dynamics and Physical Dynamics that results from their respective architectures.

Beginning from the operational concept and proceeding all the way to the architecture evaluation using executable and dynamics models, the system architects are faced with numerous decisions at each step of the architecting process. The need to characterize impact of decision variables is implicitly implied in the architecting process, although no formal methodology is provided to achieve this objective. This article fills this gap by proposing "Architecture Design Space Characterization" as an integral element of the system architecting process. The architecture design space characterization tracks the various decisions throughout the artifact development of system architecting process in Figure 1 and provides a mathematical framework to manage, assess, and quantify the most significant decision variables. The following paragraph provides a brief summary of different steps of the architecting process while the remainder of this article is dedicated to the "Architecture Design Space Characterization" discussion.

The operational concept of the system (or SoS) specifies the mission and usage scenarios of the system and is used to develop a functional architecture, which provides a set of functions along with the functional flow required to achieve the mission objectives. In the physical architecture, available technologies and platforms are assessed to provide a collection of physical capabilities that can perform the functions identified in the functional architecture (in the case of SoS these technologies, or platforms, are independent systems). The allocated architecture then imposes the requirements to execute functions on the physical assets. The functional dynamics and physical dynamics
models describes the dynamic behavior of functions and physical assets, respectively, while the executable model links the functions and physical assets as identified in the allocated architecture and simulates the architecture performance. The executable model produces the MoPs (metrics in the ROSETTA framework) or MoEs (customer requirements in the ROSETTA framework), which establishes the performance of allocated architecture to meet the mission requirements from the operational concept, given the functional and the physical dynamics model. Since there can be multiple feasible allocated architectures, mission requirements, and functional and physical dynamic considerations, the architecture design space characterization quantifies the impact of choosing different options.

Section 4 provides a detailed formulation and development of the "Architecture Design Space Characterization" step of the process in Figure 1. A reader unfamiliar with the formulation and development of the operational concepts, functional, physical, allocated architectures, executable model, and functional and physical dynamics is referred to the following Refs. 18–22.

4 | ARCHITECTURE DESIGN SPACE CHARACTERIZATION

4.1 | The need for architecture design space characterization

A system architect is faced with numerous decisions when deciding upon and designing the functional, physical, and allocated architecture
of a system to meet operational concept requirements. Selva et al.\(^{23}\) describes seven requisite decision-making tasks for developing a system architecture. These tasks are listed in Figure 2, which also provides their mapping to the corresponding artifacts of the system architecting process depicted in Figure 1. For example, Figure 2 shows that the decisions related to “mapping form and function” fall under the allocated architecture definition while decision related to “decomposing/aggregating form” correspond to the “physical architecture.”

In formulating and designing a system architecture, the system architect must formulate alternatives and generate a decision space for each of the decision-making tasks and then justify selecting a particular alternative. Parnell et al.\(^{24}\) advocates that “the decision space needs to be as large as possible to offer the most potential to create value.” It can be seen from Figure 2, that each artifact of the system architecting process would likely involve a number of distinct decisions with multiple alternatives. Hence, each artifact of the system architecting process manifests its own design space, that is, a set of decision variables each with their own independent alternatives (design space is formally defined in Section 4.2). The executable model is used for the evaluation of this design space to produce the value measures (i.e., MoPs or MoEs) and assessing system performance to comprehend impact of alternatives. It is important to note that although the decisions for each artifact may originate independent of one another, their evaluation needs to be done holistically. The challenge for system architects, then, is how to accomplish this holistic evaluation of extensive design space, and a providing methodology to do so, is the prime contribution of this article.

### 4.2 System architecture design space nomenclature

The system architecture design space consists of system architecture decisions that a system architect needs to evaluate. The presence of a decision inherently implies that there is more than one alternative for a given decision. Since a decision inherently affects the system design, in this article, a system architecture decision is called a design variable and decision alternatives are called design options which can be binary, discrete, interval, or categorical. Formally, the system architecture design space, then, is defined as:

**System Architecture Design Space** is a finite set of one or more system architecture design variables, where each system architecture design variable is comprised of two or more design options.

Let \(DV_i\) be the \(i\)th design variable of the system architecture and \(DO_{ij}\) be the \(j\)th design option of \(i\)th design variable.

Let \(DS_{\text{SysArch}}\) be the system architecture design space, then, in accordance with the above definition, the system architecture design space is given by

\[
DS_{\text{SysArch}} = \{DV_1, DV_2, DV_3, \ldots, DV_n\} \quad n \geq 1,
\]

where \(DV_i\) is given by

\[
DV_i = \begin{cases} 
\{DO_{i1}, DO_{i2}, DO_{i3}, \ldots, DO_{im_i}\} & \text{if } m_i \geq 2, \\
\emptyset & \text{otherwise},
\end{cases}
\]

where \(\emptyset\) is the empty set.

The constraints \(n\) and \(m_i\) imply that for a system architecture design space to exist there must be at least one design variable in the system architecture design space (i.e., \(n \geq 1\)) and for a \(DV_i\) to exist there must be at least two design options (i.e., \(m_i \geq 2\)) for that design variable, respectively. Hence, a design variable set by definition excludes singletons. Furthermore, \(m_i\) is the total number of design options available for the \(i\)th design variable (\(DV_i\)). If there is only one available design option (i.e., \(m_i = 1\)) for \(DV_i\), then there is no decision that needs to be evaluated for \(DV_i\), and we define \(DV_i\) to be the set that contains the empty set to indicate that there is no decision to evaluate and to ensure that the cardinality of the set is 1 for the purpose of enumerating the total number of design options in the architecture design space.

From a systems engineering perspective, Equation 2 provides a set-based design representation of the different options available for a particular solution of a given design variable, whereas Equation 1 ascribes these design options to the system architecture. A
system architecture configuration is realized when individual design options are identified for all design variables in the system architecture design space. The system architecture design space, therefore, is comprised of a morphological array with rows given by all the design variables and columns containing the corresponding design options. The morphological array of the system architecture design space given by Equation 1 is provided in Table 1. In Table 1, \( m_1, m_2, \) and \( m_n \) are the total number of design options available for \( DV_1, DV_2, \) and \( DV_n \), respectively (it should be noted that \( m_1, m_2, \) and \( m_n \) may or may not be equal).

A system architecture configuration is obtained when a single design option is selected for each row of the morphological array. Therefore, all possible system architecture configurations are given by a Cartesian product of all system architecture design variables, that is,

\[
\Omega = DV_1 \times DV_2 \times \ldots \times DV_n, \tag{3}
\]

where \( \Omega \) is a super set of system architecture configurations, which identifies an ordered set of individual design options for all design variables in the system architecture design space. Mathematically, a system architecture configuration, say \( SysArch_{Config} \), can generally be expressed as

\[
\Omega \ni SysArch_{Config} = \{DO_{1a_1}, DO_{2a_2}, DO_{3a_3}, \ldots, DO_{na_n}\}, \tag{4}
\]

where

\[
Config = \{a_1, a_2, a_3, \ldots, a_n\} \\
\quad a_1 \in \{1, \ldots, m_1\} \\
\quad a_2 \in \{1, \ldots, m_2\} \\
\quad a_3 \in \{1, \ldots, m_3\} \\
\quad \vdots \\
\quad a_n \in \{1, \ldots, m_n\}. \tag{5}
\]

A specific system architecture configuration is an element of \( \Omega \) and is obtained when specific values for parameters \( a_1, a_2, a_3, \ldots, a_n \) are identified. For example, consider the following two different system architecture configurations:

\[
\Omega \ni SysArch_A = \{DO_{11}, DO_{21}, DO_{31}, \ldots, DO_{1n_1}\}, \tag{6}
\]

\[
\Omega \ni SysArch_B = \{DO_{1m_1}, DO_{2m_2}, DO_{3m_2}, \ldots, DO_{nm_n}\}. \tag{7}
\]

The system architecture configurations, \( SysArch_A \) (Equation 6) and \( SysArch_B \) (Equation 7), differ in their selection of design options for each design variable in the system architecture design space. In \( SysArch_A \), the first design option of all variables is selected, that is, the elements of \( Config \) in Equations 4 and 5 are \( a_1 = 1, a_2 = 1, a_3 = 1, \ldots, a_n = 1 \). Whereas, in \( SysArch_B \), the last design option for all design variables is selected for the system architecture, that is, the elements of \( Config \) in Equations 4 and 5 are \( a_1 = m_1, a_2 = m_2, a_3 = m_3, \ldots, a_n = m_n \). The total number of maximum possible system architecture configurations in the system architecture design space is equal to the cardinality of \( \Omega \), which is given by

\[
|SysArch_{TotalConfigs}| = |\Omega| = \prod_{i=1}^{n} |DV_i| = \prod_{i=1}^{n} |DV_i| \in DS_{SysArch}. \tag{8}
\]

The various system architecture configurations all employ different design options for the same design variable. Consequently, the different system architecture configurations are expected to have different performance. The end goal of system architecture design space characterization is to identify the design variables that have the most impact on the system performance and to quantify the impact of variations in their design options.

### 4.3 System architecture design space taxonomy

The system architecture development process, as shown in Figure 1, provides four major artifacts for representing system architectures, which are the operational concept, the functional architecture, the physical architecture, and the allocated architecture. Instantiating and specifying a system architecture then requires the functional and physical dynamics along with the executable model. Development of each of these artifacts imposes its own particular set of design variables for the system architecture design and evaluation. Hence, the system architecture design space can be identified as a collective set of the various system design variables that arise from the system architecture development process. The decomposition of the system architecture design space into its operational, functional, physical, and allocational components is illustrated in Figure 3, which emphasizes the dependence between the constituent elements of the system architecture process. A general discussion of the various system design considerations which fall within each category of the design space and their representation using the nomenclature developed in Section 4.2 is provided in the following paragraphs.

#### 4.3.1 Operational design space

The operational design space identifies decision variables associated with the operational concept of the system. Broadly speaking, the operational design space variables are used to identify the what, where, when, who, why, and how of the system's operations as delineated in the Guide to the Preparation of Operational Concept Documents (OCD) ANSI/AIAA25.

- **What**: These are the top-level capabilities required of the system to achieve the system's mission objectives. These capabilities are described from an operational point of view and can include system...
top-level components. Necessary mission phases or modes may also be described here. These also include the external systems and any further definition of interfaces to these systems that constrains the design of the system of interest, for example, requiring data transmission to an external system to be via fiber-optic communication.

- **Where**: These are the environments, such as geographical and physical locations of facilities and interfacing systems, within which the capabilities are required to be performed and supported.

- **When**: These describe activities, tasks, flows, precedence, concurricencies, and other time/sequence-related elements necessary for the user to achieve the mission objectives in each of the various mission modes and conditions. This may also include information as to system development and operational availability dates.

- **Who**: These describe the interactions among the various human elements within the system including their interfaces with people external to the system.

- **Why**: These provide the rationale behind any established partitioning of the mission tasks between the system components and the operators, and the reasoning for specific sequences of activities or tasks. For example, an important function of an OCD is to provide the rationale behind the definition of the level of technical expertise required of the system operators. This will provide a basis for the definition of a set of system requirements and designs with a consistent level of complexity and sophistication.

- **How**: These tie together the other elements (the what, where, when, who, and why) to describe how the system is expected to be used, operated, maintained and, ultimately, retired in the given environment, under all significant conditions. The emphasis should be on concepts and should avoid any system design or implementation inferences.

The variations in the operational concept mission requirements (i.e., what, when, and how) and operational/environmental considerations (i.e., where, who, and why) can be modeled as a design variable described in Equation 2. Let $OC_i$ be the $i$th operational design space decision and $OP_{ij}$ be the $j$th operational parameter of the $i$th operational design space decision, then:

$$OC_i = \begin{cases} \{OP_{i1}, OP_{i2}, OP_{i3}, \ldots, OP_{im_i}\} & \text{if } m_i \geq 2, \\ \{\emptyset\} & \text{otherwise.} \end{cases}$$ \hspace{1cm} (9)$$

Although the formulation of a design variable (DV) and an operational concept variation (OC) is mathematically identical, the distinction between the two is important nonetheless. A design variable inherently implies a decision that a system architect has some control over, whereas, an operational concept variation may be outside the control of a system architect. However, the impact of any variations in the anticipated operational concepts need to be identified and quantified with the acknowledgment that choice of a particular operational concept parameter, $OP_{il}$, is likely to be outside the control of a system architect. This is particularly important for addressing uncertainty in the system operational environment with the end goal of assessing how the design variables in the functional, physical, and allocational design spaces are able to robustly withstand uncertain operations.

The operational design space is a finite set that constitutes the operational considerations ($OC_i$). Using the nomenclature introduced in Section 4.2 and Equation 9, the operational design space can be represented as

$$DS_{Oper} = \{OC_1, OC_2, \ldots, OC_\sigma\},$$ \hspace{1cm} (10)

where $DS_{Oper}$ is the operational design space; $OC_i$ is the $i$th operational concept decision variable as derived from Equation 2; and $\sigma \in \mathbb{N}$ is the total number of operational concept decision variables.
4.3.2  |  Functional design space

The functional design space of the system architecture includes design variables resulting from the functional architecture and functional dynamics. The functional architecture identifies the various decision variables pertaining to the functions that are to be employed and the possible functional flows between functions that are to be maintained to realize the operational concept. The functional dynamics identifies how the functions can be implemented in order to establish preliminary expectations for architecture performance.

In the context of design space nomenclature introduced in Section 4.2, the functional architecture can be viewed as establishing the design variables for the functional design space while the functional dynamics, then, provide the design options for the design variables identified in the functional architecture. For example, consider the operational concept of an aircraft tracking system: detecting an aircraft, generating measurements of the aircraft position and velocity, and developing an aircraft track are the various functions that are identified in the functional architecture. In the functional dynamics, methods for performing these functions are identified, for example, using a Kalman Filter or a Multihypothesis Tracker for developing aircraft tracks.

Let \( DV_{\text{Func} - i} \) be the \( i \)th design variable of the functional design space, then in accordance with Equation 1, the functional design space, \( DS_{\text{Func}} \) is given by

\[
DS_{\text{Func}} = \{ DV_{\text{Func} - 1}, DV_{\text{Func} - 2}, \ldots, DV_{\text{Func} - b} \},
\]

where \( b \in \mathbb{N} \) is the total number of functional design variables; each functional design variable, \( DV_{\text{Func} - i} \), is composed of the functional design options \( DO_{\text{Func} - i} \), and in accordance with Equation 2, is given by

\[
DV_{\text{Func} - i} = \begin{cases} 
\{ DO_{\text{Func} - i1}, DO_{\text{Func} - i2}, DO_{\text{Func} - i3}, \ldots, DO_{\text{Func} - im} \} & \text{if } m_i \geq 2, \\
\emptyset & \text{otherwise.}
\end{cases}
\]

4.3.3  |  Physical design space

Similar to the functional design space, the physical design space results from the decision variables that emerge during the physical architecture development and physical dynamics of the system architecture. The physical architecture defines decision variables that pertain to the individual physical components (or systems and platforms in the case of SoS) and their connectivity that collectively comprise the physical architecture. The physical dynamics provide the decision variables that specify the various options for realizing the physical architecture. For example, consider the aircraft tracking example in which the physical systems can be the sensors, the information processing systems, and the air traffic control center. The physical architecture identifies the physical existence of these systems and the connectivity between these systems. The physical dynamics then specifies individual capabilities of these systems, such as the computational power and/or storage available for performing functions, and communication frequency and network bandwidth for information exchange between different physical systems.

4.3.4  |  Allocational design space

The allocational design space consists of the decision variables that arise when functions are to be allocated to the physical components/systems. For various system applications, many different allocation strategies have been developed that provide one-to-one and one-to-many allocations\(^ {26} \) and centralized, decentralized, and/or distributed/hybrid allocations of functionality to physical resources\(^ {20,21,22} \). For defining the allocational design space, it is important to recognize that different allocation strategies may require specific functional implementation and information flow between physical resources, and therefore generate different system performance.

Utilizing the design space nomenclature introduced in Section 4.2 and Equation 1, all of the allocational decisions are represented as allocational design variables, \( DV_{\text{Alloc} - i} \), and their design choices as allocational design options (e.g., one-to-one or on-to allocations, etc.), \( DO_{\text{Alloc} - ij} \). The allocational design space, \( DS_{\text{Alloc}} \) is given by

\[
DS_{\text{Alloc}} = \{ DV_{\text{Alloc} - 1}, DV_{\text{Alloc} - 2}, \ldots, DV_{\text{Alloc} - d} \},
\]

where \( d \in \mathbb{N} \) and each \( DV_{\text{Alloc} - i} \) is given by

\[
DV_{\text{Alloc} - i} = \begin{cases} 
\{ DO_{\text{Alloc} - i1}, DO_{\text{Alloc} - i2}, DO_{\text{Alloc} - i3}, \ldots, DO_{\text{Alloc} - im} \} & \text{if } m_i \geq 2, \\
\emptyset & \text{otherwise.}
\end{cases}
\]

4.4  |  Design space quantification

4.4.1  |  Design space quantification problem formulation

It is evident from the preceding discussion that the system architecture design space can be extensive, heterogeneous, and complex. The system architect has to identify which of these design variables are significant and then quantify the impact of varying the design options of the significant design variables. The end goal is to fully characterize the implications of selecting one design option over the other, which is easy to state but challenging to achieve. At the one end of system architecture design variables, there are considerations for the operational concept such as the operational environment, the frequency of occurrence of required system activities, and the acceptable response time for completion of activities. In the middle, there are considerations of functional allocation strategies and system architecture properties. At other end, lies the logical and mathematical implementation
of behaviors and algorithms. Further adding to the complexity of design space is the interdependence of design variables. The outer loop in Figure 3 indicates that the design variables residing in each quadrant are interdependent on the others, implying that the design variables in one quadrant may not be evaluated independently. For example, the allocational design space may not be independently evaluated without taking into account the physical design space variables. Consequently, it becomes important for the system architects and decision makers to thoroughly investigate this design space and understand the holistic impact of variations in the design variables.

By definition, the system architecture design space, that is, $DS_{SysArch}$ includes all the design variables that a system architect must take into account (Section 4.2 and according to Figure 3 and as discussed in Section 4.3), these design variables are identified in the operational, functional, physical, and allocational design spaces. Hence, the $DS_{SysArch}$ can be compactly expressed as

$$DS_{SysArch} = DS_{Oper} \cup DS_{Func} \cup DS_{Phys} \cup DS_{Alloc}. \quad (17)$$

Equation 17 can be easily expanded by substituting Equations 10, 11, 13, and 15 to provide a full scope of system architecture design space characterization problem.

$$DS_{SysArch} = \{ OC_1, OC_2, \ldots, OC_n, DV\_{Func-1}, DV\_{Func-2}, \ldots, DV\_{Func-m} \}$$

Characterization of the system design space demands the system architect identifies which of the design variables in Equation 18 have a significant impact on the performance and how the system architecture performance varies with different design options of a significant design variables. For instance, let us consider that the system architect needs to characterize the performance implications of $DV\_{Func-1}$, which is part of the system functional design space. We know, from Equation 12, that $DV\_{Func-1} = \{ DO\_{Func-11}, DO\_{Func-12}, DO\_{Func-13}, \ldots, DO\_{Func-1\_m1} \}$ and $m_1$ is the total number of design options available for $DV\_{Func-1}$. The two problems that the system architect has to solve are: 1) to determine if $DV\_{Func-1}$ is a significant design variable, that is, is there any noticeable difference in the system architecture performance based on selecting different design options $DO\_{Func-11}, DO\_{Func-12}, DO\_{Func-13}, \ldots, DO\_{Func-1\_m1}$; and 2) if there is a noticeable impact, then how much is that difference and which is one the best design option. However, given the interdependence of subdesign spaces ($DS_{Oper}, DS_{Func}, DS_{Phys}$, and $DS_{Alloc}$) as shown in Figure 3, solution to this problem is more complicated than a monolithic and isolated evaluation of system architecture based on varying design options of a single design variable. The challenges that a quantification methodology needs to address for system architecture design space characterization are highlighted in Figure 4 and discussed in the following section.

4.4.2 | Design space quantification challenges

- **Design Space Dimensionality: Large**
  It can be clearly seen from the system architecture design space taxonomy discussions and Equation 18 that the total number of design variables required for system architecture evaluation typically is quite large. Theoretically, at least two physical resources must be available to form a system architecture” and based on the system architecture design space taxonomy, the system architecture design space can still contain numerous design variables (e.g., allocation of functions, selection of functional implementation, communication between physical components, etc.)

Practically, however, the system architecture will consist of many more than two components and the number of design variables increases accordingly. The dimensionality of the design space corresponds to the total number of possible system architecture configurations that a design space can generate and can be easily calculated from Equation 8. It is important to note that the system architecture design space may not include every possible design consideration, especially design considerations with fixed or predetermined solutions are excluded from the system architecture design space as defined by Equations 1 and 2.

- **Design Variables Type: Heterogeneous**
  The design variables that constitute the system architecture design space have distinct characteristics. Some design variables are categorical variables such as the functional allocation strategies, functional implementations, and algorithm choices. Some design variables are ordinal such as operational environment considerations. Furthermore, some design variables can be classified into continuous intervals such as information exchange and communication rates between physical systems, and algorithmic parameters.

- **Design Variable Cardinality: Unbalanced**
  The cardinality of a set is the number of elements in a set. The cardinality of a design variable is the number of different design options available for the given design variable. For the system architecture design space, the cardinality of the entire set of design variables typically is unbalanced, that is, the cardinality of each design variable may be different. For example, there may be only a few options of allocating a functionality to the physical systems but many options of algorithms for implementing that functionality.

- **Design Variable Interactions: Significant**
  The system architecture design space taxonomy may incorrectly allude to adequacy of independent evaluation of operational, functional, physical, and allocational design variables. An independent evaluation of a design variable means that the system architecture level impact of a given design variable can be evaluated and quantified solely by varying only its design options. This would mean that a functional design variables (for example) can be evaluated without taking into account any variations in other functional design variables and other operational, physical, and allocational design variables. Unfortunately, interactions between design variables continue to dominate the system architecture performance in many practical applications.\textsuperscript{11,13,28,29} Therefore, it is imperative for the system architect to identify and quantify these interactions, in order to fully characterize the design space of a system.
4.4.3 Design space quantification methodology

The system architecture design space quantification requires a methodology that is able to address the challenges identified in the previous section while meeting the objectives of design space characterization, that is, identify the significant design variables and quantify the impact of variations in significant design variables on the system architecture. In this article, we propose DoE as the system architecture quantification methodology. The next section provides detailed formulation of DoE for this purpose including justification of how DoE addresses the quantification challenges and satisfies the design space characterization objectives.

4.5 DoE for design space characterization

DoE utilizes statistical analysis tools to identify and quantify the key design variables and provides a methodological approach for design space characterization. Introducing DoE Rebak and Shaikh,30 states that:

The objective of experimental design is to provide the researcher or a practitioner with a statistical method that determines which input variables are most influential on the output and where to set the influential input variables so that the output is either maximized, minimized, or nearest to a desired target value.

The purpose of this section is to provide a practitioners guide for employing DoE for system architecture design space characterization. First, we discuss the DoE nomenclature and map it to the system architecture design space. Second, we discuss selection of experimental design methods that govern the strategy for conducting the experiments such that the challenges depicted in Figure 4 are satisfied. Third, we provide the mathematical DoE formulation followed by statistical tests for achieving the design space characterization objectives of identifying and quantifying significant design variables. Finally, we discuss the limitations that a system architect must take into account when employing the DoE.

<table>
<thead>
<tr>
<th>TABLE 2</th>
<th>Design space to DoE mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Design space parameters</strong></td>
<td><strong>Design of experiments nomenclature</strong></td>
</tr>
<tr>
<td>MoPs or MoEs</td>
<td>Response Variable</td>
</tr>
<tr>
<td>Design Variables</td>
<td>Factors</td>
</tr>
<tr>
<td>$(DV)_j$</td>
<td>pth Treatment level</td>
</tr>
<tr>
<td>$(DO)_{jp}$</td>
<td>of jth factor</td>
</tr>
</tbody>
</table>

4.5.1 DoE nomenclature

Some of the most important and frequently used terms in the DoE literature are response variable, factors, and treatment levels. Response variable is the output of an experiment, factors are the input variables that are expected to influence the response variable, while the treatment levels are input values of different factors. The DoE objective is to infer what treatment levels of which factors are most influential on the response variable by designing an experiment where multiple values of response variable are obtained and statistically analyzed.

In the design space quantification context, the MoEs or MoPs which come from the executable model become the experiment response variables, the design variables become the experiment factors, and design options become the factor treatment levels. A mapping of system architecture design space nomenclature to DoE nomenclature is provided in Table 2.

For simplicity of notation in the proceeding sections, all design variables and their corresponding design options will be called and represented as factors and treatment levels, respectively.

4.5.2 Selecting experimental designs and constructing the design array

The factors and the treatment levels of the system architecture design space are identified by mapping the design variables and the design options in accordance with Table 2. The next step in the DoE is to identify an underlying design for conducting the experiment. The
underlying experimental design guides the data collection of the response variable (MoP or MoE) such that the required inferences regarding factors (Design Variables) and their treatment levels (Design Options) can be obtained from the DoE. "The selection of an experimental design depends upon a number of different variables such as the total number of factors, treatments levels of each factor, available resources to conduct the experiment and the required inference regarding factors." The experimental design determines the total number of experiments that will be conducted and is the driving force behind the entire data collection process, which is then used for statistical analysis.

When the number of factors is large, which is expected in the system architecture design space, a special class of DoE methods known as the factor screening methods are used to conduct the experiment. In DoE literature, a number of different factor screening methods have been developed for various applications. However, these methods come with a set of assumptions and limitations that must be understood before their application to a given problem. This section evaluates the ability of some of the common DoE methods to meet the system architecture design space characterization challenges.

### Full Factorial Designs (FFDs):

FFD is one of the most widely applied DoE experimental design method. An FFD requires evaluation of all possible combinations of all factors at all treatment levels. An FFD design for x number of factors all with y treatment levels is called an $y^x$ FFD. For example, an FFD of five factors with three treatment levels each will be called “3^5 FFD” and will require a total of $3^5 = 243$ experiments for statistical analysis. A mixed level design, where different factors have different treatment levels, can be represented in a similar fashion. For example, an FFD of five factors with three treatment levels each and remaining factors at four levels each will be called “2^3 × 4^2 FFD” and will require a total $2^3 × 4^2 = 128$ experiments for statistical analysis.

FFDs are very powerful in the sense that all possible information about the design space is included in the experiment. Hence, the FFDs can be used to derive inferences regarding the main effects as well as the interaction effects. Furthermore, FFDs are not limited by the heterogeneity of factors (Design Variables) or mixed-level treatments (i.e., unbalanced cardinality of Design Options). However, as the number of factors and/or treatment levels increases (e.g., 100s of factors or 10s of treatment levels), FFDs can become impractical, overly expensive, or computationally prohibitive. Nevertheless, a design space spanned by up to 10,000 unique configurations can be easily analyzed via FFD in JMP, a commonly used DoE software package. For example, such a design space includes a $2^{13}$ FFD or $3^8$ FFD meaning that up to 13 factors with two treatment levels each or up to eight factors with three treatment levels each can be easily analyzed via FFD. In the light of the above, FFDs are deemed suitable for design space characterization insofar as the dimensionality of the design space (i.e., 8) does not exceed the available DoE software’s computational capabilities.

### Fractional Factorial and Plackett–Burman Designs:

In order to address the shortcoming of FFDs, fractional factorial designs are developed, which assume that a full replicate of design space is not necessary, especially for main effects and lower order factor interactions models (such as the MEPTFI model of Equation 19). These experimental designs allow a significant reduction in the number of data points required for statistical analysis as compared to the FFD. For example, where an FFD of five factors at three levels requires $3^5 = 243$ data points, a fractional factorial design of the same design space requires only 27 data points (given by the Taguchi L-27 array). The ability of fractional factorial designs to reduce the number of data points required for statistical analysis makes them a cost-effective method for conducting DoE. However, fractional factorial designs are not well-formulated to simultaneously address the four design space quantification challenges identified in Figure 4. Typically, fractional factorial designs can be formulated for up to 31 two-level factors but Montgomery notes that relaxing the two-factor interaction assumption is often required for mixed-level factor treatments in fractional factorials. This implies that fractional factorial designs can either accommodate a fixed-level design space (same number of design options for each design variable) with interactions (between the design variables in the MoE/MoP response surface equation) or a mixed-level design space (different number of design options for each design variable) with no interactions. Neither limitation makes the fractional factorial design a suitable candidate for holistic design space quantification. Similarly, another common factor screening method widely used in the DoE for large number of factors is Plackett–Burman design. However, Plackett–Burman designs are only able to screen for main effects and assume no interactions.
Sequential Bifurcation (SB) Designs:
Recognizing the lack of DoE methods that allow characterization of an extensive design space, the SB method was introduced. SB is a powerful method that allows for characterization of main effects in a large design space (~100–200 factors) but assumes no interaction between factors. Building upon the fundamentals of the SB method, “Controlled Sequential Bifurcation – X (CSB-X)” was introduced that allows for evaluation of a large design space in the presence of interactions. The only limitation of the CSB-X method is that it is not formulated to address mixed-treatment levels.

Optimal Designs:
The National Institute of Standards and Technology (NIST) Engineering Statistics Handbook recommends the use of computer-aided Optimal Designs when a generalized DoE framework is not applicable. Computer-aided Optimal Designs are a special class of experimental designs that generate experimental conditions based on a user-defined criteria tailored for a specific application. For example, a user will specify the number of factors, the type of each factor, and the required inferences (i.e., main effects and level of interactions), and the computer-aided optimal design will output the experimental configurations. Hence, the computer-aided optimal designs are capable of generating a specific experimental design that simultaneously addresses all four challenges of the design space characterization. While methods like the FFD remain generally applicable to various problem domains, an optimal design created for a specific system remains exclusive for its design space characterization. Ref. 13 provides an example of employing optimal experimental design for system architecting.

Table 3 summarizes the above discussion by illustrating DoE experimental design coverage to the four design space quantification challenges discussed earlier.

Experimental Design Array:
Once the experimental design method is identified, the specifics of the experiment and the required data that will be collected for statistical analysis is documented in an experimental design array. Each row of the experimental design array identifies an experimental condition where the columns contain the treatment levels (Design Options) of all factors (Design Variables) that are included in the experimental design. The last column contains the corresponding value of the response variable (MoP or MoE), which is obtained after conducting the experiment at the identified factor treatment level combinations. Table 4 provides an example of a 3 factorial experimental design array, indicating a total of nine experimental conditions along with the corresponding treatment levels for each experimental condition. In Table 4, X1 and X2 are the two factors with treatment levels “L1,” “L2,” and “L3” and values for column “Y” are obtained after conducting the experiment at the specified treatment levels of each factor.

4.5.3 DoE formulation
In DoE, the main effect of a factor assess the individual impact of different treatment levels (Design Options) of that factor (Design Variable) on the response variable (MoP/MoE); the interaction effect is the impact of a combination of two or more factors on the response variable. A two-way interaction corresponds to a combined effect of different treatment levels of two factors on the response variable.

Once the number of experiments and their configuration is known, all experiments are performed to collect response variable values. In order to identify significant main effects and interactions, as well as quantify their impact on the response variable, the data collected in the experimental design array are analyzed via a DoE mathematical model. The selection of a DoE model is dependent on the required inferences, for example, if only main effects are to be evaluated a main effects model is used. Since interactions are expected in the system architecture design space (as discussed in Section 4.4.2, the effects are characterized by the MEPTFI model). Let (X1, X2, X3, ..., Xk) be k number of factors where each of the factors maybe categorical, ordinal, or continuous type with two or more treatment levels. Mathematically, the MEPTFI model is given by

\[ Y_p = \beta_0 + \sum_{j=1}^{k} \beta_{jp} X_{jp} + \sum_{j=1}^{k-1} \sum_{l=j+1}^{k} \beta_{jlp} X_{jp} X_{lp} + \epsilon_p \]  

In the above equation,

\( Y_p \) is the quantitative response of the response variable for the pth experimental condition (i.e., the last column of the experimental design array); \( X_{jp} \) indicates the treatment level of factor \( X_j \) for pth experimental condition and k is the total number of factors; \( X_{jp} X_{lp} \) is the interaction term representing interaction between factor \( X_j \) and factor \( X_l \) for the pth experimental condition; \( \beta_{jp} \) is the mean of response variable due to treatment \( X_{jp} \); \( \beta_{jlp} \) is the mean of response value due to two-way factor interaction \( X_{jp} X_{lp} \); \( \beta_0 \) is the overall mean of the response variable; and \( \epsilon_p \) is the error term and accounts for uncertainty.

In Equation 19, the \( Y_p \) values for all p experimental conditions are known from the experimental design array along with the treatment levels (i.e., \( X_{jp} \) and \( X_{jp} X_{lp} \)), while all the \( \beta \) parameters and the error \( \epsilon_p \) are unknown. The coefficients \( \beta_{jp}, \beta_{jlp}, \) and \( \epsilon_p \) can all be determined by regression. Subsequently, a variety of statistical analysis tools, such as ANOVA and Range Tests, are then applied to identify and quantify main and interaction effects in the MEPTFI model. These statistical tests provide inferences to characterize the design space and are discussed next.
4.5.4 DoE statistical tests

The main purpose of employing DoE is to derive inferences regarding different factors and their treatment levels. This objective can be achieved by employing ANOVA and Range Tests once the data in experimental design array are available.

ANOVA:
In DoE, ANOVA identifies significance of main and interaction effects of different factors on the response variable. ANOVA can determine if changing the treatment levels (Design Options) of one factor (Design Variable) has a statistically significant impact on the response variable (MoP or MoE). ANOVA utilizes a linear regression to fit the experimental data provided in the experimental design array to an ANOVA model. The ANOVA model specifies the inference terms that are to be investigated via DoE. For example, the MEPTFI model of Equation 19 is an ANOVA model for investigating main effects and two-way interactions.

The unknown parameters of the ANOVA model are obtained once the model is fitted with the experimental data provided in the experimental design array, that is, \( \beta_0, \beta_j, \beta_{ij}, \) and \( \epsilon_p \) for all \( p \) experimental conditions. Using the \( \beta \) parameters values, ANOVA formulates a statistical hypothesis test to infer if changing the treatment level of one factor is expected to have a significant impact on the response variable. Let \( (\beta_1, \beta_2, \ldots, \beta_p) \) be the main effect estimate of \( u \) treatment levels of factor \( X_j \). The ANOVA hypothesis test to identify significance of a main effects is formulated as

\[
H_0 : \beta_1 = \beta_2 = \ldots = \beta_p, \\
H_a : \beta_1 \neq \beta_2 \neq \ldots \neq \beta_p 
\]  
(20)

where the null hypothesis \( H_0 \) indicates that all treatment level means are equal and hence there is no statistically detectable difference on the response variable if a treatment level of factor \( X_j \) varies. Whereas, the alternate hypothesis \( H_a \) establishes that variation in the treatment level has a statistically significant impact on the response variable. The hypothesis testing is evaluated using the F-test statistic. Given the widespread application of hypothesis testing, the mathematical details of F-test and criteria for rejection of \( H_0 \) are omitted here but can be easily found in DoE textbooks.\(^{31,36}\) Hypothesis test for evaluating interaction effects can be formulated in a similar fashion of Equation 20.

In order for the ANOVA and the hypothesis tests to be considered statistically valid, the data distribution of each factor treatment level is expected to be normally and independently distributed with equal variance.\(^{36}\) It is important to validate these assumptions prior to conducting hypothesis testing based on any ANOVA model. These assumptions are validated via residual analysis that utilizes the \( \epsilon_p \) parameter for all \( p \) experimental conditions. The residual analysis, as well as the remedial measures that may be necessary to be applied, are briefly discussed in Section 4.5.5.

The primary contributions of ANOVA is that it identifies which factor’s main and interaction effects are statistically significant. From the system architecture design space characterization perspective, this means that ANOVA identifies which design variables are statistically significant in the design space. However, ANOVA does not identify how the treatment levels differ and what type of impact (positive or negative) one treatment level has on the response variable. This information is obtained from range tests.

Range Tests:
Range tests provide pairwise statistical comparison methods, where different treatment levels of the same factor, or different combinations of treatment level for multifactor interaction are evaluated for statistical significance. The statistical significance between factor treatment levels can be established based on range tests. Hence, the range test can be used to perform sensitivity analysis by quantifying which treatment levels have the most impact on the response variable.

A number of pairwise comparison methods have been developed for the range tests such as Tukey Honest Significant Difference (HSD), Bonferroni and Scheffe multiple comparison tests. In pairwise comparison tests, different pairs of treatment levels (of the same factor or interaction) are tested against each other to determine their statistical significance. For example, Equation 21 shows the multiple hypothesis tests required for pairwise comparisons of all pairs of factor \( X_j \) with three treatment levels.

\[
\begin{align*}
\text{Test-1} & : H_0 : \beta_1 - \beta_2 = 0 & H_0 : \beta_1 - \beta_3 = 0 & H_0 : \beta_2 - \beta_3 = 0 \\
\text{Test-2} & : H_0 : \beta_1 - \beta_2 = 0 & H_0 : \beta_1 - \beta_3 = 0 & H_0 : \beta_2 - \beta_3 = 0 \\
\text{Test-3} & : H_0 : \beta_1 - \beta_2 = 0 & H_0 : \beta_1 - \beta_3 = 0 & H_0 : \beta_2 - \beta_3 = 0.
\end{align*}
\]

(21)

The total number of pairwise comparison performed and the test statistic for evaluating the hypothesis is given by the selected multiple comparison method.\(^{36}\) For example, the Tukey HSD test utilizes F-test statistic to test pairwise comparisons of Equation 21. Range test to test interaction effects of two factors can be constructed in a similar fashion.

From the system architecture design space characterization perspective, the range tests quantify which design options of the design variables have the most significant impact on the system MoPs or MoEs. Hence, by combining information from ANOVA and Range tests, the design space of a system can be fully characterized.

4.5.5 DoE limitations

The DoE is a versatile and powerful methodology for design space characterization. Nevertheless, it has a few limitations that the system architect needs to consider. These limitations along with mitigation strategies are described in the following paragraphs.

Number of Experimental Runs:
The selection of an experimental design, as discussed in the previous section, identifies the total number of samples that will be needed to fit the MEPTFI model of Equation 19. Selection of an FFD can require a large number of samples to be collected for the MEPTFI model. The total number of required samples exponentially increases with increasing number of factors included in the design space and it may become infeasible (cost prohibitive or computationally intractable) to obtain all the data required for FFD. For example, it was discussed earlier that an FFD of five factors with three treatment levels each would require \( 3^5 = 243 \) experimental evaluation. Including one more factor
in the design space will add an additional 486 experimental evaluations for the design space characterization based on FFD (3^6 = 729 and 3^6 - 3^5 = 486). It is important to consult subject matter experts to limit the number of potential design variables in the design space. Moreover, dimensional reduction approaches, for example, Principal Component Analysis, may be used to reduce the number of design variables and/or design options, although, further research is needed for their applicability to systems engineering problems. Nevertheless, from a practical standpoint and especially for simulation-based experiments, this limitation is largely dependent on the available computing hardware, both to run the experiment and to analyze the experiment data in the DoE selected framework.

Feasibility of Architecture Configurations:
The experimental design array specifies unique experimental conditions by identifying individual treatment levels of all factors included in the experimental design. Each experimental condition corresponds to a system configuration that will need to be executed for collecting the response variable value. Hence, it is imperative to validate that any experimental condition specified in the experimental design array will always result in a valid system configuration which can be executed for performance evaluation. System trade space feasibility assessment approaches, can be employed to identify infeasible configurations which can then be excluded from the experimental design selection. In cases, where it may not be possible to execute all possible experimental conditions, advanced DoE strategies such as incomplete designs can be employed for design space characterization.

MEPTFI Model Assumptions:
The MEPTFI model of Equation 19 requires that a set of assumptions must be met before any meaningful statistical analysis can be performed. These assumptions require that the residual errors (\( \varepsilon \)) in Equation 19 are normally distributed with zero mean and equal variance. It is important for the system architect to take into consideration the importance of these assumptions and employ remedial measures if necessary. A number of postexperiment standard remedial measures are available in the DoE literature which specify mathematical functions to transform the response value depending upon the nature of unsatisfied assumption. Additional details on these assumption, including the remedial measures, can be found in DoE textbooks.

5 DISCUSSION AND CONCLUSIONS

5.1 Intellectual contributions
This article has introduced a process for systems architecting that incorporates a holistic approach to architecture design space characterization into the approach described in the seminal article by. To provide a means for descriptive modeling of the system architecture and its design space options, a set-theoretic design space nomenclature was defined that spans the dimensions of the operational, functional, physical, and allocational views of a systems architecture, and a taxonomy was defined that emphasizes the dependence between the constituent elements of the system architecture process. The combination of the nomenclature and taxonomy provided a formalism to formulate the problem of design space characterization and to highlight the challenges of dimensionality, variable diversity, cardinality, and interactions. A mapping of the design space nomenclature to DoE nomenclature was developed to link the descriptive modeling of an architectural design space to DoE methods. Specific DoE methods were evaluated with respect their ability to meet the design space characterization challenges, and an example of an experimental design and response surface model was provided to illustrate the formulation of an experimental design for quantitatively characterizing an architectural design space. Two statistical tests were described for deriving inferences regarding different factors (design variables) and their treatment levels (design options), one for identifying significance of main and interaction effects of different factors (design variable) on the response variable, which is quantitative MoP or MoE, and for pairwise comparison of different treatment levels (design options) of the same factor (design variable), or different combinations of treatment level (design options) for multifactor (design variable) interaction. Finally, the article described some limitations of using design of experiments for design space characterization, and provided some mitigation strategies to manage these limitations.

5.2 Systems engineering impact
Throughout the system life-cycle but especially during early stages such as conceptual development and analysis of alternatives, system engineers are faced with multifaceted decisions involving multiple system attributes. Using the design space characterization approach developed in this article, system engineers can identify and quantify the significance of various decisions. This approach, which is built upon DoE, emphasizes the importance of holistic design space characterization by taking into account the interactions between different decisions. Since DoE is a well-matured methodology and a number of commercial software packages are already available which facilitate DoE implementation (e.g., R, SAS, JMP, etc.), system engineering practitioners can readily incorporate the ideas developed in this article for system architecting. While the objective of this article was to formally integrate design space characterization with system architecting, as summarized in Section 5.1, a number of publications attest to the utility of DoE for systems engineering applications. Systems engineers are encouraged to review these publications to gain a comprehensive understanding of applying DoE for system and SoS engineering.

5.3 Future work directions and research ideas
As systems and SoS continue to grow in complexity and capability, the need for design space characterization will become ever more pertinent. The more complex a system, the more interactions and interdependencies it will have. However, with increasing complexity, the design space characterization challenges will also continue to evolve, for example, the size of design space will increase making it...
difficult to detect and evaluate interactions. Furthermore, as systems are designed to fulfill a variety of mission needs, evaluating their performance with a single MoP or MoE may not remain feasible. Even for systems where a single measure is deemed feasible, system cost will need to be evaluated along with the performance measure(s). The research challenges that lie ahead for a comprehensive design space characterization are:

**Application of Multiresponse DoE:** Further applied research is required to formulate multiresponse DoE methods so that multiple MoPs and MoEs can be included for system design space characterization. A starting point for this research is the desirability approach\(^\text{23}\) where multiple MoPs or MoEs are included in a desirability function and experiments are conducted to optimize or attain a specified desirability.

**Integration of Cost Modeling:** The various system architecting decisions, which are solicited by the different artifacts of the system architecting process (i.e., functional architecture, physical dynamics, etc.) are all expected to have varying impact on the system cost. In future research, system cost modeling, such as Constrictive Systems Engineering Cost Model (COSYSMO),\(^\text{40}\) will be integrated into the system architecting process. The vision here is to be able to generate a system cost estimate alongside system performance in the executable model and subsequently utilize multiresponse DoE methods for cost-performance trades of the design space.

**Development of Quantitative Methods:** Although DoE provides robust quantitative analysis methods for system design space characterization, further theoretical research is needed to cope with extensive design spaces and higher order interactions anticipated in future complex systems and SoS. In future, smart algorithms based on machine learning and artificial intelligence techniques will be investigated for architecture design space characterization.

**ENDNOTE**

* This statement follows from the general system definition, which states that any system is composed of two or more components.

**ORCID**

Ali K. Raz (http://orcid.org/0000-0003-2562-1631)
C. Robert Kenley (http://orcid.org/0000-0003-1350-5350)

**REFERENCES**


AUTHOR’S BIOGRAPHIES

Ali K. Raz is a research scientist at Purdue University’s Center for Integrated Systems in Aerospace. His research interests are in System-of-Systems Engineering and Information Fusion Systems. Prior to joining Purdue University, he was a systems engineer at Honeywell Aerospace in Phoenix, AZ where he worked on Flight Control and Flight Management Systems. He holds a Bachelors and a Master’s degree in Electrical Engineering from Iowa State University and a Ph.D. in Aeronautics and Astronautics from Purdue University. He is a certified systems engineering professional (CSEP) from the International Council of Systems Engineering (INCOSE).

C. Robert Kenley is a professor of Practice at Purdue University’s School of Industrial Engineering in West Lafayette, IN (USA). He has over 30 years experience in industry, academia, and government as a practitioner, consultant, and researcher in systems engineering. He has published papers on systems requirements, technology readiness assessment and forecasting, Bayes nets, applied meteorology, and the impacts of nuclear power plants on employment. He is an expert systems engineering professional (ESEP) and a Fellow of the International Council of Systems Engineering (INCOSE).

Daniel A. DeLaurentis is a professor in Purdue University’s School of Aeronautics and Astronautics. He leads Purdue’s Center for Integrated Systems in Aerospace and he is also the director of Purdue University’s Institute for Global Security and Defense Innovation (iGSDI). His primary research interests are in the areas of problem formulation, modeling and robust system design and control methods for aerospace systems and systems-of-systems. This includes agent-based modeling, network theory, optimization, and aerospace vehicle modeling. His research is conducted under grants from NASA, FAA, Navy, the DoD Systems Engineering Research Center (SERC) UARC, and the Missile Defense Agency.

How to cite this article: Raz AK, Kenley CR, DeLaurentis DA. System architecting and design space characterization. System Engineering. 2018;1–16. https://doi.org/10.1111/sys.21439