Axiomatic Design meets Model-Based Systems Engineering (MBSE)

David S. Cochran [1][0000-0002-2891-8011], Joseph Smith[1][0000-0002-0198-8351], and John Fitch[1][0009-0007-1511-2362]

[1] Purdue Fort Wayne, Fort Wayne IN 46805, USA

Abstract. The be able to design sustainable systems relies on the ability to associate and communicate design information to meet customer needs. Model-Based Systems Engineering (MBSE) provides a means to integrate the classes of information that Axiomatic Design (AD) uses to establish a hybrid (best-of both) design approach. The strength of AD is that it offers axioms to make effective design decisions. MBSE can associate all sources of information for the Systems Engineering (SE) lifecycle. AD offers design veracity. MBSE manages information such as the logic and thought process required in design decision making, how design decisions influence risk, and the results of test plans to verify and validate a design. To advance AD and MBSE, an information metamodel is required to integrate information classes used by AD and MBSE. The benefit of integrating AD with MBSE is first that an overall design decision pattern may be used to kick-start any new product or system design. The Functional Requirements (FRs) of a design may then be associated with the decision pattern. Second, MBSE characterizes use cases very well. Customer Needs (CNs) and FRs may then be tied to the description of how a product or service is used. The information metamodel also ties risk to a chosen solution during FMEA. To address procedural errors described by Thompson, MBSE manages the identified need to distinguish between architectural FR – PS coupling, and performance FRm – DP coupling. When combined, AD-MBSE enables lifecycle design information integration, the ability to express new design viewpoints, and a single source of truth for a System of Interest. The information metamodel has been tested on the classical water faucet design illustration and the skin graft work by Gabela and Suh.

Keywords: system, systems engineering, systems architecture, axiomatic design, modeling, viewpoints.

1 Background

1.1 Axiomatic Design

Axiomatic Design represents the requirements derivation process as occurring across four domains [1]:

- Customer Needs (CN)
- Functional Requirements (FR)
- Design Parameters (DP) also known as Physical Solution (PS)

- Process Variables (PV)

As used in product development, AD seeks to maintain the independence of the FRs through the selection of DPs, i.e., to eliminate coupling between FRs that results from a chosen design solution. This simple and powerful insight has generated improvements to product/system designs across multiple industries and technology domains.

1.2 Observations Regarding AD Practice

Despite its well-published successes, AD practice varies widely across its broad global community due to a lack of standards in taxonomy, for example, practitioners may write FRs in a variety of styles and may often confuse FRs with other types of requirements [2,3]. Some FRs are stated as quantitative parameters to be improved, e.g., '*decrease infection probability*' (for an artificial skin graft product and grafting process) and others are stated as pure functions, e.g., '*prevent infection.*'

AD practice also varies widely with respect to how AD is applied. For instance, AD may be used to develop new products and systems, or as a language to express design innovations and inventions that others have developed without applying the rigor of AD's Axioms 1 and 2.

Thompson attributes difficulties in defining FRs to the limitations of AD's requirements classification scheme. For example, Thompson describes that there are only two categories for which requirements information can be captured; Functional Requirements and constraints. Thompson goes on to point out that additional information exists, but AD has no formal way of capturing the remaining information [2]. The additional design information (i.e., non-functional attributes of the system of interest and design goals (selection/optimization criteria) is unlikely to be effectively considered in the design process for two reasons. First, this additional design information may be deemed unimportant or unnecessary if the classification scheme does not explicitly provide a method or taxonomy for capturing the information. Second, the AD practitioner may choose to improvise by capturing additional design information as FRs. This improvision can lead to five common procedural errors when defining FRs as noted by Thompson [2]:

1. Mixing FRs with design parameters (DPs) - The designer confuses the "how" with the "what" and ultimately limits the design space by embedding the solution in the requirement.
2. Mixing FRs with other types of requirements – non-Functional Requirements, selection criteria, and optimization criteria are captured as FRs.
3. Mixing the FRs of the various stakeholders and of the system of interest – The designer captures FRs of the stakeholders vs. pure functions that the system must perform.
4. Mixing the FRs of the system of interest and of related systems – A poor definition of the system boundary can lead to the incorporation of FRs of related systems into the design of the system of interest. A mix of functions (FRs) that the system of

interest must perform with functions that related systems must perform will convolute the functional definition of the system of interest.
5. Defining negative FRs – The FR reads as what the system should not do instead of what the system must do.

1.3 AD's Limitations and Systems Engineering Objectives in the Design Lifecycle

In addition to design procedural issues noted in Section 1.3, AD's viewpoint in assessing design acceptability based primarily on maintaining the independence of FRs, is missing numerous key elements of a full life cycle domain-independent systems engineering methodology. Systems engineering as discipline defines design requirements for each phase of a product or enterprise systems and derivatives to achieve the following design results.

- The first desired result is to provide a design synthesis engine that conceives full solutions to a problem by combining multiple system viewpoints and elements in novel ways.
- Second, systems engineering requires the expression of a model of the system physical architecture and the interfaces between system elements. Design decompositions must have system build instructions that explicitly define the interfaces between the constituent parts and the configuration of the assembled parts. The limitation with AD decompositions is that the information required to do integration of the constituent parts (PSs/DPs) is not defined or captured. For example, a list or pile of parts is powerless to deliver the required FRs and performance because it does not define how to build the system.
- Third, systems engineering provides a method to model the dependencies and interactions between system functions in terms of control flow and item flow which define the inputs/outputs of matter, energy, or information.
- Fourth, system engineering requires a design to maintain alignment and completeness, correctness, and consistency between the system physical, functional, mathematical (performance) and ECAD models. Every design decision that answers, "How will this FR be satisfied?" further elaborates each of these four models to create a digital thread.
- Fifth, systems engineering provides an integrated approach for identifying and addressing system failure modes and risks. What could go wrong and what should be done to mitigate it?
- Six, A complete design decision-making methodology that screens out infeasible solutions and identifies best-performing solutions, taking uncertainty into account while capturing and communicating decision rationale.
- Seven, A published and standardized information metamodel that provides a schema (classes and relationships) to capture design information. The metamodel enables

suppliers to develop software tools that fully implement AD and to provide data interfaces with other engineering toolsets.
- Eight, A mechanism for accounting for emergent behaviors of a system that cannot be attributed to a single component in a system decomposition hierarchy.
- A technique for reliably and comprehensively identifying the derived requirements that flow from commitment to a specific design alternative in any design decision.

2 Model-Based Systems Engineering

This section discusses how the current practice of Model-Based Systems Engineering addresses the objectives of systems engineering described in Section 1.3, both procedural and gaps in capability, associated with AD theory and practice. The Systems Engineering Vision 2035 document, published in 2021 by the International Council on Systems Engineering (INCOSE), identifies five engineering practices that are deemed to be transitioning from a status of emerging to becoming Standard Practice in Systems Engineering [4].

- Product Line Engineering
- Agile Methods
- Design for Resiliency
- *Model-Based Systems Engineering (MBSE)*
- System of Systems

Referencing the INCOSE Systems Engineering Vision 2020 (published in 2007), the fourth edition of the INCOSE Systems Engineering Handbook defines MBSE as "the formalized application of modeling to support system requirements, design, analysis, verification, and validation activities beginning in the conceptual design phase and continuing throughout the development and later life cycle phases" [5]. MBSE, when contrasted with traditional document-based engineering approaches, is believed to offer significant benefits, including:

- Improved communication among stakeholders.
- Increased ability to manage system complexity.
- Improved product quality.
- Enhanced knowledge capture and reuse.
- Improved ability to teach and learn SE fundamentals.

The handbook provides an overview of leading MBSE methodologies, with details of two leading approaches, Function-Based Systems Engineering (FBSE) and Object-Oriented Systems Engineering Method (OOSEM). The handbook asserts that the "system model is the primary artifact of the SE process" without providing a definitive set of models that must be present for an engineering process to qualified as model based. Examples of system models, from ISO/IEC/IEEE Standard 15288 [6], include:

- Functional
- Behavioral

- Temporal
- Structural
- Mass
- Layout
- Network

Early proponents of MBSE created numerous innovations in modeling languages, visualizations and software tools that are well beyond the scope of this paper. ISO/IEC/IEEE 42010 recognized the diversity of modeling practices and representations by introducing the concept of architecture *viewpoints* and *views* [7].

The standard defines an architecture viewpoint as a "*work product establishing the conventions for the construction, interpretation and use of architecture views to frame specific system concerns*." Viewpoints frame the architectural concerns of system stakeholders and define the notation and models used to capture and communicate each concern.

The standard defines an architecture view as a "*work product expressing the architecture of a system from the perspective of specific system concerns*". Views are instances of a viewpoint applied to a specific system. The relationship between viewpoints and views may be summarized as:

"A viewpoint is a way of looking at systems; a view is the result of applying a viewpoint to a particular system-of-interest." [6]

For example, system stakeholders have a valid concern in desiring to understand and communicate:

- How a set of requirements and design goals drove the outcome of a design decision.
- How the chosen alternative creates new derived requirements that will impact the rest of the system design.

In response to this concern, an architectural viewpoint may be defined associated with Requirement-Decision-Requirement (R-D-R) Traceability that specifies a notation for visualizing this traceability thread in graphical or tabular form by including information consumed and created by the decision-making process:

- Requirements: Functional Requirements (FRs), Functional Requirement Measures (FRm's) and constraints.
- Evaluation Criteria: Factors used to evaluate the effectiveness of the alternatives.
- Decision: The fundamental question to be answered.
- Alternatives: Possible solutions to be evaluated.
- Risks & Opportunities: Ways that alternatives could fail or do better than expected.
- Mitigation and growth actions: Methods to reduce risks and grow opportunities.
- Derived Requirements: FRs, FRm's and constraints that are inherent consequences of a chosen alternative.

An instance of this viewpoint, the R-D-R Traceability *view* would be created when the R-D-R Traceability viewpoint specification is applied to a unique decision within a

project. An example of such a view from the design of a cellular manufacturing system is shown in Figure 1, below – adapted from [8].

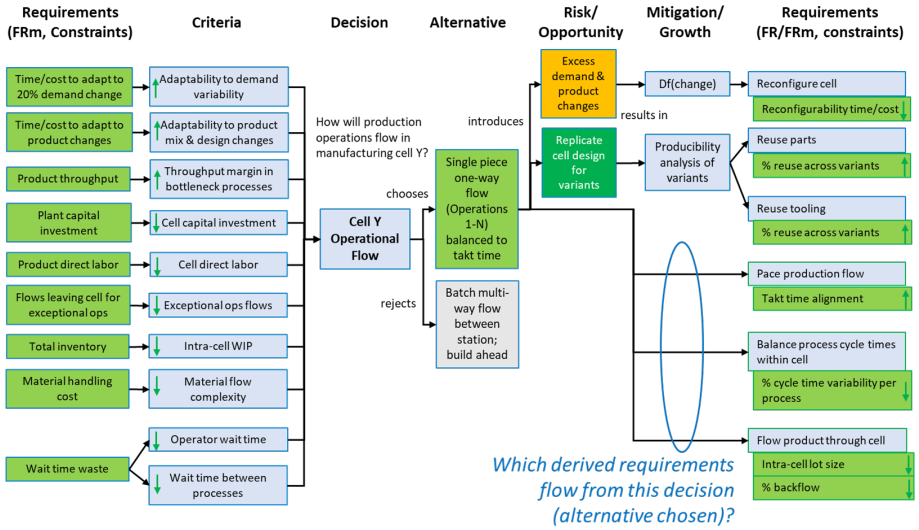

Fig. 1. Requirement-Decision-Requirement Traceability view for a Manufacturing System Design Decision.

Viewpoints and views provide a liberating structure that offers flexibility in capturing and visualizing system knowledge in a way that yields optimum insights for stakeholders across the system life cycle. These views and viewpoints break the pattern observed in document-centric engineering in which the method by which knowledge is stored and the method by which knowledge is communicated are coupled in the document artifact.

3 Class-based Information Metamodels

Views and viewpoints depend on precisely structured knowledge about the system and the thinking that drove its design. Knowledge structures (information metamodels) are typically described in terms of Entity classes, Relationships and Attributes (ERA) that support digital capture within a variety of database and modeling tools.

- Classes: The building blocks of engineering knowledge. *Requirements, decisions, components, risks*, *tasks,* etc.
- Relationships: Connections between entities that form the Digital Thread. *satisfies, performs, analyzes, results in, includes, couples*, etc.
- Attributes: "Fields" that precisely define characteristics of the entities with a class. *Priority* (of a decision), *Threshold* (value of a requirement), *Severity* (of a risk). *Selection Rationale* (of an alternative). *Target Date* (for completion of a task).

The demand for efficient information metamodels can only increase with increasing industry emphasis on Digital Engineering and the Digital Thread to address increasing system scale and complexity [9]. The engineering community needs ERA models that capture the essential information about the problem definition, design decision-making and solution descriptions and can populate an efficient set of system views that optimally engage stakeholders in various stages of the system life cycle. The goal of this set of views is collaborative thinking and fit-for-purpose insight. Different stakeholders need different views at different stages of product development. The ideal information metamodel for a specific development project includes the minimum number of entity classes, relationships, and attributes to support the highest value viewpoints and views. A compact representation of system knowledge helps to tame complexity, reduces ambiguity, enables filtering and navigation, and supports model integrity/validity checking.

Every engineering task populates instances of the SE information metamodel, i.e., new instances of each class, derivation/allocation/traceability relationships between these instances and attributes within them.

Viewpoints and views based on strict notations and modeling rules can be checked for completeness and consistency using automated or semi-automated techniques. Although such checks can detect structural defects within the model, these checks do not guarantee that the model represents the best possible system solution to the problem under analysis.

4 Development of the PFW AD/MBSE Information Metamodel

To illustrate the importance of a comprehensive, but efficient information metamodel, this paper highlights recent research on the Manufacturing System Design Decomposition, V10.0 (MSDD 10.0). MSDD 10.0 provides a design pattern for conceiving, evaluating, and creating sustainable manufacturing systems. The Purdue Fort Wayne (PFW) ERA model that is the basis for MSDD 10.0 is shown in Figure 2, below – adapted from [8].

At the highest level of abstraction, the PFW information metamodel consists of three layers:

- Requirements Layer: Stakeholder needs, use cases and associated steps (actions), and formal requirements, including FR, FRm, and constraints.
- Decision Layer: The fundamental questions/issues that demand an answer/solution and the data that inform each decision.
- Solution Layer: Physical and logical architecture of the system based on design decisions.

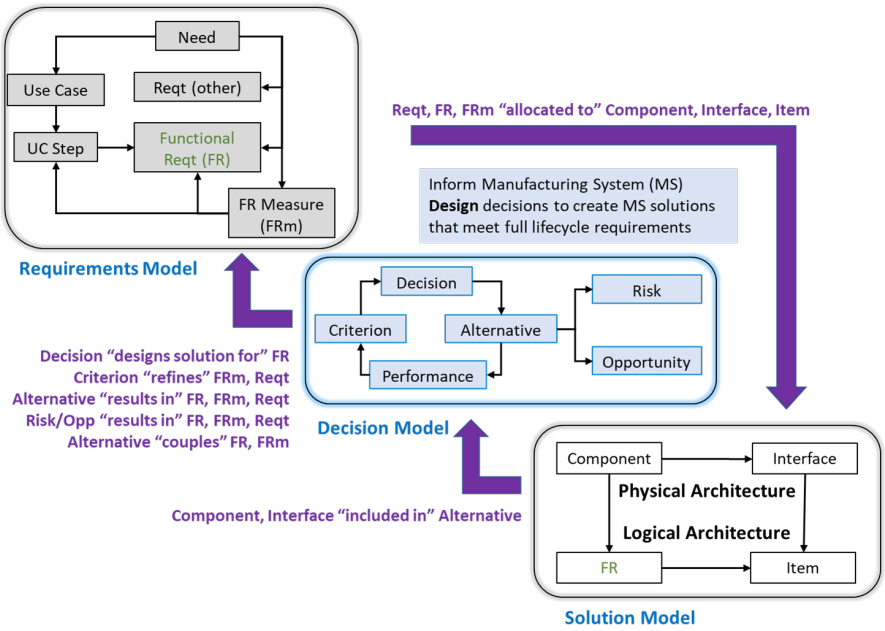

Fig. 2. Information Metamodel – Basis of the MSDD 10.0 Design Pattern

These classes of knowledge and the relationships between them form the basis for multiple system viewpoints. Examples of the high value viewpoints include:

- Requirements hierarchy: Decomposition of FRs and associated FRm's for each FR.
- Functional Flow Block Diagram (FFBD): System functions, their dependencies (control flow) and the item flow between them.
- FRm Flowdown: Mathematical relationships between FRm's at each level of the FR decomposition.
- Decision Breakdown Structure: Hierarchy of design decisions with recommended alternatives.
- System Breakdown Structure: Elements (hardware/software components, people, facilities, or data) that make up the physical system.
- Physical Block Diagram: System elements and their interfaces.
- Digital Thread – Requirements-Decision-Requirements Traceability: See Figure 1.

The ability to generate the desired views from these viewpoints demanded a significant extension to the modeling language (Vitech's Systems Metamodel) that was the basis for the MBSE tool (Vitech GENESYS) used as the software platform for this research initiative. Knowledge classes that were added include:

- Functional Requirement (FR)
- Functional Requirement Measure (FRm)
- Decision

- Criterion
- Alternative
- Performance
- Risk Mitigation

Numerous relationships were added between these new classes and between the new classes and existing classes in Vitech's Systems Metamodel. The additional knowledge coverage and precision offered by new classes and relationships can enable substantial reduction of the procedural errors identified by Thompson and "fill" many of the methodology gaps observed in AD theory.

4.1 Functional Modeling Extensions

Relative to AD theory, the most significant extension in the PFW information metamodel has been forging a clear distinction between FRs and FRm's. Removing the FR-FRm ambiguity observed in AD practice directly addresses Thompson's error #2: *Mixing FRs with other types of requirements*.

The extended PFW information metamodel limits FRs to be "pure" functions that express the transformation of inputs into outputs. With this definition, an FR is always named using a Verb-Noun (Direct Object) syntax, e.g., *Prevent bacterial infection* or *Stimulate dermal cell growth* [10]. The dependency logic between FRs is captured using control flow notation to address such logical constructs as parallelism (potential concurrency of functions), exclusivity (OR branching between either FR1 or FR2), iteration, looping or replication.

In this model, FRm's are always associated with one and only one FR. FRm's act as adverbs, measures of performance, that specify "How well" the FR must be performed to meet stakeholder expectations. The FRm, *Probability of bacterial infection*, specifies the required (threshold) value of the *Prevent bacterial infection* FR. The FRm, *Dermal cell growth rate*, specifies the threshold value of the *Stimulate dermal cell growth* FR. A FR may (and most often will) be specified by multiple FRm's to address different aspects of "goodness" (stakeholder value) such as throughput, efficacy, output quality, efficiency of the transformation process, and consistency of outputs (e.g., robustness or repeatability) in the face of variable inputs.

The one-to-many FR-FRm cardinality noted above has a significant impact on AD decomposition views in which FRm's have been misidentified as FRs. It is inefficient to identify a single Physical Solution (PS) for each FRm, i.e., to turn each FRm into a separate "How will this level of performance be delivered?" decision. Most often a PS should be chosen as the method to satisfy a FR by meeting or exceeding the levels of performance demanded by multiple FRm's, all within relevant constraints, e.g., cost, time, resource consumption or other policy-driven limitations.

4.2 Functional Modeling of System Context

A functional model may be constructed for a system of interest that elaborates the role this system plays in broader context of its use cases or mission scenarios across the

system life cycle. Such a top-level (contextual) functional model clarifies each functional interaction with stakeholders, e.g., users or maintainers, in terms of the inputs received from or outputs provided to these individuals/roles. Similarly, a functional model of the system context will explicitly define system functional interactions with external systems. As such, this model can directly resolve Thompson's error #3: *Mixing the FRs of the various stakeholders and of the system of interest* and error #4: *Mixing the FRs of the system of interest and of related systems*. "Mixing" is avoided by explicit allocation of all life cycle functions to the system of interest or to specific stakeholders or external interfacing systems in the broader system context.

4.3 Functional Modeling to Refine Decompositions

Precisely defining the control flow relationships between FRs and their inputs/outputs (item flow) clarifies the scope of each function – a process which can lead to discovery of missing or overlapping functions in the system model. Without an understanding of the functional model as a network of interacting functions, it is difficult to determine whether the intent of FR1: *Support wound healing* is completely and non-redundantly satisfied by child FRs 1.1 – 1.N, e.g., *Close wound, Form scar, Prevent bacterial infection, Aspirate wound, Prevent edema, Regenerate tissue*, …. Therefore, limiting the functional model notation to a decomposition hierarchy viewpoint increases the risk of errors and ambiguity in defining the FRs, with subsequent negative impacts on the remainder of the design process.

4.4 Functional Modeling to Distinguish Types of Coupling

Enforcing the FR-FRm distinction indirectly addresses Thompson's procedural error #1: *Mixing FRs with design parameters (DPs)*. The AD viewpoint known as the Design Matrix makes sense as a mathematical construct intended to capture a design equation with coefficients that relate the value of each **FR** to the value of all DPs. However, such math is possible and meaningful only if the Design Matrix depicts FRm-DP relationships where both are FRm's and DPs are quantitative in nature and share units that can be related in the form of an equation. In common AD practice, the DPs have been redefined as Physical Solutions (PSs), not the quantitative measures of performance that are derived from these PSs. Despite shortfalls in the notation, AD practitioners have intuitively understood that there must be a mapping between:

- Functions and the physical solutions that deliver them: PS -> performs -> FR.
- The performance required (FRm's) for each function (FR) and the PSs that exhibit that performance, i.e., satisfy the FRm's by delivering required performance against a set of DPs.

By not distinguishing FRs-FRm's and morphing DPs into solutions, the definition of coupling that is visible in the Design Matrix has become ambiguous. It could be either:

- *Architectural coupling* where the item flow between functions (FRs) is such that changing how Function A is accomplished, with a different PS alternative, will have excessive ripple effort on numerous functions that send inputs to or receive outputs from Function A.
- *Performance coupling* where the ability to achieve the performance specified for Function A (in terms of any FRm) is affected by DPs associated with more than one PS.

These two types of coupling are very different mechanisms and have differing impacts on the success of a design. Architectural coupling drives up system complexity (think "spaghetti code" or a "Rube Goldberg" design) and the life cycle cost of design changes because of the number of shared inputs/outputs across all functions. Performance coupling creates a competition/tradeoff between measures of performance in which the improvement of one FRm that is valued by stakeholders results in the worsening of another valued FRm. It appears possible that a design with excessive architectural coupling may exhibit no performance coupling and vice versa. Further research is suggested to explore this hypothesis.

4.5 Decision Modeling Extensions

To satisfy a FR (by meeting the FRm's that specify the FR), there must always be a "*How will the system deliver Function X*?" decision. Without explicitly modeling the decision and the data that informs this choice, designers:

- Increase the risk of poor solutions, i.e., alternatives that are destined to fail and disappoint stakeholders.
- May often overlook novel solutions that might represent significant leaps in stakeholder value and satisfaction.
- Have difficulty in objectively evaluating a range of possible solutions.
- Have difficulty in capturing and communicating decision rationale to stakeholders to gain their commitment (in the form of goodwill and resources) to implementation.
- Fail to capture rejection rationale for non-viable alternatives, leading to second-guessing and fruitless rework.
- Greatly increases the cost of change, i.e., revisiting a decision when requirements change or assumptions/estimates concerning the solution are invalidated.

The traditional Design Decomposition/Map used in AD displays only the FR (or sometimes FRm's misclassified as FRs) and the chosen PS to satisfy the FR. The full decision logic behind that choice is not captured nor visualized in views that can prevent the potential decision failure modes listed above.

If a PS has been chosen to fulfill a FR, a decision has been made, but likely without sufficient preservation of decision rationale to ensure decision quality. Dependence on human memory is not a winning strategy for designing complex systems or in the face of potential staff turnover.

PFW's addition of a more thorough model of decision-making information resolves most of the concerns listed above:

- Decision: Frames the question to be answered and provides the context for all other decision analysis data.
- Criterion: Provides a method for clarifying the influence of any requirement (FR, FRm or constraint) or design goal in the context of a specific decision. Defines success and provides an objective way to evaluate solution alternatives.
- Alternative: Explicit definition of possible solutions, whether physical (combination of interacting components) or otherwise (range of use cases that a product will support, or the value proposition associated with each use case).
- Performance: Estimates of the effectiveness of each alternative against each criterion. When combined, this data populates an evaluation matrix or various graphical representations of the merits of the competing solutions.
- Risks and Opportunities: Potential tiebreakers between leading alternatives based on projections of what could go wrong or what could go better than expected.

Capture of the Decision-> *chooses* -> Alternative -> *results in* ->(Derived) Requirement thread makes explicit how the chosen solution *results in* the next level requirements. All requirements are derived from answers/solutions chosen in other, typically "upstream" decisions. The Alternative -> *results in* -> Requirement (FR, FRm, constraint) relationship specifies the source of next-level requirements and localizes the potential changes to the requirements that might occur if a different alternative must be chosen in the future. Similarly, the identification of risks and opportunities supports a derivation trace to requirements added for risk mitigation (by reducing likelihood or severity) or opportunity growth (by increasing likelihood and positive impact).

4.6 Architecture Modeling Extensions

AD identifies a DP or (PS) as the means to satisfy a FR. In practice, the definition provided for a DP/PS is too imprecise to specify how to build the system from physical or software components, facilities, or human tasks. PFW's extended information metamodel includes a PS -> *includes* -> Component relationship to overcome this ambiguity. Decisions (through alternatives chosen) are the source of solution architectures; more than one decision may contribute to defining the components and component-to-component interfaces required to satisfy the system requirements. Incorporation of this relationship resolves a loose end between AD and generally accepted systems engineering and systems architecture practices and viewpoints. When combined with the functional modeling approach discussed in Section 3.1, the PFW information metamodel provide a mechanism for aligning the physical architecture decomposition with the functional architecture decomposition at each branch of the decomposition hierarchy.

5 Conclusions

The PFW experience with an extended information metamodel supports the hypothesis that MBSE constructs are complementary with AD and can increase the overall value delivered during design. Highlights include:

- FR-FRm precision is needed to distinguish between performance and architectural coupling, both of which have differing impacts on system success and require different methods to resolve.
- Functional modeling, in the form of control and item flows, is a powerful tool to improve the completeness and quality of the FRs that drive AD.
- Functional modeling of the context of the system of interest can reduce confusion between system FRs and FRs allocated to stakeholders and external interfacing systems.
- Decision modeling extensions can improve decision quality and buy-in, provide a method for deriving next-level requirements in the AD zig-zagging process, and reduce the cost of managing change.
- Information metamodel extensions enable additional high value viewpoints that can improve stakeholder engagement across the system life cycle.

6 Future Research

Additional research is suggested to further investigate the benefits of combining the fundamental elements of AD with MBSE principles and practices. Examples of such research topics include:

- Confirmation of the distinction and independence between architectural and performance coupling implied by the FR-FRm distinction.
- Evaluation of various requirements classification schemes that are common to the systems engineering and product development communities.
- Prototyping and evaluation of viewpoints enabled by AD/MBSE; clarification of the applicability and benefits associated with new viewpoints.
- Methods for "keeping alive" multiple solution alternatives through multiple layers of design decomposition and the impact of such practices on viewpoints, numbering schemes, naming conventions, etc.
- Redefinition of AD's zig-zagging process to account for synchronization of the physical architecture model, functional architecture model and mathematical system performance model at each branch of the design decomposition.
- Formalizing and refining heuristics for defining derived requirements from a chosen solution alternative.
- Methods for incorporating state/mode and state/mode transition models into the design process.
- Methods for reducing the redundancy between system risks, failure modes and hazard analysis models.
- Investigation of the potential for simplifying the information metamodel by eliminating the CN-FR-DP-PV domains as requirement classes and replacing them as subclasses that express the context of FRs and FRm's. In general, separate class from context on all entities.

References

1. Suh, N. P.: The principles of design. Oxford University Press (1990).
2. Thompson, M. K.: A Classification of Procedural Errors in the Definition of Functional Requirements in Axiomatic Design Theory. In M. K. Thompson (Ed.), Proceedings of the 7th International Conference on Axiomatic Design (Chapter ICAD-2013-16). ICAD. (2013).
3. Kulak, O., Cebi, S., Kahraman, C.: Applications of axiomatic design principles: A literature review, Expert Systems with Applications, pp 6705-6717. (2010).
4. INCOSE: Systems engineering vision 2035 – engineering solutions for a better world. https:// www.incose.org/about-systems-engineering/se-vision-2035. (2021)
5. INCOSE: Systems engineering handbook: A guide for system life cycle processes and activities, version 4.0. John Wiley and Sons, Inc. (2015)
6. ISO/IEC/IEEE: Standard 15288. Systems and Software Engineering – System Life Cycle Processes. International Organization for Standardization, Geneva Switzerland (2015).
7. ISO/IEC/IEEE: Standard 42010. Systems and Software Engineering – Recommended Practice for Architectural Descriptions of Software-Intensive Systems. International Organization for Standardization, Geneva Switzerland (2011).
8. Cochran, D. S., Smith, J., Fitch, J.A: MSDD 10.0: a design pattern for sustainable manufacturing systems. Journal of Production and Manufacturing Research, pp. 964-989 (2022).
9. Rauch, E., Vickery, A. R.: Systematic analysis of needs and requirements for the design of smart manufacturing systems in SMEs. Journal of Computational Design and Engineering 7(2):129–144 (2020).
10. Gebala, D., and Suh, N. P.: An Application of Axiomatic Design. Research in Engineering Design: Theory, Applications and Concurrent Engineering, 3:149-162 (1992).