Digital Engineering Transformation of Requirements Analysis within Model-Based Systems Engineering

Andrew R. Miller, PMP[1] and Daniel R. Herber, Ph.D.[2]

[1]https://orcid.org/0000-0001-6866-9024
[2]https://orcid.org/0000-0003-4995-7375
Department of Systems Engineering
Colorado State University
Fort Collins, CO 80523

Corresponding author's Email: miller82@colostate.edu

Author Note: [1]Andrew R. Miller is a Ph.D. candidate at Colorado State University researching applied model-based systems engineering (MBSE) aspects that drive quality of MBSE models. He has worked in the defense industry for over a decade with a particular focus on MBSE quality process and application. [2]Dr. Daniel R. Herber is an Assistant Professor at Colorado State University with research interests in the areas of model-based systems engineering, computational design, design optimization, system architecture synthesis, and combined physical and control system design (control co-design).

Abstract: As Systems Engineering (SE) has evolved into the modern age, the connectivity of data to instantiate an architecture while mapping to the requirements becomes more vital for the digital engineering approach. However, the complexity of systems typically results in difficulty understanding how architecture element data developed in models translate to programmatic risk implications. For instance, it is most important to understand what aspects of traditional systems engineering requirements analysis are transferable to digital engineering applications. Addressed within this paper is a methodology for tailoring requirements diagram content based on functional requirement analysis and the relation of data to model elements for programmatic analysis. The relationship of requirements to model architecture data elements provides an approach to deliver quantifiable metrics gathering techniques for a measurable understanding of model development maturity throughout the system lifecycle.

Keywords: MBSE, Digital Engineering, SysML, DoDAF, Requirements Analysis, Systems Engineering

1. Introduction

As SE has become more complex and intricate, the understanding of quality requirements for SE becomes more critical in the design process. To expound on this, adding phenomenology like Digital Engineering (DE) to the mix and SE struggles to understand the implication or practical application to maintain engineering quality. In this paper, the focus will primarily be on requirements analysis from a Department of Defense (DoD) perspective and translation of concepts from traditional SE to a DE approach. To embrace the application of DE, an understanding of new tools is needed to assist in the SE process. The Model-Based Systems Engineering (MBSE) approach is a key driving factor that strives to reach the interconnection of data relations to accomplish DE requirement analysis. Requirement analysis diagrams within the MBSE environment include critical features that are needed for completion, including elements, relationships, driving factors, and metrics. The methodology discussed will illustrate requirement diagram content development based on classical shall statement context. The paper will only address the functional performance requirement type with an approach to programmatic metric tracking with risk trends.

2. Department of Defense Requirements and Traditional Systems Engineering Analysis

DoD customer requirements typically come from their Joint Capabilities Integration and Development System (JCIDS) design process, including design specification and capability documents which are derived from their Initial Capabilities Document (ICD), Capabilities Development Document (CDD), System Requirement Specification (SRS), and Capabilities Production Document (CPD) (Starnell, 1991). These requirements are oftentimes expressed in a nontechnical

format that requires reformulation or translation before they can be analyzed for development. The requirements are then subdivided into various types that are then allocated to the appropriate level within the system. The various types of requirements are then allocated to the appropriate level of system design, with discipline such that the assignment is made to the appropriate level to the exclusion of all other levels above or below the assignment. Consistency, correctness, and completeness of the translation are the measure of goodness with respect to architecture and model maturity (Faudou 2016).

It is this human-centered translation where ambiguity and programmatic risk are introduced in the traditional SE process. Traditional system designers use analog or ad hoc practices in translating the nontechnical text of customer requirements' documents to technical functional requirements that are clear, concise, correct, and verifiable and often take shape as specifications. These specifications must be revised repeatedly to arrive at a suitable quality level to reduce risk. Complicating the traditional SE process within DOD is the difference in defense acquisition, as compared to some other US government and commercial entities, is that DoD capabilities typically give rise to more relevant detailed requirement information from many stakeholders, driving increased data sharing needs and enabling greater data analytics (feedback loops) in the holistic lifecycle management, but at the same time introducing programmatic and quality risks. System performance is oftentimes dependent upon the delivery of many allocated capabilities across many joint forces. That is, one might be required to allocate naval warfare performance in meeting ground forces capabilities in the overall warfighting effort. Other US government domains tend to be exclusive to their capabilities and, therefore, include constrained allocation of requirements.

3. Systems Engineering via Digital Engineering with Model-Based Systems Engineering

In DE, through MBSE, functional analysis or translation of customer requirements are defined by component functions with the functional formulation and concept selection being intertwined tightly (Biemer, 2011). Functional building blocks identify the media (i.e., capabilities, operations, enterprise, systems, segments, system functions, systems interfaces, and eventually hardware/software configuration items), the functional elements, the relation of performance requirements to other elements, configuration of the functional elements, and providing an overarching analysis and integration of all data exchanges, as well as internal interactions (Biemer, 2011). By defining the data exchanges functionally, it facilitates integration and test activities as well as reduces programmatic risks going forward in lifecycle operations, maintenance, and logistics, enabling future upgrades. Simulations are also heavily used in functional analysis (a key driver of architectures and models are data analytics and feedback loops, predicted versus actuals) to provide trade studies with varying parameters to provide more efficient and less costly experimental digital testing, not to mention management by fact rather than simply intuition, again reducing programmatic risk and improving SE quality.

MBSE tools provide the means for DE to express or extend the base Unified Model Language (UML) and Systems Modeling Language (SysML) structures into the customization realm for an industrial area or domain (i.e., business, aerospace, space & missiles, infrastructure, information technology, etc.) (Friedenthal, 2008). MBSE enables DE through the development of profiles containing key attributes or tag data that can provide relevant contextual meaning with stereotypes when applied to elements. The use of profiles and libraries that expand upon the foundational modeling languages of both UML and SysML provides enough relevant context data leading to the significant expansion of previous requirements definition, traceability, and linkage for developing new or modernizing legacy complex systems. The most common mechanism for generating customized tools is stereotypes (Friedenthal, 2008). A stereotype is a type of model element that is based on metaclass in a reference metamodel, such as SysML. Stereotypes do this through an extension relation to the newly created stereotype. Using stereotypes to create specialization gives the profile and libraries the keys to collect all the specialized attributes for different systems as needed for a particular industrial domain, thereby allowing specific programmatic metric tracking and trend analysis for the system architecture.

Another unique specialization in the realm of MBSE is the architectural frameworks. Architectures are typically specializations of the UML and SysML into profiles and provide industry standards for new domains and complex system development as opposed to the traditional analog allocation process. Architectures such as the Department of Defense Architecture Framework (DoDAF) or Unified Architecture Framework (UAF) are used to represent the critical views/viewpoints and needed for the complete system story with the appropriate depth and breadth necessary. These architecture frameworks are quickly becoming required in contracts as the means to meet architecture development and model-based compliance and are eclipsing the traditional analog or ad hoc allocation architectures. Using the DE and MBSE processers, the holistic lifecycle design and sustainment are examined much closer and far earlier to meet the needs of the customer via better-defined performance of their system but reduce the likelihood of poor quality and high risk. The example in the following section demonstrates the application of the method of decomposing a derived requirement to instantiate and justify the DoDAF architecture framework.

4. Applying the Digital Engineering Requirements Analysis Method

An MBSE requirements diagram illustrates the relation to the DoD customer requirement, which comes from the customer specification and/or the customer needs: ICD / CDD / CPD / IRD / SRS. Specifically for a functional performance requirement, the modeling elements and associated relationships show specifics about the system under design and provide the states or modes of operation to meet the intended derived functional performance requirements. The elements and relationships can show energy, matter, and data (information) flows, highlighting the requirements and specifications for the system holistically as well as interdependencies. This MBSE architecture and model development approach can be modular and repeatable for each requirement type; relying on compliance standards, trade studies, simulations, and related allocated specifications to perform the needed analyses. The key is ensuring a complete and early agreement on the defined compliance criteria (disciplined, developed requirements and management of those requirements), architectural elements, and relationships for the graphical representation of the requirements to enable needed analysis.

- Assumptions: Here are some assumptions for the architecture stereotype mapping methodology: 1. Awarded contract is defined with SRS and CDD documents; 2. DoDAF 2.02 is the selected architecture framework (Officer 2010); 3. Completion criterion is established for requirement analysis diagrams; 4. Functional performance requirement is the only requirement type shown, and 5. The textual pattern "«*stereotype*»" indicates a UML, SysML, or DoDAF stereotype.
- *Note: The functional performance requirement context must be complete enough to illustrate creation of the model architecture. The functional performance requirement should tell: 1. What the system must do; 2. How well it must do it; 3. Efficiency for the intended missions; and 4. Specific conditions under which the function and performance are applicable.

Next the key steps in the proposed DE requirements analysis method in the context of MBSE are enumerated:

1. The system requirements specification will become the customer requirement or requirements using the «*requirement*» stereotype model architecture. The customer requirements will drive the architecture content development, including requirement derivation.

2. The customer requirement will derive the «*functionalRequirement*» or system level with a populated text field. The text field is critical as well as the quality of the text, which describes the functional performance requirement. The text field can be broken down into various parts the will cause element creation in the architecture. The functional performance requirement will have a «*deriveReq*» relation to the customer requirement.

a. The initial text of the requirement is the "it" or "thing" that is used to perform the describe function and a «*System*», «*Technology*», or «*Software*» stereotype may be used. The selected stereotype may change depending on the requirement level in the system design. For example, a «*System*» element may be higher than a «*Technology*» element in the system based on structure.

1. The «*functionalRequirement*» will have an «*allocate*» relation to the system or «*OperationalPerformer*» based on the context of the requirement statement. It is possible to have more than one functional performance requirement «*allocated*» to the system. The allocation will create the allocated functional baseline for the system.

b. Functional performance requirement text will describe "what the system must do" and will cause the creation of the «*function*» element in the architecture.

1. The «*function*» element will have a «*refine*» relation to the originating functional performance requirement. The «*refine*» relation will be a required relation type to the functional performance requirement. See Step 2.d for relation created. *Note: More than one function is possible from a functional performance requirement but needs to be strictly scrutinized because this could indicate a poorly written functional performance requirement.

c. Functional performance requirement text will describe "how well the system performs" and will cause the creation of «*MeasurementSet*» that invoke a «*ActualMeasurementSet*».

1. The «*MeasurementSet*» and «*ActualMeasurementSet*» will have a «*refine*» relation to the functional performance requirement.

2. The «*MeasurementSet*» are the performance values or indications that must be met in the requirement text and are driven by using «*InformationElements*» or «*Standard*» elements. Depending on the context of the functional performance requirement, one or more «*ActualMeasurementSet*» is possible. The «*ActualMeasurementSet*» describe units of measure assigned to «*MeasurementSet*» using «*InformationElements*» or «*Standard*» elements. See Step 2.e.1 for details.

d. The functional performance requirement text "efficiency for the intended missions" will cause the creation of the «*OperationalActivity*» and «*OperationalAction*» when the system performs the function.

1. The «*OperationalActivty*» element will have the «*refine*» relation to the functional performance requirement. The «*OperationalActivity*» will be a required relation to the functional performance requirement. The «*OperationalAction*» will be contained within the « *OperationalActivity*» (see Step 3.a). An

«*OperationalActivity*» may contain more than one «*OperationalAction*» so the same «*OperationalActivity*» may be possible.

e. The functional performance requirement text "specific conditions" may include a standard, environmental, state, or mode aspect of performance.

1. The «*InformationElement*» or «*Standard*» will be related to the functional performance requirement and a «*dependency*» relation will be used. Depending on the context of the functional performance requirement, one or more «*InformationElement*» or «*Standard*» element is possible.
2. The «*OperationalStateDescription*» element will be used for state or mode data with a «*refine*» relation that will be required. *Note: More than one state or mode should not be possible from a functional performance requirement and needs to be strictly scrutinized because this could indicate a poorly written functional performance requirement.
3. The «*OperationalConstraint*» should be placed in the correct elements and are created within and will be populated in the OV-6a view of the DoDAF framework. The «*OperationalConstraint*» should be placed internally to other model elements needing an appropriate constraint based on the functional performance requirement context.

3. The ICD or CCD listed capabilities will become each «*Capability*» element.

a. A «*trace*» relationship shows «*Capability*» element relation to the «*OperationalActivity*». The «*OperationalActivity*» containing the «*OperationalAction*» will solidify what the function must "do" in the behavior exhibiting the «*Capability*». The «*refine*» relation created in Step 2.d.1 to «*OperationalActivity*» will show the «*Capability*» to be fulfilled. *Note: More than one capability could be possibly related to an «*OperationalActivity*» and, similar to before, needs to be strictly scrutinized because this could indicate a poorly written functional performance requirement.

4. Once the relations are in place and elements are instantiated to meet the textual context of the functional performance requirement shall statement, the requirement diagram is complete. This method can be performed on several types of requirements and relies on the industry standards for UML, SysML, and the DoDAF Framework.

Figure 1 shows the application of the above methodology to a Predictive Emission Monitoring System (PEMS) (Environmental Protection Agency, 2021). The customer requirement is taken from the SRS shown with the «*requirement*» element. The «*deriveReqt*» links the PEMS «*functionalRequirement*» derived from the customer requirement.

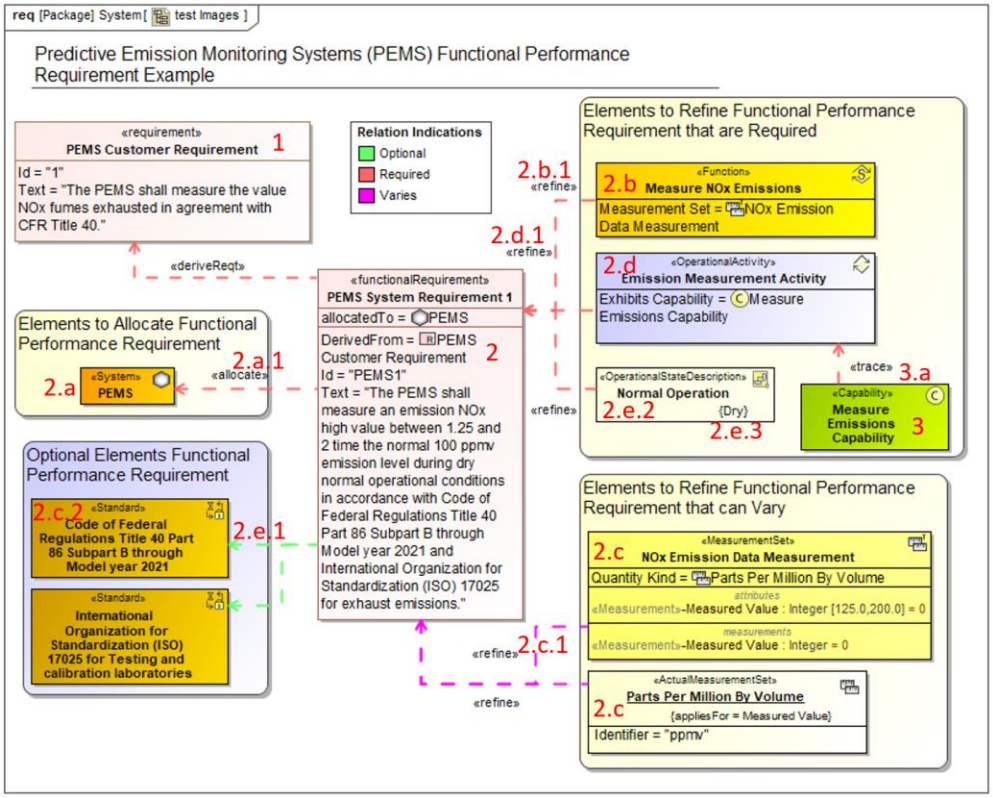


Figure 1. Example Functional Performance Requirement for PEMS (Red Text indicate steps in Method)

Table 1 shows the methodology executed that created Figure 1 based on the context of the «*functionalRequirement*». The functional performance requirement shall statement is as follows: "*The PEMS shall measure and emission NOx high value between 1.25 and 2 time the normal 100 ppmv emission level during dry normal operational conditions in accordance with Code of Federal Regulations Title 40 (CFR 40) Part 86 Subpart B through model year 2021 and International Organizational Standard (ISO) 17025."*

Table 1. Elements and Relations Created for the PEMS Example

Method Step	Element Stereotype Created	Relations Created
1	«*requirement*»	
2	«*functionalRequirement*»	«*deriveReq*»
2.a	«*System*»	
2.a.1		«*allocate*»
2.b	«*function*»	
2.b.1		«*refine*»
2.c	«*MeasurementSet*» «*ActualMeasurementSet*»	
2.c.1		«*refine*»
2.c.2	«*Standard*»	
2.d	«*OperationalAction*» «*OperationalActivity*»	
2.d.1		«*refine*»
2.e.1		«*dependency*»
2.e.2	«*OperationalStateDescription*»	«*refine*»
2.e.3	«*OperationalConstraint*»	
3	«*Capability*»	
3.a		«*trace*»
4	Complete	Complete

5. Digital Engineering Programmatic Analysis

The requirement analysis methodology defined above provides a means to take SE requirements development process and break that data into the DE approach to instantiate the complete DoDAF architecture. The method shows a promising way to create related architecture element items linking the traditional SE process to new DE best processes using built-in standardized framework data elements. The most critical piece of MBSE metrics related to development is defining what the metrics highlight from an analysis perspective (i.e., improved design, quality, robustness of the architecture, etc.). Collecting the number of various elements from within the model can demonstrate quantifiable numbers of elements. Using the notes provided in the method, quantifiable trends of counts can be shown for a model. If applied correctly, there are huge efficiencies and effectiveness benefits realized over time that can be provided to a DE implementation. Table 2 shows line trend indicators for model element metric count and the risk implication for programmatics.

6. Conclusions and Future Work

The paper presents several different aspects dealing directly with the functional performance requirement analysis method of the SE process. First, the criticality of functional performance requirement analysis for DoD in the SE process was addressed. Next, a method of tailoring MBSE model content in the DoDAF framework was presented to meet a functional performance requirement. Then, a notional example was created using the prescribed methodology with a discussion on instantiating a DoDAF architecture from the functional performance requirement shall statement. Next, a means to gather quantifiable data was discussed through the collecting of counted elements created with the methodology, including evaluation of graph line trends relating to possible programmatic risks. These presented items are based on the proposed methodology for defining quantifiable requirements analysis data and has the potential to enable transformations from a

traditional SE process to a DE data-centric process. The presented methodology can be used and expounded upon in a similar manner to address and evaluate the quality of an MBSE model.

Further work will provide a quality evaluation methodology and approach to programmatic expounding upon the one presented in this paper. This future quality evaluation framework will address DoDAF architectures using the MBSE and an alternative perspective approach. Included in the framework will be described conceptual DoDAF architecture quality analysis, including re-contextualization of Quality by Design (QbD) to the DoDAF framework, formalized taxonomy, metrics development, defined profile, integration plugin, data analysis, and programmatic implications. The approach to QbD improves the understanding of completeness for a DoDAF architecture by building better checks on the design process. The DoDAF Quality Framework can provide greater visibility into the quality system engineering process of the DoDAF application in MBSE hence improving trust in the system and data architecture models.

7. References

Starnell, P. (1991). Defense Acquisition System. An Executive Summary of DODD 5000.1, DODI 5000.2, and DOD Manual 5000.2.

Faudou, R. (2016) Requirements and Architecture Within Modeling Context, AFIS (Association Française d'Ingénierie Système). INCOSE Affiliate

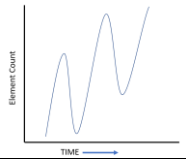
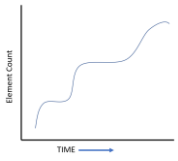
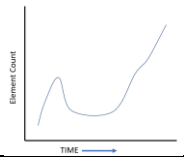
Steven M. Biemer, Alexander Kossiakoff, Samuel J. Seymour, & William N. Sweet. (2011). Systems Engineering Principles and Practice, 2nd Edition. Wiley-Interscience.

Friedenthal, S., Moore, A., & Steiner, R. (2008). Practical Guide to SysML: Systems Modeling Language. Elsevier Science & Technology.

Officer CI. (2010). DODAF - DOD Architecture Framework Version 2.02 - DOD Deputy Chief Information Officer. Office of the Department of Defense.

Environmental Protection Agency. (n.d.). EMC: Continuous Emission Monitoring Systems Information and Guidelines. EPA. https://www.epa.gov/emc/emc-continuous-emission-monitoring-systems.

Table 2. Risk Implication Table for Functional Performance Requirements

Graph Trend Line Indicators	Meaning	Risk Level	Trend Line Graphic
Spikes	*Can be Up or Down. Up indicates a large element count increase and could be indicative of "Model Bloat" or increase in useless information. Down indicates loss of data or missing model elements. Happens over very short periods.*	*High – Up possible model tool performance issues or down costly rework for lost data*	
Flat	*Flat spot meaning that data is not being created or deleted. Should depend on what is being worked in the model.*	*Low – could be lull in task execution or drop in modeling development, typically occurs before delivery milestone*	
Dips	*Dips are longer loss or changes to model element trends. Needs to be monitored to ensure baselined model content is not changing or being modified*	*Medium – possibly caused with missing content or data*	
Diverging trends	*The *Notes in the method talk about element counts. Blue indicates element count and Orange indicates related element count. Assuming a one-to-one relation the divergence needs investigation*	*Medium – each diverging area needs to be inspected to verify functional performance requirement is well written and accurate*