Rethinking the Role of Department of Defense Architecture Framework in System-of-Systems Architecture Design

Zhemei Fang and Yuxuan Liu
School of Artificial Intelligence and Automation
Huazhong University of Science and Technology
Wuhan, Hubei 430074 China
email: zmfang2018@hust.edu.cn

Jianbo Wang
China Ship Development and Design Center
Wuhan, Hubei 430064 China
email: jbwcn@hotmail.com

***Abstract*—While Department of Defense Architecture Framework (DoDAF) remains widely adopted for architecture modeling, its application to System-of-Systems (SoS) design still faces significant challenges according to feedbacks from practitioners in industry and academia. Existing research often focuses on model creation or tool support but lacks a comprehensive examination of the issues behind the unsuccessful applications. Thus this paper analyzes the root causes of unsuccessful DoDAF applications, including the perspectives of common misconceptions, inherent shortcomings, methodological inadequacies, limitations of modeling tools, and cultural and organizational barriers. Based on the challenges observed, we further explore how the Unified Architecture Framework (UAF) and SysML 2.0 could alleviate some of these limitations. Based on this analysis, we propose three improvement directions: iterative, process-driven architecture modeling, AI-assisted model generation and evolution, and domain-specific meta-model customization with consistency assurance. The study concludes that treating architecture models as evolving decision-support tools, rather than static documentation, significantly enhances their value in SoS design and provides actionable guidance for improving DoDAF and other architecture frameworks in practice.**

Keywords-architecture design; department of defense architecture framework; system-of-systems; misconceptions.

I. INTRODUCTION

Architecting is increasingly being adopted by organizations to manage the growing complexity of human-made systems, particularly large-scale SoS such as those in defense and air transportation. The latest ISO/IEC/IEEE 42010:2022 standard (Software, systems and enterprise - architecture description) [1] defines architecture as "fundamental concepts or properties of an entity in its environment and governing principles for the realization and evolution of this entity and its related life cycle processes". Meanwhile, the standard introduces the term Architecture Description Framework (ADF) (replacing architecture framework in the 2011 version) to formalize the conventions and common practices of architecture description—a tangible work product that communicates the otherwise intangible and abstract concept of architecture [1].

The ADF has evolved from the C4ISR architecture framework to DoDAF, then to the Unified Profile for DoDAF/MODAF (UPDM), and most recently to the UAF. Despite this evolution, DoDAF remains the predominant ADF in the defense sector [2]. At the same time, most commercial modeling tools have gradually aligned their underlying meta-models with the UAF meta-model, enhancing tool interoperability while still maintaining support for DoDAF-based practices. Current DoDAF models [3][4] are compatible with UAF meta-models.

However, concerns about DoDAF have been raised over the years, including inconsistencies across architectural views [5], challenges in effectively utilizing architecture models for downstream applications [6], difficulties in accommodating new technologies, such as cloud computing and big data [7]. Although UAF was introduced to address some of these challenges, it inherits many of the same weaknesses. This critique is frequently acknowledged within the Model-based Systems Engineering (MBSE) community as well [8]. Interestingly, these issues are more commonly acknowledged in informal exchanges [8] than systematically addressed in published research. This gap highlights a critical need for more rigorous investigation into the practical barriers that hinder the effective application of ADFs in real-world SoS contexts.

This paper aims to uncover the reasons behind unsuccessful application of DoDAF, as a representative ADF, in supporting SoS architecture design. The perspectives include prevalent misconceptions about DoDAF's intended role, limitations in existing modeling tool support, methodological gaps in modeling approaches, and organizational and cultural barriers to model adoption. Building on this analysis, we propose several potential directions to achieve an enhanced use of DoDAF as well as other ADFs.

The paper is organized as follows. Section II reviews related work on architecture frameworks. Section III analyzes the key challenges of applying DoDAF to SoS design. Section IV discusses improvement opportunities. Section V concludes the study and suggests future research.

II. RELATED WORK

The importance of architecture, along with the supporting ADFs that guide its formal representation, has been increasingly acknowledged across both academic and industrial domains in recent years.

Early research by Wagenhals and Levis [9] pioneered a structured methodology for developing DoDAF models

using IDEF0. Subsequently, numerous studies have adopted and extended this approach for DoDAF models development (e.g., [10]-[12]). In DoDAF model development, the Systems Modeling Language (SysML) has progressively superseded IDEF0 as the preferred modeling approach [11]. Current research and practice continue to demonstrate the framework's relevance, with active applications documented in recent works [3][4].

The U.S. Department of Defense (DoD) concluded its development of the DoDAF framework with the 2009 release (DoDAF 2.02). This transitioned to the UPDM, developed by the Object Management Group (OMG), as an interim solution. OMG subsequently established the UAF as the current standard [13]. Hause [14] indicates that the UAF was developed to address interoperability challenges by reducing disparities among architecture frameworks, modeling tools, standards, processes, data exchange formats, and domain terminology in ADF implementations.

From the 31st to 34th Annual INCOSE International Symposium proceedings, numerous implementation case studies of the UAF have been documented. For example, Martin [15] proposed an aspect-oriented approach aimed at harmonizing architectural frameworks to enhance interoperability and better support MBSE practices. Later, Martin [16] demonstrated how MBSE enhances an organization's ability to plan for capability deployments, and manage portfolios of systems, services, people, technologies, processes, and facilities critical to fielded capabilities. Carroll et al. [17] successfully implemented UAF in modeling the global copper market enterprise, noting its efficacy in fostering systems thinking beyond traditional engineering roles. Hause et al. [18] specifically addressed enterprise software architecture challenges through UAF modeling. Most recently, Martin et al. [19] and Gagliardi et al. [20] extended UAF's utility to Mission Engineering (ME), showcasing its adaptability to complex defense and aerospace applications, and the resultant modeling process and models are standardized in the U.S. DoD's Mission Architecture Style Guide (MASG) [21].

Alongside these applications of UAF, significant legacy challenges persist. Gagliardi et al. [20] highlight that "even a relatively simple Resource Architecture model requires significant time and effort to develop", emphasizing the need for careful upfront planning. Their findings suggest three critical prerequisites for effective UAF adoption: 1) scoping the modeling effort, 2) assessing modeling risks, and 3) establishing a model federation plan—all of which should be addressed prior to commencing development. Similarly, Fang et al. [22] pointed out that the relationship between DoDAF description models and architecting decisions is ambiguous—a limitation that also persists in UAF.

Modeling languages and tools also present challenges. Trujillo and Madni [23] highlight that modeling languages—particularly SysML—pose a high entry barrier, primarily due to the extensive training required to interpret increasingly complex models. In response, Morkevicius et al. [24] advocate for implementing UAF within the SysML v2 environment, anticipating that the updated specification may mitigate some inherent limitations of current SysML implementations. Regarding tooling considerations, Maier [25] indicates that a good modeling tool should manage significant redundancy in representations by using referencing instead of duplication and employing automated checks; nevertheless, there remains a clear risk of model proliferation beyond practical usefulness.

In summary, while the evolution from DoDAF to UAF has led to improved standardization and broader applicability in both defense and enterprise contexts, practical challenges remain prevalent across modeling frameworks, languages, and tools. The literature reveals a persistent tension between the theoretical promise of ADFs and their real-world implementation barriers—many of which stem from complexity, tool limitations, and organizational constraints. These gaps underscore the necessity for a deeper investigation into the root causes hindering effective ADF application, particularly in complex SoS environments. Building upon these insights, this study aims to critically examine the key obstacles to DoDAF adoption and propose actionable strategies for enhancing its practical utility.

III. Practical Challenges and Inherent Shortcomings of DoDAF in SoS Architecture Design

The unsuccessful applications of DoDAF in supporting SoS architecture design stem from a fundamental misunderstanding of its intended role, limited support from modeling tools, inadequate methodological guidance, and practical and cultural barriers to model adoption, as shown in Fig. 1. This section examines these four aspects in detail.

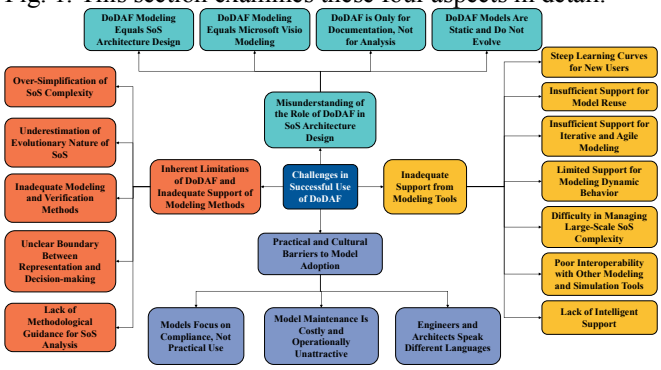

Figure 1 . Practical challenges of DoDAF in SoS architecture design.

A. Misunderstanding of the Role of DoDAF in SoS Architecture Design

Based on our practical modeling experiences and interviewing with modeling experts in industry, we summarize four common misunderstandings of the DoDAF's role in SoS architecture design.

1) Misunderstanding I: DoDAF Modeling Equals SoS Architecture Design

This misunderstanding often arises among outsiders who have unrealistically high expectations of DoDAF. They mistakenly assume that creating DoDAF models is equivalent to completing SoS architecture design.

In fact, DoDAF provides a structured set of standardized views and establishes a formal framework for representing

SoS architecture. However, a practical and effective SoS architecture design involves not only representation but also decision-making and evaluation—aspects that DoDAF alone does not fully address. Therefore, additional methodologies, such as operational simulation, trade-space analysis, and optimization, are essential to complement DoDAF for achieving a comprehensive SoS architecture design.

These inflated expectations often lead to significant disappointment, ultimately causing them to overlook the actual value of DoDAF models.

2) Misunderstanding II: DoDAF Modeling Equals Microsoft Visio Modeling

This misunderstanding often arises among practitioners who have some experiences with DoDAF modeling but have not delved into the underlying theories. They assume that creating DoDAF views is simply about drawing static diagrams, like flowcharts, without considering the underlying semantic relationships, constraints, and traceability.

In fact, DoDAF is model-based, not merely diagram-based. While it employs visual representations, it is fundamentally a structured architecture framework, not just a collection of disconnected drawings. Tools like Visio and similar diagramming software allow freeform visualization but do not enforce architectural consistency or data integrity. In contrast, DoDAF models should be developed using structured modeling tools (e.g., Cameo Enterprise Architecture, Sparx EA, IBM Rhapsody) that enforce rules and ensure consistency between capabilities, systems, and services across multiple views.

This misunderstanding can lead to superficial architecture modeling that lacks architectural rigor. Organizations may create visually appealing but structurally meaningless diagrams that fail to support real system development. Without architectural rigor, inconsistencies and logical errors may go unnoticed, ultimately undermining the effectiveness of the architecture.

3) Misunderstanding III: DoDAF is Only for Documentation, Not for Analysis

DoDAF is often misperceived as merely a documentation framework, rather than a foundation for architectural analysis and informed decision-making. This misunderstanding stems in part from the limitations of current practices and tools, which often fail to deliver on the promise of model-based analysis. Despite many tools claiming to support analytical functions, the actual use of DoDAF models for quantitative or qualitative analysis remains challenging in practice.

Several factors contribute to this gap. First, many DoDAF-compliant tools focus heavily on model visualization and reporting, offering limited support for integrated simulations, trade-off analysis, or impact assessments. Second, users may lack clear methodological guidance on how to leverage architectural description models for analytical purposes, especially in complex SoS contexts. Lastly, architecture models are often developed in isolation from operational or technical data, limiting their usefulness for real-time or predictive analysis.

As a result, DoDAF models are frequently underutilized in decision-making processes, reducing their value to stakeholders and reinforcing the notion that they are static deliverables rather than dynamic decision-support artifacts.

4) Misunderstanding IV: DoDAF Models Are Static and Do Not Evolve

Some organizations mistakenly treat DoDAF models as static, one-time deliverables rather than as evolving artifacts that must be continuously updated as the system evolves. This misconception largely arises from the inadequate support current modeling tools provide for iterative development and model maintenance.

SoS architectures are dynamic, requiring continuous updates to DoDAF models to reflect new requirements, emerging threats, and evolving technologies. Architecture models should support versioning, impact analysis, and iterative refinements throughout the SoS lifecycle.

When this need for evolution is overlooked, DoDAF models quickly become outdated and disconnected from the actual SoSs they are intended to represent, resulting in misalignment between architectural intent and operational reality.

5) Summary

The misunderstandings stem not only from a general lack of familiarity with DoDAF, but also from widespread disappointment with its practical applications. These challenges arise from inherent limitations within DoDAF and supporting methods, inadequate support from current modeling tools, and cultural resistance to adopting model-driven approaches.

B. Inadequate Support from Modeling Tools

From the perspective of modeling tools, the issues can be categorized into the following aspects.

1) Steep Learning Curves for New Users

Existing DoDAF tools often present steep learning curves, particularly for multidisciplinary teams involving architects, engineers, and operators. This hinders effective collaboration, especially when stakeholders have varying levels of modeling expertise.

2) Insufficient Support for Model Reuse

Model reuse is a fundamental benefit of architecture description modeling [23]. However, in practice, the tightly coupled nature of elements within DoDAF-based architecture models often impedes effective reuse. This rigidity limits the adaptability of existing models to new systems or evolving contexts. While some of these issues stem from tool implementations, the underlying challenges are also rooted in the structural constraints and design philosophy embedded in the DoDAF metamodel itself.

3) Insufficient Support for Iterative and Agile Modeling

SoS architecture design is typically an iterative process, yet most DoDAF tools do not effectively support version control, impact analysis, or automatic updates. Furthermore, the weak integration between different design phases (e.g., from capability planning to system design) makes it difficult to transition seamlessly from conceptual models to executable or detailed design artifacts.

4) Limited Support for Modeling Dynamic Behavior

Most DoDAF tools are primarily designed to represent static structures and relationships. While activity and sequence models offer some capability to model and analyze dynamic behaviors, they lack the flexibility needed to handle a wide range of scenarios. This limitation makes it challenging to perform simulations or visualizations that accurately reflect the operation of SoS under varying conditions, thus reducing the practical utility of architecture models in operational analysis and decision-making.

5) Difficulty in Managing Large-Scale SoS Complexity

When dealing with complex SoS architectures, comprising a large number of activities, systems, and interfaces, many tools exhibit performance bottlenecks. This includes slow user interface responsiveness and delays in rendering large diagrams. Moreover, as the interconnections between elements grow more intricate, users often find it difficult to trace dependencies, leading to confusion and decreased confidence in the models.

6) Poor Interoperability with Other Tools

Despite the growing emphasis on integrated modeling environments, current DoDAF tools often operate in silos. They lack interoperability with executable modeling tools, such as Modelica, Simulink, or AnyLogic. Data format inconsistencies and the absence of standardized exchange mechanisms hinder seamless integration, resulting in duplicated efforts and inconsistencies between architectural models and executable simulations.

7) Lack of Intelligent Support

The modeling process can be cumbersome, adding to the already heavy workload of architects and SoS engineers, who are responsible for many other tasks. Current modeling tools offer limited intelligent assistance, such as automated reasoning, consistency checking, or even model auto-generation. The integration of advanced technologies, such as large language models (LLM), holds significant potential to improve these processes by offering smarter support.

C. Inherent Limitations of DoDAF and Inadequate Support of Modeling Methods

From the perspective of inherent limitations and inadequate methodological support, five key issues can be identified: the first two stem from the intrinsic limitations of DoDAF itself, while the latter three arise from shortcomings in existing modeling methods.

1) Over-Simplification of SoS Complexity

While the goal of ADFs is to develop stable blueprints, expressed through various views, for complex SoS—similar to blueprints for building architecture—the boundaries of an SoS are far more intricate than those of a building. The diversity of stakeholders, unclear boundaries (and sometimes even objectives), varying development timelines for constituent systems, and the occurrence of complex, unexpected emergent behaviors all contribute to the difficulty of representing an SoS. As a result, ADFs tend to oversimplify the inherent complexity of SoS, making the choice of appropriate abstraction critically important.

2) Underestimation of Evolutionary Nature of SoS

SoSs are inherently dynamic, evolving continuously in response to changing requirements, constituent system upgrades, and unforeseen operational conditions. However, DoDAF often treats architecture models as static snapshots rather than living artifacts that demand iterative validation and continuous adaptation. While views such as CV-3 (Capability Phasing) and SV-8 (Systems Evolution Description) attempt to address system evolution, they largely depict it as a predefined, static process. Furthermore, many types of changes are overlooked—for example, frequent updates to OV-5b (Operational Activity Model) and OV-4 (Organizational Relationship Chart) are seldom adequately captured or represented.

3) Inadequate Modeling and Verification Methods

Although many modeling methods have been proposed over the years, some fundamental issues remain, primarily stemming from the inherent subjectivity of the modeling process. A typical example is the lack of a systematic understanding of granularity levels, which leads to inconsistent model granularity—some levels are overly detailed while others are too vague, resulting in a disorganized hierarchy. These seemingly minor issues can hinder the development of effective and reliable models.

In terms of verification, most existing methods rely on syntactic checks and rule-based reasoning [5], which are insufficient for detecting complex logical errors. This limitation undermines the reliability of the models and erodes user confidence in their correctness and utility.

4) Unclear Boundary Between Representation and Decision-making

DoDAF models are designed to structure vague or incomplete information, define and formulate decision-making problems, and guide architectural decisions [22]. However, these decision-making issues often remain obscured within the architecture models. This ambiguity creates confusion, leading to uncertainty about whether the models are flawed due to insufficient modeling experience or a lack of adequate decision analysis.

5) Lack of Methodological Guidance for SoS Analysis

While DoDAF defines a set of views, it offers limited guidance on how to use these views to conduct architecture evaluations, trade-space exploration, or impact analysis. Users are often left to interpret the views without a clear methodological framework, leading to inconsistent and ineffective practices. More critically, in many real-world applications, users struggle to identify latent deficiencies or potential shortcomings in the architecture design as represented by the models.

D. Practical and Cultural Barriers to Model Adoption

Beyond the structural limitations of DoDAF and the constraints of current modeling tools, the successful adoption of architecture models in real-world SoS projects also faces practical and cultural challenges. These issues reflect broader organizational behaviors and workflow mismatches that hinder the integration of DoDAF-based modeling into engineering practice.

1) Models Focus on Compliance, Not Practical Use

In many defense projects, DoDAF models are developed primarily to satisfy contractual or regulatory requirements rather than to support real-world design decisions. This compliance-driven mindset turns modeling into a box-checking exercise, where deliverables are created to pass reviews but rarely maintained or reused afterward. Even when the importance of architecture modeling is acknowledged, organizations often lack incentives or processes to keep these models up to date throughout the system's lifecycle. Once initial approvals are secured, model updates are deprioritized, reinforcing the perception that architecture models are static documents rather than evolving, decision-support tools. As a result, the long-term value of model-based systems engineering is significantly diminished.

2) Model Maintenance is Costly and Operationally Unattractive

The effort required to keep architecture models aligned with rapidly changing systems often outweighs the perceived benefits. Teams may prefer to directly update prototypes or source code, bypassing the architecture layer entirely. As a result, models quickly become outdated and are abandoned, viewed as an unsustainable overhead rather than a valuable asset for ongoing development

3) Engineers and Architects Speak Different Languages

A cultural gap exists between architects, who work within frameworks like DoDAF, and engineers, who focus on building and testing systems using simulation environments or programming languages. Engineers often find that DoDAF models are too high-level to support executable behavior or real system implementation in tools like Python or Simulink. This disconnect hampers collaboration and limits the effectiveness of architecture-driven development, leaving the architecture models isolated from actual system implementation.

IV. OPPORTUNITIES FOR IMPROVEMENT

Based on the identified challenges, we first evaluate whether UAF and SysML 2.0 can address some of these issues, and then propose several directions to enhance the practical application of DoDAF—applicable to UAF as well—in supporting SoS architecture design.

A. UAF's Capability to Address the Issues

As discussed in Section II, the UAF consolidates multiple architecture frameworks and offers more comprehensive views and dimensions compared to DoDAF. At its core, UAF establishes an integrated meta-model that enhances the semantic consistency and structural rigor of architecture representations. This unified meta-model also enables improved traceability from architectural elements to capability objectives by systematically linking functions, resources, and operational activities to capability definitions and performance measures.

Importantly, the OMG provides extensive support for UAF adoption, including the UAF Domain MetaModel (DMM), the UAF Modeling Language (UAFML), and a practical guide for enterprise architecture development. These resources offer more structured methodological guidance and clearer modeling practices than DoDAF, contributing to improved usability and standardization in SoS architecture design. Furthermore, UAF aligns more closely with MBSE principles and SysML [26], facilitating tighter integration between SoS architecture modeling and system lifecycle management.

Nevertheless, despite addressing fragmentation and enhancing semantic clarity, UAF still faces practical adoption challenges—particularly in terms of modeling methodology, tool maturity, and organizational constraints—as discussed in Section III.

B. SysML 2.0's Capability to Address the Issues

The current modeling language, SysML, is undergoing a significant transformation with the development of SysML 2.0. The SysML 2.0 standard focuses on three core elements, the underlying Kernal metaModeL (KerML), modeling semantics and syntax in the SysML, and the Application Programming Interface (API) and services [27]. It integrates graphical and textual modeling approaches, bridging the language gap between system architects and domain engineers. At the same time, it enhances modeling flexibility and efficiency, while supporting model sharing and automation. This revision aims to improve usability for systems engineering practitioners by introducing these more intuitive language constructs, enhanced expressiveness, and better model organization.

SysML 2.0 also defines standardized APIs that enable seamless integration with simulation engines and verification tools, significantly enhancing interoperability across the system development lifecycle. Moreover, it offers improved composability, allowing for more coherent and scalable representations of hierarchical structures—from SoSs to individual systems and components.

Moreover, its support for a formal textual syntax makes it naturally compatible with LLMs (e.g., ChatGPT, DeepSeek), enabling more interactive model manipulation, streamlined workflows, and reduced modeling complexity [28].

SysML 2.0 holds strong potential to address many of the challenges outlined in Section III; however, most of these anticipated benefits have yet to be validated in practice, and realizing them would require significant retooling of existing tools and workflows.

C. Improvement Suggestions

1) Architecture Description Models Reflect Architecting Process more than Architecture Outcomes

Rather than building complete DoDAF models upfront, development teams should focus on creating evolving, minimal viable models. Fig. 2 illustrates an iterative architecture modeling process that encompasses architecture modeling, analysis, evaluation, and decision-making. Simultaneously, enabling different stakeholders to contribute at varying levels of detail promotes better collaboration and aligns with agile development principles.

Fig. 3 demonstrates an example of iterative architecture modeling process that integrates DoDAF models, executable models (e.g., ExtendSim, Anylogic), and decision models. The decision models include qualitative decisions that help

collect constraints/rules and clarify the information for architecture models, and quantitative decision-making and evaluations based on executable simulation results. Compared to the traditional paradigm [9], the key emphasis is placed on an iterative modeling process rather than delivering a complete set of architecture models all at once. Our core argument is that architecture models should serve as a means to guide and evolve with the architecting process, rather than simply capture its final products.

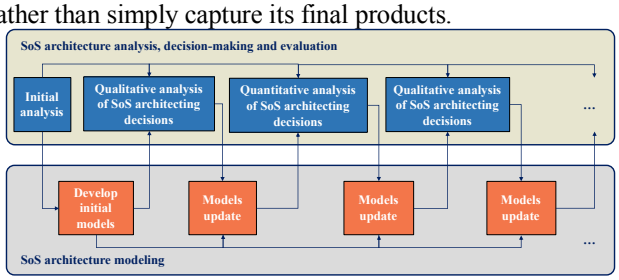

Figure 2 . Iterative architecture modeling process.

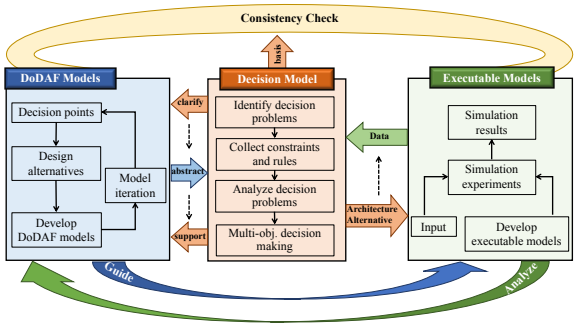

Figure 3 . An example for iterative architecture modeling process that integrates DoDAF models, executable models, and decision models.

2) AI-Assisted Architecture Modeling and Design

Recent artificial intelligence (AI) technologies offer significant potential for supporting SoS architecture design. As listed in Fig. 4, AI can support this process in four key areas: AI-assisted architecture modeling, AI-assisted architecture selection, AI-assisted architecture verification, and AI-assisted architecture evolution.

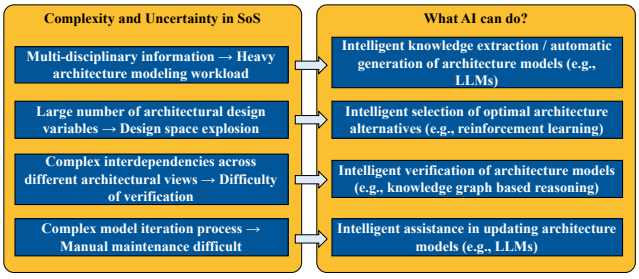

Figure 4 . Issues requiring AI assistance and potential solutions.

Among these areas, AI-assisted architecture modeling and evolution have attracted significant attention in the past two years, primarily due to the challenges associated with manual model development and maintenance, which are both labor-intensive and error-prone. Fig. 5 illustrates the generation process of architecture models (e.g., SysML or DoDAF models) using LLMs, which support the automatic generation of functional/component decompositions, activity models, and other artifacts in standard XML format. These standard XML models can then be transformed into XML structures compatible with SysML or DoDAF specifications.

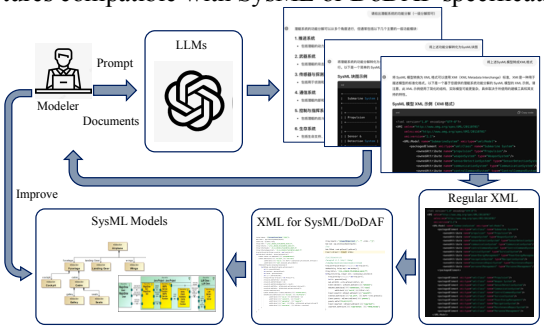

Figure 5 . Architecture model generation framework based on LLMs.

AI-driven techniques, when integrated with model version control, also show strong potential for automatically detecting inconsistencies, recommending updates, and managing complex dependencies. Furthermore, the ability to synchronize SysML/DoDAF/UAF models with real-time operational data could greatly enhance the timeliness and accuracy of model updates throughout the design lifecycle.

3) Customized Metamodel Development and Underlying Consistency Assurance

To better support domain-specific needs, organizations can develop customized meta-models that extend or specialize existing frameworks (e.g., DoDAF, UAF). These tailored meta-models allow for more precision in addressing specific requirements of a given system or domain. An integrated process of SoS architecture development and meta-model development is illustrated in Fig. 6.

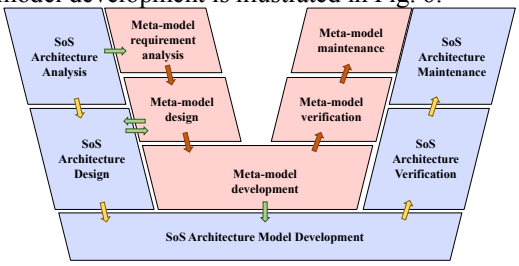

Figure 6 . SoS architecture design process with meta-model development.

It is important to note that developing customized meta-models introduces the challenge of maintaining consistency across different modeling views and with other frameworks used by different organizations. To address this, consistency assurance mechanisms must be integrated into the meta-model development process. This includes defining clear consistency rules and validation methods to ensure that models derived from the customized meta-model align with the intended system structure and behavior, while also ensuring better compliance with existing meta-models.

V. Conclusion and Future Work

This paper has analyzed the key challenges facing DoDAF in the SoS architecture design, including misconceptions, method limitations, inadequate tool support,

and organizational barriers. Our findings indicate that the core issue lies in treating DoDAF as a static documentation tool rather than a dynamic decision-support asset that must evolve throughout the lifecycle. Several key lessons emerged from this investigation. We observed that organizational and technical barriers are deeply intertwined, each exacerbating the other. A recurring difficulty was distinguishing whether problems originated from DoDAF's inherent limitations, tooling deficiencies, or methodological misapplication. While newer frameworks like UAF offer improved semantic consistency, our findings temper expectations regarding their immediate utility, as they still face challenges in method and tool maturity. The integration of AI-assisted modeling presents a promising yet challenging path forward.

Future work will focus on three directions: developing a lightweight iterative modeling plugin to integrate architectural models with decision-support tools; creating a specialized prompt engineering framework for LLMs tailored to SoS architecture tasks; and establishing quantitative metrics to empirically validate improvements in model maintenance efficiency and decision-support capability. Eventually, transforming DoDAF from a documentation exercise into an evolving intelligent decision-support process represents quite a promising direction for enhancing its practical value in complex SoS environments.

REFERENCES

[1] ISO/IEC/IEEE, "ISO/IEC/IEEE 42010:2022(E) Systems, Software and Enterprise - Architecture Description," 2022.

[2] Joint Staff, "Charter of the Joint Requirements Oversight Council and Implementation of the Joint Capabilities Integration and Development System," 2021.

[3] G. Bangjun, C. Yunfeng, W. Xinyao, and Y. Wangwang, "Modeling the Anti-UAV Swarm System Architecture Based on DoDAF," in 2023 9th Int. Conf. Big Data Inf. Anal., Haikou, China, Dec. 2023, pp. 409–412.

[4] A. Aghamohammadpour, E. Mahdipour, and I. Attarzadeh, "Architecting Threat Hunting System Based on the DoDAF Framework," J. Supercomput., vol. 79, pp. 4215–4242, 2023.

[5] M. Vinarcik, "The Problem with DoDAF Models," in 22nd Annu. Syst. Mission Eng. Conf., Tampa, FL, 2019.

[6] S. W. Mitchell, "Transitioning the SWFTS Program Combat System Product Family from Traditional Document-Centric to Model-based Systems Engineering," Syst. Eng., vol.17, no.3, pp.313-329, 2014.

[7] M. Zaman, "Dynamic Resilient Enterprise Architecture Model (DREAM) Adoption in Defense Digital Architecture Management to Mitigate Disconnected Systems and Processes," Ph.D. dissertation, George Mason Univ., 2025.

[8] [Reddit], "Change My View: Model-Based Systems Engineering in 2024 is Still More Hype Than Value," r/systems_engineering, 2024. [Online]. Available: https://www.reddit.com/r/systems_engineering/comments/1bpavpi/change_my_view_model_based_systems_engineering_in/?rdt=57765 [Accessed: Sept, 21st, 2025].

[9] L. W. Wagenhals, I. Shin, D. Kim, and A. H. Levis, "C4ISR Architectures II. A Structured Analysis Approach for Architecture Design," Syst. Eng., vol. 3, no. 4, pp. 248–287, 2000.

[10] C. Piaszczyk, "Model Based Systems Engineering With Department of Defense Architectural Framework," Syst. Eng., vol. 14, no. 3, pp. 305-326, 2011.

[11] R. Wang and C. H. Dagli, "Executable System Architecting Using Systems Modeling Language in Conjunction With Colored Petri Nets in a Model-Driven Systems Development Process," Syst. Eng., vol. 14, no. 4, pp. 383-409, 2011.

[12] M. Amissah and H. A. H. Handley, "A Process for DoDAF-Based Systems Architecting," in 2016 Annu. IEEE Syst. Conf., Orlando, FL, USA, 2016, pp.1-7.

[13] J. N. Martin and D. P. O'Neil, "Enterprise Architecture Guide for the Unified Architecture Framework (UAF) ," in Proc. INCOSE Int. Symp., vol. 31, pp. 242-263, 2021.

[14] H. M. Hause, "Rebuilding the Tower of Babel: The Case for a Unified Architecture Framework," in Proc. INCOSE Int. Symp., vol. 23, no. 1, pp. 1460–1474, 2013.

[15] J. N. Martin, "Aspect-Oriented Architecting Using Architecture Frameworks," in Proc. INCOSE Int. Symp., vol. 31, no. 1, pp. 210–226, 2021.

[16] J. N. Martin, "Extending UAF for Model-Based Capability Planning and Enterprise Portfolio Management," in Proc. INCOSE Int. Symp., vol. 32, no. 1, pp. 15-35, 2022.

[17] K. Carroll, A. Lyle, R. Lewark, C. Medina, and A. Morkevičius, "SoS - Global Solutions to Global Problems Using UAF," in Proc. INCOSE Int. Symp., vol. 34, pp. 2400-2412, 2024.

[18] M. Hause and L. O. Kihlström, "Modeling Enterprise Software with UAF," in Proc. INCOSE Int. Symp., vol. 34, pp. 2452-2475, 2024.

[19] J. N. Martin and K. E. Alvarez, "Using the Unified Architecture Framework in Support of Mission Engineering Activities," in Proc. INCOSE Int. Symp., vol. 33, pp. 1156–1172, 2023.

[20] M. Gagliardi, M. C. Hause, J. N. Martin, and M. A. Phillips, "Darth Vader's Secret Weapon: Implementing Mission Engineering With UAF," in Proc. INCOSE Int. Symp., vol. 34, pp. 1719-1747, 2024.

[21] Office of the Under Secretary of Defense for Research and Engineering, Department of Defense Mission Architecture Style Guide. Washington, DC, USA: U.S. Department of Defense, 2025.

[22] Z. Fang, X. Zhao, and F. Li, "Architecture Design Space Generation via Decision Pattern-Guided Department of Defense Architecture Framework Modeling," Systems, vol. 12, no. 2, p. 336, 2024.

[23] A. E. Trujillo and A. M. Madni, "MBSE Methods for Inheritance and Design Reuse," in Handbook of Model-Based Systems Engineering, A. M. Madni, N. Augustine, and M. Sievers, Eds. Cham, Switzerland: Springer, 2023.

[24] A. Morkevicius and G. Krisciuniene, "Towards UAF Implementation in SysML V2," in Proc. INCOSE Int. Symp., vol. 34, pp. 2452-2475, 2024.

[25] M. W. Maier, "Adapting the Hatley-Pirbhai Method for the Era of SysML and Digital Engineering," 2022 IEEE Aerosp. Conf., Big Sky, MT, USA, 2022, pp. 1-12.

[26] J. N. Martin and D. Brookshier, "Linking UAF and SysML Models: Achieving Alignment Between Enterprise and System Architectures," in Proc. INCOSE Int. Symp., vol. 33, pp. 1132–1155, 2023.

[27] Object Management Group, "OMG Systems Modeling Language (SysML) Version 2.0 Beta 2 (Revision 2024-02), Part 2: SysML v1 to SysML v2 Transformation," OMG Doc. No. ptc/2024-02-01, Feb. 2024.

[28] J. K. DeHart, "Leveraging Large Language Models for Direct Interaction With SysML v2," in Proc. INCOSE Int. Symp., vol. 34, pp. 2168–2185, 2024.