Carnegie Mellon University
Software Engineering Institute

Minimally Viable Architecture: Architecture Early in Development

Manuel A Rosso-Llopart

October 2025

TECHNICAL NOTE
CMU/SEI-2025-TN-001
DOI: 10.1184/R1/20362266

CERT® Division

https://www.sei.cmu.edu

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM25-1336

Table of Contents

List of Figures

List of Tables

Executive Summary

Utilizing architecture on an Agile project can be challenging. The primary business driver from an Agile point of view is time (i.e., schedule). Sprints, releases, and program increments are all drivers to get things done quickly. In many cases, documentation, like architecture, is viewed as secondary to "producing code."

However, leaving the architecture analysis for later in the development process bypasses the need, in many cases, for a systemic analysis of the solution that is being proposed before implementation. Certain early decisions in development will determine the nature and attributes of the final solution. This can lead to a solution that meets only some of the key systemic drivers; the architecture change that the SEI proposes in this technical note will help avoid this critical issue.

This technical note explores the concept of minimally viable architecture (MVA), which is architecture that can be used by Department of Defense (DoD) development teams as they build products. After an MVA is developed, the unified modeling language (UML) frameworks can be used to enhance the MVA throughout the life of the project as greater detail and understanding are needed to maintain the effort and ensure the quality attributes (QAs, also known as systemic properties or non-functional requirements) are met by the solution.

Overall, the SEI recommends that DoD teams begin to develop their architecture by using an MVA. This technical note offers guidance on what these teams should do with an MVA to get their certificate to field (CTF) and authorization to operate (ATO).

1 Background: Agile and Architecture

In a Department of Defense (DoD) software development project, cybersecurity staff must approve the product so that it can be fielded to the real world. This requirement means that cybersecurity is one of the key drivers and highest quality attributes (QAs) for the project, and cybersecurity staff must demonstrate that security risks have been addressed and that the product meets the policy and governance demands of the organization. All software must be compliant with software assurance policies. One primary reason for developing an architecture that captures the decisions made throughout the development of the software system is to ensure that risks associated with meeting organizational QAs have been mitigated and that the software is certifiable to field.

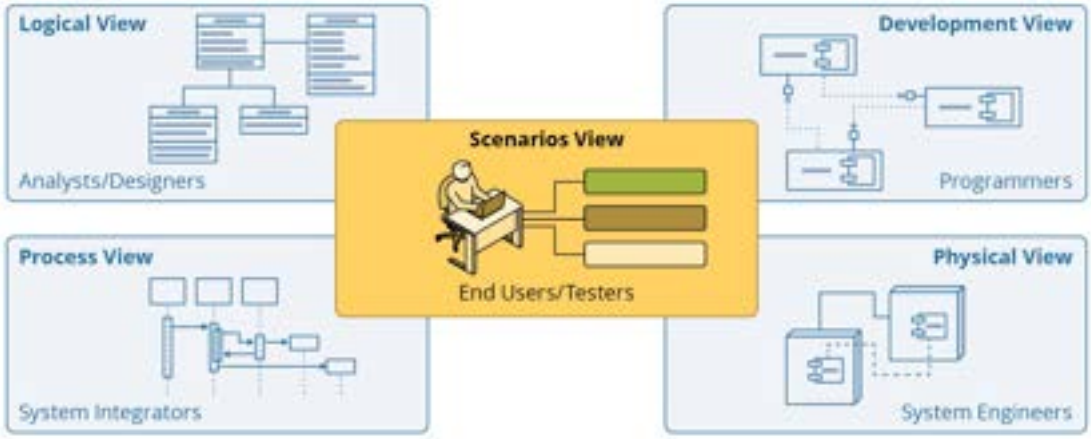
To develop such an architecture, a team should first develop a minimally viable architecture (MVA), which will provide an overview of the system and will allow the team to analyze the system and identify risks. The following non-exhaustive list identifies some key pieces of information that are used to ensure that a system meets the organization's needs and the stakeholder's operational needs:

- the key assumptions that have been made to field the system
- the interface demands that the system will make on the environment
- how the stakeholders will interact with the new system
- the technical debt (assumptions not verified, key decisions deferred) that the system will create [Kruchten 2019]
- the risks that need to be mitigated
- how the system will evolve from a current state to final product for the customer

All this information and more can be captured in an MVA.

The critical guidance in using an MVA is understanding what drives the need for one. Most teams try to clearly understand the functional requirements. They do this through the use of user stories that represent the operational need. However, to make the solution compatible with the organization's system of systems, a bigger picture view is needed. The MVA instantiates the bigger picture view.

Using the Krutchen 4+1 framework (see Figure 1), the team that is developing the model for the product that the MVA will support needs to look at the four primary areas from the 4+1 framework. The "+1" is usually already covered by the team's use of user stories and/or use cases that they create. The team normally also has already begun thinking in terms of the development view, which is represented by patterns that might be used and how components can interact. So, to fill out the 4+1 model for the MVA, the other three views, which are detailed below, should be considered.



This diagram was adapted from Botero and Piedrahita's article "An Architecture for Power Substations Communication Networks Based on SDN and Virtualization Paradigms" [Botero 2021].

Figure 1: Kruchten's 4+1 Model

The process view is an outward looking view that is related to the interfaces needed and tells the team how the new service will interact with other services. In this view, the team considers how external events will affect the new service and change the product's state of operation. Then the team should understand the logical view, which is what the new product must do to manage these inputs that can affect its states. Lastly, the team should consider the physical view in which the team establishes a clear understanding of how the environment in which the new product or service will operate [Kruchten 1995].

The order in which these views are examined is not necessarily critical; however, the need to understand gaps is critical. By looking at the new product or services from these four different views, teams can better understand the systemic properties of the proposed solution. The MVA minimally addresses each of the 4+1 views so that the teams can identify gaps in their understanding before they start building a service that creates problems that affect the quality of their product.

However, before teams develop an MVA, they should consider how components of the Agile approach and Scaled Agile Framework (SAFe) concept, which they are probably already using, can inform the creation of an MVA.

In an Agile approach,[1] teams tend to create architecture information in the "as built" phase. But architecture as a development tool is far more powerful if done as a "to be" element represented by an MVA. Taking this approach allows the team time for analysis and identification of key challenge points (see non-exhaustive list above) while building a system or system of systems.

Building architecture information in the "as built" phase is further complicated by the fact that the typical Agile approach has "time to market" as the predominant driver. In many cases, deadlines are immovable and drive the timebox in which the product is produced. But sometimes deadlines are negotiable, depending on what the customer expects in the promised minimal viable product

[1] Review the "Manifesto for Agile Software Development" for more information on the basic concepts of Agile development [GeeksforGeeks 2024].

(MVP)[2] or minimum viable capability release (MVCR).[3] These are standard concepts in the Software Acquisition Pathway (SWP) framework required by the DoD [DoD 2020].

The SAFe concept also suggests that there is a need to understand architecture and design as part of the Agile process. In the SAFe concept, intentional architecture is a set of purposeful, planned architectural strategies and initiatives that enhance solution design, performance, and usability as well as guide cross-team design and synchronized implementation. Intentional architecture is similar to an MVA [Scaled Agile 2024] and would be satisfied by an MVA. Teams that follow the SAFe concept are already gathering information to understand the system to be built; an MVA would simply reorganize this information into a model that can be used to evaluate the decisions being made in the project.

Another SAFe term is "emergent design." It provides the technical basis for a fully evolutionary and incremental implementation approach. This helps developers and designers respond to immediate user needs, allowing the structure to evolve as the system is built and deployed [Scaled Agile 2024]. Development teams engage in emergent design as part of the Agile process.

Balancing the intentional architecture and the emergent design is essential because teams that look at design only during development lose the big picture provided by the intentional architecture, and the MVA allows the team to keep the "big picture" in mind.

In general, the Agile development process (represented by SAFe and current DoD development guidelines) is an evolutionary process, with a product being built rapidly over time and stakeholders giving feedback on the product. Architecture, like development, is an evolutionary process supported by MVA development. Figure 2 shows scaled Agile roles and domains that influence the development of a system. It also indicates the evolution of a system from enterprise down to service. The architecture evolves as decisions are made on how to satisfy operational needs.

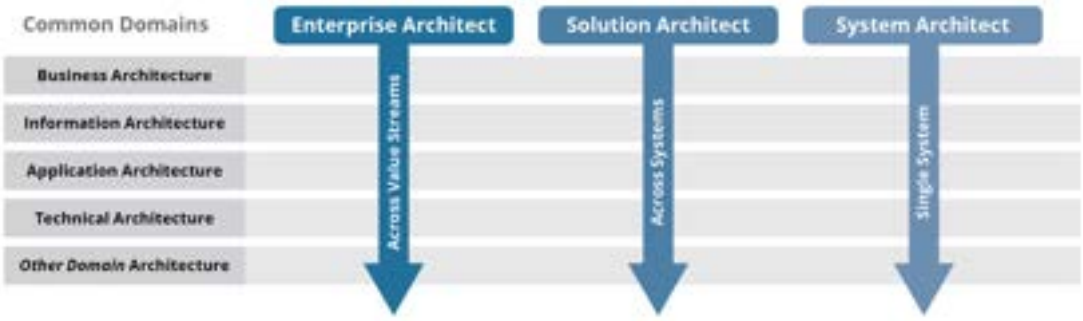


Figure 2: Image Adapted from Capgemini's "Agile and Architecture – Lessons from the Trenches" [Capgemini 2020].

[2] A minimal viable product (MVP) is an early version of the software that includes the basic capabilities and is delivered or fielded to users who evaluate and provide feedback on it. Insights from MVPs help shape the software's scope, requirements, and design.

[3] A minimum viable capability release (MVCR) is the initial set of features suitable to be fielded to an operational environment that provides value to the warfighter or end user in a rapid timeline. The MVCR delivers initial warfighting capabilities to enhance some mission outcomes. The MVCR is analogous to a minimum marketable product in commercial industry.

To implement a balanced approach to design, architects and teams should prioritize the following:

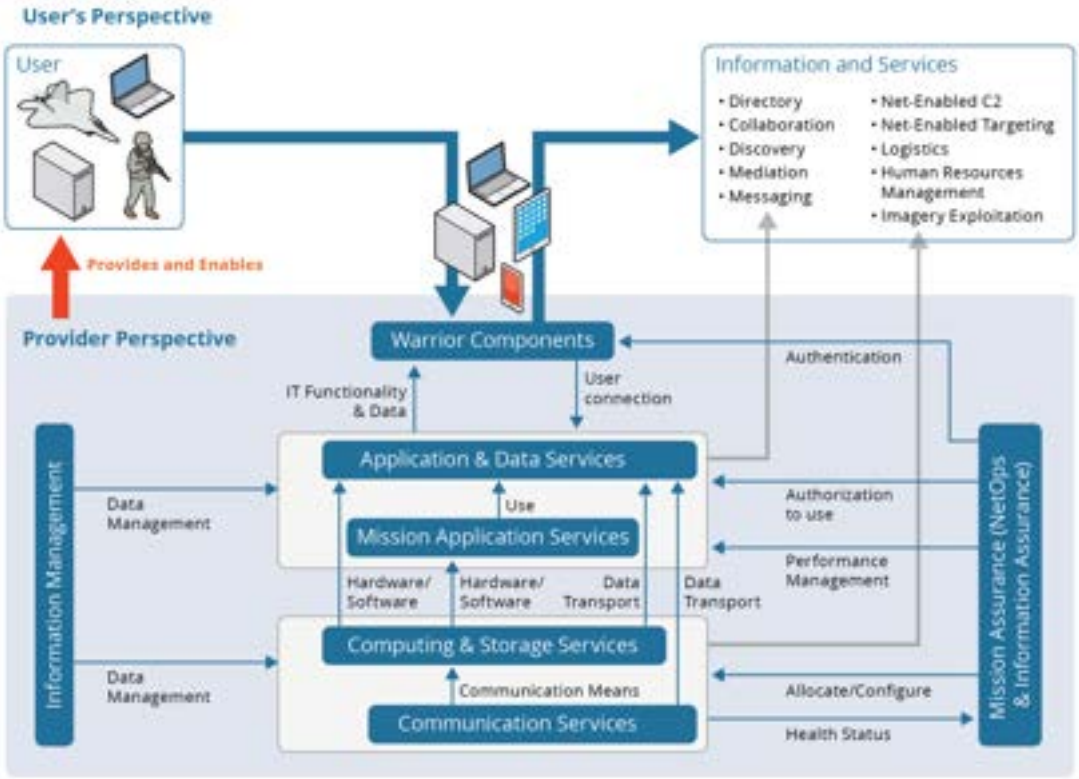
1. understanding the system's goals and requirements before designing the system
2. identifying the areas where intentional design is necessary to build a solid foundation
3. being willing to adapt and evolve the design as the system is developed
4. encouraging experimentation and learning to discover new and innovative solutions
5. striving for consistency and cohesion in the design to maintain the system's quality

By balancing intentional design (i.e., an MVA) and emergent design, the team can build a flexible, adaptable, and responsive system to change business needs while maintaining a clear direction and vision for future development.

2 Recommendations for Building and Using an MVA

2.1 A DoD Software Project Needed for Cybersecurity Approval[4]

For a team to know the value of the architecture, the team must understand the context in which the software will operate. This context is normally shown in a high-level diagram, which some people call the high-altitude view. Figure 3 is an example of a high-level view that shows the context in which a solution will exist. The reference to "Authorization to Use" in the diagram might give context to the developers of the domain in which their solution will operate. (An example area of interest in Figure 3 is highlighted with a red circle).



This Diagram was Adapted from the DoD's "DoD Information Enterprise Architecture v2.0" [DoD CIO 2012].

Figure 3: Example High-Level Diagram

The team must have a very clear understanding of the systemic properties that are critical for the success of their products. Table 1 summarizes example QAs that represent design principles and the systemic properties that must be met at the organizational level. Development teams must adhere to these kinds of non-functional requirements.

[4] Note that the recommendations in Section 2.1 use the 4+1 model [Kruchten 1995].

Table 1: Summary of Example Design Principles [5]

System Property	Description	Quality Attributes (QAs)
Hierarchical Approach to Design	The hierarchal approach drives the concept of ranking components and is instantiated to support the relationship in the example's components. The hierarchal approach is a software engineering methodology in which the entire system is seen as a tree structure and tree layers that are decomposed into logical modules or subsystems at different levels in the hierarchy.	Scalability Simplicity (-) Adaptability
Observability	Observability is used in system engineering and design to describe the extent to which the internal state or conditions of a complex system can be understood by an observer who only has knowledge of the system's external outputs.	Fault Tolerance Portability (-) Runtime Performance
Modularity	The concept of a modular design aligns with the design principle of hierarchal design. In modularity, platform as a service (PaaS) can provide enterprise services. These services—which will be flexible in design when adding, removing, and updating specific services—increase cohesion and reduce coupling as much as possible in the system.	Extensibility Modifiability (-) Performance
Increased Speed	This design principle of increased speed (or performance) was identified to ensure that development teams consider general design concepts that increase the speed of process and data flows at critical points in the design of the system.	Development Performance (-) Runtime Performance
Increased Availability	Availability is a term used in system engineering and design to describe the percent of time that a system is available to perform its task and function under normal conditions.	Availability Reliability (-) Security

2.2 Agile Development and an MVA

This section offers suggestions on how to better utilize architecture in an Agile environment to produce and maintain an MVA.

The following is a basic overview of how to construct an MVA and what should be included in the process as an MVA is instantiated:

1. The Agile team creates an example overview to provide a big picture of where the solution fits into the system.
2. The Agile team discusses what should be in the MVA based on the 4+1 Krutchen views (defined below) [Kruchten 1995].
3. The team conducts a review of the common DoD modeling language frameworks that can be used to represent the MVA.
4. The team determines how these framework components are needed to support the 4+1 architectural views [Kruchten 1995]. (Cybersecurity is a critical QA that must be met.)

After the steps above are completed, the team will have the information needed to support their MVP, which they will present to the customer.

[5] The full descriptions and original table are available from the Unites States Cyber Command's source titled, "JCWA 2.0 TEM Output - Joint Cyber Warfighting Architecture (JCWA)," which is not publicly available but was provided separately to those individuals who have a Common Access Card (CAC).

This adoption of an MVA does not require a policy change but rather suggests how DoD teams can better support the cybersecurity requirements associated with obtaining a certificate to field (CTF) and an authorization to operate (ATO) or maintain a continuous ATO (cATO).

Organizations that use Agile and SAFe practices normally conduct these practices together. The MVA will not impact these practices, including program increment (PI) planning, sprint planning, etc., other than to suggest that time be applied to ensure that the MVA is maintained and available to support cybersecurity assurance practices.

2.3 MVP Process with MVA

The 4+1 framework is an example of an architecture framework that can be used to capture architectural information.[6] It allows the teams to see the "product" from different points of view that can help identify gaps in understanding [Kruchten 1995]. Using this framework will enable the team to reduce technical debt and mitigate architectural risks earlier in the development process when changes can have the highest value and lowest cost.

Before beginning the steps outlined below, the team should validate with the customer a big-picture diagram that team members use to scope the project. Having this concept diagram allows the team to begin the MVA process.

The basic MVA process is

1. Develop the basic representation of the solution throughout the sprints. For example, Architecture Monday may be a technique that the team uses throughout the development cycles. To meet the needs of the customer, the design will evolve in each sprint toward the final goal.
2. Focus on creating minimal views that allow the team to look for gaps and identify risks in the solution (e.g., Krutchen 4+1 [Kruchten 1995]).
3. Use the MVA to guide and capture decisions being made. The MVA can be used to understand how selected design patterns fit the solution. (Expand the MVA as needed to validate captured decisions.)
4. Verify that the created product matches the patterns used to solve the problem.

As the MVP represents the first decisions the team makes toward a customer deliverable, the QAs must be considered to meet all stakeholder needs. These first products produced by the team instantiate design decisions on which the rest of the product will be delivered. These features, which show the customer what can be done, "harden" architectural decisions over time. The impact on the architecture, which can be identified in the MVA, will help provide the correct quality to meet all stakeholder needs. (For example, see the organizational needs listed in Table 1.)

[6] See Appendix A for a list of other available architecture frameworks that can be used in developing an MVA.

2.4 Deliverables As Part of the MVA

Philip Krutchen first defined what can be considered an MVA in his 4+1 model, which is depicted in Figure 3. In Kruchten's assessment of an MVA, he argues that there is a minimal set of views that is needed to begin the analysis of the architecture to determine if it meets all stakeholders' needs [Kruchten 1995].

Kruchten's 4+1 view model describes software architecture from multiple stakeholder views, including the following:

- **The logical view** describes when the required system is decomposed into a set of key abstractions, taken (mostly) from the problem domain. (This view is about how the software will work internally.)
- **The process view** depicts how the main functional abstractions map onto executing processes and threads of control. (This view is about how the software will work with other systems.)
- **The physical view** reflects distributed aspects by showing how the software maps onto the hardware (physical environment).
- **The development view** focuses on the actual software module organization in the development environment [Baragry 2001].

Table 2 shows how the 4+1 model can be related to the common UML DoD structure Department of Defense Architecture Framework (DoDAF) [DoD CIO 2010a]. UML diagrams are typically used by teams to identify key decisions made for development. The 4+1 model simply organizes the information into groups for an MVA that allows analysis of decisions and identification of gaps.

Table 2: Comparison of the 4+1, DoDAF, and the UAF Viewpoints

View Name	Corresponding UML Diagrams	Example: Minimum Diagram Views from DoDAF representation	+ Minimum + Diagram Views from UAF representation
Logical View	Class Diagram, Object Diagram, Component Diagram, Package Diagram, Composite Structure Diagram	OV-1, SV-1, SV-4, CV-1	St-Sr, Sm-Ov, Op-Is, Sv-Sr
Process View	Activity Diagram, State Machine Diagram, Sequence Diagram, Timing Diagram, Interaction Overview Diagram	Sequence Diagram, Flow diagram	Sv-Pr, Sv-St
Physical View (Deployment View)	Deployment Diagram	OV-1, OV-5b SV-10(b), SV-4	St-Sr, Rs-Sr
Development View (Implementation View)	Component Diagram, Package Diagram,	SvcV-1 Services or functional view, SvcV-4	Rs-Pr, Sv-St
Use Case View	Use Case Diagram	Scenarios, User Stories, OV-6	Sc-Sr, St-Sr

A Comparison of the 4+1, DoDAF, and the UAF viewpoints [Kruchten 1995, DoD CIO 2010a, UAF 2025]. (View names are the 4+1 elements.)

See Appendix B for detailed examples of how the DoD Architecture Framework (DoDAF) model [DoD CIO 2010a] matches with the views from the 4+1 model [Kruchten 1995].[7]

[7] No Magic, a company founded in 1995, created Cameo, a tool for diagrammatically representing models. This company embraced several of the framework standards for representing engineering models, such as DoDAF, UAF, and others. They have been able to take many of the standards and represent the information in summary pages, which can be found in the appendices of this technical note. No Magic was acquired by Dassault Systèmes in 2018 [No Magic 2025].

3 Conclusion: Balancing Demands for Architecture

Development teams should use architecture principles early in their process to validate decisions being made throughout the Agile development process. Using architecture principles improves the correctness and completeness of the architecture by holistically evaluating the needs of all stakeholders. For example, it allows the cybersecurity assurance process to proceed and support the team in obtaining the ATO and CTF. Reviewing the software architecture provides a way for the team to evaluate decisions made, which is critical to ensuring that the software product will meet the needs that the customer has identified. By ensuring that the architecture continually evolves and the minimally viable critical release is shared, the team is able to create an MVA that shows how the product has evolved to meet the warfighter's mission needs.

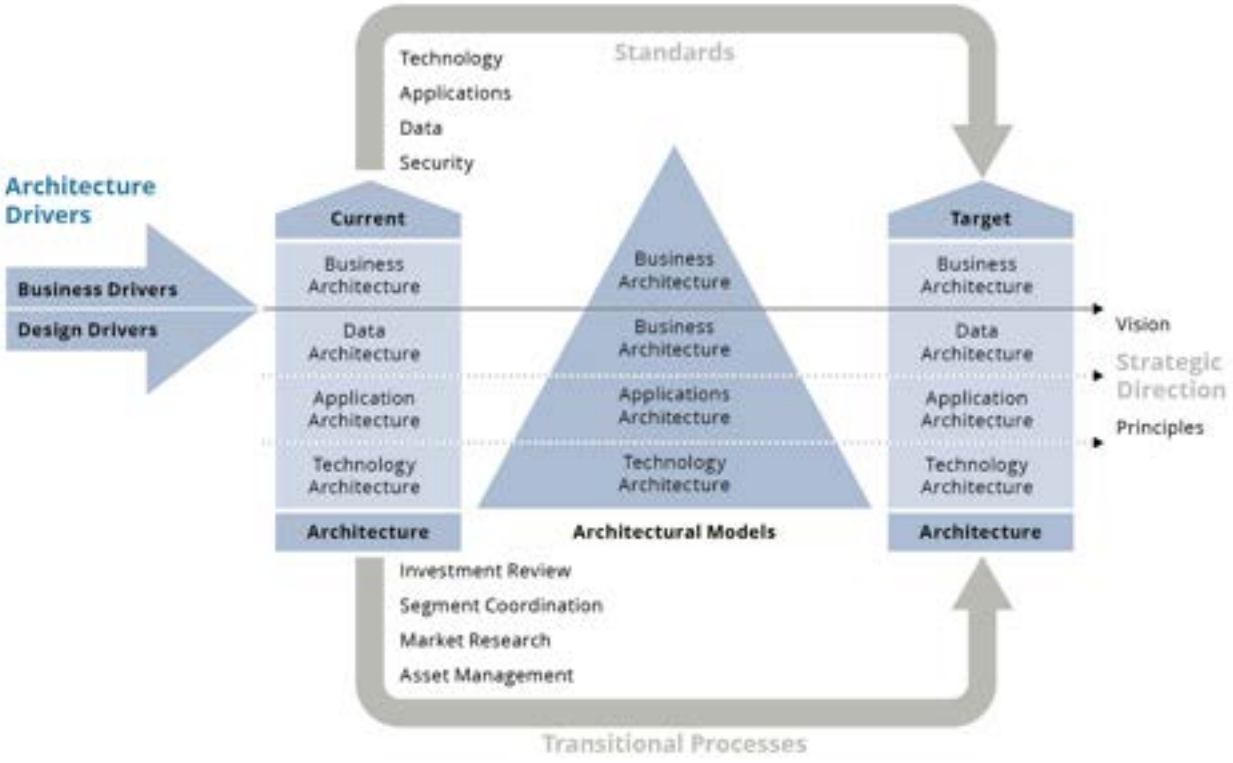
To change the way the team manages information and to support software development, teams should do the following:

1. Develop the concept of an MVA that coincides with the MVP.
2. Provide a context view of the product being developed so that the environment is well understood.
3. Use the 4+1 Krutchen framework for developing the MVA [Kruchten 1995].
4. Use an industry standard framework to represent the architecture views of which the MVA will be composed.

These steps comply with the basic principles in the Software Acquisition Pathway [DoD 2020] and SAFe principles for how architecture is part of Agile development efforts [Scaled Agile 2024].

Appendix A: Other Architecture Frameworks

This appendix provides examples of enterprise architecture frameworks.[8] Figure 4 gives an overview on how to select an architecture framework [Conexiam 2025].



This diagram was adapted from Conexiam's article "The Three Types of Enterprise Architecture Frameworks" [Conexiam 2025].

Figure 4: How to Select an Architecture Framework

The following frameworks are other common enterprise architecture frameworks that could be used. When selecting a framework, the best rule to follow is to use the right tool for the right environment:

- **Zachman Framework** is a fundamental structure that provides a comprehensive view of an organization and its information systems. The framework shows how various components are interconnected [Visual Paradigm 2025].
- **TOGAF** (The Open Group Architecture Framework) is a modular, scalable, and configurable framework that's one of the most common in business today. It includes an architecture capability framework to ensure that the organization's architecture is aligned with its business strategy and goals [Conexiam 2025].

[8] See the articles "A Minimum Viable Product Needs a Minimum Viable Architecture" [Bittner 2022] and "What Is Minimum Viable Architecture? Why Is It Important?" [Protecto 2021] for additional resources on architecture frameworks.

- **DoDAF** is the foundational structure for the DoD that provides a visualization infrastructure for stakeholders [Bizzdesign 2025].
- **ArchiMate** is a graphical modeling language and framework that provides a standardized notation to describe, analyze, and visualize relationships between architectural domains [SAP 2025].
- **MODAF** (Ministry of Defence Architecture Framework) is used by the UK Ministry of Defense to develop and document enterprise architectures for defense-related systems [Soto 2023].
- **FEAF** (federal enterprise architecture framework) is a framework issued by the U.S. Chief Information Officers (CIO) Council that provides guidance on categorizing and grouping IT investments [Tharpe 2020].
- **SAP Enterprise Architecture Framework** is an expansion of the TOGAF framework that combines SAP's reference architecture library and the option to customize the methodology [Sheppard 2024].

Appendix B: Detailed Example of How to Use DoDAF to Comply with 4+1 Modeling

This appendix provides additional information on how to use the DoDAF framework [DoD CIO 2010a] to fill in the 4+1 model [Kruchten 1995]. The goal of this appendix is to help teams understand where there is overlap between these approaches and the work they are already doing. Often, when developing an MVA, a workflow simply needs to be reorganized to fulfill the requirements in the 4+1 framework. (This appendix uses the DoDAF nomenclature, but UAF can be substituted [UAF 2025].)

The following is a high-level example of how the DoDAF or UAF frameworks can pair with the 4+1 model. The framework that is best understood by the team can be used to represent the information needed. For example, using UAF

- the concept of Op (Operation) fills the 4+1 logical view
- the concept of Sv (Service) fills the 4+1 process view
- the concept of Rs (Resource) fills the 4+1 physical view
- the concepts of St (Strategic) and/or Sc (Security) fill the 4+1 development view

Below is in-depth information on views from the 4+1 model and relevant language from the DoDAF to further clarify how these models can match to one another. There is also overlap across some of these sections (e.g., Overview [OV] can help support the 4+1 views identified in both sections 3.1 and 3.3); identifying this overlap can assist teams in determining where work can be reused.

Please note: Not all content of the frameworks needs to be used. Teams can start an MVA by using small parts of the framework for each of the following areas: logical, process, physical, and development. If elaboration is necessary, teams can add more components from the framework to the view as needed to ensure that the ideas are fully understood and the complexity is captured.

3.1 Logical View

3.1.1 OV-1: High Level Operational Concept Graphic

The following language is from the DoDAF viewpoints and models [DoD CIO 2010b].

> *The OV-1 [Figure] describes a mission, class of mission, or scenario. It shows the main operational concepts and interesting or unique aspects of operations. It describes the interactions between the subject architecture and its environment, and between the architecture and external systems. OV-1 is the pictorial representation of the written content of the AV-1 Overview and Summary Information. Graphics alone are not sufficient for capturing the necessary architectural data.*
>
> *The OV-1 provides a graphical depiction of what the architecture is about, and an idea of the players and operations involved. An OV-1 can be used to orient and focus detailed*

discussions. Its main use is to aid human communication, and it is intended for presentation to high-level decision-makers.

The intended usage of the OV-1 includes:

- *Putting an operational situation or scenario into context.*
- *Providing a tool for discussion and presentation; for example, aids industry engagement in acquisition.*
- *Providing an aggregate illustration of the details within the published high-level organization of more detailed information in published architectures.*

3.1.2 SV-1: Systems Interface Description

The following language is from the DoDAF viewpoints and models [DoD CIO 2010d].

SV-1 addresses the composition and interaction of Systems. For DoDAF V2.0, the SV-1 incorporates the human elements as types of Performers - Organizations and Personnel Types.

SV-1 links together the operational and systems architecture models by depicting how Resources are structured and interact to realize the logical architecture specified in an OV-2 Operational Resource Flow Description. An SV-1 may represent the realization of a requirement specified in an OV-2 Operational Resource Flow Description (i.e., in a "To-Be" architecture), and so there may be many alternative SV models that could realize the operational requirement. Alternatively, in an "As-Is" architecture, the OV-2 Operational Resource Flow Description may simply be a simplified, logical representation of the SV-1 to allow communication of key Resource Flows to non-technical stakeholders.

A System Resource Flow is a simplified representation of a pathway or network pattern, usually depicted graphically as a connector (i.e., a line with possible amplifying information). The SV-1 depicts all System Resource Flows between Systems that are of interest. Note that Resource Flows between Systems may be further specified in detail in SV-2 Systems Resource Flow Description and SV-6 Systems Resource Flow Matrix.

Sub-System assemblies may be identified in SV-1 to any level (i.e., depth) of decomposition the architect sees fit. SV-1 may also identify the Physical Assets (e.g., Platforms) at which Resources are deployed, and optionally overlay Operational Activities and Locations that utilize those Resources. In many cases, an operational activity and locations depicted in an OV-2 Operational Resource Flow Description model may well be the logical representation of the resource that is shown in SV-1.

The intended usage of the SV-1 includes:

- *Definition of System concepts.*
- *Definition of System options.*
- *System Resource Flow requirements capture.*
- *Capability integration planning.*
- *System integration management.*
- *Operational planning (capability and performer definition).*

SV-1 is used in two complementary ways:

- *Describe the Resource Flows exchanged between resources in the architecture.*
- *Describe a solution, or solution option, in terms of the components of capability and their physical integration on platforms and other facilities.*

3.1.3 SV-4: Systems Functionality Description

The following language is from the DoDAF viewpoints and models [DoD CIO 2010e].

The SV-4 addresses human and system functionality.

The primary purposes of SV-4 are to:

- *Develop a clear description of the necessary data flows that are input (consumed) by and output (produced) by each resource.*
- *Ensure that the functional connectivity is complete (i.e., that a resource's required inputs are all satisfied).*
- *Ensure that the functional decomposition reaches an appropriate level of detail.*

The Systems Functionality Description provides detailed information regarding the:

- *Allocation of functions to resources.*
- *Flow of resources between functions.*

The SV-4 is the Systems Viewpoint model counterpart to the OV-5b Activity Model of the Operational Viewpoint.

The intended usage of the SV-4 includes:

- *Description of task workflow.*
- *Identification of functional system requirements.*
- *Functional decomposition of systems.*
- *Relate human and system functions.*

3.2 Process View

3.2.1 SV-10b: Systems State Transition Description

The following language is from the DoDAF viewpoints and models [DoD CIO 2010f].

The SV-10b is a graphical method of describing a resource (or system function) response to various events by changing its state. The diagram basically represents the sets of events to which the resources in the Activities respond (by taking an action to move to a new state) as a function of its current state. Each transition specifies an event and an action.

The explicit time sequencing of service functions in response to external and internal events is not fully expressed in SV-4 Systems Functionality Description. The SV-10b can be used to describe the explicit sequencing of the functions. Alternatively, SV-10b can be used to reflect

explicit sequencing of the actions internal to a single function, or the sequencing of system functions with respect to a specific resource.

The intended usage of the SV-10b includes:

- *Definition of states, events and state transitions (behavioral modeling).*
- *Identification of constraints.*

3.3 Physical View

3.3.1 OV-1: High Level Operational Concept Graphic (See Above) & OV-5b: Operational Activity Model

The following language is from the DoDAF viewpoints and models [DoD CIO 2010c].

OV-5b describes the operational, business, and defense portion of the intelligence community activities associated with the Architectural Description, as well as the:

- *Relationships or dependencies among the activities.*
- *Resources exchanged between activities.*
- *External interchanges (from/to business activities that are outside the scope of the model).*

An operational activity (Tasks) is what work is required, specified independently of how it is carried out. To maintain this independence from implementation, logical activities and locations in OV-2 Operational Resource Flow Description are used to represent the structure which carries out the Operational Activities. Operational Activities are realized as System Functions (described in SV-4 Systems Functionality Description) or Service Functions (described in SvcV-4 Services Functionality Description) which are the how to the Operational Activities what, i.e., they are specified in terms of the resources that carry them out.

The intended usage of the OV-5a and OV-5b includes:

- *Description of activities and workflows*
- *Requirements capture*
- *Definition of roles and responsibilities*
- *Support task analysis to determine training needs*
- *Problem with space definition*
- *Operational planning*
- *Logistic support analysis*
- *Information flow analysis*

3.4 Development View

3.4.1 SvcV-1: Services Context Description

The following language is from the DoDAF viewpoints and models [DoD CIO 2010g].

> *The SvcV-1 addresses the composition and interaction of Services. For DoDAF V2.0, SvcV-1 incorporates human elements as types of Performers - Organizations and Personnel Types.*
>
> *The SvcV-1 links together the operational and services architecture models by depicting how resources are structured and interact to realize the logical architecture specified in an OV-2 Operational Resource Flow Description. A SvcV-1 may represent the realization of a requirement specified in an OV-2 Operational Resource Flow Description (i.e., in a "To-Be" Architectural Description), and so there may be many alternative SvcV models that could realize the operational requirement. Alternatively, in an "As-Is" Architectural Description, the OV-2 Operational Resource Flow Description may simply be a simplified, logical representation of the SvcV-1 to allow communication of key Resource Flows to non-technical stakeholders.*
>
> *It is important for the architect to recognize that the SvcV-1 focuses on Resource Flow and the providing service. This differs from a SV-1 System Interface Description which focuses on the System-to-System point-to-point interface, for which the Source System and Target System have an agreed upon interface. For the SvcV-1, the focus on the provider and the data provided is a Net-Centric Data Strategy tenet appropriate for a publishing/subscribe pattern. This pattern is not the only type of service that can be captured in the SvcV-1.*
>
> *Sub-services may be identified in SvcV-1 to any level (i.e., depth) of decomposition the architect sees fit. The SvcV-1 may also identify the Physical Assets (e.g., Platforms) at which Resources are deployed, and optionally overlay Operational Activities and Locations that utilize those Resources. In many cases, an operational activity and locations depicted in an OV-2 Operational Resource Flow Description may well be the logical representation of the resource that is shown in SvcV-1.*
>
> *The intended usage of the SvcV-1 includes:*
>
> - *Definition of service concepts.*
> - *Definition of service options.*
> - *Service Resource Flow requirements capture.*
> - *Capability integration planning.*
> - *Service integration management.*
> - *Operational planning (capability and performer definition).*
>
> *The SvcV-1 is used in two complementary ways:*
>
> - *Describe the Resource Flows exchanged between resources in the architecture.*
> - *Describe a solution, or solution option, in terms of the components of capability and their physical integration on platforms and other facilities.*

3.4.2 SvcV-4: Services Functionality Description

The following language is from the DoDAF viewpoints and models [DoD CIO 2010h].

The SvcV-4 DoDAF-described Model addresses human and service functionality.

The primary purpose of SvcV-4 is to:

- *Develop a clear description of the necessary data flows that are input (consumed) by and output (produced) by each resource.*
- *Ensure that the service functional connectivity is complete (i.e., that a resource's required inputs are all satisfied).*
- *Ensure that the functional decomposition reaches an appropriate level of detail.*

The Services Functionality Description provides detailed information regarding the:

- *Allocation of service functions to resources.*
- *Flow of resources between service functions.*

The SvcV-4 is the Services Viewpoint counterpart to the OV-5b Operational Activity Model of the Operational Viewpoint.

The intended usage of the SvcV-4 includes:

- *Description of task workflow.*
- *Identification of functional service requirements.*
- *Functional decomposition of Services.*
- *Relate human and service functions.*

It is important to note that one usage of the SvcV-4 can support a net-centric (service-oriented) implementation in describing the producing services and consuming services. The Services Functionality Description information can support the registration of services in net-centric (service-oriented) implementation.

Appendix C: DoDAF and UAF Models

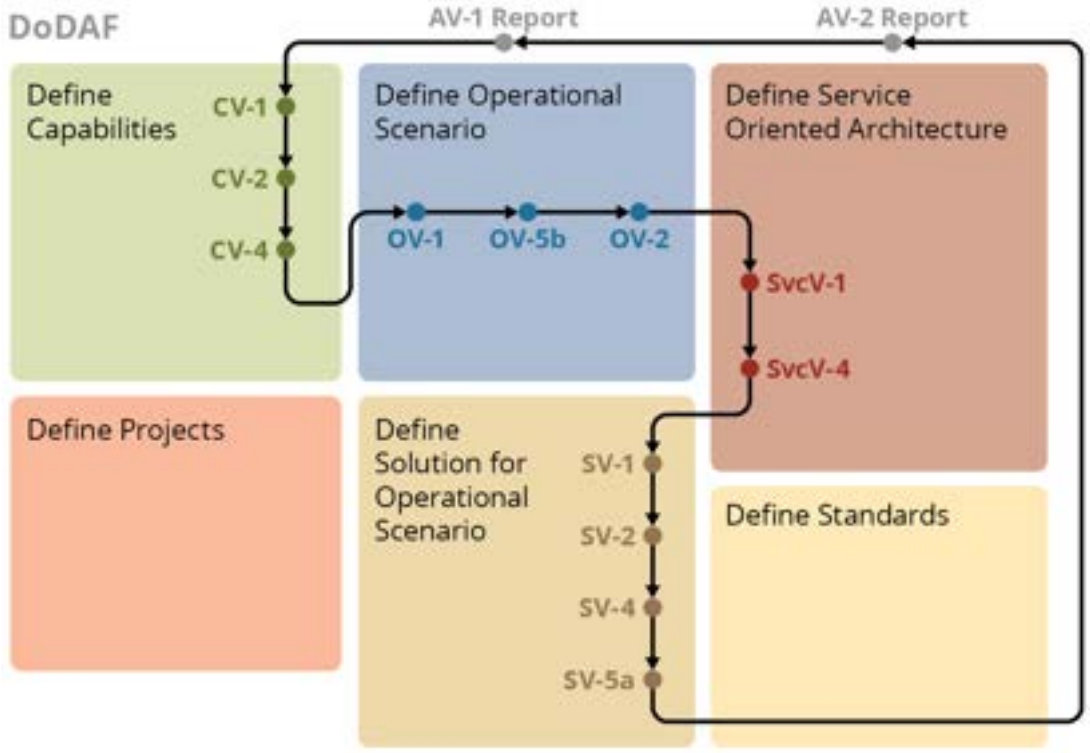
This appendix explores what the proposed MVA development process can look like when compared to the process for developing an extensive architecture using the complete DoDAF [DoD CIO 2010a], UAF [UAF 2025], and 4+1 [Kruchten 1995] models. Overall, the concept of using UML is critical, and the specific modeling language is just the approach taken for a UML. (See Table 3 for the simple relationships between 4+1 and a typical UML.) Starting development with an MVA is the right decision because extensive architecture is only necessary if the complexity that must be represented exposes risks that must be fully understood. For many developments, this full model architecture requires too many resources.

Table 3: Relationship of UML to 4+1 Views

UML Architectural Views	4 + 1 Architectural Views
Use Case View	Scenarios
Design View	Logical View
Interaction View	Process View
Implementation View	Development View
Deployment View	Physical View

This table, adapted from the original by JSWare.io., shows the relationship of UML to 4+1 views [JSWare 2025].

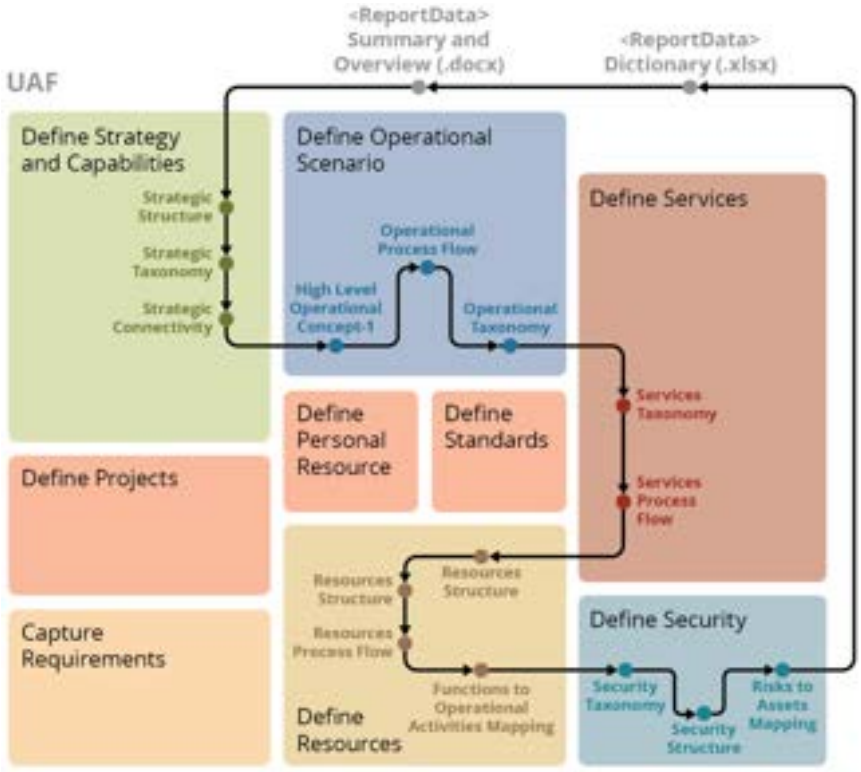
Therefore, while using industry standard modeling for architecture representation is generally recommended, the SEI recommends adapting these standard models to meet an organization's specific needs when developing an MVA. The following diagrams represent how an organization can adapt the complete models to address its needs. Each diagram has black lines and highlights to denote the minimum path that a development team can follow through the UML modeling framework to develop an MVA.



This diagram was adapted from the original, which was created by No Magic [No Magic 2025]. Black lines represent the minimum set of DoDAF areas that need to be included in the MVA to satisfy Krutchen 4+1 views [Kruchten 1995].

Figure 5: Minimum DoDAF Areas That Must Be Included in the MVA to Satisfy Krutchen 4+1 Views

The dark line path in Figure 5 shows the elements from the DoDAF model that can be used to instantiate an MVA. It also indicates that first passes can be further elaborated in the different areas of the model if needed to better understand the complexity of the proposed solution. However, the team does not have to engage in a heavily model-driven documentation practice to get started, they can produce a light MVA for system property analysis, as discussed in this technical note.

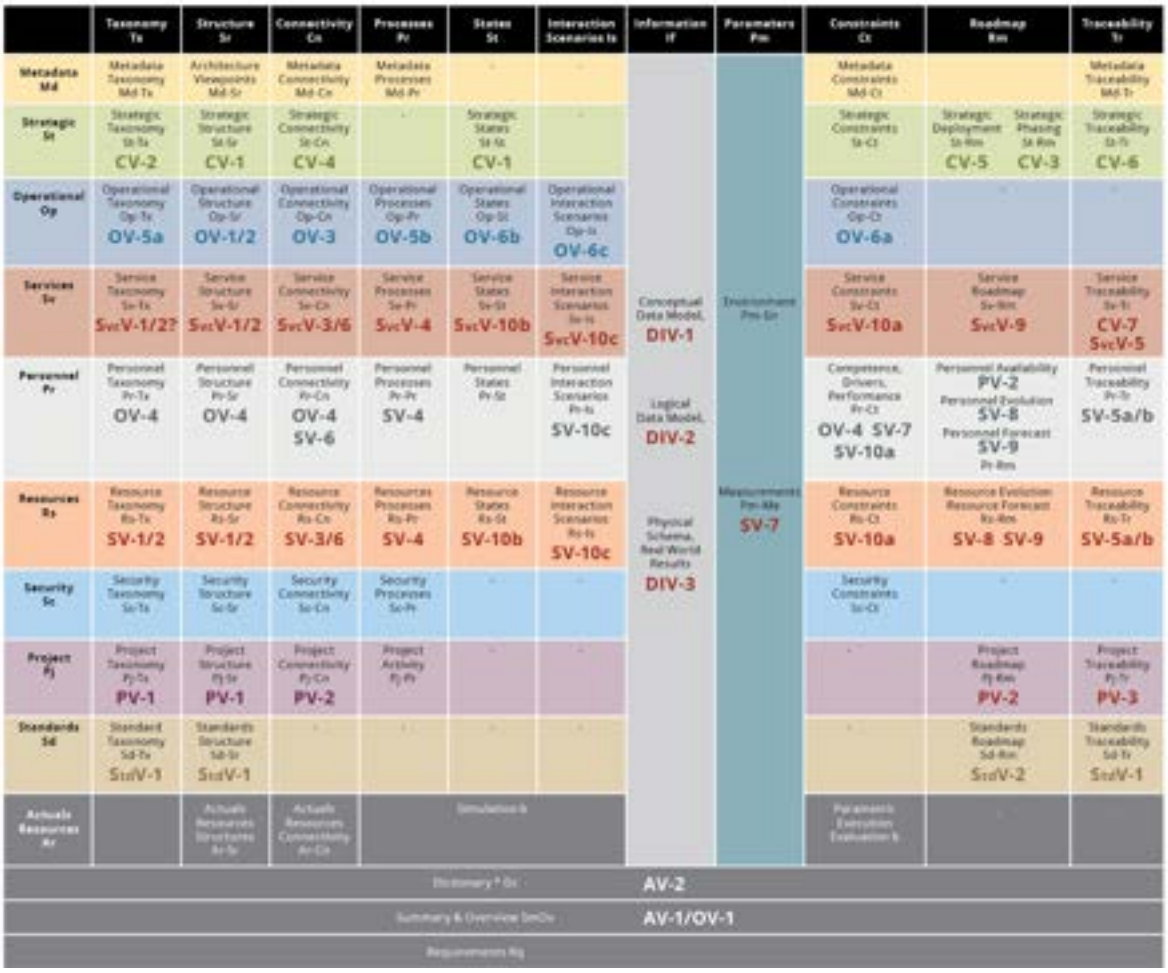


The black line in this diagram represents the minimum items that would be needed for an MVA to include the full "human" view of the DoDAF [DoD CIO 2010a] and UAF modeling frameworks [UAF 2025]. This diagram was adapted from the original that was created using No Magic [No Magic 2025].

Figure 6: Minimum Items Needed for an MVA to Include the Full "Human" View of the DoDAF

The dark line in Figure 6 shows the elements from the UAF model that can be used to instantiate an MVA. It also indicates that first passes can be further elaborated in the different areas of the model if needed to better understand the complexity of the proposed solution. However, the team does not have to engage in a heavily model-driven documentation practice to get started; they can produce a light MVA for system property analysis, as discussed in this technical note. This modeling has the added advantage of explicitly identifying security modeling that can help with cybersecurity assurance validation of the solution.

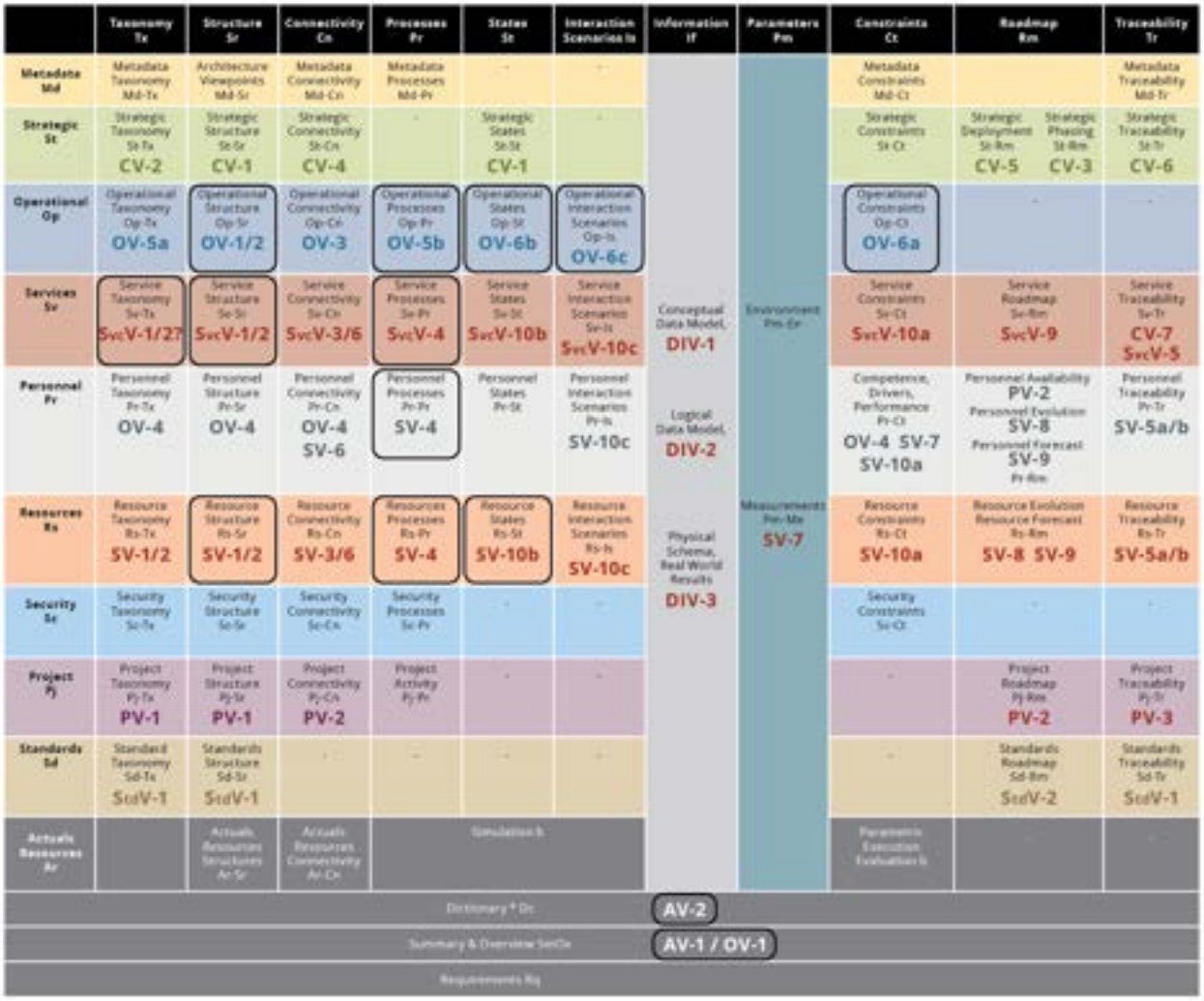
Table 4: How the DoDAF and UAF Frameworks Map to One Another



This table, which was adapted from the original by Steven Dam [Dam 2018], shows how the DoDAF and UAF frameworks map to one another. The abbreviations in bold font are the elements from the DoDAF superimposed on the UAF table of elements.

Table 4 and Table 5 show that modeling language with DoDAF and UAF can be used to generate "simple" models in support of the 4+1 framework and the MVA. Many times, these models are viewed as overwhelming and a huge "up-front cost." By using an MVA, the development team can conduct analysis that helps the project immediately while also avoiding excessive costs and effort later [Pearson 2014].

Table 5: Elements Normally Needed for an MVA



This table, which is the same as Table 4, outlines areas that highlight the elements normally needed for an MVA.

References

URLs are valid as of the publication date of this report.

[Baragry 2001]
Baragry, Jason & Reed, Karl. Why We Need a Different View of Software Architecture. Pages 125-134. In *Proceedings Working IEEE/IFIP Conference on Software Architecture, Amsterdam, Netherlands.* August 2001. DOI: 10.1109/WICSA.2001.948419. https://ieeexplore.ieee.org/document/948419

[Bittner 2022]
Bittner, Kurt & Pureur, Pierre. A Minimum Viable Product Needs a Minimum Viable Architecture. *InfoQ.com*. June 8, 2022. https://www.infoq.com/articles/minimum-viable-architecture/

[Bizzdesign 2025]
Enterprise architecture frameworks. *Bizzdesign*. June 2, 2025 [accessed]. https://bizzdesign.com/wiki/eam/enterprise-architecture-frameworks/

[Botero 2021]
Botero, Juan Felipe & Leal, Alexander. An Architecture for Power Substations Communication Networks Based on SDN and Virtualization Paradigms. *Revista Facultad de Ingeniería Universidad de Antioquia*. March 2021. 10.17533/udea.redin.20210321

[Capgemini 2020]
Agile and Architecture – Lessons from the Trenches [blog post]. *Capgemini*. April 28, 2020. https://www.capgemini.com/insights/expert-perspectives/agile-and-architecture-lessons-from-the-trenches/

[Conexiam 2025]
The Three Types of Enterprise Architecture Frameworks [blog post]. *Conexiam*. June 2, 2025 [accessed]. https://conexiam.com/the-three-types-of-enterprise-architecture-framework/#:~:text=The%20TOGAF%C2%AE%20Standard.%20The%20TOGAF%20Standard%20is,fundamental%20concepts%20and%20a%20set%20of%20guides

[Dam 2018]
Dam, Steven H. Ontology for Lifecycle Development and Management: Implementing DoDAF and UAF. *SPEC Innovations*. October 25, 2018. https://ndia.dtic.mil/wp-content/uploads/2018/systems/Thurs_21419_Dam.pdf

[DoD 2020]
Department of Defense (DoD). *Operation of the Software Acquisition Pathway*. DoDI 5000.87. DoD. October 2, 2020. https://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodi/500087p.PDF

[DoD CIO 2012]
Department of Defense (DoD) Chief Information Officer (CIO). *DoD Information Enterprise Architecture v2.0*. January 2012. https://www.slideserve.com/genesis/dod-information-enterprise-architecture-v2-0

[DoD CIO 2010a]
Department of Defense (DoD) Chief Information Officer (CIO). *The DoDAF Architecture Framework Version 2.02*. August 2010. https://dodcio.defense.gov/Library/DoD-Architecture-Framework/

[DoD CIO 2010b]
Department of Defense (DoD) Chief Information Officer (CIO). *DoDAF Viewpoints and Models: Operational Viewpoint: OV-1: High Level Operational Concept Graphic*. August 2010. https://dodcio.defense.gov/Library/DoD-Architecture-Framework/dodaf20_ov1/#:~:text=The%20OV%2D1%20provides%20a,high%2Dlevel%20decision%2Dmakers

[DoD CIO 2010c]
Department of Defense (DoD) Chief Information Officer (CIO). *DoDAF Viewpoints and Models: Operational Viewpoint - OV-5a: Operational Activity Decomposition Tree and OV-5b: Operational Activity Model*. August 2010. https://dodcio.defense.gov/Library/DoD-Architecture-Framework/dodaf20_ov5ab/

[DoD CIO 2010d]
Department of Defense (DoD) Chief Information Officer (CIO). *DoDAF Viewpoints and Models: Systems Viewpoint - SV-1: Systems Interface Description*. August 2010. https://dodcio.defense.gov/Library/DoD-Architecture-Framework/DoDAF20_sv1/

[DoD CIO 2010e]
Department of Defense (DoD) Chief Information Officer (CIO). *DoDAF Viewpoints and Models: Systems Viewpoint - SV-4: Systems Functionality Description*. August 2010. https://dodcio.defense.gov/Library/DoD-Architecture-Framework/dodaf20_sv4/

[DoD CIO 2010f]
Department of Defense (DoD) Chief Information Officer (CIO). *DoDAF Viewpoints and Models: Systems Viewpoint - SV-10b: Systems State Transition Description*. August 2010. https://dodcio.defense.gov/Library/DoD-Architecture-Framework/dodaf20_sv10b/

[DoD CIO 2010g]
Department of Defense (DoD) Chief Information Officer (CIO). *DoDAF Viewpoints and Models: Services Viewpoint - SvcV-1: Services Context Description*. August 2010. https://dodcio.defense.gov/Library/DoD-Architecture-Framework/dodaf20_services1/

[DoD CIO 2010h]
Department of Defense (DoD) Chief Information Officer (CIO). *DoDAF Viewpoints and Models: Services Viewpoint - SvcV-4: Services Functionality Description*. August 2010. https://dodcio.defense.gov/Library/DoD-Architecture-Framework/dodaf20_services4/

[GeeksforGeeks 2024]
Manifesto for Agile Software Development. *GeeksforGeeks, Sanchhaya Education Private Limited*. August 2, 2024. https://www.geeksforgeeks.org/software-engineering/agile-manifesto-for-software-development/

[JSWare 2025]
Architectural Views [blog post]. *JSWare.io*. July 7, 2025 [accessed]. https://www.jsware.io/uml2/architectural-views.html

[Kruchten 2019]
Kruchten, Philippe et al. *Managing Technical Debt: Reducing Friction in Software Development*. Addison-Wesley Professional. 2019. ISBN: 978-0-13-564593-2. https://insights.sei.cmu.edu/library/managing-technical-debt-reducing-friction-in-software-development/

[Kruchten 1995]
Kruchten, P. B. The 4+1 View Model of Architecture. *IEEE Software.* Volume 12. Number 6. November 1995. Pages 42-50. 10.1109/52.469759

[SAP 2025]
SAP. *LeanIX GmbH*. June 2, 2025 [accessed]. https://www.leanix.net/en/wiki/ea/enterprise-architecture#:~:text=ArchiMate%20is%20a%20graphical%20modeling%20language%20and,visualize%20the%20relationships%20between%20various%20architectural%20domains

[No Magic 2025]
Welcome to No Magic Documentation. *No Magic, Inc*. June 2, 2025 [accessed]. https://docs.nomagic.com/site/nm

[Pearson 2014]
Pearson, LaTonya. The Rising Costs of Defects [blog post]. *Segue Technologies Inc.* March 31, 2014. https://www.seguetech.com/rising-costs-defects/

[Protecto 2021]
What Is Minimum Viable Architecture? Why Is It Important? [blog post]. *Protecto*. May 9, 2021. https://www.protecto.ai/blog/what-is-minimum-viable-architecture-why-is-it-important#:~:text=Simply%20put%2C%20Minimum%20Viable%20Architecture,good%20enough%20to%20be%20released

[Scaled Agile 2024]
Extended SAFe Guidance. *Scaled Agile, Inc*. May 2, 2024. https://framework.scaledagile.com/advanced-topics/extended-safe-guidance/

[Sheppard 2024]
Sheppard, Neil. Comparison Of Top 5 Enterprise Architecture Frameworks. *LeanIX GmbH*. September 2, 2024. https://www.leanix.net/en/blog/5-enterprise-architecture-frameworks#:~:text=5%20SAP%20Enterprise%20Architecture%20Framework.%20The%20SAP,methodology%20and%20add%20features%20from%20other%20frameworks

[Soto 2023]
Soto, Manuel. Enterprise Architecture Frameworks. *LinkedIn Corporation*. November 24, 2023. https://www.linkedin.com/pulse/enterprise-architecture-frameworks-manuel-soto-xcwwc/

[Tharpe 2020]
Tharpe, Bunny. Types of Enterprise Architecture Frameworks: ArchiMate, TOGAF, DoDAF and more [blog post]. *Quest Software ALL*. March 19, 2020. https://blog.erwin.com/blog/enterprise-architecture-frameworks/#:~:text=FEAF.%20A%20Federal%20Enterprise%20Architecture%20Framework%20issued,the%20US%20Office%20of%20Management%20and%20Budget

[UAF 2025]
Unified Architecture Framework (UAF). *Object Management Group*. March 2025. https://www.omg.org/spec/UAF/

[Visual Paradigm 2025]
What is Zachman Framework? [blog post]. *Visual Paradigm*. September 17, 2025 [accessed]. https://www.visual-paradigm.com/guide/enterprise-architecture/what-is-zachman-framework/

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	September 2025	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Minimally Viable Architecture: Architecture Early in Development	FA870225DB003

6. AUTHOR(S)

Manuel A Rosso-Llopart

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2025-TN-001

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
SEI Administrative Agent AFLCMC/AZS 5 Eglin Street Hanscom AFB, MA 01731-2100	n/a

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

This technical note explores the concept of minimally viable architecture (MVA), which is architecture that can be used by Department of Defense development teams as they build products. After an MVA is developed, the unified modeling language frameworks can be used to enhance the MVA throughout the life of the project as greater detail and understanding are needed to maintain the effort and ensure the quality attributes (also known as systemic properties or non-functional requirements) are met by the solution.

Overall, the SEI recommends that DoD teams begin to develop their architecture by using an MVA. This technical note offers guidance on what these teams should do with an MVA to get their certificate to field and authorization to operate.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Minimally Viable Architecture (MVA), Minimal Viable Product (MVP), Agile, 4+1 model	34

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. Z39-18 298-102