2022 NDIA MICHIGAN CHAPTER
GROUND VEHICLE SYSTEMS ENGINEERING
AND TECHNOLOGY SYMPOSIUM
DIGITAL ENGINEERING / SYSTEMS ENGINEERING TECHNICAL SESSION
AUGUST 16-18, 2022 - NOVI, MICHIGAN

GROUND VEHICLE SYSTEM REFERENCE ARCHITECTURE MODEL (DIGITAL THREAD)

Kevin W. Griffin, Giuseppe D. Suffredini, Robert J. Kanon, Surender K. Dua, Jihsiang J. Yeh, Eric J. Alexander, Mark R. Feury, Russell D. Kouba

SAIC, GVSC
Sterling Heights, MI, Warren, MI

ABSTRACT

Digital Engineering (DE) strategy is defined by the Department of Defense and establishes five goals [1]. *One of the goals includes providing an enduring, authoritative source of truth, which moves the primary means of communication from documents to digital models and data. This enables access, management, analysis, use, and distribution of information from a common set of digital models and data. As a result, stakeholders have the current, authoritative, and consistent information for use over the lifecycle. The DE Model Based Systems Engineering (MBSE) Reference Architecture Framework (RAF) defines, at a minimum, the digital model authoritative source of truth, model structure, stakeholder needs, systems and subsystem context, process model elements, architecture types, views, viewpoints, and supporting methodologies and best practices. This framework is defined using the Systems Modeling Language, semantics, and constructs. The RAF structure is expressed to support DE transformation and help improve MBSE best practices.*

1. INTRODUCTION

The engineering community is transitioning to Digital Engineering (DE) processes, methods, and practices to achieve better quality products and reduce product development costs and risk through connecting previously disjointed artifacts and modernizing traditional systems engineering (SE). The DE transformation is an incremental and evolutionary process requiring many elements, such as engineering methods, practices, information technology, network, computing, modeling, simulation, data management, training, and much more.

Some important aspects of the DE ecosystem are defined as follows: 1) Digital Thread - the flow of information about a product's performance and use from design to production, sale, use and disposal or recycling. 2) Digital Twin - a digital replica or a representation of a physical object (e.g. aircraft engine, person, vehicle) or an intangible system (e.g. marketing funnel, fulfillment process) that can be examined, altered and tested without interacting with it in the real world and avoiding negative consequences. 3) Physical Twin - The physical twin is a relative term that describes the real-world physical system-of-systems or platform vehicle products associated with design, software, hardware, component elements. The current physical assets which contain cables, connectors, test equipment, hardware, application software, embedded real-time software, power, electronics, etc. represent the physical twin.

Traditional systems engineering (SE) processes have relied on document-based artifacts to capture much of the system specification and design information. Spreading this information across many different documents results in lost precision, inconsistent data, and difficulties in reuse and maintainability. The following approach applies DE and Model-Based Systems Engineering (MBSE) best practices providing the following benefits:

- Accelerates and ensures a more complete, consistent, and traceable design
- Identifies and mitigates risk early
- Promotes safety and hazard analysis studies
- Improves quality and mission assurance
- Improves management of program resources
- Maintains configuration control throughout the lifecycle
- Reuses existing engineering methods and practices

This approach was applied recently to the GVSC Virtual Prototyping (VP) process in order to define and standardize core VP functions across the key modeling and simulation (M&S) activities. Specific VP functions include requirements capture (RC), concept development (CD), performance modeling (PM), virtual experimentation, virtual mock-ups, and System Integration Labs (SILS). For background, VP is a virtual product development process by which advanced M&S enables Warfighters up-front, virtual operational evaluation of Next Generation ground vehicle concepts and emerging technologies. Example VP functions for requirements capture, concept development, and performance modeling are included below to highlight this ongoing activity with GVSC.

2. REFERENCE ARCHITECTURE MODEL DESCRIPTION

A Reference Architecture Framework (RAF) model structure is a key element of the DE model authoritative source of truth and a core attribute to support DE transformation. The following sections outline the key elements, which define the RAF to support SE and the ground vehicle modeling structure including frame of reference. This RAF establishes a common DE Model Based Systems Engineering (MBSE) vocabulary, ontology, processes, methodologies, and best practices to facilitate systems engineering and product lifecycle development. It also establishes and supports integration with open system frameworks, industry and military standards, systems safety, and DE ecosystem. The framework can be deployed across programs to quickly start the DE MBSE modeling effort and can be tailored or extended to meet specific program requirements.

3. PROCESS VIEW

The RAF contains Process Views to define both engineering and MBSE processes, methodologies, work instructions, guidelines, and model structure content. Without defined processes, a complex model developed by multiple users would lack the consistency needed to provide enduring utility. Process views (Figure 1) provide the specific details and guidance necessary for developing an enduring model. The model,

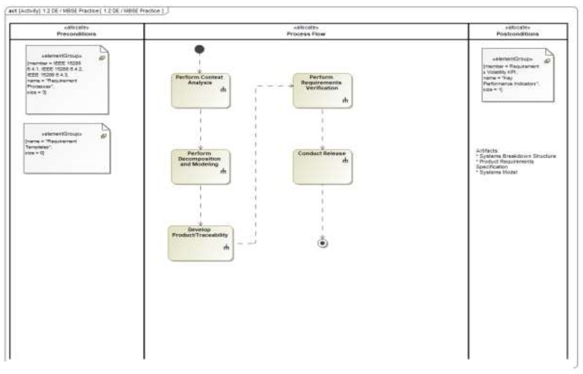

Preconditions
Process Flow
Postconditions
Perform Context Analysis
Perform Decomposition and Modeling
Develop Product/Traceability
Perform Requirements Verification
Conduct Release
Artifacts
* Systems Breakdown Structure
* Product Requirements Specification
* Systems Model

Figure 1 – Process View model

which represents the authoritative source of truth, captures the current state and developmental history of the technical baseline. Such a model allows for traceability up and down the system life cycle, and captures knowledge as the system evolves over time.

The RAF process views implemented in the DE model extend the authoritative source of truth to include industry and organizational system and software engineering best practices to support product life cycle development. It includes reference information, artifacts, and key performance measures to perform and assess the quality of the engineering and modeling practices. In addition, the process views allow for the communication and deliberation of best practices, and documents them for continued use, assessment, and refinement.

The process views (e.g. Process Model) represents various industry engineering processes, such as IEEE 15288 Systems and Software Engineering – System Life Cycle Processes [4], IEEE 12207 Systems and Software Engineering – Software Life Cycle Processes [5], process life cycle stages, and specific organizational business models, which aid in the engineering and modeling practices. Figure 2 depicts the RAF process views.

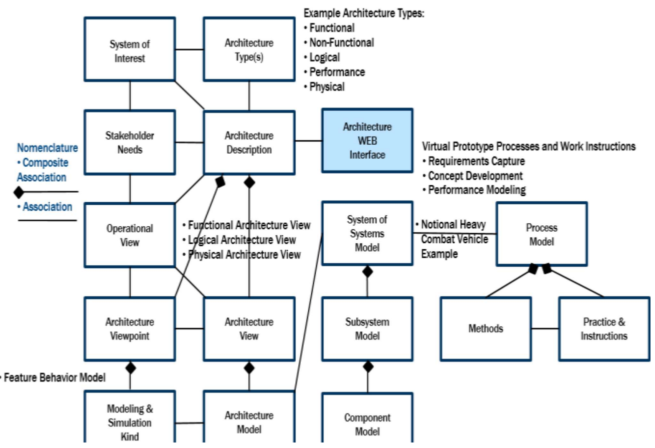


Figure 2 – Reference Architecture Framework (RAF) views

4. OPERATIONAL VIEW

The models in Operational Viewpoint describe the tasks and activities, operational elements, and resource flow exchanges required to conduct operations. The operational view models are of various types, depending upon the operational view it depicts.

Some of the important operational view models are described below:

- The top-level operational view model describes a mission, class of mission, or scenario. It shows the main operational concepts and also interesting or unique aspects of operations.
- The operational resource flow view model shows the resource flow exchanges between operational activities.

- The operational activity view model shows operational activities and their relationships among activities, inputs and outputs.
- The operational rules view model describes operational activity and identifies business rules that constrain operations.

The development of new technologies impacts the operations and their relationship with other activities. Further the improvement in processes also impacts how the operations are conducted. In such cases, operational models may have materiel constraints and requirements that need to be addressed. Therefore, it may be necessary to include some high-level system architectural data to augment information onto the operational models.

Operational view models are useful for development of user requirements, capturing future concepts, and supporting operational planning processes. The model may also be useful for boundary definition, which requires some degree of stakeholder engagement; identification of the boundaries of capabilities, thus rendering the functional scope and organizational span.

5. STAKEHOLDER NEEDS VIEW

The stakeholders are individual, groups or organizations which support the system product lifecycle from inception to deployment. The stakeholders in complex systems context are warfighters, maintainers, test personnel, service providers, trainers, external organizations, external systems, and other sources identified to support the systems.

Stakeholder needs are the primary inputs for the development of the complex systems. The initial stakeholder concerns do not serve as stakeholder requirements, since they often lack definition, analysis, consistency, and feasibility. The stakeholder needs are refined to obtain a better understanding of the system perspective. Stakeholder inputs lead to the following requirement types to express system context and capabilities from an operational perspective:

1. Capability requirements
2. Context requirements
3. Operational mode requirements
4. Operational requirements
5. Operational scenario requirements, and
6. Validation requirements

The Stakeholder needs view (see Figure 3) is represented by the “Use Case” diagrams in MBSE Systems Modeling Language (SysML). These diagrams depict the interactions between the various stakeholders and the system context. In use case diagrams an actor is depicted as a human figure, which defines the role played by the stakeholder. An actor models a type of role played by an entity that interacts with the system (e.g., exchanging signals and data). Actors may represent roles played by human users, external hardware, or other subjects, which have direct interaction with the system context. If needed, the use cases are decomposed into supporting use cases to depict these interactions. In the system’s context, actors represent stakeholders. The use cases depict the relationships, associations, and interactions between various actors, system boundaries and the supporting use cases.

The requirements are classified into two types described as below:

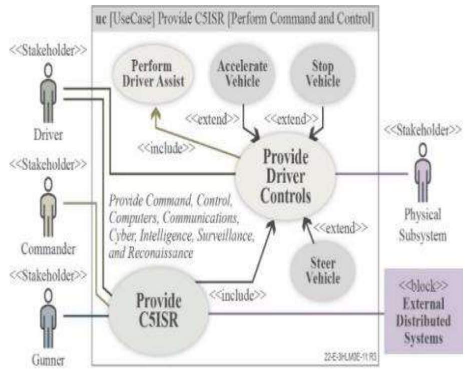


Figure 3 – Stakeholder Need view

a) Functional Requirements describe the functions or services provided by the system and lists specific conditions and actions to be accomplished, which contribute to the goals or objectives, an action, task, or activity to be performed to achieve a desired outcome.

b) Non-functional Requirements define the criteria for the operation of a system and includes the quality attributes of the system; includes quality goals, quality of service, constraints, non-behavior requirements, and technical requirements. The types of quality attributes include scalability, reliability, availability, regulatory, recoverability, capacity, maintainability, serviceability, security, environmental, and data integrity.

Table 1 below defines the inputs and analysis tools used to support the VP requirements capture methods and practices. The requirement analysis supports the initial set of operational and stakeholder requirement artifacts which is the basis of the digital thread.

VP Process (RC) Input	Tools
Singular functional requirements	DOORs, MS-Excel
Singular non-functional requirements	
Requirements traceability allocations	

`

Table 1 – Requirements Capture process and tools

6. SYSTEM VIEW

The System View model shows the system's functionality, boundary and its relationships with the surrounding systems, resource flows across the systems and other details. Some of the important system view models are described below:

- Systems Interface Description identifies the systems, subsystem, components and their interconnections.
- Systems Resource Flow Description provides a description of resource flows exchanged between systems.
- Systems Functionality Description shows the functions (activities) performed by systems and the system data flows among system functions.
- Systems Resource Flow Matrix provides the details of system resource flow elements being exchanged between systems and the attributes of that exchange in a matrix format.

The System View can be decomposed (see Figure 4) to Subsystem View by breaking down the system into lower level subsystems, if needed.

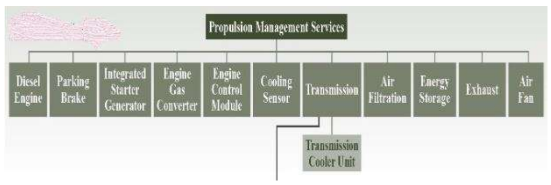


Figure 4 – Functional Decomposition

The different System View models can also be applied to Subsystem View models, where needed.

Table 2 below defines the typical VP Concept Development process and tools. The result of this analysis work will verify the functional requirements and decomposition model artifacts.

VP Process (CD) Identifiers	Tools
Science/Technology Contributions	Tech Surveys, DOORS, MagicDraw Photoshop, Rhino 3D, Blender 3D, Creo, COTS, NX, Solidworks, Pugh Analysis SpaceClaim
Architecture Refinement & Translation	
Industrial Design Ideation Iterations	
3D CAD Solid Model	
Space, Weigh, and Power Assessment (SWaP)	
System Level Requirements Trades	
Work Breakdown Structure (PWBS)	

Table 2 – Concept Development process and tools

7. ARCHITECTURE TYPES

Architecture types define functional, logical, and physical model abstractions of systems according to the modeling guide including libraries for stereotypes, interfaces, signals, and item flows.

Functional architecture (see Figure 5) defines a solution-independent representation of the system design to identify system functions that will be allocated to various system components. The system is decomposed into subsystems (e.g., a further decomposed system within the larger system context) with supporting black box functionality including behavior and performance. Subsystems establish boundaries, interfaces and behavior to support the mission and operational performance as described by the stakeholder requirements, use cases, and include sub element parts (e.g., hardware, software, and equipment) and may represent additional system entities (e.g., Commercial off-the-Shelf [COTS], Government off-the-Shelf [GOTS], and other purchased items as defined in the system product structure). System functions identify the services to be performed by the system as actions. SysML activity diagram (act), state machine diagram (stm), and sequence diagram (sd) elements are used to model functional decomposition. Further development of the functional allocation, decomposition, and derivation occurs where the system is described in the form of activities or actions, which include the following: triggered event, event condition, actor, action, object, output event, and intended behavior. The system context is modeled with operating scenarios (e.g., sunny day and rainy day), subsystems, blocks, black box structure, and activities. Functions including performance constraints are derived and allocated in the functional architecture. The MBSE SysML provide value types for datasets (e.g., interfaces, signals, item flows, and enumerated system elements).

When developing the Functional Architecture, the Performance Modeling

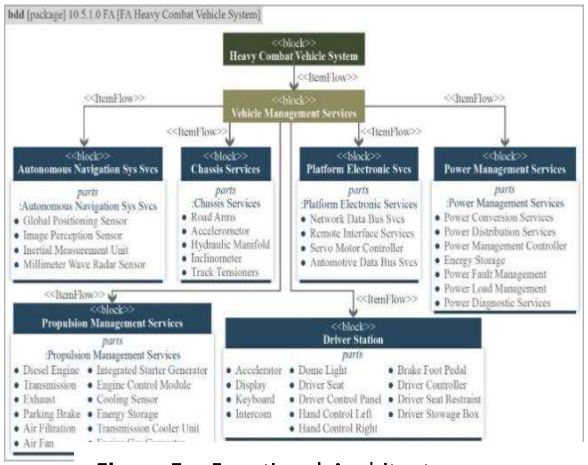


Figure 5 – Functional Architecture

function will assess and define the concept design. Mobility and powertrain performance models are two example analyses. Table 3 below describes the process steps and supporting tools used.

PM Mobility and Powertrain Performance Processes	Tools
Data Collection and Verification	MAT (or NRMM/VehDyn),
MAT/NRMM Mode development & simulation	

Output data processing and simulation	Photoshop, Rhino 3D, Blender 3D, Matlab, Simulink, GT-Suite, MS Powerpoint, MS Excel MS Outlook,
Propulsion System Model development and Simulation	
Prepare customer briefing	
Present results	

Table 3 – Performance Modeling – Mobility and Powertrain process and tools

The logical architecture (see Figure 6) represents an intermediate abstraction and encapsulation of system elements between functional and physical architectures. Logical entities are developed and modeled to represent systems components and supporting behavior for hardware, software, services, and parts. Typically, the logical architecture is developed to support

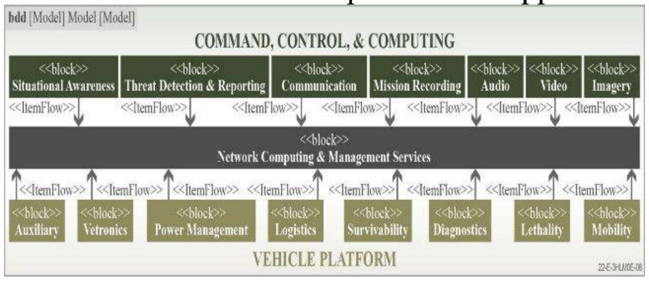


Figure 6 – Logical Architecture

allocation of system requirements to software and hardware elements. It is also widely used to define the data bus communication, computing, and network logical elements. For instance, a data-centric model representing the logical architecture is developed to define the data packets, which are transmitted and received on the asynchronous automotive Controller Area Network (CAN) and Ethernet buses. The MBSE SysML modeling tools provides the ability to model the data bus communication between various system components. This offers significant benefits because the data bus model can be easily transferred to software units and executed on computing hardware including real-time embedded hardware. This reduces the effort and risk to develop data communication using hardware equipment versus simulation. Logical data models are developed using the MBSE SysML modeling tool and then transferred to the Unified Modeling Language (UML) software modeling tool environment. The logical architecture models support interchangeability between the systems and software engineering domains. The SysML tools provide a seamless integration between SysML and UML models. The logical architecture is developed to support software code generation, testing, and deployment to physical computing hardware to enhance the workflow and product development between systems and software engineering methods and practices. The physical architecture (see Figure 7) represents the collection of items (e.g., equipment, parts, services, and interfaces) to support system design and implementation. Depending on the availability of the information, the physical architecture is developed by using design drawings, data sheets, product specifications, interface control documents, and any source of information useful in developing the system signals and data interfaces. The physical architecture also accounts for the systems constraints such as; performance, throughput, redundancy, failure modes and effects, reliability, hazards, and cybersecurity related to the stakeholder needs including services. Physical systems and components are implemented in the DE and MBSE SysML as blocks with <<physical>> stereotype nomenclature. The product structure as defined to support the systems hierarchy and acquisition strategy identifies and includes COTs, GOTs, and Modified off-the-Shelf (MOTs), internally developed items, and other supporting products obtained via the supply chain. The physical architecture derived viewpoints typically include supporting signal flows to support ground vehicle system interfaces (e.g., power, data, signal, network, communication, command, control, and thermal).

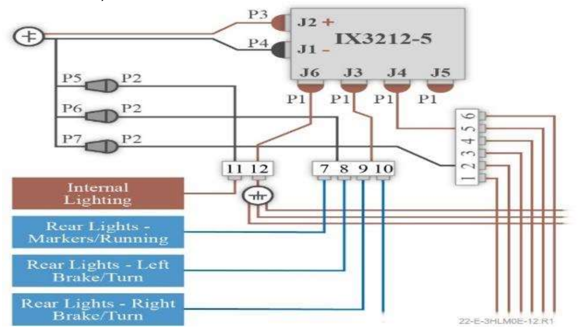


Figure 7 – Physical Architecture

The RAF includes product structure which provides a mapping for the physical architecture items and system design elements including the hierarchy of the system, subsystem, components (e.g., hardware, software). The DE and MBSE SysML dataset model including data dictionary provides the definition of the specific signal interfaces, stereotypes, item flows, and supporting data.

8. ARCHITECTURE VIEWPOINT

A view is a representation of one or more structural aspects of an architecture that illustrates how that architecture addresses one or more concerns held by one or more of its stakeholders. There may be a one to many relationship between the architecture type and architecture viewpoint. Each viewpoint can address functional analysis, safety/hazard analysis, and enablement for modular-open architecture definition.

Specific architectural viewpoints that encompass SAE J3094 Surface Vehicle Recommended Practice, define the architectural structure of a ground vehicle system model by partitioning it into subsystem models and by defining subsystem interfaces required to enable plug-and-play operation of simulation models. Applying this recommended standard to the architecture framework will allow for the enablement of seamless plug-and-play reusability of models for simulating the functional behavior of a ground vehicle system and its subsystems.

The requirements of applying critical safety and hazard analysis to electric and/or electronic systems is a key directive in today's complex systems. Applying the functional architecture in support of both ISO 26262 Road vehicle [2] and MIL-STD-882E System Safety [3] analysis enables the ability to eliminate hazards, where possible, and minimize risks where those hazards cannot be eliminated. Adherence to this standard ensures that sufficient levels of safety are being met and maintained throughout the vehicle lifecycle.

Lastly, the reference architecture framework approach also include open modular-open architecture definition. Specifically adhering to the Department of Defense Architecture Framework (DoDAF) and viewpoints, as well as, applying modular operating system architecture viewpoints such as Modular Open Systems Approach (MOSA), Vehicular Integration for C4ISR/EW Interoperability (VICTORY), and Future Airborne Capability Environment (FACE). This approach allows for a reference architecture framework that supports modular design and will allow system level components to be added, removed or replaced throughout the life cycle of a major system platform. This will then allow the system to afford opportunities for enhanced innovation and competition.

9. ARCHITECTURE MODEL

The foundation for the architecture model is a functional architecture view. This view is provided by the RAF and permits the system engineer to focus on the functional behavior of the features of the system under design. The feature model ensures identified use cases and stakeholder needs are developed and analyzed early in the system life cycle to ensure a strong and clear understanding of the expected system behavior. A feature's functional behavior is designed and executed to verify that performance constraints are met, failure modes and effects are analyzed, and confirmation of necessary feature interfaces.

The RAF functional architecture view provides the safety team a SysML standard model-based canvas to develop a safety strategy early in the system life cycle (see Figure 8), and to apply a model-based safety approach as well. Using a hazard ontology, the safety analysis occurs to identify and capture potential hazards and risk assessments directly in the model. Item definition is the first phase of the safety analysis, including development of a description of the item with regard to functionality, interfaces, environmental conditions, and hazards. Next, a hazard analysis and risk assessment take place which determine the Automotive Safety Integrity Level (ASIL) for each hazard and get assigned to safety goals. From this phase, detailed safety requirements get derived from the safety goals as well as a functional safety concept to mitigate program risks. This model-based safety analysis ensures consistency between the system design and safety artifacts, enable continuous improvement of the structure and behavior of the system being designed, and prevents late detection of errors in the program lifecycle.

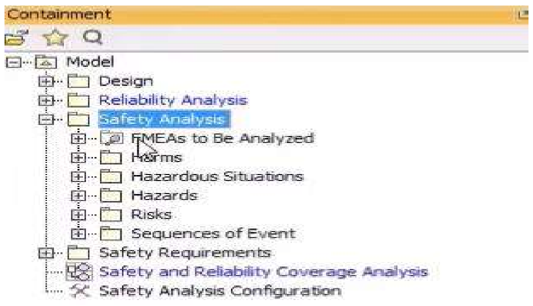


Figure 8 – SysML Package Structure

As product development moves from the feature level to subsystem to system level, the integration, verification, and safety validation continues as well. Typical safety tasks at the system level include validation of any assumptions for the ASIL classification, human behavior, aspects of the functional safety concept implemented by other technologies, and assumptions concerning effectiveness and performance of external measures. The RAF systems engineering model permits the safety analysis, risk mitigation, strategies, concepts, requirements, and documentation to be fully captured, linked, and traced to other model elements. Spreadsheet-like tabular displays permit all stakeholders the ability to visualize safety information easily and within a format that conveys information pertinent to their particular perspective.

System stakeholder needs and system requirements are captured within a requirements management tools like DOORS Next Generation. Development of a digital thread from the requirements data to the systems architecture data permits the systems engineers to work within the modeling environment and develop relationships between modeling elements and requirements. Utilization of the Cameo Data Hub plug-in provides the ability to create and maintain this digital thread (Figure 9).

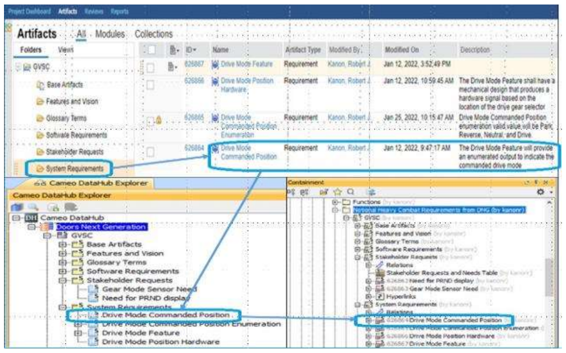


Figure 9 – Digital Thread

The logical architecture view provides the flexibility to support the allocation of feature functions to a design independent of physical constraints. The usage of the SAE J3049 Surface Vehicle Recommended Practice provides a proven standard to be the starting point. This logical view provides integration and validation capabilities enabling subsystems and components not only to connect properly through interfaces, but to execute via simulation to analyze their expected collective behavioral functions across a higher level of abstraction. This permits analysis and validation of subsystem and component requirements as feature functions are integrated across the logical architecture. This step in the process also uncovers emergent requirements that are discovered through the system design process, much earlier in the system life cycle that when production begins. Earlier discovery of these issues results in higher quality, decreased errors, and cost savings over the program life.

The logical architecture is focused on building a system that meets the customer requirements, defining connections and communication, and fully investigating the scope of the problem while demonstrating that the solution works.

Lastly, the physical architecture specifies specific devices that the functional elements will actually reside on, and provides enough detail to implement the architecture. The physical architecture will identify the hardware, software, sensors, and structure, and will also satisfy all non-functional requirements including reliability and availability.

10. MODELING AND SIMULATION

Modeling and executable simulation (see Figure 10) is used to define and verify behavioral analysis, as well as functional validation of the system of interest and its requirements. To facilitate experimentation

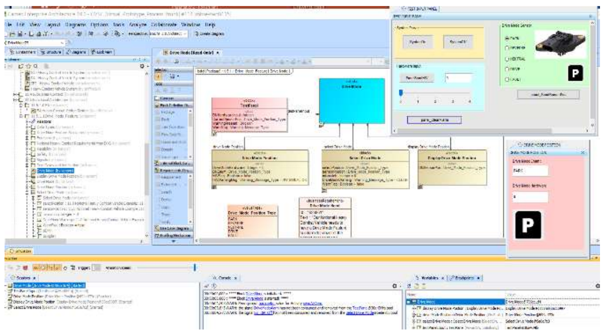


Figure 10 – Modeling and Simulation

and assessment of the system as part of the DE/MBSE process, model representation is constructed in a common system language (i.e. SysML).

Ultimately the creation of these models will create the authoritative source of truth with digital traceability up and down the system life cycle.

The SAIC Digital Engineering Validation Tools provides modeling consistency to reduce errors, aid analyses, and improve quality. Use of validation rules for Cameo Enterprise Architecture and IBM Rational Rhapsody have an immediate and measureable impact on model quality. These tools are available to industry at no expense and contains 201 validation rules for both language and style. These customizations also include model-based style guides, videos, and example models. The style guide and language semantics are automatically enforced resulting in coherent and consistent high quality system models.

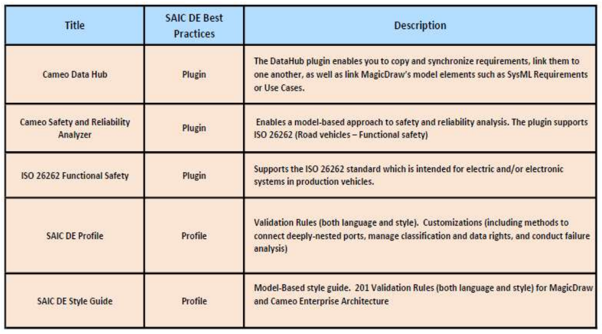
Model creation provides the digital thread from requirement to functional behavior. Executable simulation of these models are used to verify and validate functional and operational requirements. Simulation models are part of requirements analysis, hazard analysis, and failure mode effect/criticality analysis. However, at all times, the model always remains the authoritative source of truth that gets modified when customer requirements change and new design decisions are made.

11. MODELING ELEMENTS

When creating the model elements within the RAF structure, the use of profiles and plugins allow for extended usage of model elements within the SysML language and adds support for additional specification, analysis, design, and validation of a broad range of systems and specifications. Profiles offer the ability to customize and ensure consistency across multiple users, and often contain packages of stereotype modeling elements. A stereotype defines a new kind of SysML model element that extends an existing element by adding properties, constraints, or semantics. A plugin establishes the toolset environment (e.g. CAMEO) to enable a model- based approach to standards and methods.

The following (Figure 11) lists a few of the profiles and plugins that are in currently incorporated as part of SAIC DE best practices:

Leveraging the use of profiles and plug-in



Title	SAIC DE Best Practices	Description
Cameo Data Hub	Plugin	The DataHub plugin enables you to copy and synchronize requirements, link them to one another, as well as link MagicDraw's model elements such as SysML Requirements or Use Cases.
Cameo Safety and Reliability Analyzer	Plugin	Enables a model-based approach to safety and reliability analysis. The plugin supports ISO 26262 (Road vehicles – Functional safety)
ISO 26262 Functional Safety	Plugin	Supports the ISO 26262 standard which is intended for electric and/or electronic systems in production vehicles.
SAIC DE Profile	Profile	Validation Rules (both language and style). Customizations (including methods to connect deeply-nested ports, manage classification and data rights, and conduct failure analysis)
SAIC DE Style Guide	Profile	Model-Based style guide. 201 Validation Rules (both language and style) for MagicDraw and Cameo Enterprise Architecture

Figure 11 – Profiles and plug-ins

environments allows for enhanced, consistent, and quality level model elements further strengthening the model's authoritative source of truth within the RAF environment.

12. TOOL INTEGRATION

The catalyst for execution of this project is a fully-integrated Digital Engineering Ecosystem. Industry leading tools that provide superior capabilities and adhere to recognized standards have been selected for the design and development of the tooling environment. The collaboration aspect is provided by Aras Innovator (see Figure 12), which supplies the Product Lifecycle Management (PLM) capabilities, but also serves as a digital data backbone to connect data from multiple engineering disciplines and permit construction of effective digital threads permitting efficiencies and traceability. In addition, Innovator provides an excellent simulation process and data management core capability to enable simulation tools to leverage data easily from the SysML models, and to maintain the data associated with simulation workflows.

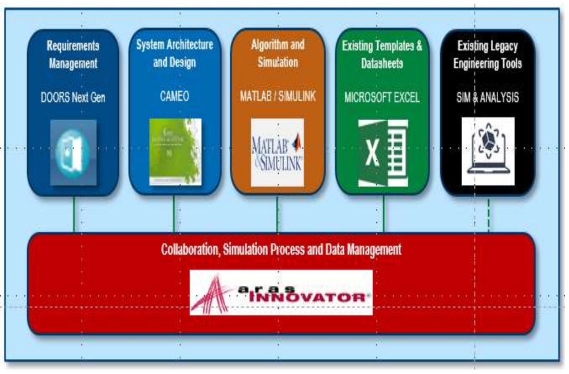

Requirements Management
DOORS Next Gen
System Architecture and Design
CAMEO
Algorithm and Simulation
MATLAB / SIMULINK
MATLAB &SIMULINK
Existing Templates & Datasheets
MICROSOFT EXCEL
Existing Legacy Engineering Tools
SIM & ANALYSIS
Collaboration, Simulation Process and Data Management
aras INNOVATOR

Figure 12 – Integrated Digital Environment

Stakeholder needs and system requirements get captured and managed within the DOORS Next Gen tool from IBM. The data captured within DOORS is synchronized to ARAS Innovator, and therefore now available to any other engineering discipline, such as the system modeling tool Cameo Enterprise Architect. Within the Cameo environment, requirements can be viewed, added to diagrams, connected to other modeling artifacts. The digital threads make needed data and information available to various team members within the tool that is needed for each to perform their particular tasks on the project. Simulation engineers can access requirements and system modeling data within their simulation tool, such as MATLAB Simulink. Existing system engineering tools can also be connected to the digital ecosystem through a data connector, which is accomplished via an existing connector like Microsoft Excel, or other authoring or simulation tools. These connectors can also be developed via an Aras Innovator open architecture approach ensuring the digital ecosystem is never locked into a single tool or vendor, and ensuring collaboration for distributed teams across the entire product lifecycle. Aras' Innovator solution provides all the remaining PLM capabilities after the design is complete, including product data management, manufa cturing, operations, quality, and sustainment features.

13. CONCLUSIONS

The DE transformation strategy is essential in supporting DOD goals and objectives to improve engineering processes, practices, and methodologies including tool integrations and deployment. The RAF provides the core foundation, structure, and content to establish the DE MBSE authoritative source of truth and integration throughout the product life cycle to reduce risk and improve product quality. The major elements identified in the RAF aid in the development of the DE models applying the MBSE SysML language and ontology. The RAF accelerates modeling and product development and provides the building blocks to perform engineering analysis and distribution of information in a consistent format improving communication.

14. REFERENCES

[1]Department of Defense, "Digital Engineering Strategy," Office of the Deputy Assistant Secretary of Defense for Systems Engineering, Washington, D.C., June 2018.

[2]ISO 26262, "Road Vehicles – Functional Safety," International Standard, second edition, December 2018.

[3] MIL-STD-882E, "Department of Defense Standard Practice System Safety", May 2012

[4]IEEE 15288, "Systems and Software Engineering – System Life Cycle Processes", International Standard, first edition, May 2015.

[5]IEEE 12207, "Systems and Software Engineering – System Life Cycle Processes", International Standard, first edition, November 2017.