
R. William Maule, Ph.D.
Information Sciences Department, Naval Postgraduate School, Monterey, CA, USA

Abstract - Applications and services deployed across complex, multi-layered architectures require comprehensive modeling frameworks to effectively manage the software lifecycle—from concept to code. The typical process evolves from high-level concepts to requirements and specifications and then design, development, deployment and maintenance. Too often the modeling process breaks with the code, especially in the deployment and maintenance phases. This makes the understanding of complex architectures exceedingly difficult, especially as we evolve into the “Internet of Everything” with the resultant requirement to address diverse end-user devices. This paper presents a methodology that integrates best practices from government and commercial modeling and provides for integration of commercial tooling to support system and software lifecycles from concept models through to technical engineering models suitable for code generation.

Keywords: Modeling, Lifecycle, Enterprise, Systems

1 Introduction

Modeling frameworks can support the design and development of complex enterprise systems. Frameworks can be especially advantageous for system-of-systems architectures where a modeler can quickly be overwhelmed with the complexity of integration. Comprehensive frameworks might address distributed enterprises and assist with current efforts to support "bring your own device" (BYOD) in "Internet of Things" (IoT) scenarios—which collectively require that future enterprises be flexible enough to integrate any device yet robust enough to secure every device. This paper examines some of the current issues faced by modelers of diverse enterprise systems and discusses some tools to help mitigate complex modeling problems.

2 Model Frameworks

IEEE Standard 1516-2000 establishes modeling and simulation high-level architecture rules to help ensure consistent framework deployment in the community [1]. For example, the Object Model Template provides a documentation standard for describing data used by a particular model. The Federate Interface Specification describes a generic communications interface for interoperability between simulation models. In addition to industry frameworks are frameworks that are more specific and often aligned with functional areas or disciplines.

Even within specializations or disciples these framework-based models can be developed from a number of complementary perspectives. Object-oriented frameworks work at the code level and offer an efficient means for code reuse [2]. Frameworks have been developed to support multi-dimensional engineering at a macro level that includes mission areas, technical domains, and supporting lifecycles [3]. Model frameworks have been targeted to address issues of abstraction in sequence-based activities [4]. Frameworks have also been narrowly focused to address issues within specific disciplines, such as a framework to support concurrent design refinement from behavioral and functional modeling in rapid prototyping systems architecture [5].

Recognizing the need for frameworks to model components in complex heterogeneous systems, researchers have proposed meta-model frameworks to integrate semantics across models to provide consistent designs [6]. The addition of services into models with modular self-contained components and requirements to support loose coupling, dynamic runtime service discovery, and late binding has given rise to frameworks specific to service architecture [7, 8].

3 Services and Processes

The need to address services within a modeling framework adds to complexity. Service-based engineering tools and algorithms tend toward fine-grain process analysis and offer an additional means for framework and model development. In this context the focus would be on frameworks for component assessment and methods for component performance modeling.

Cumulative or composite services multiply the number of component interactions that must be modeled due to this increased number of components and processes [9, 10]. This need is accentuated in widely distributed system-of-system architectures. Resource availability in composite, cumulative services becomes a primary concern as the paradigm shifts from single process to integrated services [11, 12]. Adding to the complexity of modeling composite services is the cumulative impact of operations in each of the OSI layers, e.g., process, service, routing, transformation, etc. [13, 14].

This requires that attributes be modeled at different levels of abstraction [15] to address coupling between services and clients [16] and to support measurements to include process throughput, reliability and availability [17]. Heuristics must
address the impact of component variations in service reuse across different architectures [18, 19, 20] and the impact of these service interactions on user requirements [21, 22] in component relationships, object interactions, and rules.

4 DoD Model Framework

The U.S. Department of Defense (DoD) uses a highly detailed modeling framework with dozens of integrated models to help with lifecycle modeling. While not sufficiently comprehensive to provide end-to-end lifecycle modeling, from concept to code, the DoD Architecture Framework (DoDAF) provides a base into which other needed frameworks and models can interface. DoDAF applies the IEEE 1471 definition of an architecture description to define a standard approach to describing, presenting, and integrating architecture.

Services represent operational functionality, with information exchanges modeled as data types that traverse functionality through Capability Viewpoints (CV) that model requirements, Data and Information Viewpoints (DIV) to model data relationships, Project Viewpoints (PV) to model relationships between operational and capability requirements in services design; and Services Viewpoints (SvcV) to model performers, activities, services, and their exchanges [23].

Enhancements to DoDAF can help with the sequencing of modeling products [24] by validating consistency in the products [25], enhancing event specification capabilities [26], and providing support for simulation-based testing [27]. Integrations with Activity Based Methodology (ABM), System Modeling Language (SysML), and Business Process Modeling Notation (BPMN) have strengthened DoDAF.

DoDAF-2 shifted the underlying modeling paradigm to data-driven “viewpoints” with an extensible data model [28]. DoDAF-2 also added DoDAF Meta-Model (DM2) which provides a high-level view of data elements and enables the modeler to describe a model in XML (vice a physical/visual model) to speed model exchange to better support reuse of architectural information [29].

Processes for integration with coding frameworks and models such as the Unified Modeling Language (UML), UML Profile-based Integrated Architecture (UPIA), and SOA Modeling Language (SoaML) have been advanced [30]. The research herein extends this research through a first step at a workflow for the integration of models and supporting frameworks as required for a comprehensive end-to-end lifecycle and toward an eventual Framework-of-Frameworks for Context-to-Code (FoF-CtC) modeling.

5 Framework Integration

Any discussion of industry model integration must begin with the Unified Modeling Language (UML). A history of UML is not needed for the readers of this paper but some notes on the extensions as applied to larger meta-models is appropriate. Specifically, the UML Profile-based Integrated Architecture (UPIA) provides a useful extension to DoDAF in a FoF-CtC methodology. UPIA can be integrated with Model-Driven Software Development (MDSD) methodologies to enable generation of operational code from models [31].

The UPIA SOA Design viewpoint can model service specifications, service ports, and service consumers and providers at both operational and system levels of abstraction. UPIA is DoDAF-2 XML-compliant and can import and export DoDAF-2 Physical Exchange Specification (PES) architectural data. Most of the concepts defined in DM2 can be modeled in UPIA and there are utilities for model validation. Models advanced in this paper will implement UPIA to bridge UML and DoDAF and therein provide a data point for framework integration. Comprehensive architecture models will evolve from high-level operational and systems models to low-level code.

The required framework integration for end-to-end lifecycle modeling is further strengthened through SoaML which can address low-level variables for modeling code development. SoaML is an open source specification project from the Object Management Group (OMG) to describe a UML profile and meta-model for services within a SOA. Existing models and meta-models (e.g., TOGAF [32]) for describing system architectures were considered insufficient to describe SOA in a precise and standardized way. UML was too general and needed clarification and standardization specific to SOA. Additionally, a means was required to operationalize SOA as advanced in the OASIS Reference Model (OASIS-RM). SoaML was adopted as a means to provide SOA-specific tooling and to instantiate OASIS-RM.

Researchers have found that SoaML adds value to UML for large-scale SOA deployment [33]. Extensions have been added to provide support for multi-agent systems [34], mobile applications [35], service requirement variability [36], security-aware invocations [37, 38], and pattern propagation between design and performance models [39]. However, while designed to bridge IT and business models, SoaML is somewhat vague regarding implementation methodology to other business-level languages, such as Business Process Modeling Notation (BPMN) [40] – hence, the need for framework integration.

6 Model Integration

The following discussion extends methodologies for model integration through examples, moving from high-level concept models, through performance-based models, to low-level code models. Whenever possible the models and their supporting frameworks are referenced against DoDAF to show how commercial tools and their model outputs can interface in day-to-day, practical usage.

For example, SoaML is modeled through UPIA and referenced against DoDAF to address anticipated future requirements for secure distributed SOA clouds. In other words, UPIA is presented as a means to associate SoaML models within the DoD modeling framework. The author has taken liberties in the application of commercial models to DoD frameworks and this does not represent DoD policy.

First is a commercial modeler that has been adopted and modified to meet DoD-specific requirements. The Joint
Communication Simulation System (JCSS) uses DoD-specific systems nomenclature. JCSS is a Defense Information Systems Agency (DISA)-endorsed Commercial Off-the-Shelf (COTS) Modeling and Simulation (M&S) tool based on the OPNET Modeler product line. The intent in providing this software to the DoD community is to develop a common modeling base across Command, Control, Communications, and Computer (C4) systems communities [41]. As the name implies, this software is focused on communications planning and on simulation of communications effects on networks.

In operation, modeling begins with construction of a network topology. Then, information systems, databases, and communication-specific devices are added—such as routers, satellite antennas, multiplexors, etc. Expected transmission capacity is modeled, including bandwidth that will be available and expected demands on that bandwidth. Simulations using “what-if” analysis help determine if communications capacity is sufficient to accommodate expected network traffic. Of course, this implies that the modeler has access to accurate data—such as past network performance for similar environmental constraints and transmission conduit, the load generated by the applications, and the degree of latency tolerated by the systems.

JCSS can generate reports to help analyze models with utilization statistics, failure reports, and network optimization analysis. Discrete event simulation can be run as a “state machine” in which change in the state of a machine, network connection, or transmission capacity can help predict impact on systems and applications—with traffic metrics for jitter, latency, and queuing delay.

JCSS reports can be rendered into a DoDAF OV-3 Operational Information Exchange Matrix or DoDAF SV-6 Systems/Services Data Exchange Matrix. The higher-level views generated in JCSS can serve as DoDAF OV-1 High-Level Operational Concept Description models, and the more detailed views can present DoDAS SV-2 Systems Communications Description models (see figure 1).

Another tool, not technically a DoD modeler but a used regularly within the DoD community to help understand complex mission-based relationships is the System Tool Kit (STK) from AGI Corporation which excels in model development for space defense industries and is widely used for satellite and aircraft modeling and simulation. Recent product enhancements have introduced opportunities to include air, land, and sea assets in the models with high-resolution 3-D visual simulation [42]. As an output format in a DoD environment the STK would be used for operational modeling.

If properly constructed, through STK, we can create precise, realistic events. The reports that we can generate are more in the area of resource allocation vice the JCSS communication performance reports. In a perfect world, with time permitting, we would use both tools. We would model then simulate the scenario in STK with primary resources and assets for 3D visual analysis, and we would simultaneously model communication specifics within JCSS. We would use STK for asset evaluation—for example, Unmanned Aircraft System (UAS) sensor packages within geographic areas. Or, area coverage of Unmanned Vehicle (UAV) flight patterns for search and rescue. Or, analysis of communications coverage for mobile users in isolated terrain with geo-stationary satellite constellations supplemented by Beyond Line-of-Sight (BLOS) radios. Prior to execution we would model the communications in JCSS and during execution capture the traffic data, import that data to JCSS, and use that data to refine our STK event simulations and resource allocation models.

STK does not render directly to DoD but the visual representations can be captured for OV models. The reports can be used as data points for DoDAF resource matrix models. Figure 2 shows the STK user interface with rendering in 2D and 3D windows—which are active for both modeling and simulation.

We can integrate STK with a large number of complementary modeling packages for an easy exchange of
ideas, such as Keyhole Markup Language (KML) for visualization to Google Earth. Or, we can export model specifications to the open source System Modeling Language (SysML) for systems engineering specification and model validation. We can use QualNet software from Scalable Network Technologies to capture live communications data into STK simulations. As such, STK fits nicely within the concept of an end-to-end lifecycle modeling framework.

As referenced earlier, UML-based modeling tools, suites and frameworks would be a critical component of a comprehensive end-to-end lifecycle analysis. These tools and their models represent the far end of the modeling spectrum—code generation. So, in evaluating a FoF-CtC approach, modelers such as STK would simulate the concept or mission and receive performance data from QualNet. JCSS and the supporting OpNet modelers and tool suites would occupy the middle tier for communications performance assessment. UML-based frameworks and modeling suites would be for low-level code modeling. In the FoF-CtC proposed herein all are integrated with DoDAF-2 such that each model and approach is mapped to a corresponding DoDAF model or viewpoint.

While UML itself is an open standard and governed accordingly, the company that brought UML to market is Rational which is now part of IBM. The System Architect (SA) suite of modeling tools from Telelogic is also now part of IBM, positioned under Rational. SA has the most robust support for DoDAF of any model tooling available and excels at the lower-level modeling. In the FoF-CtC, SA will serve as the technical means to integrate low-level system and service models with UML-compliant code-generation models as applied to DoDAF.

Figure 3 shows the DoDAF model options within the SA graphical user interface. There are two SA versions optimized for DoDAF architectures: SA for DoDAF with the MITRE Activity Based Method (ABM) option, and SA C4ISR—which has been renamed to SA for DoDAF—to build open architecture models around structured IDEF techniques. Integration DEFinition (IDEF) models are also significant in the DoD and would be a component of a FoF-CtC. Similar to DoDAF, IDEF models range from high-level functional models to low-level object-oriented models.

Rational Software Architect (RSA) supports the software engineering phase of code modeling and is also built on UML and is capable of DoDAF-like models but without pre-defined DoDAF output modes such as SA. In other words, we need to understand the DoDAF model that we are designing and then use RSA to develop that model vice the SA process where we would select the DoDAF model and then automatically receive the necessary tooling. RSA is built on the Eclipse open-source software framework which is the industry standard for Model-Driven Development (MDD) in both the DoD and commercial sectors.

Figure 4 shows the RSA user interface in the traditional Eclipse design pattern—with the model in the center and the development tooling around the model. In the figure we have selected Integrated Architecture Modeling (IAM) with UPIA SOA design options to develop SoaML models—as indicated in the selection in the left column. RSA can be applied in the requirements specification phase and models simulated to help communicate system dynamics and evaluate software against different network topologies [43].

Another level of modeling valuable in FoF-CtC lifecycle assessment but without a direct correlation to DoDAF viewpoints addresses Return on Investment (ROI) analysis wherein processes are modeled, assigned cost variables, simulated and assessed. Figure 5 shows a Business Process Management (BPM) model specific to a SOA processes using WebSphere Business Modeler (WBM) [44]. The simulation component includes a module that calculates ROI for a proposed innovation—in this instance, for a complex role- and attribute-based security process.
7 DoDAF Integration

A discussion of DoDAF is beyond the scope of this paper. There are many excellent online references, and the source DoDAF site is available online. The framework is extensive with multiple models in model categories that address capabilities, data and information, operations, projects, services, standards, and systems [45].

The intent herein is to use DoDAF as a baseline for framework integration since it seems to be the most comprehensive framework available. Yet, DoDAF is insufficient at the end-to-end lifecycle extremes, both at the highest conceptual levels and the need to visualize and simulate complex mission threads, and at the other extreme for low level code development, deployment, monitor, management and maintenance.

As such, DoDAF serves as the baseline and our starting point for FoF-CtC lifecycle integration. Since our interests for this paper are in service architecture, a couple of examples that apply the commercial tooling, models and frameworks previously discussed to DoDAF will be given.

To start, figure 6 presents a UML use-case for a distributed service architecture. While not really a UML use-case, and not really a DoDAF Operational Viewpoint (OV), the integration provides value in our FoF-CtC integration since a parallel development of our DoDAF modeling with the commercial tool suite for code development will enable an evolution from concept to code—all within the confines of a reference framework.

Figure 6. Use-case for secure distributed service architecture.

The System Viewpoint (SV)-2 is a prominent DoDAF model to show how a system operates and its interfaces. In the SV-2 we model data flows between systems, identify the protocols and the networks, and specify the system ports. A model may be created for each resource flow, or we can model all resource flows on one diagram. For example, figure 7 models a Navy cloud which consists of telecommunications centers and user systems. Within the cloud are nodes which provide services. A source node hosts authoritative data sources and physical servers host virtual machines.

Services Viewpoint (SvcV) models offer a precise means to model interactions in a service architecture. In these models we visualize interactions and describe services and resources required for development and execution [46]. As such, the SvcV series of models become a backbone for a FoF-CtC service architecture lifecycle.

Figure 7. SV-2 for backbone cloud with subordinate clouds.

SvcV models can articulate use-case performers, system activities, and data exchanges in support of operational capabilities. Both structural and behavioral factors, and the impact of such factors on the architecture, can be modeled. The SvcV models work hand-in-hand with the SV models discussed earlier. The SV models focus on system and component interaction, the SvcV models a step lower to service, object and process interaction.

A final example of framework integration is the hybrid model in figure 8 which builds from a SvcV-10b to model a resource (or function) response to events—taking action to move to a new state as a function of a current state—with each transition an event and an action. This can be correlated with a UML state chart to model change in a sequence of service functions. Behavior is modeled as the traversal of a graph of specific states, all interconnected by transition arcs that are triggered by events. A SoaML model represents the SvcV-10b state machine. State change in this instance refers to change in our SOA due to changes in data elements or attributes—such as publication topics or subscriptions—or changes in security for any process.

The oval in the diagram are UML collaboration use-cases for participant interaction—which is where our state change occurs. The connection boxes with handles we use to identify service interface ports and protocols. The service contract is the means to ensure QoS and enforce contracts. The collaboration shows how participants work together to provide services. Each participant plays a role in the service contract to help verify and ensure that underlying constraints are honored.

Figure 8. SvcV-10b SoaML service state diagram.
8 FoF-CtC Lifecycle

Figure 9 provides the cumulative FoF-CtC integration lifecycle as developed throughout this paper and presented as an initial step toward model and framework integration across disciplines and between modeling communities. The intent is to address limitations of current approaches which do not really provide a comprehensive end-to-end modeling solution from concept to code—to include deployment, monitor, performance assessment and maintenance.

In the examples presented, UML-based frameworks were integrated with the DoD framework as a step toward comprehensive end-to-end modeling. For example, a SvcV model represented data flows in standard DoD notation, and then a corresponding UML model was integrated to address tooling requirements for the generation of computer code from the models. Together the integrated models, along with complementary tooling and frameworks, become the core of our service architecture and a baseline for FoF-CtC lifecycle integration. Supporting models and commercial tools were presented as extensions that provide visualizations of the capabilities advanced in the collective models, as represented in the FoF-CtC lifecycle.

Future research might integrate additional frameworks or commercial tooling to fill identified gaps in the current FoF-CtC or strengthen the base of applicable models in the identified categories. Additionally a workflow for model application, drawing from the FoF-CtC, might present a means for ready integration and usage of models across system and software development lifecycles.

9 Conclusion

The FoF-CtC lifecycle methodology provides a means to model from high-level operational concepts to an integrated enterprise, evolving from goals or objectives to modeling specifics which realize specific functions, processes and services. Examples were presented to illustrate a means to integrate government and commercial modeling approaches to optimize the strengths of each and begin a dialog on framework and model integration.

10 Acknowledgment

The author wishes to acknowledge OPNAV for support of this research.

11 References
