20th ICCRTS

Executable Architectures Concept and Methodology
Paper 002

Topic 1: Concepts, Theory, and Policy
Topic 5: Modeling and Simulation
Topic 9: C2-simulation Interoperability

Mr. Eui Soon Kim
Korea Institute for Defense Analyses
37 Hoegi-ro, Dongdaemun-gu, Seoul 130-871,
South Korea

+82-10-5081-2462 kes3738@naver.com

Executable Architectures Concept and Methodology

Mr. Eui Soon Kim
Korea Institute for Defense Analyses
37 Hoegi-ro, Dongdaemun-gu, Seoul 130-871,
South Korea

Abstract

The static architectures are converted into executable architectures for conducting detailed dynamic behavioral analysis. A survey of research papers show that many terminologies are referred to and used for the explanation of executable architectures. This results from the simple definition of an executable architecture.
This paper surveys related papers and suggests a clear definition of an executable architecture and arranges transition methodologies. Architecture-based analyses employ the transition of architecture products into intermediate architectures and then executable models for execution simulation programs. An executable architecture can then be defined to be the union of an executable model and a simulation program.
The modeling relation involves the use of Colored Petri Net (CPN), Discrete Event Simulation (DES) and Unified Modeling Language (UML)/Systems Modeling Language (SysML). The dynamic relation involves the use of Business Process Model (BPM), Matrix Laboratory (MATLAB) and Discrete Event System Specification (DEVS). The simulation relation involves the use of federation, simulation software, MATLAB and DEVS. The consecutive application of the modeling, dynamic and simulation relation results in the executable architecture.
The methodology can be classified into three types. This paper also compares the types and recommends one type for the reuse and composability.

1. Introduction

Static architecture products of the Command and Control (C2) system only show that operational activities/system functions must be capable of producing and consuming information and data. They do not provide details on event

sequencing or how or under what conditions information/data is produced and consumed. They also do not explicitly identify, for each activity/function, the number (capacity) of roles/systems needed or their ordering for the case when multiple roles/systems perform the same activity/function (who/which operates on the first input, who/which operates on the second, etc). Static architecture products can't be used to carry out a dynamic analysis as to how operational activities and system functions interact.

To overcome these limitations, operational activities, information, and resources should be formulated as a dynamic model which would be converted as an executable simulation (xS). The static architecture products go through modeling and simulation (M&S) and they become an executable architecture (xA).

The objectives of developing executable architectures of the C2 system from DoD-developed static architectures are mainly to identify bottlenecks in C2 processes and communications networks and estimate optimal C2 activities process times and to identify operators in organizations as well as nodes in the communication systems (as networks) that are overloaded and re-distribute the C2 activities where appropriate [7].

The use of architecture-based network simulation to study denial of service attacks is well known. However, M&S techniques can be used to evaluate intrusion detection systems, place and configure security appliances and to design appropriate access control mechanisms [12].

A survey of research papers and articles shows that many terminologies are referred for the explanation of an executable architecture. This results from its simple definition such as "An executable architecture is a dynamic model of Activities and their event sequencing performed at Operational Nodes by Roles (within Organizations) using Resources (Systems) to produce and consume Information"[1]. The definition needs to be re-defined to have all the properties and relations with a static architecture. This paper surveys related papers and suggests a clear definition of an executable architecture. This leads to the arrangement of Modeling and Simulation (M&S) methodologies.

Typical M&S methodologies can be categorized into the three groups. This paper compares the three groups and recommends one from reusability and

composability point of view.

2. Concept of an Executable Architecture

2.1 Mixed Terms and Analytical Models

The survey shows that the three terms "architecture, model, and simulation" (in the right side of Figure 1.) appear with some adjectives (in the left side of Figure 1.) in various forms such as a static architecture, a static model, an integrated architecture, an integrated model, a dynamic model, an executable architecture, an executable model, an executable simulation, an executable simulation model etc. Figure 1 provides the terms in two sets. The arrows and colors denote such relations.

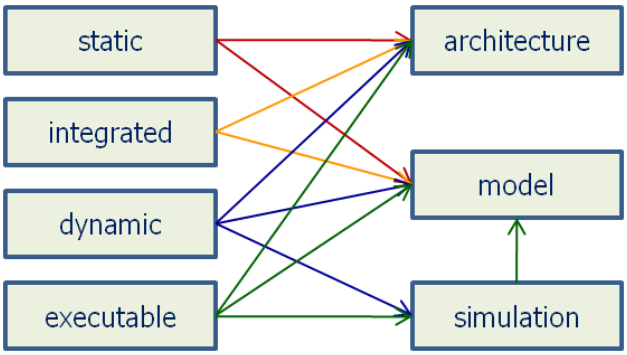

Figure 1. The Mixed Relations of Terms

The models mentioned above are analytical models depending on the modeling technique. For example, the models can be a CPN model, BPM, a UML model, a SysML model, an IDEF model or, other mathematical models.[1] A part of them is correlated to combat simulations and communication/network models. The mixed terms and related analytical models cause some confusion about how they are related and what they are. The concept of an executable architecture can be plainly defined by making these terms and models clear.

2.2 Definition of an Executable Architecture

The framework for M&S as described by Zeigler, et al. [4] establishes entities and their relationships that are central to the M&S enterprise; see Figure 2. The

[1] CPN: Colored Petri Net, IDEF: Integrated Definition

entities of the framework are source system, experimental frame, model, and simulator; they are linked by the modeling and the simulation relationships [2].

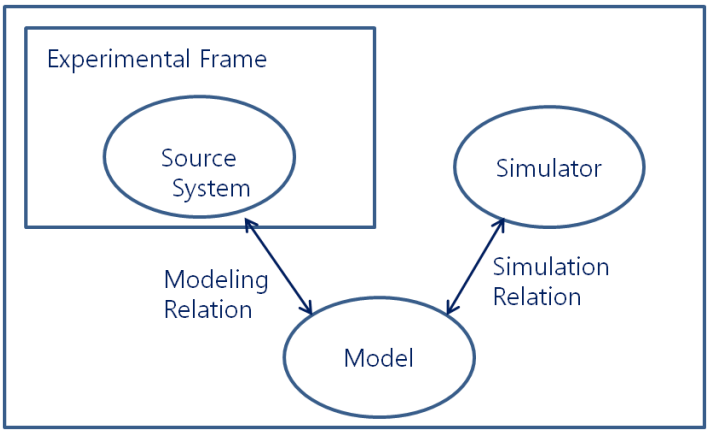

Figure 2. Framework entities and relationships

This framework for M&S can be applied to the mixed terms and analytical models. Figure 3 depicts the complete mapping among them by introducing a new entity called an intermediate architecture (*mA*). The entities are an integrated architecture (*iA*), an intermediate architecture, an executable model (*xM*), and an executable simulation (*xS*). These entities are linked by the modeling relation (M_R), the dynamic relation (D_R) and the simulation relation (S_R). The statics entities of *A, iA and mA*, while capturing enormous amounts of information about the Operational Architecture (OA) and System Architecture (SA) fail to provide a good vehicle for conducting dynamic "behavioral" analysis of how the systems are supposed to interact with each other [7].

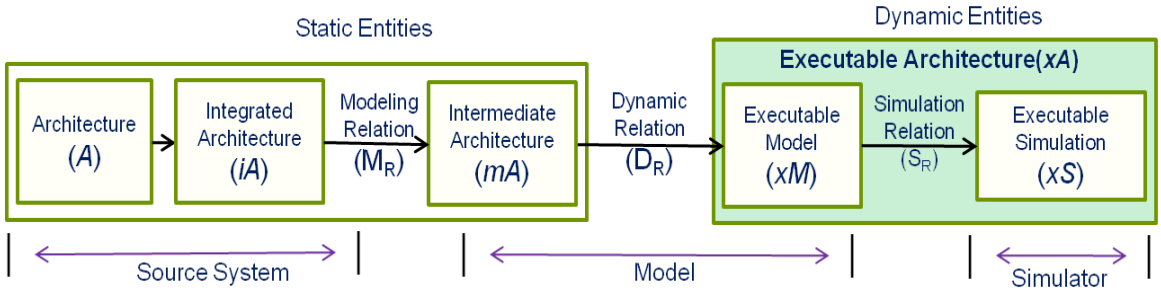

Figure 3. The Concept of an Executable Architecture

Figure 3 shows the transition of an integrated architecture into an intermediate architecture by M_R. The second step is the transition of an intermediate

architecture into an executable model by D_R. An integrated architecture cannot be directly used to create an executable model. An intermediate architecture makes this transition easy through the M_R transformation. Dynamic elements are added to a static architecture by D_R, which provides a dynamic model.

The third step is the transition of an executable model into an executable simulation program by S_R according to the analysis purpose. Therefore an executable architecture is the M&S transformation output of an integrated architecture, which includes both an executable model and an executable simulation. The relation is as follows:

$$mA = M_R(iA)$$
$$xM = D_R(mA)$$
$$xS = S_R(xM)$$
$$xA = xM \cup xS$$

3. Conversion Methodology for Executable Architecture

The survey shows that there is a wide variety of the modeling relation (M_R), the dynamic relation (D_R) and the simulation relation (S_R). Firstly, the conversion methodologies for each relation are reviewed. Secondly, typical consecutive application methodologies of all the three relations are sought and compared.

3.1 Modeling Relation (M_R)

The modeling relation (M_R) involves the use of CPN, DES and UML/SysML; see Figure 4.

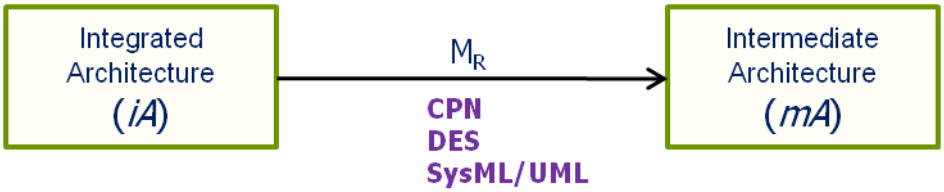

Figure 4. The Modeling Relation

Each modeling technique has its own strength and weakness. Each is applied depending on the analysis target. While CPN is certainly a valuable tool for

understanding the dynamic behavior of a system, it falls short in its ability to model a combat environment where the rules of engagement (ROE) are changing and the enemy model is learning and evolving [3]. DES uses numerical analysis to analyze systems where the state variable(s) changes only at discrete points in time [4]. DES can be a useful modeling tool for modeling things such as queues (which may be seen in logistics analysis or missions involving the movement of information or resources) [4]. The UML and SysML are used to visually express and communicate system structure and behavior with activity flows and information exchanges.

3.2 Dynamic Relation (D_R)

The dynamic relation (D_R) involves the use of BPM, MATLAB/Simulink and DEVS; see Figure 5. D_R takes the charge of populating dynamic properties into *mA*. They include event time, event priority, event sequence, etc. The number of personnel, the number of systems and the performance of systems can be varied in the dynamic model.

The dynamic model is formulated as three types, which are BPM, MATLAB/Simulink model and DEVS model. Especially an intermediate architecture in UML version 2 (UML2.0) can be automatically converted into DEVS models. The tool for automatic conversion has not yet been developed.

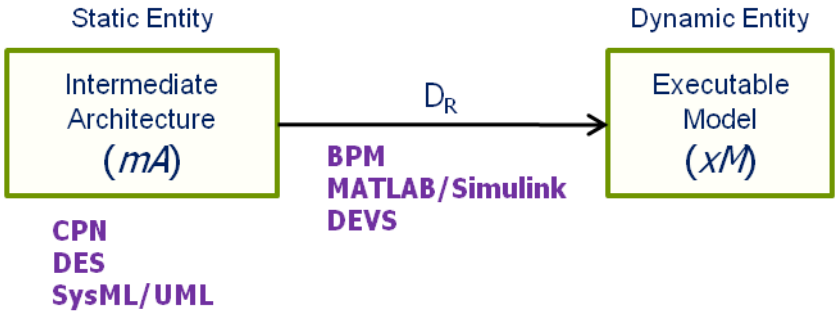

Figure 5. The Dynamic Relation

3.3 Simulation Relation (S_R)

The simulation relation (S_R) involves the use of federation, simulation software, MATLAB/Simulink and DEVS; see Figure 6. MATLAB is a high-level technical computing language and interactive environment for algorithm development,

data visualization, data analysis, and numerical computation. Simulink is good for modeling and designing dynamic systems (it runs under MATLAB). DEVS provides a means to evaluate the component behavior in a finite time frame. Incorporation of DEVS in architecture products will make the design process more tractable and controllable. As a result DEVS has the characteristics of reusability and composability.

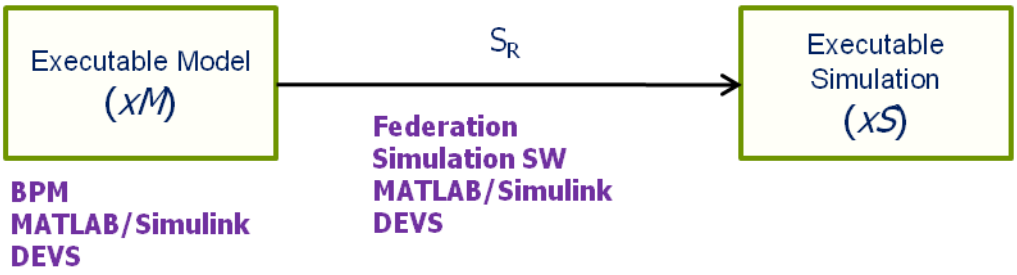

Figure 6. The Simulation Relation

The executable model in MATLAB/Simulink is automatically converted into the executable simulation inside the tool. So is the executable model in DEVS.

The executable model in BPM is converted into the executable simulation by linking combat simulations and communication/network models via the High Level Architecture (HLA) / the Run-Time Infrastructure (RTI)

. Simulation software can be directly used to convert an intermediate architecture in CPN model and SysML model into an executable simulation. This will be further explained in Section 4.

4. Executable Architecture Methodology

The executable architecture can be created by the consecutive application of three relations, M_R, D_R, S_R; see Figure 7. Each relation has some tools or models as in Figure 7.

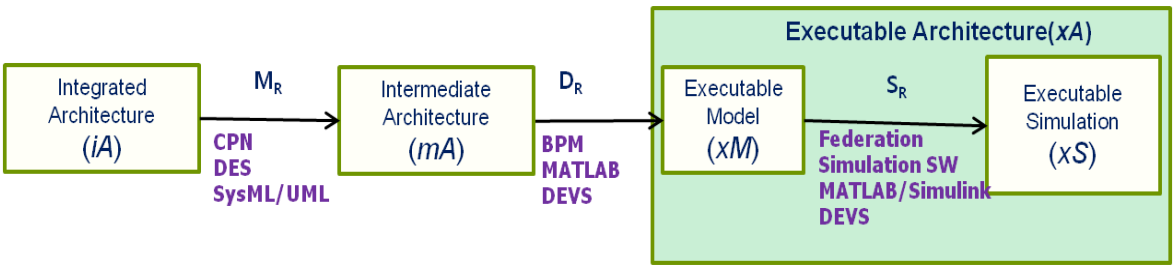

Figure 7. Executable Architecture Methodology

There may be numerically many combinations of three relations. The combinations can be grouped into three types considering the properties of tools or models. The three types are as follows:

○ Type 1: CPN → BPM → federation
○ Type 2: SysML/UML → MATLAB/DEVS → MATLAB/DEVS
○ Type 3: CPN/SysML → - → simulation software
(Type 3 does not require D_R)

4.1 Type 1: CPN → BPM → federation

Type 1 employs CPN as M_R, BPM as D_R and federation as S_R. The federation consists of BPM, the combat simulation, and the network model linked together through HLA.

The integrated architecture is converted into an intermediate architecture using CPN by mapping elements within operational views (OVs) into CPN model components. The activities from the OV-5 become Transitions in the model [5]. The Places on the model come from Input, Control, Output, Mechanism (ICOMs) off the OV-5 and OV-6a. The attributes from the OV-7 become Tokens in the model. The CPN model does not use the System Views (SVs) and the Technical Views (TVs).

The CPN model is not able to effectively model a variable environment like a battlefield. CPN models the logical behavior of the system. We may analyze how data flows through functions, yet we do not have enough information to determine how long it takes [6].

CPN cannot evaluate the effects of changing individual system performance and communication systems, or the effects of changing the Concept of Operations (CONOPs) that guide blue force operations against an enemy that has CONOPs of its own way. To overcome these limitations CPN is converted into a timed CPN model or BPM to reflect dynamic properties. A timed CPN model or BPM alone is restricted from various kinds of analyses. As a result a timed CPN model or BPM needs a federation of simulations in a different paradigm.

The simulation paradigm will be using combat simulations and C2 models. Combat simulations are of two kinds - a deterministic model and a stochastic model. The employment of a federation mode for a deterministic simulation to the evaluation of target architectures is not as useful as one with a stochastic model.

As an example, a BPM, EAGLE as a deterministic Army combat simulation and a communication model using the Network Simulator version 2 (NS-2) are linked together via RTI of HLA as in Figure 8 [7][2]. As another example, a BPM, the agent based combat simulation such as the System Effectiveness Analysis Simulation (SEAS) as an Air Force stochastic combat simulation, and the Optimized Network Engineering Tool (OPNET) are linked together. The entities in one model must relate to entities in another model. A mapping of relationships is also necessary to establish how events in one model are related to events in another model.

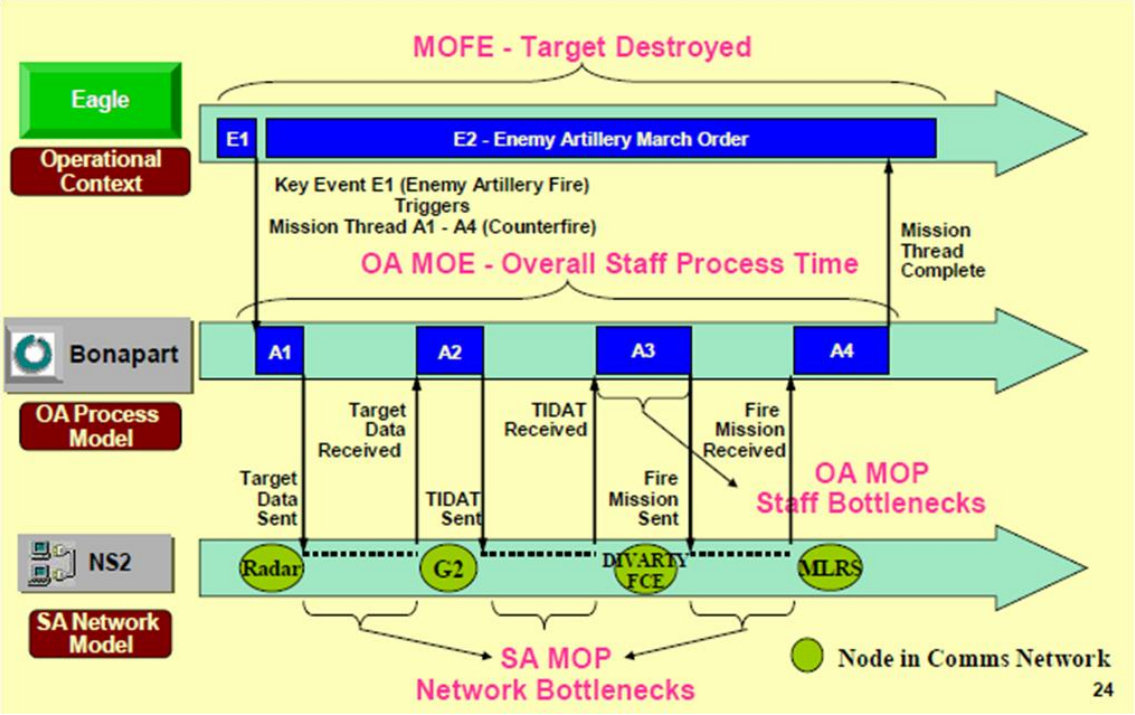

Figure 8. Modal Interactions and Sample Measures of Merit

4.2 Type 2: SysML/UML → MATLAB/DEVS → MATLAB/DEVS

Type 2 employs SysML/UML as M_R, and MATLAB/DEVS as D_R and S_R.

---

[2] Some of the characteristics of the network model could be cyber and trust characteristics

UML elements are mapped from architecture products. The translation tables as in Table 1 and 2 show the mapping of various OV/SV products and UML elements [6].

SysML elements are also mapped from architecture products. The mapping Table 3 shows the mapping of various OV/SV products and SysML elements [8].

The intermediate architecture in SysML/UML can be used as a vehicle for conveying the necessary information for M&S, but it does not provide dynamic results. It must be converted into an executable model in dynamic concept to get useful analysis results.

The elements in Simulink have a close relation to SysML/UML entities, making the conversion feasible. The mapping relation is provided in Table 4 [9].

Table 1. DoDAF-DEVS extended translation table focusing focusing on OV

DoDAF Elements			UML Elements	DEVS Elements (Generated Using XML)	
	Name	Description			
Operational View	OV-1	Top-level operational view	• Use-case diagrams	• Activity component identification • Top-level entity structure	DEVS model repository
	OV-5	Operational activity model	• Use-case • Activity sequencing diagrams • Data flow diagrams	• Activity component updating • Hierarchical organization of activities • Input-output pairs • Port identification	
	OV-6	Operational timing and sequencing diagrams	• Time sequencing diagrams • State machine diagrams	• DEVS atomic model creation (initialize function, internal and external, transition functions, time advance, and output functions) for activity components • Entity identification • Activity-entity component mapping	
	OV-2	Operational node connectivity	• Composite structure diagrams	• Coupling information • Hierarchical component organization	
	OV-8	Activity component description	• Composite structure diagrams • State charts	• Activity component update • Activity port identification and refinement	
	OV-3	Operational information matrix		• Input-output transaction pairs • Message formats • Activity interface and coupling information	DEVS system test suite
	OV-9	Activity interface specifications	• State charts • Composite structure diagrams	• Acitvity-entity interface • Entity structure refinement • Activity-entity port mapping and refinement	
	OV-7	Logical data model	• Packages (only for xUML) • Class diagrams	• Entity identification • Hierarchical structure	
	OV-4	Organizational relationship chart	• Class diagrams	• Entity identification • Hierarchical entity structure	DEVS model repository

Table 2. DoDAF-DEVS extended translation table focusing focusing on SV, TV

DoDAF Elements			UML Elements	DEVS Elements (Generated Using XML)	
	Name	Description			
Systems View	SV-4	System functional description	• Use-case description • Activity sequencing diagrams	• Hierarchical functional components organization	DEVS model repository
	SV-5	System functional traceability matrix (based on OV-5)		• Coupling info refinement	
	SV-10	System state description and event trace (based on OV-6)	• Sequence diagrams • State charts	• DEVS atomic model transition functions refinement	
	SV-6	System data exchange matrix		• Input-output pair refinement	
	SV-1	System interface description (based on OV-2)	• Composite structure diagram	• Port assignment refinement • Entity refinement	
	SV-2	System communication description	• Deployment diagrams	• Coupling info refinement (hierarchical management)	
	SV-7	System performance parameters matrix		• Experimental frame	DEVS system test suite
	SV-3	System-systems matrix		• Hierarchical model organization • Entity refinement	DEVS model repository
	SV-11	Physical schema	• Class diagrams	• Hierarchical model organization	
Technical View	TV-1	Current standards	• Timing response	• Basic DEVS model for COTS component	
	TV-2	Future standards		• Improved DEVS model for desired functionality	

The intermediate architecture in UML can also be converted into an executable model in DEVS. The UML element is mapped to the DEVS element(s), with the translation table shown in Table 1 and 2 [6] demonstrating the mapping of various UML elements and DEVS elements. The reason for choosing the DEVS formalism as a means to M&S is its expressive power and modularity support. The automatic tool for converting UML2.0 model into DEVS can be developed and applied.

An executable model in MATLAB/Simulink and DEVS can be automatically converted into simulation frameworks.

Table 3. The mapping between the System of systems Architecture Development Process (SoSADP), DoDAF, and SysML diagrams for an integrated, NCOW architecture

SoSADP	DoDAF Product	SysML Diagrams							
		Requiremen	Structure		Behavior			EngrAnalysis (Performance)	
		Requirement Diagram	Block Diagram	Parametric Diagram	Activity Diagram	Sequence Diagram	Use Cases	Modeling & Simulation	Context Diagram
SoS Problem	Textual information (not AV-1)								
Needs Analysis									
Analyze SoS Needs	Textual information (part of AV-1)								
Develop MOEs				√					
Mission Analysis	AV-1, OV-1								
Determine Threats	OV-1 (High-level Operational Concept Graphic) AV-1 (Overview and Summary Information)						√		
Define Scenarios							√		√
Define Missions							√		√
Requirements Analysis	OV-6, OV-5, SV-4, SV-5								
Perform Operational Requirements Analysis	SV-4 (Systems Functionality Description)	√							
Perform Functional Analysis	SV-4 (Systems Functionality Description)	√							
Perform Non-functional Analysis		√							
Flowdown Requirements	SV-4 (Systems Functionality Description)	√						√	
Develop MOPs									
Perform Thread Analysis with Data & Messages	OV-6c (Operational Event Trace Description)				√	√			
SoS Architecture Alternatives	OV-3								
Define SoS Force Composition Options			√						√
Identify Critical Elements									
Identify Existing Systems									
Postulate Future Systems									
Perform Functional Embedding	SV-1 (Systems Interface)								
Define SoS Comm. Structures			√						√
Define SoS C2 Structures	OV-4 (Organizational Relationships Chart)		√						√
Define SoS Architecture Options	SV-1 (Systems Interface)		√						√
Define CONOPS									
Develop concepts of operations	OV-5 (Operational Activity Model)				√	√			
Develop Internal Threads with Data & Messages	SV-1 (Systems Interface)				√	√			
Cost and Risk Analysis									
Model Cost									
Identify Risk									
SoS Architecture Ranking									
Perform M&S				√				√	
Conduct Performance Analysis				√				√	
Select SoS				√				√	
Rank SoS Architecture Alternatives				√				√	

Table 4. Relationship between Simulink concepts and UML elements

Simulink concept	**UML 2** concept
Primitive block	Class
Subsystem block	Class, containing properties corresponding to contained blocks
Line	Connector
Branch	Connector
Port	Port

4.3 Type 3: CPN/SysML → - → simulation software

Type 3 employs CPN/SysML as M_R, and simulation software as S_R. In this type even without D_R, the intermediate architecture in CPN/SysML can be directly converted into executable simulations.

The CPN model can be developed in Arena by using mapping rules provided in Figure 9 [2]. The SysML diagram can be converted into Extendsim executable simulation model. The Extendsim is designed specially to represent the flow of messages and data that traverse the network in a specific sequence of events, called a thread [8].

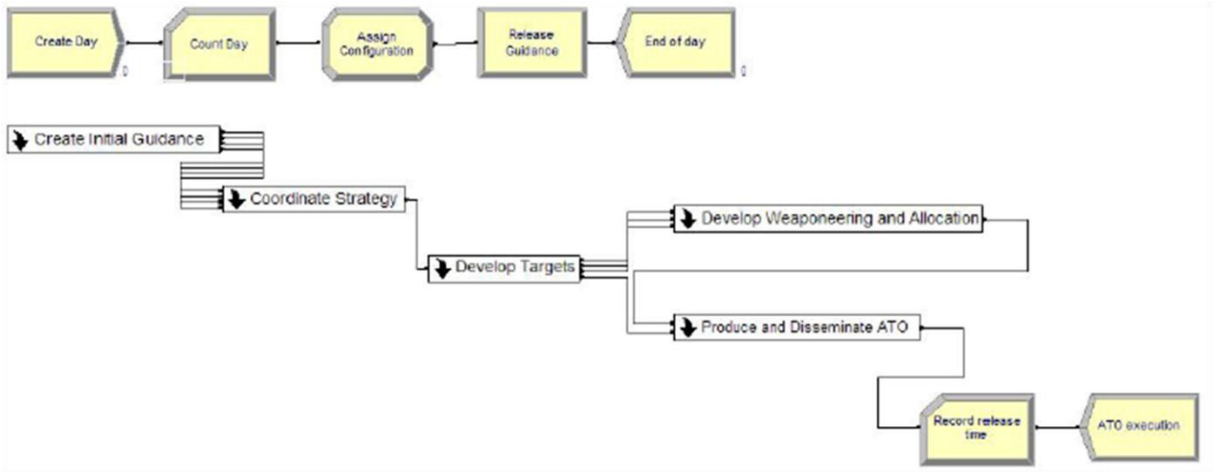

Figure 9. Top Level View of the Arena Implementation of the CPN

5. Comparison of Executable Architecture Methodology

Each type of methodology has a different level of granularity for the battlefield functions. Type 1 uses a federation of BPM and other models for battlefield functions such as combat, communication, logistics, etc. The functional models have been developed independently of BPM. As discussed here, the linking of BPM and the functional models should follow HLA Compliance, mapping requirements, allocation of mission thread activities to federates, and additional programming needed to implement the technical interactions among the models, etc. There remain technical challenges, which when overcome will likely improve the utility of executable architectures.

Types 2 and 3 can include the battlefield functions within executable architectures. They must interact with operational scenarios, including force laydown,

operational tempo, potential Tactics, Techniques and Procedures (TTP) changes, etc.

Type 1 is adequate for a mid and long term period, while Types 2 and 3 are adequate for a short term period.

These are summarized in Table 5.

Table 5. Comparison of Executable Architecture Methodology

Type	Conversion Technique			Form	Battlefield Function	Term
	M_R	D_R	S_R			
1	CPN	BPM	HLA	federation	detailed	Mid & long
2	UML/ SysML	MATLAB/ DEVS	MATLAB/ DEVS	Non-federation	High-level	short
3	CPN/SysML	-	Simulation SW			

6. Recommendations

The development of executable architectures would better be approached incrementally. The first step is to develop the models in high level descriptions down to the totally reusable architecture-based information sets. The second step is to develop the models to a specific documentation required to answer a particular question or solve a problem [10]. The third step is to develop the models in a system engineering level of detail, with enough rigor to inform and support the test/evaluation and M&S communities.

By modeling communications networks in the form of an "as-is" architecture, an executable architecture for cyber security can be created. This is asserted in [12] as follows:

> *One practical example of using executable architectures to support operational planning involves defending against distributed denial of service (DDoS) attacks. A denial of service attack floods a network with so much traffic that legitimate traffic is blocked. This is analogous to jamming a radio network. A distributed DoS attack is one that is launched from many stations instead of a single station. (Mirkovic and Reiher 2004) classify DDoS defense mechanisms as preventive, reactive, cooperation*

degree and deployment location. An executable architecture can be used to evaluate each type of mechanism. One prevention strategy is to place ""forward deployed"" firewalls on the outbound ports of the main routers as described in (Chatam 2004). The performance impacts of various firewall configurations and placements are readily displayed though an executable architecture. A typical reactive strategy is to simply reconfigure the network and reroute traffic to a server that is (hopefully) not under a DDoS attack. One autonomous means to mitigate a DDoS attack is to use a dual-queue system, which automatically starts dropping traffic that comes from untrusted hosts at the onset of an attack (Fletcher and Eoff 2004). All of these partial solution strategies to defend against DDoS attacks can be systematically evaluated through an executable architecture.

Simultaneous events are only partially supported in UML2.0. I recommend UML2.0 as M_R, and DEVS as D_R and S_R. This is asserted in [11] as follows:

The DEVS formalism excels at modeling complex discrete event systems. A framework capable of simulating a DEVS model is presented via UML state machines. A set of rules is enumerated for the creation of UML models. Adherence to these rules results in models that are both DEVS and UML compliant. Resultant UML models are executable within DEVS simulation frameworks such as DEVSJAVA. Such an approach to modeling in UML represents a significant improvement over alternative approaches since it enables earlier simulation and verification of a design.

Another reason for choosing the DEVS formalism as a means to M&S is its reusability and composability. To-Be architecture products can be easily developed on the basis of As-Is architecture products with minor changes and additions.

This approach can be applied to determine the contribution of a C2 system or capability to the overall capability of a fighting force.

Reference

1. Ring, Steven J., Lamar, B., Heim, J., Goyette, E. (2005), "Integrated Architecture-Based Portfolio Investment Strategies"(10th ICCRTS), MITRE

2. Beal, Robert J., Hendrix, Jeremy P., McMurray, Garth P., Stewart, William C. (2005). "Executable Architectures and their Application to a Geographically Distributed Air Operations Center"(AFIT/GSE/ENY/05-M03). Air Force Institute of Technology.
3. Behre, Christopher. "DoD Enterprise Architecture Conference Applied Joint Mission Thread." USJFCOM.
4. DeStefano, G. V. (2004). "Agent Based Simulation SEAS Evaluation of DODAF Architecture" (AFIT/GOR/ENS/04-05). Air Force Institute of Technology.
5. Griendling, K., Marvis, D. N. (2011). "Development of a DoDAF-Based Executable Architecting Approach to Analyze System-of-Systems Alternatives." IEEE. Paper #1389, ver. 1.
6. Mittal, Saurabh. (2006). "Extending DoDAF to Allow Integrated DEVS-Based Modeling and Simulation." JDMS, vol. 3, Issue 2. pp. 95-123.
7. Pawlowski, T., Barr, P. C., Ring, S. J. (2004). "Applying Executable Architectures to Support Dynamic Analysis of C2 Systems." Command and Control Research and Technology Symposium. San Diego. (June 15).
8. Ruegger, K. L. (2008). "Architecting a Net-Centric Operations System of Systems for Multi-Domain Awareness." Master's Thesis, Naval Postgraduate School, CA.
9. Ryan, M. H., Hanoka, W. J. (2012). "A Study of Executable Model Based Systems Engineering From DoDAF Using Simulink" (AFIT/GSE/ENV/12-S05DL). Air Force Institute of Technology.
10. Zinn, A. W. (2004). "The Use of Integrated Architectures to Support Agent Based Simulation: An Initial Investigation" (AFIT/GSE/ENY/04-M01). Air Force Institute of Technology.
11. Mooney, J. (2008). "DEVS/UML - A Framework for Simulatable UML Models." Master's Thesis, Arizona State University.
12. Hamilton, Jr., J. A. (2013). "Architecture-Based Network Simulation for Cyber Security." Proceedings of the 2013 Winter Simulation Conference.