24th Annual INCOSE International Symposium (IS2014)
Las Vegas, NV, June 30 – July 3, 2014

Synthesizing and Specifying Architectures for System of Systems

C. Robert Kenley, Timothy M. Dannenhoffer, Paul C. Wood, and Daniel A. DeLaurentis
Purdue University
School of Aeronautics and Astronautics
Center for Integrated Systems in Aerospace
Neil Armstrong Hall of Engineering
701 W. Stadium Ave., West Lafayette, IN, 47907-2045
kenley, tdannenh, pwood, and ddelaure@purdue.edu

Abstract. Based on our experience with defining and evaluating system of systems and architectures using agent-based modeling, we describe the steps that we have employed in the context of a well-known process for defining system architectures and identify which steps of the process capture the unique characteristics of a system of systems. We describe a particular method that we developed to automate the generation of the communications links needed for executable simulation models when evaluating a large architectural design space, and we review model-based systems engineering methods that are applicable to specifying systems of systems and that support developing executable agent-based simulation models.

Introduction

The typical reaction of many long-time practitioners of systems engineering after hearing about the challenges in systems of systems is to ask, "What is it that I should be doing for systems of systems that is different from what I always have done when engineering a system?" In this paper, we provide our answer to this question by

1. relating the practices that we have employed for generating and evaluating C2BMC architectures in the context of a well-known process for synthesizing system architectures, and
2. reviewing applicable model-based systems engineering methods such as UML and Petri nets that are relevant to specifying architectures for systems of systems and showing how they apply to our C2BMC example.

The Missile Defense C2BMC System of Systems Problem

The US Missile Defense Agency is responsible for developing an integrated Ballistic Missile Defense System (BMDS), which is a system of systems that links land-, sea-, air-, and space-based assets to defend the United States, its friends and allies, and its deployed forces from ballistic missile attack. A Command and Control, Battle Management, and Communication (C2BMC) capability is a critical set of functionalities that link the various individual systems into a system of systems, ensuring the highest capability for protection against all types of ballistic missile threats in all regions of the world and in all phases of flight. It is an "acknowledged" system of systems (Dahmann and Baldwin 2008) in that there are objectives, management, funding, and authority for the system of systems; however, systems

retain their own management, funding, and authority in parallel with the system-of-systems effort.

We have been developing, refining, and analyzing candidate system-of-systems architectures for the BMDS C2BMC capability that employ worldwide, on-demand, secure network services that permit global employment of multiple independently capable weapon and sensor systems. The architectures are required to operate both strategically and regionally within the US force structure and potentially to share data and resources with allied forces.

A Reference Process for Synthesizing an Architecture

The distinguishing characteristics for systems of systems are operational and managerial independence (Maier 1998). This paper focuses on the methods used to define operational architectures for systems of systems that are able to operate independently when deployed. Managerial independence is the context within which the technical methods described in this paper are employed. Those interested in the managerial aspects of acquiring and deploying systems of systems should refer to Dahmann et al. (2011), which defines a "management architecture" called the wave model, and to Acheson et al. (2012), which describes an approach to analyzing the management architecture based on the wave model.

We applied a well-known process for synthesizing system architectures that was originally articulated by Levis and Wagenhals (2000) and has been incorporated into Buede's textbook with the addition of a step for defining the allocated architecture (2009).

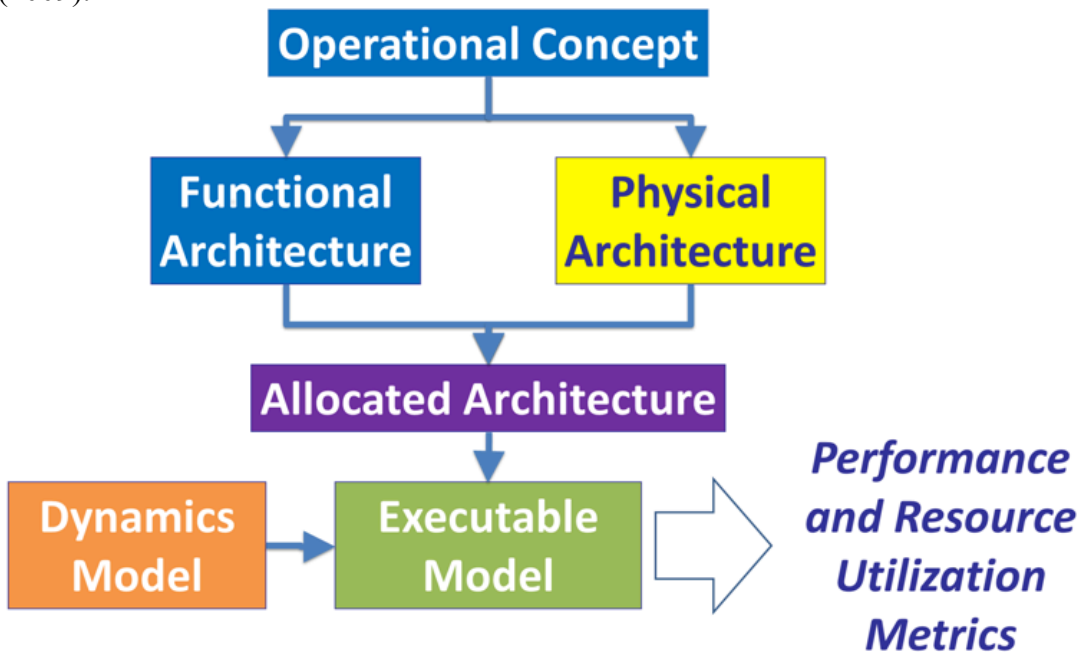

Figure 1. Reference process for synthesizing SoS architectures

Figure 1 shows the process that we applied, which is a combination of the process described by Levis and Wagenhals and by Buede. An overall operational concept for the system of systems is defined that captures the mission requirements and usage scenarios that are employed to develop a functional architecture, which is a set of functions that are arranged to execute in a way that facilitates achieving the mission objectives. Available technologies are reviewed to develop a physical architecture, which is the collection of physical capabilities that perform sensing, data processing, communications, and target engagements, and that are arranged in a communications

network. The allocated architecture is the result of placing the requirement to execute functional capabilities on the physical components. Multiple possible allocated architectures can be defined from a functional and physical architecture. It is the primary goal of systems of systems architecting to define feasible SoS architectures; to evaluate the ability of the architectures to satisfy mission requirements and the resources required to procure and operate the SoS. The dynamics model describes the dynamic behavior of the allocated architecture. It is the basis for developing an executable model that simulates the behavior of the allocated architecture, which is used primarily to obtain the performance and resource utilization metrics for the operational SoS. In theory, the executable model could also be used for determining resource utilization metrics such as procurement and operational costs, procurement schedules, and other SoS-wide measures such as complexity; however, in practice separate models are usually developed to obtain these metrics.

Applying the Reference Process to Missile Defense

A US National Research Council (2012, 132-145) report describes a high-level operational concept for missile defense identifies seven systems that participate in the missile defense system of systems: command level (national command authority), intelligence, surveillance sensors, combatant commander, battle manager, fire unit, and tracking and discrimination. Within each of the systems identified by the National Research Council, there may be multiple, heterogeneous physical systems in their own right such as land-based, airborne, and sea-based interceptors, each with their own fire control capabilities, and land-based, airborne, space-based, and sea-based sensors that collect tracking and discrimination data. The C2BMC capabilities within this system of systems encompass the functionalities listed for the battle manager, for tracking and discrimination, and for decision and control portions of the fire unit, i.e., selecting assets and committing interceptors.

Figure 2 shows a high-level functional architecture that we developed for implementing C2BMC capabilities that is consistent with the operational concept and supports meeting the overall mission to negate hostile threats and minimize collateral damage. It shows a flow of information for tracking, discrimination, and typing of targets, information for sensor tasking and interceptor engagement, and information for kill assessment.

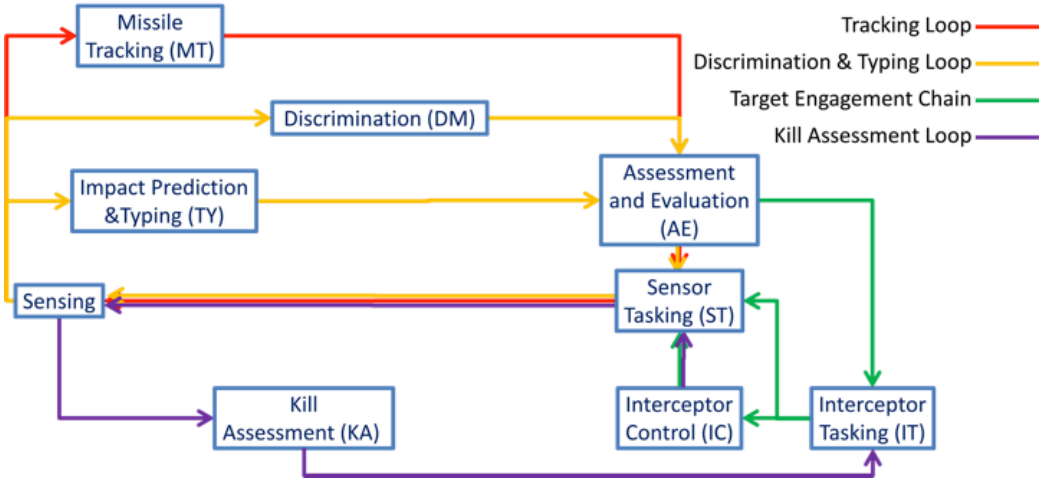

Figure 2. Control and information flow for the functional C2BMC architecture derived from the operational concept

The entities that we use to define the physical architecture of the C2BMC capabilities belong to one of two classes, platforms, and communications links. Platforms are defined by their initial location and the trajectory of their location over time (in case they have the capability to move), their data processing resources, and the communication links by which they are able to interface. The communication links that connect the platforms to form the physical architecture are defined by their communication protocols (e.g., UDP or TCP) and their capacity or bandwidth. A summary of the types of platforms and communication links used in generating C2BMC physical architectures is shown in Table 1.

Table 1. C2BMC Physical Architecture: Platforms and Communication Links

Class	Physical Entity	Relevant Attributes
Platform	• Aircraft • Satellite • Ground Station • C2 Node • Interceptor	• Location and Trajectory • Processing Resources • Interfaces to Communications Links
Communications Link	• Satellite • Wireless • Fiber	• Communication Protocols and Capacities

The functional and physical architectures for the C2BMC system of systems are indistinguishable from architectures that would be developed if a systems engineer were developing architectures for a C2BMC system that is managed and operated by a single entity. It is in defining the allocated architecture that the distinguishing trait of operational independence is exhibited.

Two views provide insight into operational independence: the perspective of a single platform and the perspective of the entire set of allocated functional capabilities of the system of systems.

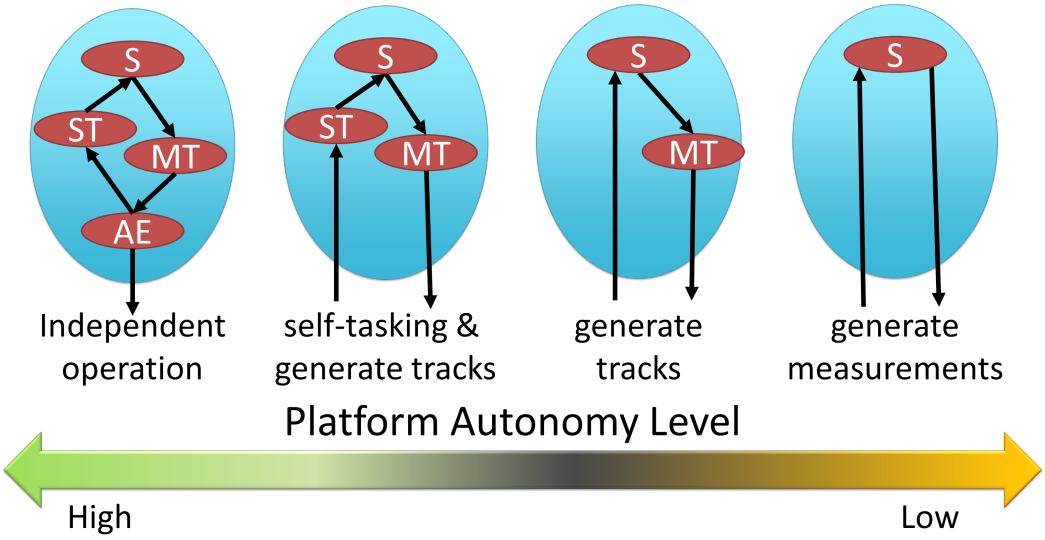

Figure 3. Example of options for allocating functions to a platform with a tracking sensor

From the perspective of a single platform, the concept of autonomy level aids in characterizing the levels of operational independence that a platform has for a given

allocated architecture. Figure 3 shows how a platform with a tracking sensor can range from a high level of autonomy to a low level of autonomy. At the highest level of autonomy, the platform operates independently with its own functionality for sensor tasking (ST), sensing (S), missile tracking (MT), and assessment and evaluation (AE). A self-tasking platform receives priorities from the assessment and evaluation functionality of other systems. If more than one external system is providing an uncoordinated set of priorities to the platform, it is considered to be operationally independent; however, if only one system is providing priorities to the self-tasking platform, it is not operationally independent. The two types of platforms that generate tracks and generate measurements receive commands from external systems to direct their sensing and, similar to self-tasking platforms, are operationally independent if they receive an uncoordinated set of sensing commands from multiple systems.

From the perspective of the entire set of capabilities, the concept of centralization vs. decentralization of functionality is another aid to characterize the level of operational independence. In a system of systems, a functional capability can be allocated to multiple physical entities with the functionality performed by each physical entity being executed according to the inputs (matter, energy, and information) that it receives from external entities and any local energy, matter, or information resident in the physical entity.

Table 2, which is adapted from Mane and DeLaurentis (2012), shows an example possible allocations of missile tracking, assessment and evaluation, and sensor tasking functionality that complete the tracking loop in the functional C2BMC architecture presented in Figure 2 to physical architectures that have multiple sensors for collecting data and a single command and control (C2) node. In a centralized architecture, all functionality is allocated to a single C2 node, and none of the sensors is operationally independent of the C2 node. A decentralized architecture has all functionality allocated to each of the multiple sensors in the architecture, and all of the sensors are operationally independent of the C2 node. For the architecture with centralized tracking and prioritization, each sensor may have its own sensor tasking or its own missile tracking functionality that would enable it to be operationally independent. A central managerial authority could prescribe the sensor tasking and missile tracking functionality at each sensor to ensure that operational independence would not be present. Similarly, for the architecture with centralized tracking, operational independence is determined by the entity that prescribes the missile tracking functionality.

Table 2. Example of Centralized vs. Decentralized Tracking

	Location of Functionality According to Architecture Centralization			
Functions	Centralized	Centralized Tracking and Prioritization	Centralized Tracking	Decentralized
Missile Tracking (MT)	C2	C2	C2	Sensors
Assessment and Evaluation (AE)	C2	C2	Sensors	Sensors
Sensor Tasking (ST)	C2	Sensors	Sensors	Sensors
Sensing (S)	Sensors	Sensors	Sensors	Sensors

In our dynamics models of the allocated systems of systems architecture, each function is modeled as an agent (DeLaurentis 2005). The agents interact as a network of multiple independent entities. Agents exhibit their dynamic behavior by performing functions allocated to physical platforms and by communicating via the links defined for the physical architecture's network. An individual agent, as shown in Figure 4 (adapted from Joslyn and Rocha (2000)), has its own initial set of objectives and desires, which controls its behavior. It also has an initial set of knowledge, beliefs, and information, which may include information about the physical properties and energetic state of the platform on which it resides. The agent then decides on actions to be taken and executes those actions. The actions may transfer matter, energy, or information to the environment, which includes other agents that it interacts with, as well as contextual entities such as targets and the ambient conditions. The agent receives inputs in the form of matter, energy, and information from the environment and updates its knowledge, beliefs, and other information, which may indirectly update its objectives and desires.
The executable model for a C2BMC architecture was implemented using DAF (the Discrete Agent Framework, (Mour et al. 2013)). DAF is a MATLAB-based framework that provides the underlying infrastructure for agent-based simulation that moves messages around and maintains the simulation environment (locations, time, etc.). We developed executable MATLAB code for all of our dynamics models of the agents to simulate their functionality and operational effects, such as computational and communications latency that are a consequence of the physical properties of the allocated architecture.

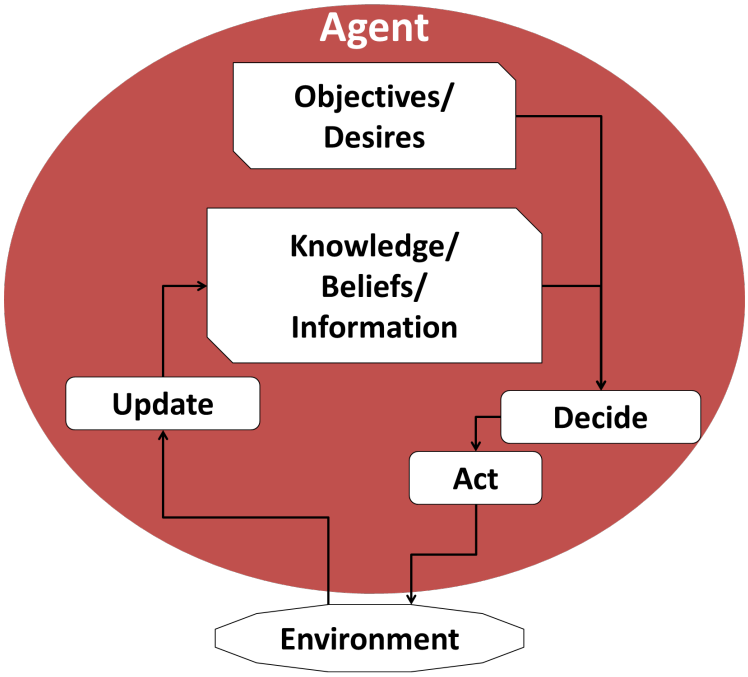

Figure 4. The dynamics model for an agent

Generating Communication Architectures

As Maier (1998) pointed out, the architecture for a system of systems is defined by the interfaces and for those that exchange information at the interfaces, such as C2BMC systems of systems, the communications network describes these interfaces. DAF was created as a flexible framework for testing the functionality of agent configurations and evaluating the allocation of this functionality in a

system-of-systems context, and part of this flexibility comes from defining network connections between physical platforms. DAF achieves this goal by placing complete control of agent allocation and network connections in the architect's hands with explicit definitions of agent arrangements and interconnects. While this approach provides the ultimate flexibility, it can create an extra burden on the developer to manage communication agents that link functionality across platforms. We implemented a model builder in DAF for defining communication networks in C2BMC architectures to reduce this burden.

Designers have to specify the connections for each particular functional allocation of C2BMC components. The model builder separates the connections into two parts, a physical network part and an agent data path part. The first is specified as a list of communication agents that define the point-to-point links available as physical assets. The architect then specifies which agents are to be connected, ignoring the complexities in the network path. DAF will automatically create the communication path between any two agents allowing the developer to ignore network infrastructure when defining logical agent-to-agent connections. Previous versions of the model builder required an architect to maintain the network links in addition to the agent-to-agent connections. This created a bookkeeping burden and required the developer to consider independent layers simultaneously to create a functioning architecture. Errors in this process were somewhat common and can be difficult to trace, and the updated model makes it much easier to successfully create architectures.

The automated process requires a few assumptions and a link allocation algorithm. The model builder represents each architecture by a series of architecture design variables rather than an explicit list of connections. The only input required from the architect for this representation is the specific sensor laydown to be used in an architecture. The tool automatically connects the sensor platforms to ground stations, based on the assumption that each ground station is connected to only a single type of sensor. The agents are then automatically wired together in the correct way and on the correct ports based on the architecture variables for the selected architecture. A physical path between two logical agents must be created, and the approach taken is to use a shortest path algorithm. The distance can be defined in several ways, but the assumption made in the implementation is that the least number of network links should be traversed to make a connection. An alternative would be to define distance as latency in the system, which in the case of C2BMC systems would make routes favor fiber connections over lower speed satellite links. The network paths are defined in a static route table that is cached at initialization time for speed. Future improvements may make this dynamic so that re-routing around congested or disabled links becomes a simple process.

By automating the creation of these architectures for various sensor numbers and types, the tool allows for performing more efficient architectural analyses over a large design space. For BMDS, we have used the tool to investigate a large range in the number of sensors on different platforms across the full spectrum of sensor platform autonomy.

Model-Based Methods Applicable to Specifying Architectures for System of Systems

The specification of architectures for system of systems using model-based systems engineering methods should support dynamics models and executable models that account for

- actions of operationally independent systems that are modeled as agents, and
- the interactions of concurrent, asynchronous activities undertaken by operationally independent systems.

In this section, we review the key literature on proposed approaches to applying of UML (and by extension, SysML) and applying Petri nets to support the goals for dynamics modeling and for developing executable models.

Park, Kim, and Lee (2000) propose a comprehensive approach to specifying agent-based modeling using UML 1.1. The have two classes of models, intra-agent models and inter-agent models. The intra-agent models include the goal model (object model of a goal hierarchy), the belief model (object model of beliefs and external message protocols), the plan model (updates beliefs; and determines actions to take and messages to send), and the capability model (logic for actions to be taken by the agent). The inter-agent models include the agent mobile model (how an agent coordinates its actions to perform a task with other agents; it assumes a coordinator agent) and the agent communication model (how message exchanged between agents including sequence diagram of agent actions and messages). Their example that they use to illustrate their approach does not assume complete autonomy among the agents nor does it assume concurrency; however, their method is applicable to completely autonomous agents and the ability to handle concurrency was addressed by the release of UML 2.0.

Before the release of UML 2.0, several approaches were developed to convert specifications of systems from various modeling languages to Petri nets that are executable models for simulating the interactions of concurrent, asynchronous activities. Peleg, Yeh, and Altman (2002) describe how to map a business-process workflow model of a biological system to a Petri net. Wagenhals, Haider, and Levis (2003) converted the UML 1.3 specification C4ISR to a colored Petri net.

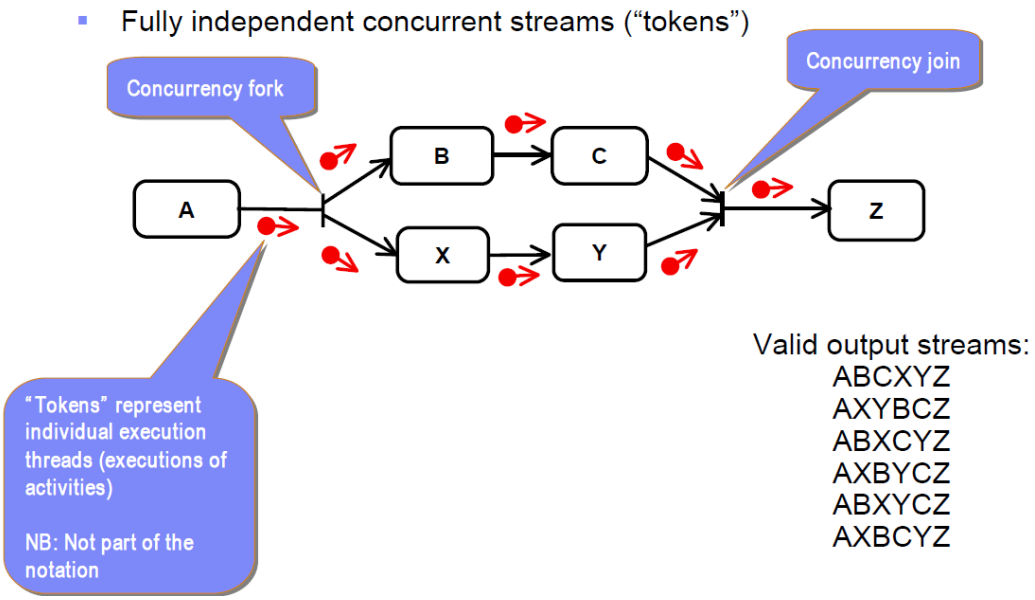

Figure 5. UML 2.0 extended concurrency model from Quatrani (2005)

When UML 2.0 was released in July 2005, it used a Petri-like semantics to allow for concurrency that include tokens, which are a feature of Petri nets, as shown in Figure 5. Störrle (2005) provides a detailed description of mapping UML 2.0 activity diagrams to colored Petri nets, and Staines (2008) describes how to maps UML 2.0 activity diagrams to a Fundamental Modeling Concepts (Knöpfel, Gröne, and Tabeling 2006) Petri net diagram. Sinclair (2009) applies hierarchical and timed

colored Petri nets to modeling and simulation of system of systems and proposes to extend UML in a way that highlights its underlying Petri-like features and adds explicit constructs for hierarchical and timed colored Petri nets.
Bauer and Odell (2005, 156) indicate that "UML has no 'off-the-shelf' constructs to express: goals, agent, groups, multicasting, generative functions, such as cloning, birthing, reproduction, parasitism and symbiosis, emergent phenomena, and many other nature-based constructs that are helpful for representing agent structures." Nonetheless, there are many examples of using UML to define goals, agent, groups, and multicasting that can be followed to specify the systems within a system of systems as agents, and there is adequate guidance on modeling the dynamics of concurrent, asynchronous activities using UML activity diagrams, and converting them to Petri nets to create an executable model.

Specifying C2BMC Tracking Architectures

Figure 6 uses a UML activity diagram to specify a completely centralized tracking architecture that has two sensors (S1 and S2) and allocates missile tracking (MT), assessment and evaluation (AE), and sensor tracking (ST) to the C2 node. The diagram specifies that each platform (S1, S2, and C2) iteratively executes its assigned functionality and that the inputs are <<optional>>, i.e., performing an iteration proceeds according each platform's internal schedule rather than waiting for external inputs to trigger an iteration. Inputs that arrive during an iteration are processed as inputs in the subsequent scheduled iteration.

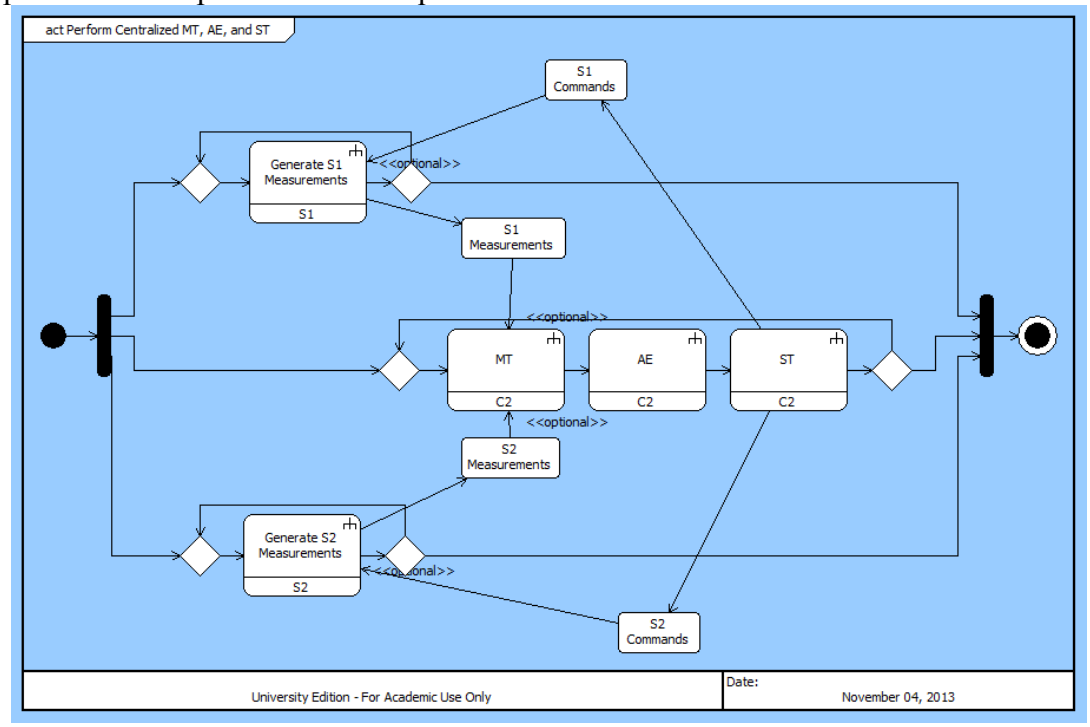

Figure 6. UML Activity Diagram for Completely Centralized Tracking Architecture

Figure 7 follows the approach of Park, Kim, and Lee (2000) to specify the intra-agent dynamics model of a generic agent. It defines functions that

- update the knowledge, beliefs, and information of the agent using inputs from its environment;
- decide on actions that achieve its objectives and desires;
- and take action that produces outputs that affect its environment.

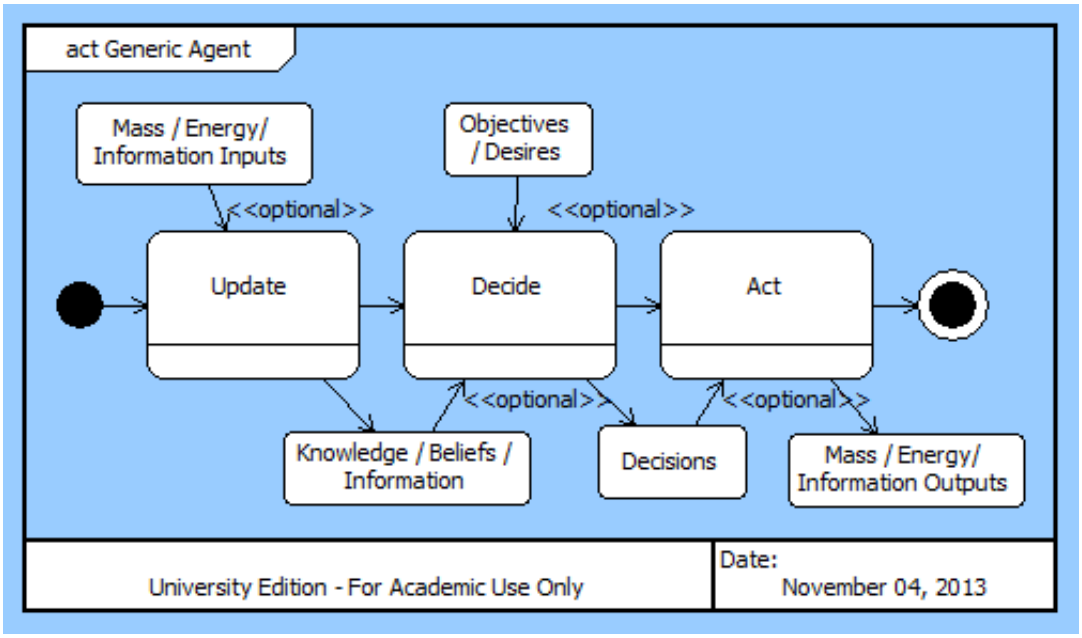

Figure 7 UML Activity Diagram for Generic Agent

Figure 8 applies the pattern from Figure 7 to specify the activity diagram for a missile-tracking agent. Missile tracking incorporates sensor measurements into its tracking database and makes a series of decisions based on parameters that specify its objectives. The series of decisions conclude with a decision on which tracks (labeled as "Firm Tracks" in the diagram) it will forward to assessment and evaluation. This pattern also was used to specify AE, ST, and sensor measurement agents.

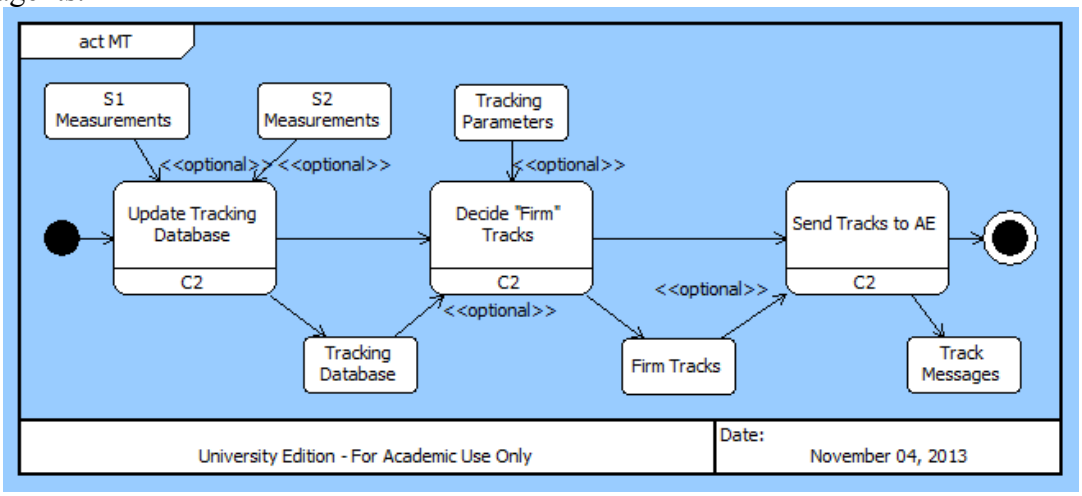

Figure 8. UML Activity Diagram for a Missile Tracking Agent (MT)

Figure 9 specifies an alternative tracking architecture that allocates only missile tracking to the C2 node and allocates separate instances of assessment and evaluation (AE1 and AE2) and sensor tasking (ST1 and ST2) to each sensor. The feedback from the C2 node to the sensors is specified as track messages generated by missile tracking. If this feedback were not specified, each sensor would be allowed more autonomy and would consider only the measurements generated locally to make its AE and ST decisions.

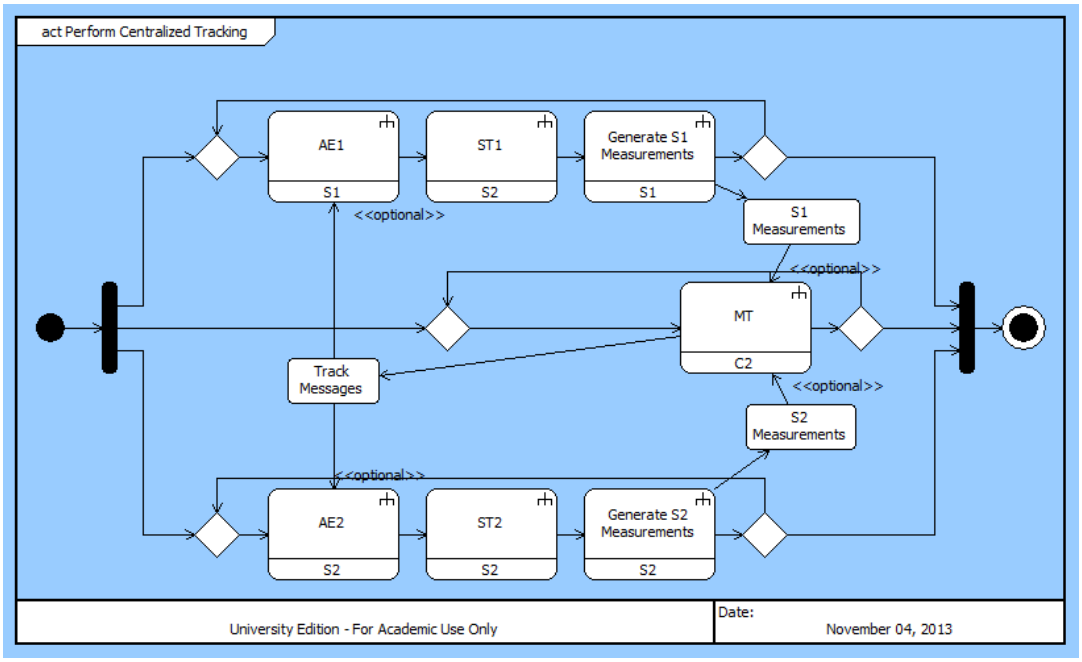

Figure 9. Activity Diagram for Centralized Missile Tracking

Conclusions and Future Directions

Synthesizing and specifying architectures for system of systems can use well-known processes and methods for model-based systems engineering with some adaptations. These adaptations are necessary to capture the operational independence of the constituent systems that derives from the autonomous, concurrent, asynchronous actions of the constituent systems. In our experience, the process for synthesizing architectures can follow the same top-level processes used for centrally operated systems. We believe that the dynamic modeling of a system of systems as a collection of autonomous agents is well suited for capturing the emergent behavior that derives from complex interactions of systems of systems.

Because the architecture of a system of systems is essentially its communications network, it is wise to develop methods that ease the burden of manually synthesizing network architectures when generating a large design space of allocated architectures.

UML 2.0 (and by extension, SysML) is able to specify the concurrent, asynchronous actions that give rise to the complex interactions of systems of systems. A pattern for agent-based models for the dynamics can be created in UML, and it can be used to specify the independently operating constituent systems within a system of systems. Due to its Petri-like syntax, UML 2.0 provides a specification language that allows for developing executable Petri nets for simulating the behaviors and exploring the design space for a system-of-systems architecture. In the future, we will evaluate Petri net tools that can enhance our toolset that we use to synthesize architectures.

Acknowledgement

This publication was developed under work supported by the US Missile Defense Agency (MDA) under contract No. HQ0147-10-C-6001. It has been reviewed by MDA and approved for public release (13-MDA-7638, 14 December 13). The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or

implied, of the US Missile Defense Agency. The US Missile Defense Agency does not endorse any products or commercial services mentioned in this publication.

References

Acheson, Paulette, Louis Pape, Cihan Dagli, Nil Kilicay-Ergin, John Columbi, and Khaled Haris. 2012. "Understanding System of Systems Development Using an Agent- Based Wave Model." *Procedia Computer Science* no. 12 (0):21-30. doi: http://dx.doi.org/10.1016/j.procs.2012.09.024.

Buede, Dennis M. 2009. *The engineering design of systems: models and methods*. Hoboken, US-NJ: Wiley.

Dahmann, J. S., and K. J. Baldwin. 2008. Understanding the Current State of US Defense Systems of Systems and the Implications for Systems Engineering. Paper read at Systems Conference, 2008 2nd Annual IEEE, 7-10 April 2008.

Dahmann, Judith, George Rebovich, JoAnn Lane, Ralph Lowry, and Kristen Baldwin. 2011. An Implementers' View of Systems Engineering for Systems of Systems. Paper read at Proceedings of IEEE International Systems Conference, at Montreal, Quebec.

DeLaurentis, Daniel 2005. "Understanding Transportation as a System-of-Systems Design Problem." In *43rd AIAA Aerospace Sciences Meeting and Exhibit*. American Institute of Aeronautics and Astronautics.

Joslyn, Cliff, and Luis M Rocha. 2000. Towards semiotic agent-based models of socio-technical organizations. Paper read at AI, Simulation and Planning in High Autonomy Systems (AIS 2000) Conference, at Tucson, US-AZ.

Knöpfel, Andreas, Bernhard Gröne, and Peter Tabeling. 2006. *Fundamental modeling concepts: effective communication of IT systems*. Hoboken, US-NJ: J. Wiley & Sons.

Levis, Alexander H., and Lee W. Wagenhals. 2000. "C4ISR architectures: I. Developing a process for C4ISR architecture design." *Systems Engineering* no. 3 (4):225-247. doi: 10.1002/1520-6858(2000)3:4<225::AID-SYS4>3.0.CO;2-#.

Maier, Mark W. 1998. "Architecting principles for systems-of-systems." *Systems Engineering* no. 1 (4):267-284. doi: 10.1002/(sici)1520-6858(1998)1:4<267::aid-sys3>3.0.co;2-d.

Mane, Muharrem, and Daniel DeLaurentis. 2012. Sensor Platform Management Strategies in a Multi-Threat Environment. Paper read at Infotech@Aerospace 2012, 19 - 21 June at Garden Grove, California.

Mour, Ankur, C. Robert Kenley, Navindran Davendralingam, and Daniel DeLaurentis. 2013. Agent-Based Modeling for Systems of Systems. Paper read at 23nd Annual International Symposium of the International Council of Systems Engineering, at Philadelphia, US-PA.

National Research Council. 2012. Making Sense of Ballistic Missile Defense: An Assessment of Concepts and Systems for U.S. Boost-Phase Missile Defense in Comparison to Other Alternatives. The National Academies Press.

Park, Sooyong, Jintae Kim, and Seungyun Lee. 2000. "Agent-oriented software modeling with UML approach." *IEICE TRANSACTIONS on Information and Systems* no. 83 (8):1631-1641.

Peleg, Mor, Iwei Yeh, and Russ B. Altman. 2002. "Modelling biological processes using workflow and Petri Net models." *Bioinformatics* no. 18 (6):825-837. doi: 10.1093/bioinformatics/18.6.825.

Quatrani, Terry. 2005. Introduction to UML 2.0. Paper read at MDA, SOA, and Web Services Workshop, March 21-24, at Orlando, US-FL.

Sinclair, Kirsten. 2009. *The Impact of Petri Nets on System-of-Systems Engineering*, Durham University.

Staines, T. S. 2008. Intuitive Mapping of UML 2 Activity Diagrams into Fundamental Modeling Concept Petri Net Diagrams and Colored Petri Nets. Paper read at 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems.

Störrle, Harald. 2005. "Semantics and Verification of Data Flow in UML 2.0 Activities." *Electronic Notes in Theoretical Computer Science* no. 127 (4):35-52. doi: http://dx.doi.org/10.1016/j.entcs.2004.08.046.

Wagenhals, Lee W., Sajjad Haider, and Alexander H. Levis. 2003. "Synthesizing executable models of object oriented architectures." *Systems Engineering* no. 6 (4):266-300. doi: 10.1002/sys.10049.

Biographies

C. Robert Kenley is a Research Scientist in Purdue's School of Aeronautics and Astronautics in West Lafayette, IN (US). He has over thirty years' experience in industry, academia, and government as a practitioner, consultant, and researcher in systems engineering. He has published papers on systems requirements, technology readiness assessment and forecasting, Bayes nets, applied meteorology, and the impacts of nuclear power plants on employment.

Timothy Dannenhoffer is a Graduate Research Assistant in Purdue's School of Aeronautics and Astronautics. He is a recent graduate from the North Carolina State University in Raleigh, NC (US) where he completed his Bachelor's in Aerospace Engineering and Applied Mathematics. He is currently a member of the Center for Integrated Systems in Aerospace and the System-of-Systems Laboratory. His primary research interests are in the areas of agent-based modeling, design space exploration, and multidisciplinary optimization.

Paul C. Wood is a Graduate Research Assistant in Purdue's School of Electrical and Computer Engineering. He graduated from Tennessee Technological University in Cookeville, TN (US) with a Bachelor's in Electrical Engineering with a focus in digital systems. He is currently a member of the Dependable Computing Systems Laboratory (DCSL) and the Center for Education and Research in Information Assurance and Security (CERIAS). His research interests are in the areas of modeling and simulation for agent-based systems, dependable networks and systems, and information assurance.

Daniel DeLaurentis is an Associate Professor in Purdue's School of Aeronautics and Astronautics. He leads Purdue's Center for Integrated Systems in Aerospace and its largest recent project with the Missile Defense Agency's Enhanced C2BMC program developing agent-based modeling and simulation for development of advanced battle management architectures. His primary research interests are in the areas of problem formulation, modeling and robust system design and control methods for aerospace systems and systems of systems. This includes agent-based modeling, network theory, optimization, and aerospace vehicle modeling. His research is conducted under grants from NASA, FAA, Navy, the DoD Systems Engineering Research Center UARC, and the Missile Defense Agency.