2004 Command and Control Research and Technology Symposium
The Power of Information Age Concepts and Technologies

C2 Assessment Tools & Metrics Track

Applying Executable Architectures to Support Dynamic Analysis of C2 Systems

Tom Pawlowski, PhD
The MITRE Corporation
245 Sedgwick Ave
Ft Leavenworth, KS 66027

Work: 913-684-9139
Fax: 913-684-9166
pawlowst@mitre.org

Paul C. Barr, PhD
The MITRE Corporation
12 Christopher Way
Eatontown, NJ 07724

Work: 732-578-6307
Fax: 732-578-6019
poobarr@mitre.org

Steven J. Ring
The MITRE Corporation
202 Burlington Rd Rte. 62
Bedford, MA 01730

Work: 781-271-8613
Fax: 781-271-7213
sring@mitre.org

Applying Executable Architectures to Support Dynamic Analysis of C2 Systems

Tom Pawlowski, PhD
The MITRE Corporation
245 Sedgwick Ave
Ft Leavenworth, KS 66027

Work: 913-684-9139
Fax: 913-684-9166
pawlowst@mitre.org

Paul C. Barr, PhD
The MITRE Corporation
12 Christopher Way
Eatontown, NJ 07724

Work: 732-578-6307
Fax: 732-578-6019
poobarr@mitre.org

Steven J. Ring
The MITRE Corporation
202 Burlington Rd Rte. 62
Bedford, MA 01730

Work: 781-271-8613
Fax: 781-271-7213
sring@mitre.org

Abstract

This paper describes the development of a methodology to import key products of the DoD Architecture Framework into an executable form to conduct a dynamic analysis of the Command and Control (C2) system or capability represented by the architecture. Dynamic analysis of these models enables a user to assess the impact of change and determine measures of performance and effectiveness.

The research team successfully implemented the methodology in a demonstration that made a three-way link among a business process model, a communications network model, and a combat simulation representing the system's operational environment. We linked these models together via the Runtime Infrastructure (RTI) of the High Level Architecture (HLA). Basic HLA interactions that we established allowed key events in the combat simulation to initiate one or more business processes in the process model. As a business process proceeded in time, the process model requested from the communications model a delay time that represented the time required to pass information through the network from one node to another. We extracted measures of performance and effectiveness from the simulation runs to conduct the dynamic analysis of the system in question.

Background

In a large modern enterprise, a rigorously defined framework is necessary to capture a vision of the "entire system" in all its dimensions and complexity. This includes all aspects of a "system of systems." The Clinger-Cohen Act of 1996 requires all Federal agencies to develop and document Information Technology (IT) architectures by using a framework to guide the descriptions of their architectures. A framework provides the rules, guidance, and basis for developing and presenting architecture descriptions in a uniform and consistent manner and is intended to ensure that the architecture descriptions can be understood, compared and related across organizational boundaries. To accomplish this, a framework defines numerous products to capture specific architectural views. Architecture products are those graphical, textual, and tabular items that are developed in the course of building a given architecture description that describe characteristics pertinent to the architecture's purpose.

The current framework products provide "static" representations of information for their various views. These static products, while capturing enormous amounts of information about the Operational Architecture (OA) and System Architecture (SA), fail to provide a good vehicle for conducting detailed dynamic "behavioral" analysis of how the systems are supposed to interact with each other.

A major issue for all DoD and federal agencies is not only how to accurately document C4ISR or Information Technology capabilities using architectures, but also how to conduct a useful analysis of these capabilities to determine the performance and effectiveness of the systems providing the capability in question. This analysis is currently limited to a static analysis where one can examine the static architecture framework products to examine connectivity and other routine requirements. The current DoD architecture framework provides no clear means to examine these systems in a dynamic fashion as they function in their operational environment.

The methodology that our research team developed began to address this problem by providing a means to import the static architecture products into a dynamic form that can then be executed with a federation of simulations. By extracting key performance and effectiveness measures from the simulation runs, one can conduct a dynamic analysis of these represented systems. Ultimately, this approach can support the Joint Capabilities Integration and Development System (JCIDS) process where future capabilities are defined. It can also support the budgeting and acquisition processes where investment strategies must be applied to determine the most cost effective solutions to meet the approved capability requirements.

Objectives

The objectives of developing executable architectures from DoD-developed static architectures are:

- To determine the contribution of a system or capability to the overall capability of a fighting force.
- To identify blocked resources and provide for alternatives of the systems development as well as those associated with communications and networking.
- To identify bottlenecks in processes and communications networks and estimate optimal command and control activities process times.
- To identify operators in organizations as well as nodes in the communication systems (as networks) that are overloaded and to re-distribute the command and control activities where appropriate.

Approach

Our approach was to use the inherent functionality of the tools and only develop additional tools as necessary to import the static architecture products into the tools with a simulation capability.

We modeled the business processes as queuing problems using a Timed Petri Net. The Timed Petri Net provides a means to examine the delays created by the tasks, processes and activities in the model. This makes it is easier to identify bottlenecks or to compare alternative optimization strategies such as changing the organizational structure, or altering the information system to improve process flows. Time characteristics are flexible so that the user can determine exactly when resources are available. Timed Petri Nets integrate organizational, process and information views in the same model, thereby allowing in-depth analysis of how processes, the organization and the information systems interact.

As activities in one operational node pass information to another activity at another operational node, we want to include any delay time created by the flow of data from one node to the other through the communication network. By modeling the communication systems and their networks, we can determine these delays and incorporate them into the overall processing time for the appropriate mission thread.

The timing of when various business processes occur in relation to one another is dependent on the operational environment in which those processes occur. In our approach, we used a combat simulation to provide the representation of the flow of the battle. This battle flow provides the context and timing that shows when the mission threads (business processes) in the process model are initiated. By realistically representing the initiation of the mission threads, we can identify those situations when there is a contention for a resource or system among two or more mission threads. This contention will require one or more mission threads to wait on the resource or system, thereby increasing the overall processing time for that mission thread.

Figure 1 represents the general approach to apply executable architectures to support dynamic analysis of a system or capability. This approach assumes that Popkin's System Architect is the de facto standard tool within DoD for developing Information Technology architectures. Given this assumption, we were interested in extracting key DoD Architecture Framework products from System Architect. Those key architecture products include Operational View (OV) products (Operational Node Connectivity Diagram (OV-2), Operational Information Exchange Matrix (OV-3), Organizational Relationship Chart (OV-4), and Operational Activity Diagram (OV-5)) and System View (SV) products (Systems Interface Description (SV-1) and Systems Communications Description (SV-2). We then import them into a middleware tool called the Integrated C4ISR Analysis and Management System (ICAMS). ICAMS is then used to generate, in the appropriate format, the data needed to populate the business process modeling tool and the network communications modeling tool. We also use ICAMS to correlate the entities and their relationships in the combat simulation with the same entities and relationships captured in the other two modeling tools. Ultimately, the goal is to use the same source documents (architecture products) to populate all three models (process model, communications network model, and combat simulation). Because we are using a combat simulation that already has its own means of generating entities, it is not practical to implement this part of the methodology for the combat simulation.

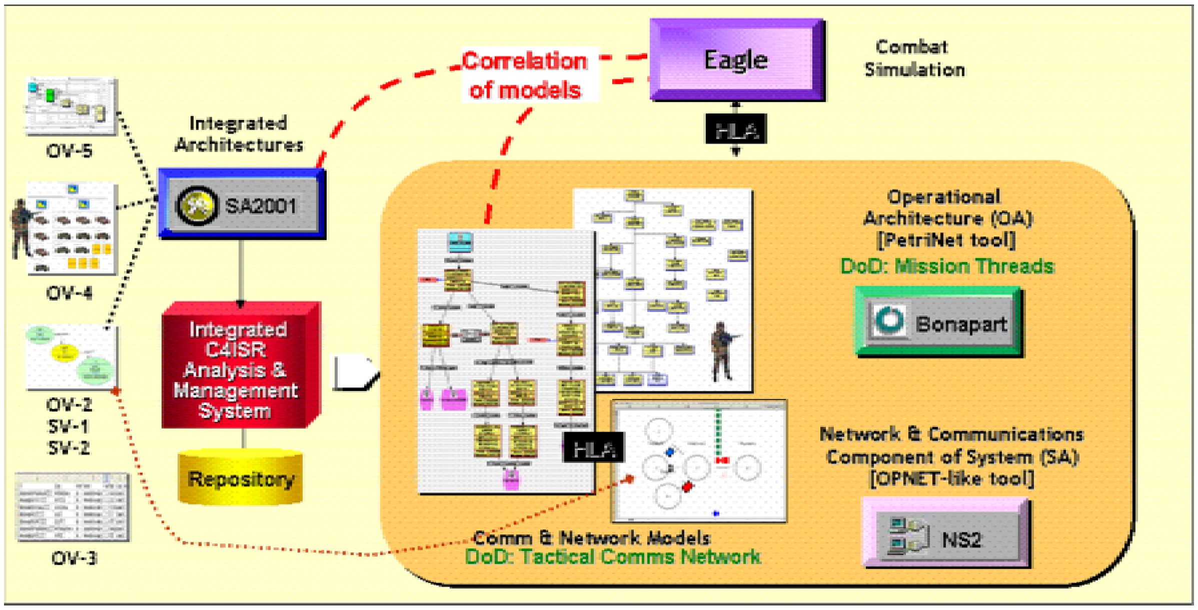

Correlation of models
Eagle
Combat Simulation
HLA
OV-5
Integrated Architectures
SA2001
OV-4
Integrated C4ISR Analysis & Management System
OV-2
SV-1
SV-2
Repository
OV-3
Operational Architecture (OA) [PetriNet tool]
DoD: Mission Threads
Bonapart
HLA
Network & Communications Component of System (SA) [OPNET-like tool]
NS2
Comm & Network Models
DoD: Tactical Comms Network

Figure 1. General Approach

When the models are fully populated in the tools, a mapping must occur that clearly identifies how the entities in one model relate to entities in another model. In most cases, the mapping will be straight forward: we map a unit or staff cell explicitly represented in one model to the explicit representation of that unit or staff cell in another model. However, in some cases the mapping will be less clear. For example, a staff cell that is explicitly represented in the process model might only be implicitly represented by a unit headquarters in the combat simulation. The unit headquarters could actually represent a number of staff cells in the mapping process.

In addition to mapping the entities, a mapping of relationships is necessary to establish how events in one model are related to events in another model. In the combat simulation we can expect certain events to trigger specific processes in the process model. This type of relationship needs to be identified prior to running the simulations so that when the event occurs, the combat simulation sends the appropriate interaction to initiate the process.

When the mapping is complete, we run the simulations as a time-managed HLA federation. However, before this occurs, we need to identify measures of performance (MOP) and measures of effectiveness (MOE). We can extract some measures directly from a model. We calculate other measures based on output from multiple models. During the simulation run, we collect from each of the models data that will support our MOP and MOE. An important part of this process is determining which data in which model will provide the measures of interest. From the combat simulation we must calculate measures of force effectiveness (MOFE) to support any analysis of the impact of the system on the unit's mission accomplishment.

Procedure

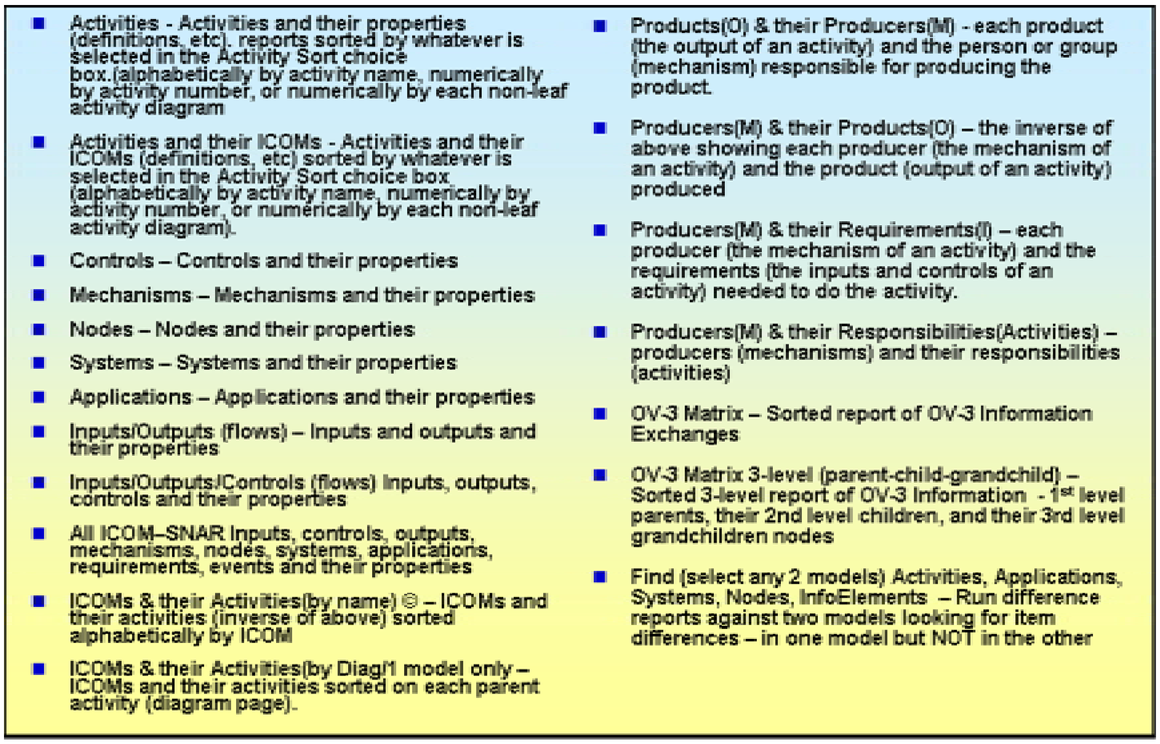
Applying the concept of an integrated architecture, we developed the procedure as:

1. Create the appropriate architecture framework products (OV-2, OV-3, and OV-5) in the framework modeling tool (System Architect).
2. Export the OV-2, OV-3, OV-5 data into the middleware repository (ICAMS).
3. Conduct the static analysis and reconcile the terms and definitions in the architecture model.
4. Export the data from the repository (ICAMS) and import into the communications network simulation tool and business process modeling tool.
5. Complete the process model and communications network model.
6. Test the process model and network model linked together.
7. Integrate the combat simulation with the process model and network model.
8. Test the federation consisting of the combat simulation, process model, and network model linked together through HLA.
9. Conduct the dynamic analysis applying previously defined MOP, MOE, and MOFE.

To begin, we apply the Activity-Based Methodology to generate the initial architecture framework products. The Activity-Based Methodology was developed under the direction of the Office of the Assistant Secretary of Defense for Networks and Information Integration (OASD-NII) as a means of standardizing the approach to the development of architecture products that produce an integrated architecture. Ring, et al., provides a detailed description of the methodology. Under the methodology, we assume that developers of the architecture products are using Popkin's System Architect as the primary tool to generate the products.

After the initial architecture products are developed, we export them from System Architect into the middleware repository called Integrated C4ISR Analysis and Management System (ICAMS). ICAMS is a Microsoft Access-based, MITRE developed tool that performs several functions. First, it understands the formats of the tools we're using and allows us to move data from one tool to another by changing the format of the data file. Second, ICAMS allows us to examine multiple architectures at the same time, so that we can compare terminology, definitions, and conduct any ‘data cleaning’ that might be necessary to ensure any required consistency across architectures. Third, it allows us to conduct a static analysis of the architecture that we’ve imported from System Architect.

With the static analysis we can examine the architecture products and ensure that nodes are activities are properly configured and connected as well as ensuring consistency in terminology and definitions. Figure 2 describes some of the reports that ICAMS can generate to support our static analysis.



- Activities - Activities and their properties (definitions, etc). reports sorted by whatever is selected in the Activity Sort choice box.(alphabetically by activity name, numerically by activity number, or numerically by each non-leaf activity diagram
- Activities and their ICOMs - Activities and their ICOMs (definitions, etc) sorted by whatever is selected in the Activity Sort choice box (alphabetically by activity name, numerically by activity number, or numerically by each non-leaf activity diagram).
- Controls – Controls and their properties
- Mechanisms – Mechanisms and their properties
- Nodes – Nodes and their properties
- Systems – Systems and their properties
- Applications – Applications and their properties
- Inputs/Outputs (flows) – Inputs and outputs and their properties
- Inputs/Outputs/Controls (flows) Inputs, outputs, controls and their properties
- All ICOM–SNAR Inputs, controls, outputs, mechanisms, nodes, systems, applications, requirements, events and their properties
- ICOMs & their Activities(by name) ☺ – ICOMs and their activities (inverse of above) sorted alphabetically by ICOM
- ICOMs & their Activities(by Diag/1 model only – ICOMs and their activities sorted on each parent activity (diagram page).
- Products(O) & their Producers(M) - each product (the output of an activity) and the person or group (mechanism) responsible for producing the product.
- Producers(M) & their Products(O) – the inverse of above showing each producer (the mechanism of an activity) and the product (output of an activity) produced
- Producers(M) & their Requirements(I) – each producer (the mechanism of an activity) and the requirements (the inputs and controls of an activity) needed to do the activity.
- Producers(M) & their Responsibilities(Activities) – producers (mechanisms) and their responsibilities (activities)
- OV-3 Matrix – Sorted report of OV-3 Information Exchanges
- OV-3 Matrix 3-level (parent-child-grandchild) – Sorted 3-level report of OV-3 Information - 1st level parents, their 2nd level children, and their 3rd level grandchildren nodes
- Find (select any 2 models) Activities, Applications, Systems, Nodes, InfoElements – Run difference reports against two models looking for item differences – in one model but NOT in the other

Figure 2. Reports for Static Analysis

After we complete the static analysis and modify any architecture data to correct any problems, we use ICAMS to generate an import file in a form that the business process modeling tool can read. This may be an MS Access database, a text file, or an MS Excel spreadsheet. Whatever the format, it is important for ICAMS to generate the correct data to create the appropriate objects in the modeling tool. In the case of tools we used for this project, ICAMS generates an MS Access database following the form of a Bonapart template. This template captures the appropriate architecture data in the object form used by Bonapart. It also establishes the appropriate relationships among the objects. These relationships include identifying what Role (job title in Bonapart) performs which Activity (task).

Currently ICAMS lacks the capability to completely import the entire architecture into the process model or communications network model to provide a fully executable architecture. So, some post-importing work is necessary. Typically in the process model, this entails configuring the mission threads to more accurately reflect the key business processes of the organization and documenting some of the attributes not available in the architectures but necessary to execute it as a simulation. These attributes include those needed to support the mapping of entities between models as well as those that support the interactions between the models.

It is important that after you have completed the modifications to the process and communications models to support mappings and interactions, you test the links to ensure that the federation and its interactions are performing as required. Testing will consist of runs of the federation with artificial data, which would normally come from the combat simulation, to initialize the mission threads in the process model. The testing should ensure that the roles and nodes in the process model are correctly mapped to the nodes in the network model. When you are satisfied that the two models are working properly together, you can integrate the combat simulation with them.

There are two main actions that you must perform to integrate the combat simulation with the other two models. First, you must map the key events in the combat simulation with the appropriate mission thread(s) in the process model. To do this, you must determine what key events would cause each mission thread to begin. Likewise, there is a corresponding mapping from the end of a mission thread back to one or more events in the combat simulation. This mapping identifies what should happen in the scenario in the combat simulation because of the processing done in the mission thread. As an example, this may include timely effects on a target due to speedy sensor to shooter mission thread.

Second, you must identify what entity attributes in the combat simulation need to be updated in the process model or network model. The primary attributes of concern are those related to the nodes in the network model. Whichever entities in the combat simulation represent nodes in the network model, you should ensure that the status and locations of those entities are passed to the network model through an HLA object attribute. Other attributes of concern are the status of entities in the combat simulation that represent roles performing activities in the process model. If the status of the entity changes, such as the entity is destroyed, the process model must adjust the mission thread accordingly to reflect the fact that a role is no longer able to perform an activity.

After integrating the combat simulation with the other two models, you must once again test the federation to ensure that the mappings are correct and that the proper interactions are occurring. When the federation is working properly, you need to determine how many runs of the simulations you need. You should apply proper design of experiments and related analysis procedures to determine the approach to running the simulations and adjustments to any parameters that these models may have.

One of the critical aspects of conducting the dynamic analysis is determining the measures you want to extract from the simulation runs. Determination of the appropriate measures will depend largely on the analytical questions you are trying to answer. With the focus of our effort on C4ISR system architectures, we must examine those measures that best address the performance and effectiveness of these information processing systems. We can examine the three simulations and determine what measures we can extract from them to help conduct our dynamic analysis.

Measures of Merit

Most combat simulations have been developed to provide a variety of Measures of Effectiveness to determine mission success and other critical outcomes resulting from a system's performance. Classic measures such as Force Exchange Ratios and Total Red Losses help assess a system's effect on the outcome of the battle. Although many of these measures are indirect indicators of a system's impact on the mission, combined with other measures from the other simulations, we can develop a clearer picture of the contribution of the system to mission success.

In addition to these classic Measures of Force Effectiveness, we can examine other aspects of the combat simulation as well as the process models and communications network models to determine other effectiveness and performance measures. We can categorize these measures into three different areas: time-related, resource-related, and reliability-related.

Time-related measures include the time it takes to complete an activity or group of activities (i.e., a process), and the time it takes to send information from one node to another. We can measure these specifically as delay times caused by one factor or another. In the case of the time to complete an activity or process, the delay can be caused by a resource bottleneck, that is, a human or system resource that is busy performing other activities and therefore is causing undue delays in the processing of the activity or process in question.

Delays in sending information from one node to another may have several causes. One may be the delay due to the interdependence of activities within a process. If one activity required information from more than one other activity, it can see delays if one of the source activities is taking a long time to be performed. There are also delays in sending information or data due to bottlenecks in the communications network.

Resource-related measures can assess the utilization of a resource by measuring how busy it is to identify whether the resource is a bottleneck (over utilized) or is idle (underutilized). We can also measure the operational costs of the resource by determining what costs are attributed to the resource when it performs its designed function. Additionally, we can examine the marginal utility of adding another resource to support the operation. This measure will balance the benefit gained by adding the resource with the cost of that resource.

Reliability-related measures can assess the health of the operation and its recoverability from a failure. We can measures the health of the operation in terms of the impact of a single point of failure. These measures may be qualitative assessments such as ‘mission failure’, ‘loss of life’ or ‘task failure’. We can also measure the health of the operation in terms of the availability of alternate or back-up systems when they’re needed. If an alternate system is not available when it is needed the impact could again be described in the same qualitative measures as the impact of a single point of failure. Recoverability of the system can be measured in terms of the time to recover from a failure, the adaptability to changes in its environment (measured in time, quality, mission success, or losses), and the ability of the system to possess graceful degradation. This can be measured in terms such as, whether the mission tasks were completed prior to shutdown, or whether the mission was accomplished prior to the organization’s status changing to combat ineffective.

Methodology Guidance and Standards

To apply the methodology, users should follow the guidance and standards described in the section.

HLA Compliance

Each of the simulations that are used in the methodology must be HLA (IEEE 1516) compliant. These tools must be able to share object attributes and interactions through the Runtime Infrastructure software. Additionally, the simulations must run in a time-managed mode where each simulation remains time-synchronized with the others in the federation. This requirement ensures that any measures of performance or effectiveness involving time are properly calculated and accurate.

Required Architecture Products

In addition to the set of integrated architecture products, several additional products are needed to implement the methodology:

- OV-4 Organizational Relationships Chart
- SV-2 Systems Communications Description

Again, these products are assumed to be produced using System Architect, however, any architecture modeling tool is acceptable that can generate these products and allow them to be imported into the business process and communications network modeling tools. Additionally, the OV-6 a, b, & c products will need to be created through the business process modeling tool, which must be capable of creating the instantiation of these products in the form of a mission thread.

Combat Simulation Functionality

The combat simulation must have the functionality that allows it to send staff_process_model (SPM) interactions to the business process model. Key events within the combat simulation are the triggers that initiate these interactions that will start a mission thread in the process model. The combat simulation may need to be modified to send the appropriate HLA interactions to the business process model to identify when a key event has occurred. These interactions must be documented in the .fed file so that the business process model can subscribe to them.

You must determine which key events will cause a process or mission thread to start. After identifying these key events, you need to ensure that the combat simulation will generate an HLA interaction when that key event occurs. The interaction must have all the critical information to pass to the process model. This information includes a reference identification number that allows the combat simulation to know which key event the process model is referring to when it returns an interaction to the combat simulation. The interaction should also include operationally pertinent data that the process model or network model needs to execute its functions in the federation. Likewise, you must determine what actions in the combat simulation will occur as a result of interactions it receives from the process model.

Mapping Requirements

There are several mapping requirements that must be accomplished in order to implement the methodology. There is a mapping between the key events occurring in the flow of the battle represented in the combat simulation with mission threads represented in the business process model. A mapping file captures this relationship. The business process modeling tool reads the mappings in this file to translate the HLA interactions coming from the combat simulation into inputs that start one or more mission threads.

Additionally, there is a mapping between the operational nodes in the business process model and the system nodes in the communications network model. This mapping is used by the business process model when sending an HLA interaction to request a delay time from the communications network model. The business process model identifies which nodes are passing and receiving data and it translates the operational nodes into the system nodes recognized by the communications network model. The system node ID numbers are sent through the HLA interaction from the business process model to the communications network model.

Allocation of Mission Thread Activities to Federates

One of the issues that we discovered as we worked with a current set of mission threads from the Army was that some of the activities are appropriately represented in the business process model, but others are more appropriately represented in the combat simulation. These activities typically are those that are primarily physical actions where the entity performing the activity is explicitly represented in the combat simulation and can perform the physical actions in the combat simulation.

For example, one of the activities in a mission thread might be “Conduct a Fire Mission.” The entity performing that activity could be a Field Artillery firing battery. Rather than represent the time it takes to conduct the fire mission as a random draw from a probability distribution in the business process model, it is more realistic to have the firing battery conduct the fire mission in the combat simulation. The combat simulation would then return the amount of time the fire mission required to the business process model which would add that time to the overall process time for the mission thread.

This type of representation requires additional interactions between the business process model and the combat simulation, but it will more accurately represent the processing time required to complete the mission thread in question.

Implementation of the Methodology

For the research effort, we used tools that were readily available to us and that met the initial criteria for use in the methodology.

For the combat simulation we use Eagle, an Army ground combat simulation that is focused on Corps and Division-level operations. For this project, Eagle provided the flow of the battle or operational environment that dictated when the various mission threads (business processes) would occur in relation to one another. As sensors such as JSTARS, Guardrail, or Q37 Radar detect targets, this event initiates mission threads in the process model that represent the staff processes to handle that target detection.

For the business process modeling tool, we used Bonapart. Bonapart is an object-oriented business process modeling tool. It has a built in Petri-Net simulation engine that allows business process models to be executed as a simulation. The business processes modeled in Bonapart typically begin with an Input that receives Petri-Net tokens which follow the path of the business process through a series of activities that pass information in the form of the token until the business process is complete. Bonapart also contains the organizational charts and representations of the key processes that depict people and activities.

For the communications network modeling tool, we chose Network Simulator, also called NS or NS-2. NS-2 is a freely available, open source event driven network simulation developed at the University of California at Berkeley. NS simulates a variety of communications networks. It implements network protocols such as Transmission Control Protocol (TCP) and Unreliable Data Protocol (UDP); traffic source behavior such as File Transfer Protocol (FTP), Telnet, Web, Constant Bit Rate (CBR) and Variable Bit Rate (VBR); router queue management mechanisms such as Drop Tail, Random Early Detection (RED), and Class-based Queuing (CBQ); routing algorithms such as Dijkstra and more.

Due to various factors, we had additional programming that was needed to implement the technical interactions among the three modeling tools. Within Eagle, it was necessary to identify the key events that would trigger the mission threads in Bonapart. After identifying these key events, we had to modify Eagle to generate an HLA Staff Process Model interaction when the key event occurred. Eagle needed to publish these interactions through the Federated Object Model (FOM) and Bonapart needed to subscribe to them.

The main programming effort of this project focused on developing the functionality needed to allow Bonapart to interact with both Eagle and NS-2. The first challenge was to make Bonapart HLA compliant.

A key element of the methodology is the mapping of entities and relationships among the modeling tools. This mapping provides the capability to allow the combat simulation to pass the key events as interactions to the business process model that initiates the appropriate business process (mission thread). To accomplish this, there is a web-based application that reads the set of mission threads from the Bonapart model and displays them with a selection of Staff Process Model interactions from the FOM (fed file). The mapping occurs after the user selects the interactions that will initiate each of the mission threads in model. When this mapping is saved, a file is generated that contains this mapping. The BonaStep code reads this file when the Bonapart simulation is started.

Eagle and Bonapart must interact in order for the federation to achieve its purpose. Eagle will provide the occurrence of key events that will trigger the business processes (mission threads) in Bonapart. We must predetermine the key events that are of interest to us. These are the events that will trigger one or more business processes in Bonapart. An example might be the key event of a sensor detecting a target. This event might start the 'Process a Target' mission thread that requires information processing by several staffs (e.g., Intel, Operations, and Fire Support).

When the key event occurs, Eagle sends an HLA interaction of the type that will initiate the mission thread we want started. When the mission thread starts it will begin executing the activities performed by the appropriate staff cells. When one activity is finished and it is going to send information to another activity that will be performed at a different operational node, Bonapart send an HLA interaction that NS-2 understands. This interaction requests a delay time between the sending node and the receiving node, given the current load on the communications network. NS-2 determines that delay time and sends that value back to Bonapart through another HLA interaction.

For this all to work, there must be several mappings across the simulations. First, there is a file that shows the mapping of the key events in Eagle to the individual mission threads in Bonapart that each key event initiates. When a key event occurs and Eagle sends the corresponding HLA interaction, Bonapart finds the selected event in the mapping file and then starts the corresponding mission thread. There is a similar mapping file containing information about the Bonapart and NS-2 models. When a specific activity in a mission thread is executed in Bonapart, the program consults the mapping file and looks up the appropriate communications model in NS-2. Then NS-2 determines the node to node delay time and passes it back to Bonapart.

Technical Issues

There have been a myriad of technical challenges to overcome over the course of this project. This section describes a sampling of the critical challenges encountered and any recommended approaches to solving them.

HLA Compliance of Modeling Tools

Bonapart presented a challenge of its own, since it is not inherently HLA compliant. Although Pikos (formerly Pro UBIS GmbH), the developer, was very helpful, there was much to overcome. Pikos added "hooks" into Bonapart that we can use to access the information that HLA needs. Pikos modified earlier versions of BonShek and BonaStep to support the HLA compliance requirements and to allow interactions to occur between the Bonapart simulation and the RTI software. However, some desired functionality still is not possible. Pikos has once again been contacted and is working to add this functionality to support MITRE's effort.

As this methodology is applied using other modeling tools, it will be necessary to make those tools HLA compliant if they are not already compliant. Part of this functionality needs to take interactions from the combat simulation and determine which process to initiate in the business process model.

Implementation of DoD Architecture Framework in System Architect

Since Popkin released its original C4ISR module, many of the original assumptions and the definitions of several of the Framework products have since changed and there were areas of the C4ISR module that did not align with the current Framework document and the Core Architecture Data Model (CADM). These misalignments led to a collaborative effort with OASD(NII)'s Global Information Grid (GIG) architects to define the *Activity-Based Methodology for Integrated Architectures*. Given the need to bring the original System Architect C4ISR module up to date with the present Framework and CADM, and to implement the Activity-Based Methodology, an extension to the tool was designed and developed. This extension to System Architect, called the "*System Architect Extension*" or "*The Extension*", adjusts and aligns the C4ISR module by modifying and extending the underlying System Architect meta-model schema to the current DoD Framework and CADM data model and implements the *Activity-Based Methodology for Integrated Architectures*. This most current version of System Architect provides the functionality to correctly implement the DoD Architecture Framework.

Mapping Combat Simulation Events and Business Processes

After importing the architecture information from System Architect into the appropriate modeling tool (business process or combat simulation), it is necessary to map the key events, that will occur in the flow of the battle in the combat simulation, to the business processes. This may be a one-to-one, a one-to-many, a many-to-one, or a many-to-many mapping. In the one-to-one mapping, one key event will initiate only one business process. In a one-to-many mapping, one key event will initiate multiple business processes. An example is a chemical attack alert that would initiate several different processes in terms of defensive actions that individual units would take including notifying and checking on subordinate units and issuing subsequent orders to respond to the attack.

The mapping needs to be completed as part of the preliminary setup of the federation. This requires the appropriate subject matter experts (SME) to identify which key events will trigger which business processes. Once the SMEs have identified these mappings, the technical setup of the mappings must be completed. The technical challenge is how to capture the SMEs' input and how to make this mapping information accessible to the modeling tools as they run as federated simulations.

Mapping Mission Thread Activities to Models

Current versions of mission threads (business processes), as captured in operational architectures, show the flow of information from one operational entity to another. One of the challenges in using these mission threads is determining in which model the activity should occur. Although the mission threads represent the transfer of information, the activities themselves can represent a physical act, cognitive act, or some combination. Likewise some of these activities are staff functions while others are activities performed by sensors, weapons systems or other platforms whose main activity is a physical action as a result of receiving information from other entities. When the main activity is a physical action, consideration should be made to represent it in the combat simulation rather than the business process model.

When activities within a mission thread are determined to be represented within the combat simulation rather than the business process model, the appropriate interactions must be developed to allow the sending federate to pass the correct information to the receiving federate so that the appropriate activity can be performed.

Supporting Dynamic Changes

The primary issue here is the problem of the architecture framework's inability to handle fault tolerance. Currently, the DoD Architecture Framework assumes away any significant changes to the business processes that might occur due to a loss of one or more entities performing the operational activities in a process. The other Architecture Frameworks in the

Federal government have the same problem. The fault tolerance issue must be fixed to allow a more realistic portrayal of the processes when one or more entities fail.

Future Work

There are a number of areas where this research must continue in order to improve the utility of executable architectures in supporting dynamic analysis.

Integration of Dynamic Costs

Executable architectures must address not only the performance and effectiveness of the capability represented in the architecture, they must also address the dynamic cost of that capability as it operates to accomplish its mission. The dynamic costing must capture the costs that are a result of the capability operating in its operational environment and performing its designed functions. The categories of dynamic costing must be identified and the methods of measuring these various costs determined so that they can be included in the overall costs of the systems to support investment strategy, the acquisition process and other appropriate applications.

Handling Stale Information

Some of the messages that can build up in the queue at an operational activity may be messages with stale information, i.e., information that is no longer valid because it represents a condition that has changed, information that has been updated by a later message, or information that is so old that it has no bearing on the activity being performed.

The process modeling tools must handle stale information directly to better reflect the case where a person or system performing an activity would normally disregard and/or dispose of a message with stale information thus reducing the size of the message queue. This would result in a more accurate reflection of whether the person or system is, in fact, a bottleneck in the process.

Handling Fault Tolerance

With the focus of this research on supporting a military domain, we must account for the cases where military units, staff or individuals representing operational nodes may be destroyed in the course of the battle. When these nodes are destroyed in the combat simulation, we must account for these changes in the process model and communications network model. Our methodology and the tools we use must be capable of dynamically changing the flow of information and data to reflect the loss of a node and the adjustment of the organization to provide an alternate route or backup node to handle the information/data flow.

Summary

This project has an impact for DoD organizations, Simulation-Based Acquisition (SBA), and the DoD and Federal architecture communities. First, it allows DoD organizations to gain a capability to develop and analyze DOD C4ISR architectures and adapt them to changes. It also helps SBA by reducing the cost of dynamically exploring multiple alternative solutions. Finally, it helps all Federal architecture interests by providing a universally applicable methodology for dynamic analysis, helping standardize architecture efforts across DoD and potentially across the Federal Government, steering architecture efforts to more rigorous methodologies supporting both static and dynamic analysis, and providing input to next generation of architecture planning tools.

References

1. DOD Architecture Framework, V1.0, Vol. I and II, 15 August 2003.
2. Dickerson, C., Soules, S., "Using Architecture Analysis for Mission Capability Acquisition," Paper #123, 2002 Command and Control Research and Technology Symposium, 2002.
3. Hamrick, M., Pawlowski, T., Ring, S., "Multi-tier Simulation of Executable Architecture Views & Command Post Configuration Tool," Briefing, 27 June 2000.
4. Introduction to Petri Nets, http://www.daimi.au.dk/PetriNets/introductions/.
5. Kuhl, F., Weatherly, R., Dahmann, J., "Creating Computer Simulation Systems," Prentice Hall PTR, 1999.
6. Network Simulator, NS-2, http://www.isi.edu/nsnam/ns/.
7. OASD/NII, "An Activity-Based Methodology for Integrated Architectures and Analysis," Briefing, November 2003.
8. Ring, S. J., "An Activity-Based Methodology for Integrated Architectures and Analsysis," Paper #077, 2004 Command and Control Research and Technology Symposium, 2004.
9. Ogren, J. W., "Eagle Time Management," Paper 97F-SIW-073, Fall Simulation Interoperability Workshop, 1997.
10. Pawlowski, T., Barr, P., Ring, S., Williams, M., Segarra, S., "Executable Architecture Methodology for Analysis, FY03 Report," MITRE Technical Report 03W-0000081, February, 2004.
11. Pro UBIS, GmbH, "Reference Manual: Bonapart," 1999.

Author Biographies

THOMAS PAWLOWSKI, a Principal Operations Research Analyst for the MITRE Corporation. His work has focused on several areas including command and control systems, modeling & simulation, tools to support military training, and executable architectures. Dr. Pawlowski has worked for MITRE for the past five years at the Leavenworth, Kansas site. Prior to that, he spent 26 years in the Army as a Field Artillery officer and as an Operations Research Analyst. He retired as a Colonel in 1998. He graduated from the U.S. Military Academy at West Point and also has a Masters degree in Operations Research from the Naval Postgraduate School and a PhD in Industrial Engineering from Georgia Tech. He is a graduate of the Air Command and Staff College, the Army's Command and General Staff College, the Air War College, and the Army War College.

PAUL BARR, represents over five decades of design and academic experience in the fields of Computer Architecture, Computer Networking, Radar Signal Processing, Systems Programming, Operating Systems and Modeling and Simulation. He has been with the MITRE Corporation for the past 18 years and is presently at the New Jersey site as a Technical Manager. He currently provides consultations and individual contributions in several key software areas. Concurrent with his design experience, Dr. Barr has lectured in the graduate school of New England Universities (Boston University, MIT, and Northeastern University) in the field of Computer Architecture and Operating Systems. He is also an Adjunct Professor at Stevens Institute of Technology, in the department of Systems Engineering, lecturing on special topics in the field of modeling and simulation of complex systems.

STEVEN RING, a Principal Information Systems Engineer at the MITRE Corporation in Bedford, MA. He has over 3 decades experience including technical and managerial roles in commercial/military product development and integration. Mr. Ring received his MS in Systems Engineering from Case Institute of Technology in 1971. He has focused on applying information and knowledge-based repository technology to DOD architecture development and integration in support of interoperability and simulation based acquisition. He has contributed in the areas of techniques, methodologies, and tools for analyzing and validating both static and dynamic DOD architecture models. He currently is leading an OSD and Air Force sponsored effort in developing a tool-independent approach to integrated DOD architectures for assessment and analysis that also facilitates the transition to dynamic executable architectures.