

Executable Architecture Analysis Modeling for Network Testing and Evaluation in an HLA and TENA environment

Johnny Garcia
SimIS, Inc.
200 high Street suite 402
Portsmouth, VA 23704
Johnny@simistech.com

Keywords:
DODAF, Executable Architectures, Open Standards

ABSTRACT: *The Net-Enabled Command Capability (NECC) is the Department of Defense (DoD) C2 program to integrate existing and emerging C2 capabilities into a net-centric, enterprise-based, interoperable, joint architecture, to deliver integrated applications and databases to support the National Military Command System, Joint Force Commanders (JFC), Service/Functional components, and unit level commanders' decision making processes. NECC is critically dependent upon the Core Enterprise Services provided by the Net-Centric Enterprise Services (NCES) Program and establishment of the Global Information Grid (GIG). NECC is not a single system; rather it will be a system of integrated enterprise services. While the initial focus is on establishing shared situational awareness capabilities provided by the Global Command and Control System – Joint (GCCS-J) and the Global Command and Control System Family of Systems (GCCS FoS) as the initial NECC baseline, NECC is much broader and enables data sharing among many IP-enabled systems. NECC will employ time-phased evolutionary acquisition and spiral development to speed the delivery of advanced joint C2 capabilities to its users. Users will be able to participate and influence the NECC capability development process to ensure successful migration of GCCS FoS capabilities to NECC. NECC will evolve over time through ongoing concept development and experimentation. Part of this solution is applying executable architectures to support operational environments utilizing an open standards-based solution. This method can support federal communities with measurable & repeatable environments to virtually reproduce the environment and segments of training, experimentation or operational events to support sensitivity analysis of conditions with replaceable battlespace components. The creation of an Executable Architecture Analysis Modeling (EAAM) capability would offer the ability to adjust or replace components in the architecture to create and examine alternatives. By adjusting or changing the flow of activities, we can, in effect, adjust doctrine or TTP and see the effects on mission performance using a simulation environment that is linked to process and communication models. By applying simulations to operational processes, the federal government can reduce time and costs by testing certain aspects of a system's performance within the simulation environment prior to spending the resources to physically build the capability. When used as an approved surrogate for a live operational test environment, such a simulation offers the test community opportunities for greatly increased flexibility and significantly decreased cost and complexity.*

1. Background

The development of software and knowledge-based systems has simply not matured to the point where it can be characterized as following a discipline yet alone be considered equivalent to engineering methodologies in other technical domains. Programming could be viewed as the moral equivalent of the construction of an artifact (e.g., building, vehicle, etc) for disciplines like civil or aerospace engineering that occurs once design and engineering analysis has been completed. What software "engineering" lacks is a specification methodology that can guide "construction" or programming per se. Most attempts by Computer Scientists have evolved from

knowing how to program or code: pseudo code, specification languages, prototyping, etc.

The entire discipline of software development needs a common basis for specifying the design of the target system, one that can be shown to the stakeholders of that system (to include purchasing agency, users, testers, and etc.) and which they can understand without significant training. To draw an analogy, a building designer (AKA architect) creates an architectural drawing which becomes an artifact for discussing that design with everyone involved in the project and evolving it over time. Once that architectural drawing is finalized, required engineering details and specifications are worked out and

eventually provided to construction specialists who actually build the desired structure. Other development efforts follow a similar approach, e.g., aircraft, ship building, integrated circuits, civil engineering projects, etc. In software we have not achieved a clean division of responsibilities between design and construction and more importantly fail to achieve stakeholder buy-in, not because the system concept is too complicated, but because current software design artifacts just do not lend themselves to obtaining feedback and discussion of the system to be built until it is actually built.

System software needs architectural frameworks and standards for specifying them which can meet these needs. The Department of Defense has tackled this challenge through its Department of Defense Architecture Framework (DoDAF). While DoDAF may not offer the ultimate common solution for Computer Science or Software Engineering, it serves as a useful artifact for examining how such a framework could be used to support the development of better software that implements the systems needed to support a knowledge-rich business or decision making environment. One specific need is to insure any architectural specification can provide more than a static representation of the system, that it can somehow be simulated or executed as a way to evaluate system concepts and dialog amongst designers as well as with the other stakeholders. Today this is what other designers or “architects” do, they create design artifacts which can be shown to stakeholders using 3-D computer visualizations or dynamic simulations that demonstrate how the end product will look and work.

Research on architectural specification methodologies for software objects, particularly large ones intended to implement complex knowledge-based systems, needs to focus on approaches that will allow dialog during design and support evaluation of that design before it is actually implemented in code through simulations. Other engineering disciplines have achieved this maturation of process and purpose. Even our brethren Computer Engineers use design artifacts and then simulate them prior to actually building and testing (Yi et al 2006). Executable architectures are one approach that shows promise in meeting this goal but require additional exploration and evaluation.

2. Department of Defense Architecture Framework (DODAF)

DoDAF defines standard products to capture specific architectural views. DoDAF products are those graphical, textual, and tabular items that are developed in the course of building a given architecture description that describes characteristics pertinent to the architecture's purpose. The

current DoDAF prescribes products that provide "static" representations of information for various views. These static products, while capturing enormous amounts of information about the operational, system, and technical architectures, fail to provide a good vehicle for conducting detailed dynamic "behavioral" analysis of how the systems are supposed to interact with each other.

Before you can use executable architecture descriptions for any analysis purposes you must start with an architecture that is integrated, unambiguous, and consistent, then define what an integrated architecture is. A single architecture description is defined to be an integrated architecture when products and their constituent architecture data elements are developed such that architecture data elements defined in one view are aligned with architecture data elements referenced in another view [DoDAF WG Vol.I, 2003]. By alignment we mean that there is a set of symmetric operational and systems architecture elements that have similar meanings, associations/ relationships, properties, and characteristics. An integrated architecture can be defined among multiple architectures when similar or related single architectures, each based on the same set of DoDAF integrated products and constituent aligned architecture elements can be combined for further development and analysis purposes. A final and yet more relevant definition to the research from a memo published by Deputy Secretary of Defense, “an integrated architecture has been defined as an architecture consisting of different views or perspectives (operational view, system view, and technical view) that facilitates integration and promotes interoperability across Family-of-Systems/System-of-Systems and compatibility among related mission area architectures”. Integrated architectures usually have associated with them a time frame, whether by specific years (e.g., 2005-2010) or by designations such as “as-is”, “to-be”, “transitional”, “objective”, “epoch”, etc. In all cases, this reduces to either inventories of current capability (“as-is”) or blueprints of future capability (“to-be”) based on some future need or objective.

Domain experts, program managers, and decision-makers need to be able to analyze these architectures to locate, identify, and resolve definitions, properties, facts, constraints, inferences, and issues both within and across architectural boundaries that are redundant, conflicting, missing, and/or obsolete. The analysis must also be able to determine the effect and impact of change (“what if”) when something is redefined. In most “as-is” architectures, details about activities, nodes, roles, systems, etc. are fully known and architectures analysis can be readily accomplished.

Integrated architecture elements are keys to the creation of executable architectures. “Executable architecture” refers to the use of dynamic simulation software to

evaluate architecture models. These executable architectures differ from the typical simulations because they are often generated directly from the architecture models via a semi-automated or automated process. Several purposes can be achieved with these specialized tools.

- The architecture model itself can be verified for internal self-consistency.
- Operational concepts can be simulated, observed dynamically, verified and refined.
- Operational plans can be examined and assessed.
- Tradeoffs between systems can be assessed.
- Architecture measures can be evaluated, which can support cost-benefit analyses and quantitative acquisition decisions.

There are some key factors in the process of constructing and using executable architectures. First, the aspect of automated or semi-automated generation directly from the architecture models is not simply for convenience. Rather, the driving factor is the accuracy of the executable model, in terms of consistency with the existing architecture models. Many typical simulation efforts diverge from the actual architecture models over time, leading to either the architecture being ignored in favor of the implemented design within the simulation, or, the simulation falls into disuse, as it is not able to keep up with the pace of the architecture modifications. There are currently no standards for the format or process for constructing executable architectures. Research has been done, but additional research is still required.

Most executable models assume a distributed, message-passing paradigm for the architecture operations, which is very applicable in most of the situations encountered in current practice. However, the architecture data elements and the attributes required to construct executable models are specified in the DoD Architecture Framework (DoDAF) products [DoDAF WG Vol.II, 2003] [2], but still no standard process exist. It is also important to make the most of the executable architecture concept. The process by which these types of tools are applied must be integral to the overall systems engineering process. In other words, the development process must be configured to rely upon the results of the executable efforts for validation and refinement. Efforts to construct executable architectures for their own sake have generally not been beneficial to their programs. Executable architectures have immediate implications for process improvement, but also directly support the investment decision process by providing realistic and repeatable cost-benefit analyses. 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.

3. Executable Architecture Approaches that use DODAF Techniques

Most studies in the area of executable architectures consist on designing, comparing, and suggesting an approach that is disparately similar with other type's of architecture modeling techniques. In the literature related to executable architectures, there are only a few examples that deal with executable architectures and DODAF. The MITRE Corporation developed Executable Architecture Methodology for Analysis (EAMA) [4] integrating a combat model, a process model, and a communications model to represent the primary components of an architecture and implement these models in a simulation environment. The Joint Forces Command (JFCOM) developed Process Architecture and Analysis Model (PAAM) [5] for Experimentation, PAAM is an analytical and operational tool used to examine the effectiveness of current and future operational architectures, systems and processes. These capabilities and the author's approach are similar techniques; therefore, to perform a comparison of this article within the existing literature, two points will be considered. The first point is related to approaches that strive to solve the problem of integrating "Static" DODAF products to create integrated architectures. In essence, the general philosophy of these techniques is related to using operational and systems architecture models to generate executable architectures. Fundamentally, the second point is to bring to light DODAF shortfalls in the context of generating executable architectures.

A small number of authors and developers have created approaches that deal with Executable Architectures and integrated architecture techniques. In this regard, [Pawlowski 2004] the MITRE developed Executable Architecture Methodology for Analysis (EAMA) was developed to integrate a combat model, a process model, and a communications model to represent the primary components of architecture and implement these models in a simulation environment. The combat model provides the operational context for a system. The process model represents the mission threads that must be executed using the system to accomplish the mission. The communications model represents the network in which the data sent or received by the system must travel. The objective of EAMA is to:

- 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 for system 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.

The U.S. Joint Forces Command (JFCOM) developed Process Architecture and Analysis Model (PAAM) for Joint Experimentation and is a capability created to examine the effectiveness of current and future operational architectures, systems and processes (Garcia, Browning 2006). PAAM is developed to integrate transformational C4ISR processes and work flows analysis and assumptions using several Department of Defense common architectural standards (JTA, DODAF); and several commercial and open standards bodies architectural standards (UML, TOGAF). PAAM was developed to support analysis of operational architectures and organization before building or implementing them. The US Navy created a capability Global Engineering Methods Initiative for Integration and Interoperability (GEMINII), which enables decision makers to understand the impact of their acquisition, decisions (Charles, Turner 2004) [6]. It captures capability-based analytical data, which helps to manage complexity in an almost ad hoc development environment. GEMINII meets the need for traceability and repeatability. It is both a process and a toolset based on achieving desired capabilities through activity decomposition, integrated architectures and semi automated analysis of inter-system dependencies. Our assessment of the three papers/projects; (Pawlowski, Charles, Turner and Garcia) concluded that well-defined architectures are an essential part of engineering assessments. They allow decision makers to look for the mix of assets that best optimizes the balance between cost and capability. The acquisition community has determined that decision-makers need the ability to perform detailed technical analysis while maintaining traceability and repeatability; all of this is driven by validated DODAF models

Several scientist and engineers have pointed out that integrated executable architectures are difficult to attain due to the DoDAF's limitations. They used existing DODAF techniques, a software engineering tool and a new method of using architecture templates to create congruently integrated processes, methods, and tools effectively and efficiently to solve interoperability problems by identifying information exchange requirements (IER) and deliver key interface requirements and key functional performance requirements for each system in the warfighter domain. Additionally, the architecture templates gave the operators and technicians the ability to capture and create the process models

needed. The approach offered and compared by represents a suitable and alternative method for creating executable architectures. In this regard, the strength of the argument is that it can use data-centric build sequence tailorable using DODAF "static" products, architecture templates and COTS tools and lead to better and more accurate results.

4. State of the Art

The purpose of this paper is to present the current state of the art for Executable Architecture within a network telemetry environment specifically for use with protocols like High level Architecture (HLA) and Test and Training Enabling Architecture (TENA). We discuss current use of methods, standards and processes for executable models development. One of the keys to the required research is based it on utilizing the Department of Defense Architecture Framework (DoDAF) models generic six step process of building an architecture description. Figure 1 lists the six step process of building an architecture description

While DoDAF provides a process for producing static models of systems a base reference model for the domain for the systems being modeled is needed, such as the Levels of Information Systems Interoperability (LISI) Reference Model. Each higher level of the LISI represents a demonstrable increase in capabilities over the previous level of system-to-system interaction -- in terms of the data transferred, the applications that act on that data, the infrastructure required, and the procedures (e.g., policies and processes) for information management. MIL-STD-499 described the systems engineering process as requirement analysis, functional analysis/allocation, synthesis and system analysis & control and IEEE 1471 is the recommended practice that addresses the activities of creation, analysis, and sustainment of architectures of software-intensive systems, and the recording of such architectures in terms of architectural descriptions. A conceptual framework for architectural description is established. MIL-STD-499 has evolved to the EIA 632 standard. Figure 2 depicts the evolution of the EIA 632 standard.

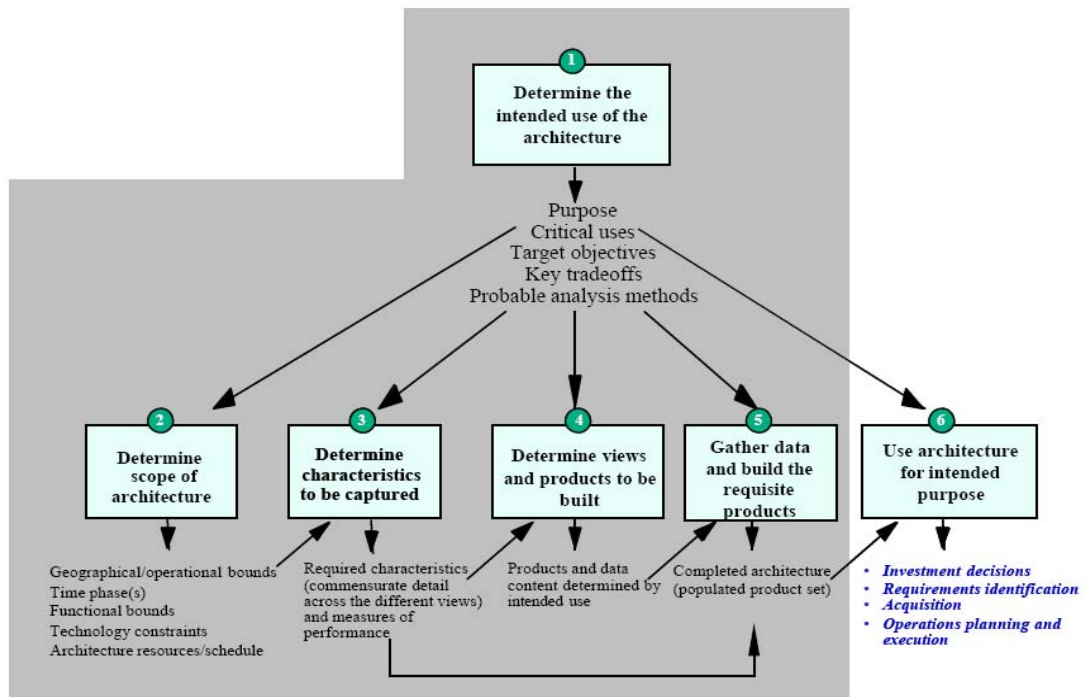


Figure 1.0 the Six-Step Process of Building an Architecture Description

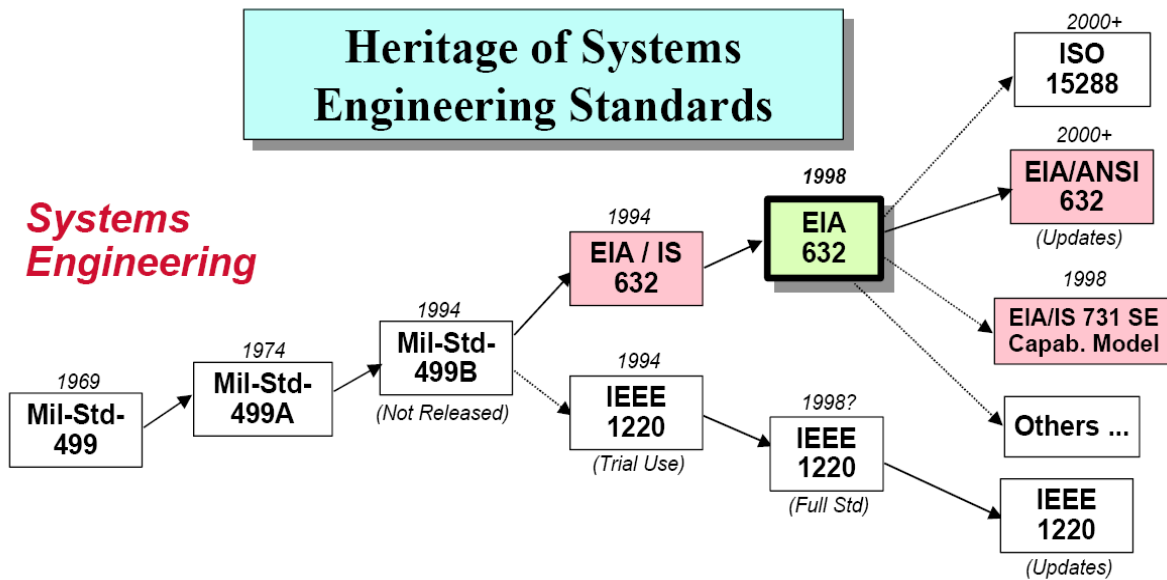


Figure 2 Electronic Industries Alliance (EIA) 632 Heritage

4.1 DODAF Shortfalls to Enable Executable Architecture Techniques

The current DODAF begins at a systemic level where most “business” requirements have been identified and the scenarios are presented as given, rather than as potentially “out of the box” alternative scenarios to be modeled, projected, and examined for business impact analysis. DODAF 1.5 only addresses “build, implement, deploy, operationalize” decisions. In addition, it only does so at the information systems level (or more correctly the “data grouping” systems level). The DODAF currently does not include Monte Carlo simulation, trade-off analysis, game theory projections or other complexity modeling analytical support tools (Markovian or analytical hierarchical processes support). Dr. Alexander Levis, the Chief Scientist of the Air Force, notes that this is a worthy goal, but the framework lacks explicit guidance on how to achieve it. He presents his own ideas on how to do this: “The derivation of an executable model of the architecture from the three views and the associated integrated dictionary provides a basis for understanding the interrelationships among the various architecture products and establishes the foundation for implementing a process for assessing and comparing architectures” (Levis and Wagenhals, 2000:226). An executable architecture (referred to by Levis as an executable model) is defined by the DoDAF as “use of dynamic simulation software to evaluate architecture models” (DoDAF, 2003:7.3). Levis goes on to assert the value of the executable model, “To fully describe and understand the dynamic aspects of the system requires an executable model” (Levis and Wagenhals, 2000:228). Essentially, the static diagrams presented in the architecture do not adequately convey, “the logical, behavioral, and performance properties of the architecture.” Thus, to Levis, the creation of the executable model is essential to be able to portray a dynamic system and measure its utility versus other systems.

To implement the executable model, Levis (along with Handley and Zaidi) has advocated the use of the Colored Petri Net (CPN) (Handley et al, 2000). CPNs are a means to create an executable model that “combines all the information in the various static models or views into one model that can illustrate dynamic behavior: the flow of data, the conditions under which functions are performed, and the order in which they are performed. Behavior analysis and performance evaluation can then be carried out using scenarios consistent with the operational concept.” (Handley et al, 2000:15).

5. Evaluation of the Impact of the Content on the State of the Art

So, why do we need standards and models in the first place? Standards are shared across industry, which greatly facilitates successful interoperability of our individual efforts. Standards tell us “what” to do with respect to the processes for engineering systems based on the life cycle perspective. Models, in this case give specific guidelines on best practices for how to do the “what” better than has historically been done. In (Lee et al, 2005) [the researchers report on utilizing EIA 632, then applying the DODAF six step guidelines to develop the process for the project. The standard process of creating an executable architecture is not addressed here or in any other study, all studies and implementations of integrated executable architectures is conducted “on the fly” and no common process has been created or followed.

Pawlowski (2004) proposed EAMA; based on that work other researchers have discussed, designed, and proposed different approaches to deal with executable architecture issues. For example, Garcia (2006) proposed the Executable Architecture Analysis Modeling Method that will enable the Naval Test & Evaluation command the ability to conduct dynamic, persistent, extensible, measurable, repeatable and interactive testing of processes, architectures, and components. EAAM can adjust components in the architecture to create and examine alternatives. This method is an effective way to test and validate strategies, tactics, and procedures while experimenting with organizational structure and functions. Also, EAAM can provide the capability to measure and assess the effects of doctrine, tactics, techniques and procedures (TTP), and processes on mission performance. By adjusting or changing the flow of activities in the process model, EAAM, in effect, adjusts doctrine or TTP and sees the effects on mission accomplishment using a combat simulation that is linked to the dynamic process model.

6. Approach Proposed

Key elements of this approach are (1) analyze the use of executable architecture (2) integrate a combat model, a process model, and a communications model to represent the primary components of architectures and (3) implement these models in a simulation environment. The combat model provides the operational context for a system. The process model represents the mission threads that must be executed using the system to accomplish the mission. The communications model represents the network in which

the data sent or received by the system must travel. The proposed method applies executable architectures to support verification analysis of a system or capability within one common operational perspective. A major issue is not only how to accurately document capabilities using architectures, but also how to conduct a useful analysis of these capabilities to determine 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 architecture frameworks provide no clear means to examine these systems in a dynamic fashion as they function in their operational environment. A successful solution must start with current situational awareness data provided by a Common Operational Picture (COP). These capabilities can support this research with measurable and repeatable environments to virtually reproduce the environment and play back segments of events to support verification analysis of conditions with replaceable components to simulate the system or system of systems.

Broader impacts of the research will provide contributions that allow the advancement of executable architecture techniques for testing and experimenting with proposed technical and operational architectures. These experiments will help uncover the relationships and processes in the operational, systems and network architectures in simulated environments and provide a method for optimizing the true cost of the architecture. Similar related contributions may also be applied to training and creation of model simulations and architecture models. These contributions may enhance our scientific and technological understanding of architecture model validation techniques for interoperable solutions. The architecture verification modeling approach for executable architecture consists of the following steps:

1. Starting with the operation architecture descriptions, we formulate the activity models for the operational scenarios within the simulation.

2. After validating the activity models with subject matter experts, the activities are mapped to existing systems.

3. Next is the development of the logical deployment architecture. The logical architecture may be presented at several levels of detail. Physical or network architecture may also be identified as appropriate. Sequence diagrams may be used to describe interaction between services or systems in the architecture.

4. The logical data model provides useful information on data flow. The system data exchange matrix and data schema will deliver further estimates on the data sizes and data transmission latency.

5. Finally, all of the above architecture information is combined to construct the executable architecture model. The executable model is constructed based on the activity model, with additional details for data flow, sequencing and timing from the process model. Discrete event simulation techniques will be used to implement these methods. The method will validate the architecture logic and will provide quantitative decision making information for the end-to-end process. The results of the executable architecture analysis will produce performance parameters. Figure 4 shows the above steps in architecture modeling for the proposed method. The use of the integrated architectures concepts will be the main data driver of the architecture framework verification process and the executable architecture model for EAAM will initialize its data sets utilizing only the required elements of integrated architectures to provide a concept of operations picture, activity-functions service mapping, functions/services, and operational activity diagrams. Then, systems data will come from the logical deployment and network architecture, sequence diagrams, logical data model and finally the interface definition. These elements are shown in figure 3. Figure 4 shows 3 integrated models all executing simultaneously. The SV-10c controlling the chronological sequence of interactions and the systems involved in those interactions is shown on the top-right. The Operational Activity Diagram is shown on the bottom also depicts how the 3 models are integrated and how they are dependent on each other. Figure 5 depicts the findings of the execution for analytical view.



Figure 3 shows required elements of integrated architectures to provide a concept of operations picture, activity-functions service mapping, functions/services, and operational activity diagrams.

Integrated and Dependent Views

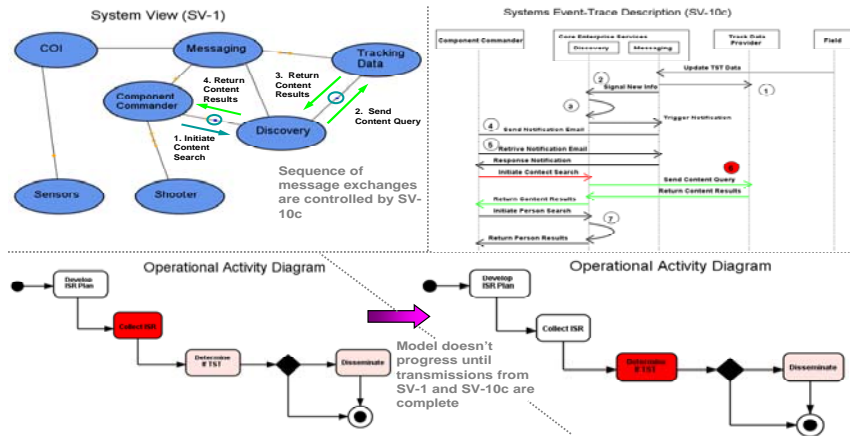


Figure 4 shows the Integrated and Dependent Views

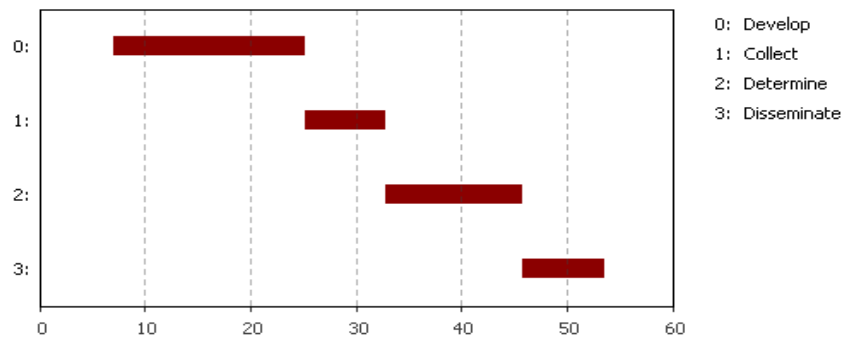


Figure 5 shows the findings

7. Summary

Research into developing executable capabilities for network telemetry and communication modeling has significant potential to mature (perhaps transform) the discipline of architecture development into a state where it more closely resembles the approaches which other technical disciplines follow when designing, specifying, and then constructing the artifacts which are the focus of their work. Architectural specifications which support collaboration amongst designers and programmers and with stakeholders who will procure, test and use such systems are needed. However, such architecture specifications and their consequential products need to be dynamic to support these collaborative dialogs and allow non computer scientists to understand how the system is intended to functions. Executable architectures hold the potential to achieve that goal and are worthy of further research and development at the investigatory level supported by federal research funding agencies.

8. References

- [1] Yi, Joshua J, Eeckhout, Leiven, Lilja, David J., Calder, Brad, John, Lizy K, Smith, James E. The future of Simulation: A Field of Dreams, IEEE Computer, Nov 2006, Vol 39, No 11, p.22-29.
- [2] DOD Architecture Framework, V1.0, Vol. I and II, 15 August 2003.
- [3] Else, Steven. "as-is," "could be" and possible transition consideration, center for Government Transformation, 2005.
- [4] Pawlowski, T., Barr, P., Ring, S., Williams, M., Segarra, S., "Executable Architecture Methodology for Analysis, FY03 Report," MITRE Technical Report 03W-0000081, February, 2004.
- [5] Garcia, J, Browning, J., "Innovations in Process Modeling as Applied to JFCOM Joint Experimentation Directorate Division Experiment Management Teams" Spring SIW 06S-SIW-015, March 2006.
- [6] Charles, Philipp, Turner, Phil "Capabilities based acquisition ... from theory to reality" CHIPS, summer, 2004.

Author Biography

Johnny Garcia is founder and CEO of SimIS, Inc. and a Modeling and Simulation Ph.D. Candidate at Old Dominion University. He has over 17 years of engineering experience that includes systems architecture design, software development, database development, C4I systems development, logistics systems development, and new technology insertion for the Department of Defense. His Dissertation Thesis is in the creation of an Executable Architecture Analysis Modeling method proposed in this paper.
Distribution: Authorized to U.S. Government Agencies and their contractors only; Reason: administrative/operational use. Other requests for this document shall be referred to the Program Manager for Test & Evaluation/Science and Technology (T&E/S&T), Test Resource Management Center, 1225 South Clark Street, Suite 1200, Arlington, VA 22202