Simulation Interoperability Workshop, Orlando, Fall 2001
Paper 01F-SIW-026

Object-Oriented Analysis of a DII COE Simulation Product Line Architecture

Ronald B. Sprinkle
The AEgis Technologies Group
12565 Research Parkway
Orlando, FL 32765
rsprinkle@AEgisTG.com

William P. Sudnikovich
Atlantic Consulting Services
167 Avenue at the Common, Suite 4
Shrewsbury, NJ 07702
wsudnikovich@acsinc-nj.com

Francis H. Carr
The MITRE Corporation
7515 Colshire Drive
McLean, VA 22102
carr@mitre.org

Keywords: Army Enterprise Architecture (AEA), Command Control Communications Computers and Intelligence (C4I) Interoperability, Defense Information Infrastructure Common Operating Environment (DII COE), Joint Common Database (JCDB), Modeling and Simulation (M&S), Product Line Architecture (PLA), Object-Oriented Analysis (OOA), Software Architecture, Technical Architecture

ABSTRACT: *The Army has articulated a vision in which simulations will support C4ISR systems through the integration of simulation infrastructure into the Defense Information Infrastructure Common Operating Environment (DII COE) software architecture. Identification of a specific simulation infrastructure product set is the key to developing the technical steps required to achieve this vision.*

Integrating simulation into the DII COE in a systematic fashion requires the following: 1) reuse by the simulation infrastructure of existing DII COE C4ISR software segments 2) identification of new segments required to provide DII COE based simulation capability and 3) identification of new simulation-enhanced C4ISR functionality not available today in either C4ISR or simulation domains through new DII COE segments. As simulation-enhanced C4ISR systems will use intelligent agent software, there are relevant Future Combat Systems (FCS) implications. FCS C2 systems will need to interact with intelligent agent-based robotic forces and will encounter similar challenges identified for future simulation-enhanced C4ISR systems.

This paper describes a general Object-Oriented Analysis based approach, which identifies DII COE segments as software products in a Product Line Architecture. This paper concludes with recommendations for use of the DII COE Simulation Product Line Architecture in achieving the Army simulation to C4ISR interoperability vision.

1. Introduction

1.1 Opportunity

A significant portion of the U.S. Department of Defense is at work attempting to solve interoperability challenges or reduce interoperability costs. The Army is building Future Combat Systems and the Objective Force on the concept of network centric warfare. Without a system-of-systems solution we will be unable to perform the seamless data interactions required to build network centric warfare systems. Efforts such as the Software Blocking Policy, the Environmental Database (EDB) Integrated Product Team (IPT) and the Simulation to C4I Interoperability (SIMCI) Overarching IPT (OIPT) [1] are working to solve the interoperability challenge at the system-of-systems level.

Further, as M&S applications continue to increase in power and utility and those capabilities transition into "embedded" C4ISR capabilities, the difficulties inherent in specifying, tracking, interfacing, and assessing this evolving "system-of-systems" will grow.

The U.S. Army vision for simulation & C4ISR interoperability, as specified in [2], prescribes the need to develop a simulation infrastructure in the DII COE for C4ISR systems. That paper and other SISO activities set the stage for a plan to solve some of the major C4ISR and simulation interoperability challenges.

A plan to satisfy these interoperability challenges must define simulation-enhanced C4ISR system components; must specify their operational context to the degree required by the designer; must identify and specify interfaces; and must provide a mechanism by which multiple simulation infrastructure components can be sequenced and prioritized for acquisition or development.

This paper presents the results of an object-oriented analysis of a simulation-enhanced DII COE product set and makes recommendations for the future acquisition of simulation infrastructure components.

1.2 Thesis

The integration of simulation infrastructure into the DII COE software layered architecture, depicted in Figure 2, is necessary but not sufficient to ensure future C4ISR simulation interoperability. A coherent system-of-systems level capability must be prescribed and does not occur from "market forces" or when left to chance.

One way to form that system of system functionality for families of simulation-enhanced C4ISR systems is to identify the needs of the user and specify a finite set of software application products to support them.

Through the identification of those software application products, the solution space of the interoperability challenge is broken into discrete components. When we focus on the software products as a set, we look to understand their interdependencies in providing the simulation-enhanced C4ISR functionality. It is these interdependencies that help us define the architectures, data modeling and standards depicted in Figure 1 that are required to build true interoperability.

Furthermore, software products are the culmination of a myriad of interoperability design choices such as those represented by the subjects of Figure 1 blocks B, C & D. The user cannot directly interface with Common Data/Object Models, Common Standards or Architectures. Software products must represent interoperability design choices to the user through

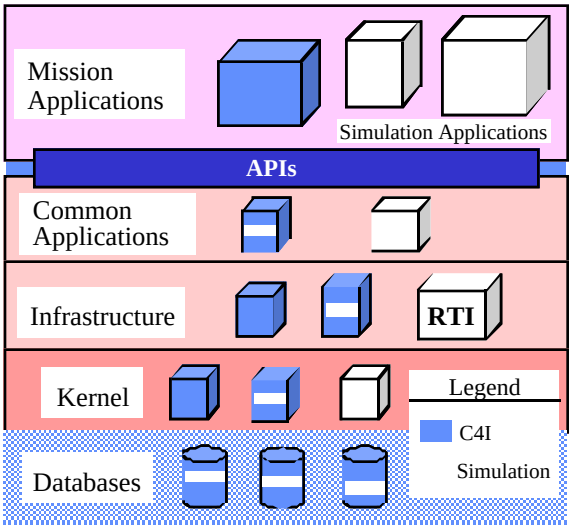

Figure 2. DII COE Software Layered Architecture

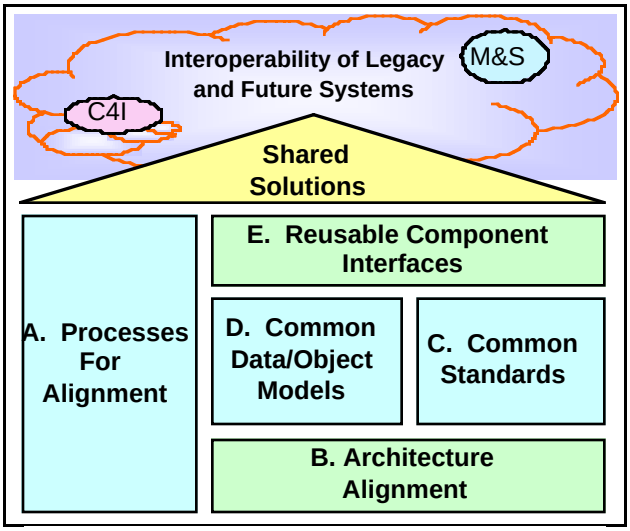

Figure 1. "House" of Interoperability

increased functionality.

The standards, architectures, data models and process efforts can only impact interoperability if they are somehow manifested in software products.

1.3 Scope

This paper is concerned with collective interoperability of Army systems with M&S applications as described in [3]. Joint systems can be similarly analyzed using the methods described herein, but this paper does not specifically address Joint systems. The Joint system uses are potentially different from Army systems and therefore the corresponding system interactions warrant a separate analysis.

As described in [2], this paper's scope is bounded by those systems supported by the DII COE and the simulations that must either connect to or be embedded within these systems.

The analysis in this paper, and therefore its result, does not encompass the simulation-enhanced C4ISR system's full spectrum of use. The scope of the analysis is restricted to the use of simulation on board a future C4ISR system used in standalone mode. We expect that the extrapolation to LAN/WAN distributed modes, to include interaction with external simulation(s), will result in the addition of architecture components but not modification of the architecture itself.

The remainder of this paper is organized as follows. Section 2 discusses the simulation-enhanced C4ISR system. Section 3 presents the Product Line Architecture (PLA) overview. Section 4 outlines the analysis method.

Section 5 presents the detailed section on the OOA method and how its use results in classes in the PLA. Section 6 looks at the PLA and its uses in detail. Section 7 concludes with recommendations and considers the impact of the presented architecture.

2. The Simulation-enhanced C4ISR System

This section discusses the impact and potential uses of simulation to enhance C4ISR system capability. We cite some of the increased functionality based on the current and future systems, how interoperability is approached and the role of reuse.

For the purpose of our discussion, M&S applications can be one or more functional components that are either embedded within, or externally linked to a C4ISR system to aid or enhance the operational capability of the base system.

2.1 Enhancement of C4ISR systems

The following paragraphs provide examples of common operational functions that are required within military service organizations. Prior to the availability of C4ISR systems, these functions were traditionally carried out by manual, human intensive activities. As computer hardware and software evolve, these activities are gradually being automated. Just as Tolk discussed in [4] we believe that the addition of M&S capabilities to the existing suite of C4ISR applications will benefit users with more powerful tools. More importantly, the successful application of M&S is critical for current and near term C4ISR systems to deliver the performance and functionality expected of them.

Situation Assessment. Situation Assessment gives military staff visibility into current force deployment, activities, and status. Additional capabilities might provide insight into similar information for opposing forces. Sample tools that aid in this function are simple message storage and retrieval mechanisms, automated alert systems, and common data stores. It may also involve more sophisticated visualization tools such as Geographic Information Systems (GIS), Digitized Topgraphic Support System (DTSS), or Joint Mapping Tool Kit (JMTK).

Course of Action (COA) Development & Analysis. COA development and analysis is the ability to develop and measurably assess the results of taking one or more possible actions as a response to a given military situation. In addition to using the same capabilities described above, typical components used in COA development and analysis include Synch Matrices, Force Ratio tools, GIS, DTSS, or JMTK.

Mission Rehearsal. The Mission Rehearsal capability attempts to project the results of applying a particular COA to the current situation. Simulating the activities based on the current situation assessment, or modeling the aggregate effects of actions taken may produce these results. Currently, methods used for mission rehearsal are principally intellectual activities of mission staff, with some assistance being provided by SA and COA software. Mission rehearsal tools have a limited presence on C4ISR systems and there is significant interest in development of such components.

Execution Monitoring. Execution Monitoring provides the capability to assess the current military situation and compare it against COAs that had been developed earlier. Future capabilities might retain a number of COAs, with specific branches and sequels that could be considered. They might also continuously evaluate potential COAs for optimal circumstances which when encountered, flag them for Command and / or planning staff. Currently, tools provided to aid this function represent simple alerts set on C4ISR systems that monitor the flow of information, and signal when critical resources or items of interest are encountered.

Training. It is envisioned that all future C4I systems will be fielded with the capability to perform training, either embedded in the system itself or linked to some distributed training capability. As can be seen in Table 1, conducting training encompasses all of the functionality expected to be required to perform tactical tasks and helps illustrate the need and potential benefit of common products.

The operational functions just discussed are excellent examples of simulation-enhanced C4ISR system functionality. Our next step is to define in more detail what products are needed to gain these C4ISR system capabilities. An initial effort at identifying applications and the functions they support is shown in Table 1.

Table 1 also illustrates that there are simulation-enhanced applications (left column) common across multiple mission functions (top row). It is clear that these applications, when instantiated into software components, must fit into some framework that facilitates interactions among the applications and the system functions they support.

2.2 Interoperability and Re-use

Table 1 illustrates that a number of C4I system applications can be reused across mission functions. Current re-use of this kind has evolved both as a part of a planned development strategy by DoD and Service

developers, and also as a result of ad-hoc experimentation with these tools in training and field situations. Through the careful development of use cases as part of the Product Line Architecture set, further reuse can be realized. Scarce development resources can be focused on components with the highest potential value. C4I systems will be enhanced with simulations as a collection of new DII COE components and modifications to existing components. This is no different than other C4I systems applications.

Table 1. Functionality Cross Matrix

Functions / Applications	Situation Assessment	COA Development	COA Analysis	Mission Rehearsal	Execution Monitoring	Training
Display	X	X	X	X	X	X
Data Collection	X	X	X	X	X	X
Scenario Generation		X		X		X
Exercise Control	X			X	X	X
Terrain Analysis	X	X	X	X		X
Overlay Builder	X	X		X	X	X
Synch Matrix		X		X	X	X
Wargame Simulation		X	X	X	X	X
AAR Tool		X	X	X	X	X

The Army Enterprise Architecture (AEA) [5] provides the context within which approaches to system implementation and interoperability are defined. The AEA is the Army's framework used to guide information technology investments, acquisitions, and fielding of integrated systems-of-systems capabilities. It includes programs to develop integrated architectures incorporating Operational Views (requirements), Systems Views (system-of-system laydowns), and Technical Views (technical standards) for Army tactical units, functional areas, and installations. A similar approach is presented here, defining requirements, looking at the system and its components and focusing on common products that will be created within a common framework or standard. A critical part of this task is identifying the individual products and components that provide the future system-of-system capabilities/functions.

2.3 Future Applications

As the C4I systems of today transition to the systems of the future, solving the interoperability issue with a well thought out, rigorous process based on accepted, standard architectures, common products and components is critical to success. An example of a current research and development effort looking at future command and control can be found in the Agile Commander Advanced Technology Demonstration [6] program being executed by the Army's Communications-Electronics Command (CECOM) Research, Development and Engineering Center (RDEC). The Agile Commander ATD is exploring new architectures and designs to demonstrate the emerging technologies feasibility (to include simulations) and their future command and control system applications.

We envision that, like today, the military scenario building of the future is concentrated around the development of plans. The difference is that building a plan in the future is affected by the introduction of information technologies as pointed out in the March 2001 PHALANX [7]. Plans creation may be done graphically, worked in tight iteration with Course of Action development and then shipped to subordinate units as a simulation run file. The subordinate unit executes the run file in their on-board simulation to study the time-varied 3D execution of the plan in concert with their sister, subordinate and higher units.

The system-of-systems that will be the basis of the Objective Force is the Future Combat Systems (FCS)[8]. It is envisioned that some components of the FCS will be autonomous, robotic platforms. These platforms, while expected to revolutionize warfare, present unprecedented challenges in terms of command and control, integration and interoperability. Simulation-enhanced C4ISR systems in this future force will be critical to its success. What, in fact, is different about interfacing and controlling a robotic platform with a C4I system compared to doing the same with a wargame simulation?

Simulation technology is playing a major role in each of these programs. It is imperative that we start to build future systems in a manner that will bring to bear the benefits anticipated by the network centric system-of-systems concepts. Critical to this success is a common framework and infrastructure to support interoperability between simulations and future C4I systems.

3. Product Line Architecture Overview

The European Software Institute [9] defines PLA as "The common architecture of a set of related products. The product-line architecture captures the essential commonality in the product-line and provides for flexibility and adaptability for specific requirements." A

Product Line Architecture, as specified in [10] "defines element types, how they interact, and how the product functionality is mapped to them". Wittman & Harrison represent the product line perspective in [11] with "For OneSAF the product line concept is driven by the need to support multiple user domains with a variety of end state uses". These definitions are particularly applicable since we intend to design the simulation infrastructure to be useful to not one specific C4ISR system, but a set of C4ISR systems.

We use a PLA because it is oriented towards the development of products and we are correlating the components of the DII COE to products. We believe that the specification of PLA products will result in a more focused overall effort thereby increasing the probability that the simulation-enhanced C4I capabilities will become a reality.

An example of a PLA, identified in [12] is a "Car Periphery Systems (CPS), which combines all products which use, or could use, radar or ultra-sonic sensors to detect objects in the immediate vicinity of or approaching a vehicle". Individual products require an intense amount of domain expertise to produce so their independent development is practically dictated. Still, the concept of their integration onto a vehicle requires that they integrate well, that improvements to the products do not invalidate their common architecture.

How do we go about developing a PLA? The next section introduces the method we used.

4. Method

It is clear that we intend to be constrained to supporting system functionality within the DII COE. The DII COE, through the specification of software capabilities as reusable segments, supports a component-based architecture. The question to answer is "How do we systematically specify the components needed for a simulation-enhanced C4ISR system?"

4.1 The Decision

We can begin to answer this question by observing some characteristics of DII COE software segments. DII COE software segments are:

- Reusable software with well defined interfaces.
- Unaware of their users. They simply provide the needed interface for their use.
- Comprised of private data and methods on that data that gives them their characteristic signature and allow them to perform their intended function.

We recognize these observations as some of the most fundamental characteristics used to describe software objects. We also recognize that DII COE segments are in many ways not unlike software objects [13,14]. For this reason we choose to use the Object-Oriented Analysis (OOA) method to analyze, specify and communicate the components needed for the simulation-enhanced C4ISR system. Making the selection of OOA has the advantage of the use of a well understood and rich design language in the Unified Modeling Language (UML). With UML we can communicate our specification of objects to the world of systems engineers and software designers.

Furthermore, the object-oriented framework is well studied and gives us options for patterns of designs from which to choose our solution space decomposition [15]. Supporting this claim, the book providing one of our PLA definitions, is a member of the well-known Object Technology Series by Addison-Wesley [9].

4.2 The Method Overview

The development of a Product Line Architecture has a flavor of creativity in that there are not any cook book recipes to follow, resulting in the finished product, just right, every time. The developer must combine the correct amounts of domain experience, UML expertise and System/Architecture design experience.

This section outlines the steps selected to develop the Product Line Architecture for the C4I-to-Simulation integration domain. Figure 3 depicts the primary steps.

Step 1: Actor Selection. In UML, users of the system are known as actors. Actor selection is done in close coordination with the system users, in this case, users from the Central Technical Support Facility (CTSF) at Ft. Hood, TX. Actor selection also involves projecting how

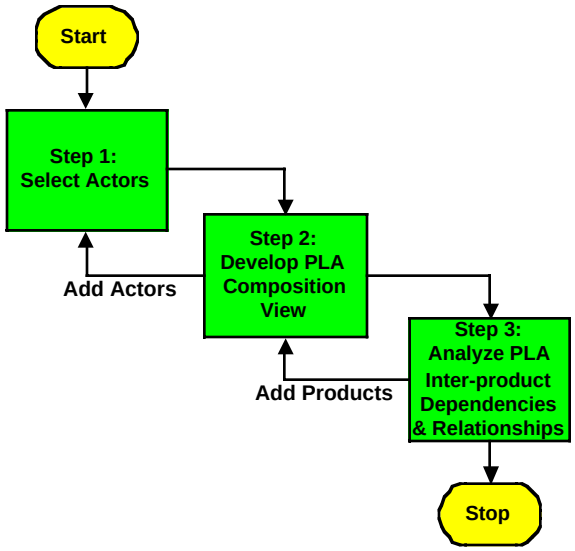

Figure 3. Basic PLA Development Steps

future systems may be used. We will come back to Step 2: Develop PLA Composition View since it is described in detail in the following diagram. Step 3: Analyze PLA Inter-product Dependencies & Relationships is part of the follow-on effort to the work covered by this paper. Step 3 consists of conducting an analysis of each of the PLA software products as classes with the intent of capturing all of the important relationships and data. The potential exists for many useful design products to result from activities in Step 3. Examples include, although not limited to; specification of the DII COE software segment APIs, interaction specifications (perhaps contributing to HLA FOM interaction and object models), and potential object models.
Since we also desire to understand dependencies between the classes (or products), it is important to show how the products work together to provide system level functions.

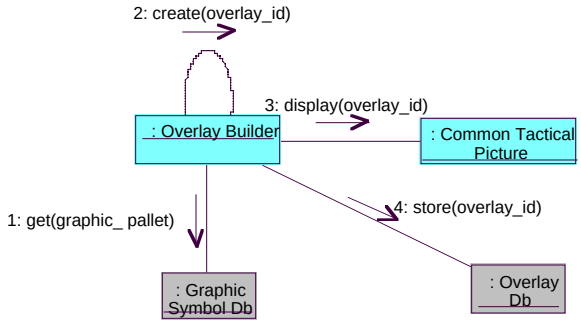

Figure 4 Example Collaboration Diagram

Object Collaboration Diagrams accomplish this. Object Collaboration Diagrams, such as the one depicted in Figure 4, are used to "realize" the functions required to support a use case. Developing an Object Collaboration Diagram for each use case also ensures that each of the systems functions are addressed within the object/products identified.

The development of these different diagrams (Class, Object Collaboration and Use Case) is the focus of Step 2 in Figure 3. It is an iterative process where the developer changes focus from one diagram type to the next. The primary purpose of Step 2 is the development of the PLA Composition View. However, the developer can seldom complete all aspects of one diagram type without discovering a facet (use case or class) that needs to be added to another diagram or view. In this way each diagram and especially the PLA Composition View, continues to be enriched in detail as the process interactively reveals more depth and breadth to the content of the architecture.

In our analysis we used the process depicted in the flow chart of Figure 3 until each use case had its own Object Collaboration Diagram. The process of developing an Object Collaboration Diagram for each generated use case causes the discovery of new classes and sometimes, new use cases. Since our focus is to identify products in the PLA, each new object type identified in an Object Collaboration Diagram equates to the identification of another class in the PLA that performs a system level capability.

5. Object-Oriented Analysis

While there is no requirement in this paper for the reader to be familiar with OOA, some familiarity with OO terms is assumed. The germane elements of the OOA method are highlighted as steps in defining the PLA. The first step in OOA begins with Use Case identification.

5.1 Step 1: Actor Selection

The Object-Oriented Analysis of any system-level design begins with an analysis of the system from the use perspective. Properly done, use case analyses recognize the external stimuli that drive the design of the system to respond to the user needs. Like any complex problem, decomposition of the problem space is of critical importance to the simplification of the solution space. A simulation-C4I PLA is no different. Problem space decomposition begins with the selection of the system Actors.

Selection and distinction between the different roles performed by the actors on the system, enables unambiguous and ideally, non-redundant identification of use cases. In Figure 5 we see the relationships between the actors enhances the understanding that each actor inherits the use cases of the actor above it, thereby

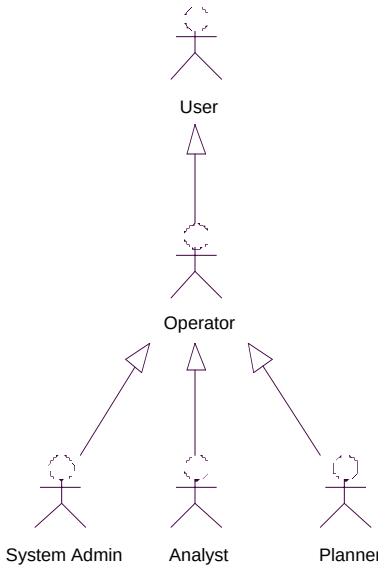

Figure 5. Actors & Actor Relationships

cementing what we already know to be true; that more sophisticated users build their actions upon a foundation of simpler operations with the system.

Actors. Since we are developing a Product Line Architecture, we take care not to select actors that are specific to any one C4ISR system, but apply to the class of C4ISR systems for which the PLA is intended. The following actors and their definitions were identified with help from users of the ABCS systems:

- User. This is the most basic actor. The role of the User actor performs actions related to system operation that does not relate to C4I/Simulation software such as turning on the computer, printing adjusting the monitor, setting the screen saver, etc.
- Operator. Operator is a basic user who specializes in the operation of the C4ISR/Simulation System software. Starting it, sending messages, etc. are actions performed by the Operator actor.
- Analyst. This role primarily encompasses actions related to information retrieval and subsequent analysis.
- Planner. This role focuses on forming plans and task-organizing assets for future operations. Creating action plans, task organizations, tactical control graphics, and measures of effectiveness are activities performed by this role.
- System Administrator. System Admin performs specialized activities such as software installation and hardware maintenance.

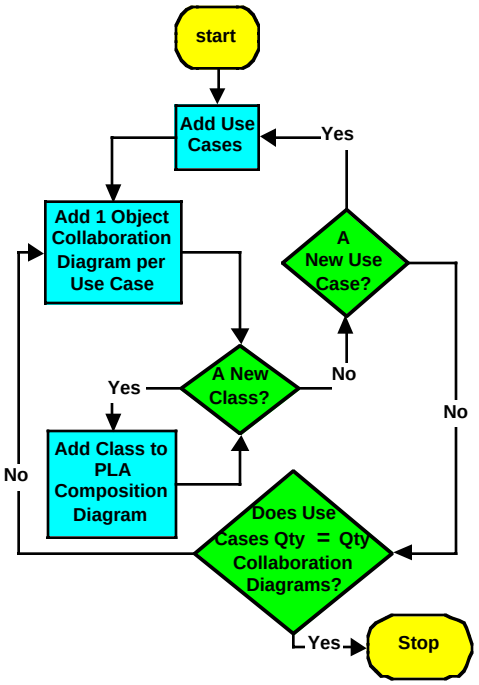

Figure 6. PLA Composition View Development Process

Having identified the actors involved with the simulation-enhanced C4ISR system, we turn to the use cases themselves. Naturally, these are developed and organized around the perspective of the actor.

5.2 Step 2: Develop PLA Composition View

The goal of Step 2 is the development of the PLA Composition View. The process to develop the PLA Composition View is patterned after the diagram in Figure 6. Accordingly, we begin by addressing use cases.

5.2.1 Use Cases

The planner develops potential current and future operations. The planner is concerned with the setup of the military scenario and the context within which the scenario must run. We call this context the exercise scenario. The use cases supporting this genre of interactions are listed in Table 2.

Table 2. Planner Actor Use Cases

USE CASE HIERARCHY
1.0 Create Course of Action
1.1 Create Military Scenario
1.1.1 Create Ops Plan
1.1.1.1 Create Task Organization
1.1.1.2 Create Graphic Overlay
1.1.1.3 Create Unit Instructions
1.1.2 Input Unit Status
1.1.3 Create Comms Plan
1.2. Create Exercise Scenario
1.2.1 Input Data Collection Plan
1.2.2 Create Mission Event List
1.2.3 Set Measures Of Effectiveness
1.2.4 Create Network Laydown
1.2.5 Map Models To Units

The indention and the numbering within Table 2 indicates parent/child relationships such that Create Course of Action may include (or is extended by) Create Military Scenario and Create Exercise Scenario. Create Military Scenario may include Create Ops Plan, Input Unit Status and Create Comms Plan. In the same fashion Create Ops Plan may include Create Task Organization, Create Graphic Overlay and Create Unit Instruction. From the Planner hierarchy of use cases, we can see that the two main Planner use cases of the simulation-enhanced C4ISR system supporting Create Course of Action are Create Military Scenario and Create Exercise Scenario.

At this point, we have captured some interesting use cases. We continue this exercise for User, Operator, Analyst and System Administrator Actors to complete the use cases required for the system. We have shown the results of the process used to develop the Planner use cases. Since method execution for the rest of the actors is no different than that shown for the Planner, listing of the rest of the use cases is omitted.

The next step in the development of the Product Line Architecture is to assemble, in an Object Collaboration Diagram for each use case, the objects required to support the simulation-enhanced C4I system functionality.

5.2.2 Collaboration Diagram Development

Collaboration Diagrams show which classes are involved in implementing a particular use case. They show this by depicting a set of objects, links between the objects and messages sent between the objects for a particular scenario.

The result of developing a Collaboration Diagram for the

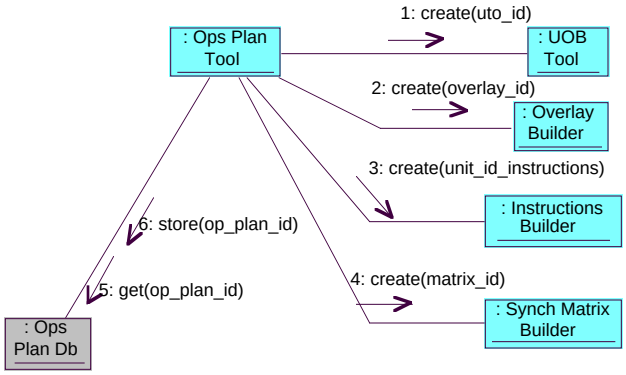

Figure 7. Create Operations Plan Collaboration Diagram

Create Ops Plan use case from Table 2 is shown in Figure 7. For this example, we envision that a tool such as the Ops Plan Tool would assemble an Operations Plan consisting of different part types, such as textual instructions, graphic overlays, etc., of the 5 Paragraph Order format. We imagine that this is done using independent software applications, much like this document is produced using MS Word, Rational Rose, MS PowerPoint and MS Visio.

We then step through the thought process of how these applications would interact to produce an Operations Plan for use by the simulation-enhanced C4I system. Figure 7 shows very basic messages or links between the objects representing software applications that assist building and storing the Operations Plan.

5.2.3 Composition View Class Diagram

The process described in Figure 6 ultimately results in the addition of new classes to the PLA Composition View. The Composition View is a UML class diagram where the "has a" relationship forms the primary relationships between classes in the diagram. Through the "has a" relationship we depict what software applications are a part of the PLA. The "has a" relationship also helps us recognize clusters of classes, or software products, that collectively provide some major system capability. Figure 8 depicts the composition of the Military Scenario simulation-enhanced C4I capability, as a "cluster" of products bordered in a dotted red line at the bottom of the diagram.

The development of the Create Ops Plan collaboration diagram in Figure 7 results in the identification and addition of the Synch Matrix Builder and Instructions Builder products. These products were added as classes to the PLA Composition View and are shown circled in the Military Scenario portion of Figure 8.

Although Step 3 from Figure 3 analyzes PLA inter-product relationships, we have not completed that step. It serves to further identify relationships among the PLA software application products and is left to future efforts.

From Step 2: Develop PLA Composition View, we have gained a significant understanding of the relationships and dependencies between software application products. We use this insight to demonstrate how the dependencies among software application products are used to organize and reach the Army vision of DII COE integrated simulation architecture

6. The Product Line Architecture

Since the purpose of the analysis is to understand what products comprised the PLA, we are concerned with an architecture view, shown in Figure 8, that consists of the primary classes (software application products) and their relationships. The diagrams focus for relationships is on the line with the diamond on one end. This forms the "has a" relationship between classes. For example the C4I/Sim System box "has a" Execution Monitor, AAR, etc. Except the "depends on" dashed line with arrow all other relationships are labeled.

The products in Figure 8 combine in clusters to form the major simulation-enhanced C4ISR system capabilities of Military & Exercise Scenario Development, AAR, Reports Development and Execution Monitoring. As an example, Military Scenario Development is identified in

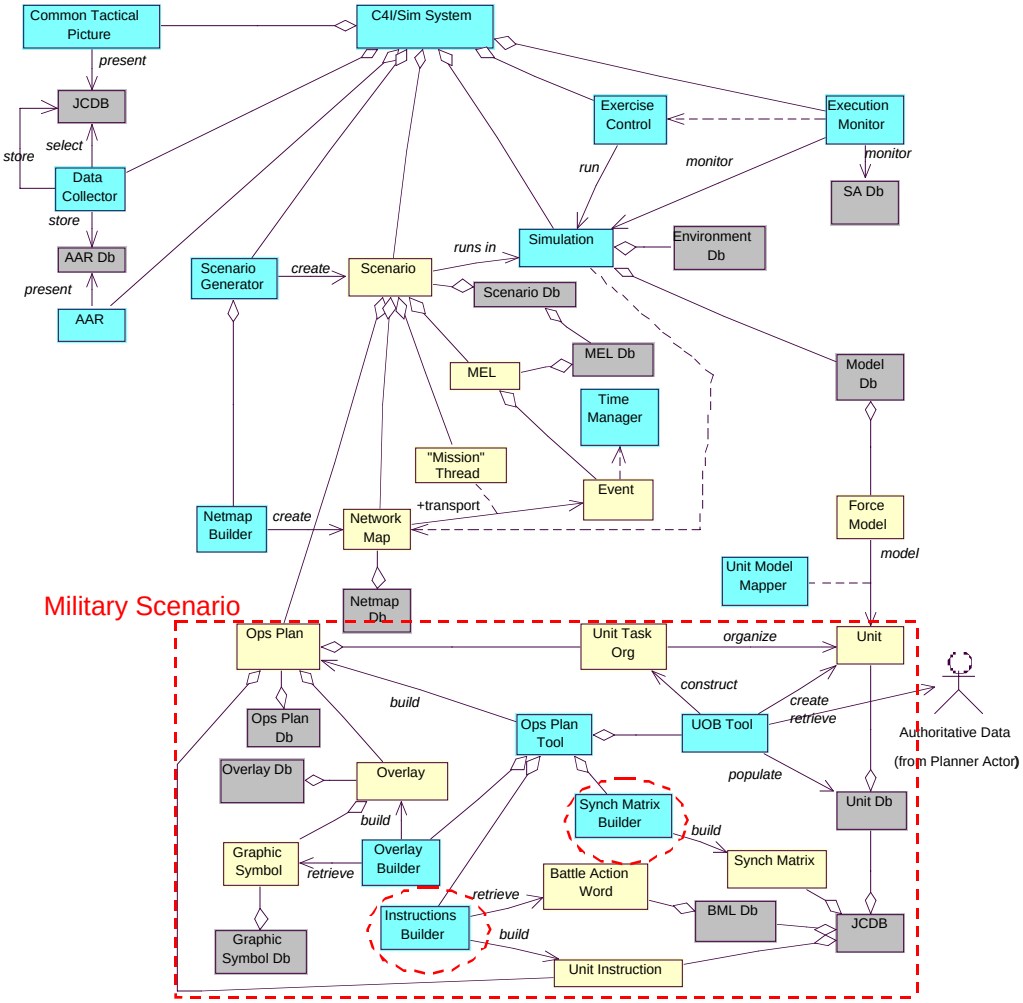

Figure 8. Product Line Architecture Composition Class Diagram

the dashed red box at the lower portion of Figure 8. The figure also shows executable software products shaded in blue, non-persistent data artifacts shaded in yellow and persistent data stores shaded in gray. At the bottom of Figure 8, the software product "Instructions Builder" retrieves the data artifact "Battle Action Word" which is contained (stored) in the persistent data store (database) called "BML db". In this way, Figure 8 classes depict system composition.

6.1 Using the Product Line Architecture

There are really two immediate uses of the PLA. The PLA is used to understand the list of products that must be in place to support the simulation-enhanced C4ISR system functionality. The PLA is also used an input to determine the time ordering of product deployment into the hands of the user. There are other uses not as immediate, but still of primary importance such as the development of software product interface specifications. Our immediate uses take precedence however, so the others uses of the PLA will be left to future work.

We use the PLA to identify which products must be developed, which products can be adapted or modified from existing products and which products already exist and can be used as is. This is the same make/buy analysis that most systems engineers perform during system design. They identify system components and any COTS/GOTS availability/suitability for system use. Preliminary product make/buy analyses for the C4I/Simulation PLA are shown in Table 3.

In addition to the make/buy analyses presented in Table 3, we have listed all of the products identified in the analyses to date and their dependency upon databases. It is startling to note how often databases are reused among the different products. The implications of that reuse are clear. *Data modeling is a crucial element to the*

interoperability of the different products and the simulation-enhanced C4ISR systems that they support.

While critically important, dependencies upon databases are not the only dependencies recognized among the products through the OOA method. The process of developing the Object Collaboration Diagrams and adding classes to the PLA Composition View establishes each software product and some of the relationships between those products as critical to the execution of at least one use case. These *relationships translate to development dependencies*, especially true where some products support the capabilities of many other products.

Table 3. PLA Make/Buy Analyses

PRODUCT	DATABASE	ADOPT/ADAPT /DEVELOP
AAR Tool	JCDB/AAR Db	Adapt
Common Display	JCDB	Adopt CTP
Data Collector	JCDB	Adapt DCM
Exercise Control	JCDB	Develop/Adapt
Execution Monitor	JCDB/SA	Develop
Instructions Builder	JCDB/BML	Develop
Ops Plan Tool	Ops Plan Db	Develop
Overlay Builder	Overlay Db/JDCB	Develop
NetMap Builder	NetMap Db	Develop
Scenario Generator	Scenario Db	Adapt
Simulation	Env./Model Db	Develop/Adapt
Synch Matrix Builder	JCDB	Develop
Time Manager	N/A	Adapt
UOB	Unit Db/JCDB	Adopt
Unit Model Mapper	Model/Unit Db	Develop

Products with the most dependencies are selected for early deployment. We adopt/adapt/develop these products first so that the dependencies of the products that follow will be satisfied.

Figure 7 illustrates the Ops Plan Tool dependency upon the Unit Order of Battle (UOB) tool as the UOB Tool "create(uto_id)" method is called. Rational Rose™ automatically collects each class dependency from all Object Collaboration Diagrams where a UOB Tool method is called. Classes with the most method calls and the highest quantity of dependencies can be easily determined. Each use case has a collaboration diagram and the Rational Rose™ tool tracks each dependency from the collaboration diagrams to each class. *Requirements traceability from system use case to class (product) is built in.*

As an intermediate step, MS Project was used to organize the products as tasks and the dependencies identified with Object Collaboration diagrams as task links. These finish-to-start task links establish the MS Project time ordered sequence of products depicted in Figure 9.

The colored horizontal bands in Figure 9 are based on the technical blocks of the house chart in Figure 1. Each band in Figure 9 is built upon the interoperability foundation of the bands beneath it. If, for example, the foundational work in Architectures is not executed first, the interoperability of related work in Common Standards, Common Data/Object Models and Shared Solutions & Interfaces is in jeopardy.

Looking deeper into the Figure 9 one can begin to see that the same relationship existing between the blocks is also manifested in the relationships between the products. For example, the product "NetMap Builder" depends on a standard for network specification, in this case called

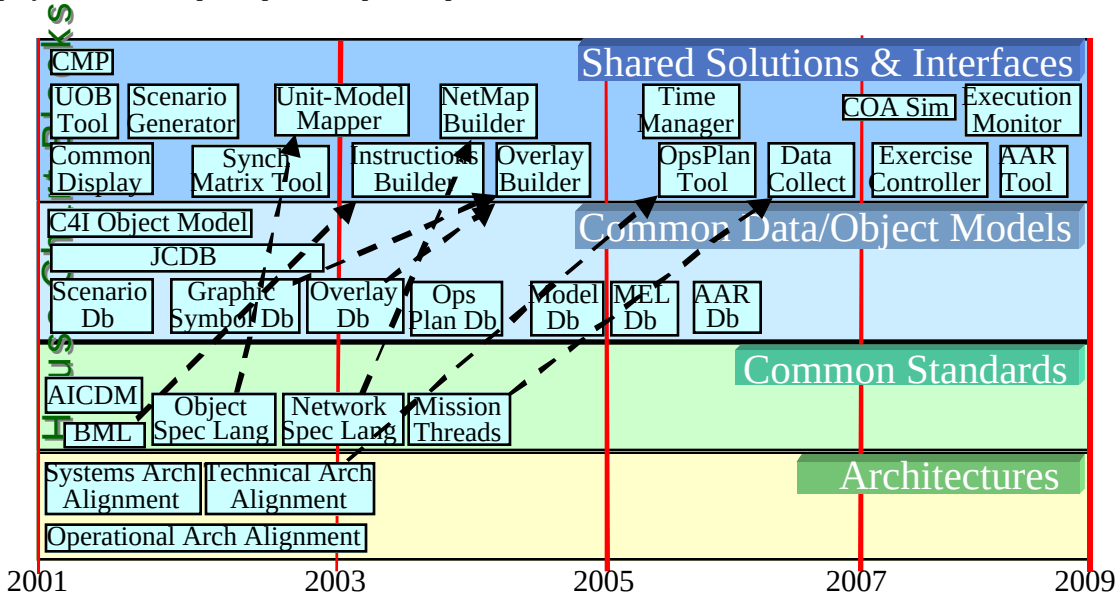

Figure 9. Product Development Timeline

"Network Spec Language". The "Network Spec Language" may be incomplete if it leaves out newer network components included in the Systems Architecture. Some of the dependencies between are depicted with the arrows between software components.

As the PLA analysis becomes more mature and fully populated, the products and timing represented in Figure 9 will solidify. Furthermore, SIMCI will synchronize the timeline represented by Figure 9 with the US Army's new Software Blocking Policy. *By default, this will force product acquisition and development timing priorities to align with the US Army Transformation Plan.*

7. Conclusions & Recommendations

This paper frames a method and a plan to attain vision for the future of simulation interoperability with the C4ISR domain over the next 10 to 20 years. The DII COE paradigm has most of the technical and process components needed for a comprehensive solution. It is still necessary to further specify the products and their time phased deployment for the effective creation of simulation-enhanced C4I system functionality. Object-Oriented Analyses to determine a Product Line Architecture for the DII COE provide a well-suited method to plan and define the integration of simulation infrastructure and functionality into simulation-enhanced C4ISR systems. Our purpose has been to introduce the method, define the solution space and to identify the results to date.

We have discussed how the need for a suite of components that comprise the DII COE-based simulation infrastructure is well defined by the functional requirements of the simulation-enhanced future C4ISR systems. The observation that the DII COE components are similar to objects enables the use of the OOA method. The realization that DII COE components are also the products in a DII COE-based simulation Product Line Architecture logically connects the analysis method to support the description of the desired PLA end-state.

7.1 Conclusions

Using the OOA method as our process for defining the PLA we conclude the following:

- Standards, architectures, data models and process efforts can only impact interoperability if they are manifested in software application products.
- Given the strong correlation between products and DII COE components, developing a PLA using OOA enables a straightforward identification of a DII COE-based simulation infrastructure.
- The PLA method supports systems design by identifying make/buy decision targets (products). Product capability is traceable to system use cases.
- So much of the PLA is dependent upon common databases that the specification of a common data model is essential for product, simulation and C4ISR systems interoperability. This confirms the direction in data modeling that SIMCI has been heading for the last 2 years.
- The PLA method allows a systematic way to time phase the introduction of the simulation infrastructure products into the DII COE. This is an enabling function to achieving the interoperability vision for C4ISR and simulation systems.

7.2 Plans for Future Work

Our recommendations fall out naturally from the analyses and conclusions:

- Complete the use case study and subsequent OOA of the PLA for the RDA, TEMO, ACR domains, for distributed/embedded C4ISR-based simulations and for umbilical simulations externally supporting C4ISR systems.
- Continue to specify the PLA API level information. This specification will assist in identification of whether to adopt or adapt existing products, or if new products must be developed.

In closing, since the vision driving this paper is based upon the DII COE, it is not only an Army vision but is also a Joint vision. Furthermore, parallel efforts [16] suggest that this vision can be an international one that is shared between the U.S. and its allies. We believe that our approach to attaining the vision through the use of the PLA method is broad enough that it can be used by the Joint and international communities and that the conclusions and recommendations herein are valid for consideration by those communities.

8. Acknowledgements

The authors would like to express deep appreciation to the many individuals who reviewed this paper and improved it with their thoughtful insight. We acknowledge Dr. Michael Hieb and Dr. Andreas Tolk for their work in formulating the vision. Ron Sprinkle was supported by the US Army Simulation, Training and Instrumentation Command (STRICOM), Bill Sudnikovich was supported by the US Army Communications-Electronics Command (CECOM) and Frank Carr was supported by the Defense Modeling and Simulation Office (DMSO) while writing this paper. Both Mr. Sprinkle and Mr. Sudnikovich are Architects to the Overarching Integrated Product Team for Simulation to C4I Interoperability (SIMCI OIPT) [1]

and acknowledge the support and vision of the OIPT in the creation of this paper.

9. References

[1] SIMCI WWW Site, Army Overarching Integrated Product Team for Simulation to C4ISR Interoperability: "http://www.simci.org/", 2000.

[2] Hieb, M.R. and Sprinkle, R.: "Simulation Infrastructure for the DII COE Architecture: The Army Vision", Paper 00F-SIW-035, 2000 Fall Simulation Interoperability Workshop, 2000.

[3] Hieb, M.R. and Staver, M.J.: "The Army's Approach to Modeling and Simulation Standards For C4I Interfaces", Paper 98F-SIW-259, 1998 Fall Simulation Interoperability Workshop, 1998.

[4] Tolk, Andreas: "Heterogeneous Synergy – A Vision for the Next Generation of Warfighter IT Systems", Paper 00F-SIW-013, 2000 Fall Simulation Interoperability Workshop, 2000.

[5] ODISC4, "The Army Enterprise Architecture Master Plan, Volume I, Strategy", 30 Sept 1997.

[6] Agile Commander Advanced Technology Demonstration (ATD), http://www.c2d.c3sys.army.mil:443/agile.htm, June 2001.

[7] Hayes, Robert E, "C4ISR Framework of the Future", PHALANX Vol. 34 No. 1, March 2001.

[8] Kern, LTG Paul J., "Future Combat Systems", Briefing, DarpaTech 2000 Symposium, Dallas, TX, 2 Sept 2000.

[9] Product-Line Architecture, Information Services Dictionary, http://www.esi.es/Help/Dictionary/Definitions/Product-line_architecture.html, Feb 16, 2001.

[10] Hofmeister, Christine, Nord, Robert and Soni, Dilip. "Applied Software Architecture", Addison-Wesley, 2000.

[11] Wittman, Rob, Harrison, Cynthia, "OneSAF: A Product Line Approach to Simulation Development", Paper 01E-SIW-061, 2001 European Simulation Interoperability Workshop, 2001.

[12] McGregor, John, "A Product Line Process for the Production of Platform Software", Bosch, 22nd International Conference on Software Engineering (ICSE 2000), Limerick (Ireland), June 4-11, 2000.

[13] Booch, Grady, Rumbaugh, James & Jacobson, Ivar, "The Unified Modeling Language User Guide", Addison-Wesley, 1999

[14] DII COE Integration & Runtime Specifications, Version 4.1, http://dod-ead.mont.disa.mil/cm/general.html, October 3, 2000.

[15] Gamma, Eric, Helm, Richard, Johnson, Ralph & Vlissides, John, "Design Patterns", Addison-Wesley, 1998

[16] Tolk, A. and Kunde, D., "Decision Support Systems-Technical Prerequisites and Military Requirements", Proceedings of the 2000 Command and Control Research and Technology Symposium, June 2000.

Author Biographies

RON SPRINKLE is the Orlando Director of Operations and the C4I Technology Lead for the AEgis Technologies Group, Inc. During his 13 years as defense contractor, he has developed software and hardware systems on numerous programs to include the Line Of Sight Anti-Tank (LOSAT), the ARPA Reconfigurable Simulator Initiative (ARSI), Army Experiment 3 (AE3), the BattleLab Reconfigurable Simulator Initiative (BLRSI), Fire Support Combined Arms Tactical Trainer (FSCATT) and the Joint Simulation System (JSIMS). As a U.S. Army Reserve Officer, LTC Sprinkle has served 7 active duty years and Battalion S-3 and Battalion XO tours. Mr. Sprinkle is a graduate of Oregon State University with a BS in General Engineering and of Southern Methodist University with a MS in Computer Science.

WILLIAM SUDNIKOVICH is a Project Manager for Atlantic Consulting Services in Shrewsbury, NJ. He is responsible for planning, directing and consulting on simulation activities in support of the US Army CECOM and other Army activities. Prior to joining ACS, during a nineteen year career at CECOM, Mr. Sudnikovich was the architect of the CECOM SINCGARS Radio Model (SRM), the Tactical Internet Model (TIM) and was instrumental in establishing M&S activities within the CECOM RDEC. He was active contributor to the development of the IEEE 1278 DIS standards and has previously served as Chairman of the C4I Forum of the SISO Simulation Interoperability Workshops. Mr. Sudnikovich holds a B.S. degree in Computer Science from Rutgers University and a M.S. degree in Computer Science from Fairleigh Dickinson University.

FRANCIS H. CARR is a Lead Software Engineer with the MITRE Corporation and has been with the Information Systems and Technology Division since February 1996. His software development experience reaches back to 1975, and includes work in Artificial Intelligence, Reliability Engineering applications, Mathematical Systems Modeling, and Business systems development. At MITRE, Mr. Carr has worked with both civilian (FAA) and military simulations, and has been involved with research, analysis, design, and development of Simulation and C4I Interfaces. His current efforts include exploration of Thin Client Architecture use with ABCS systems, development of the Technical Requirement Specification for the DII COE M&S TWG, refinement of the C4I-M&S Technical Reference Model, and advising activities for DII COE M&S TWG, DISA, DMSO, and other Sponsors. Mr. Carr is a graduate of

Boston University and lately from George Mason University with an MS in Software Engineering.