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

"The DoD AF Views As Requirements Vehicles in an MDA Systems Development Process"

Authors: Michael P. Bienvenu (The MITRE Corp.), Keith A. Godwin (The MITRE Corp.)

Working with the Joint SIAP Systems Engineering Office (JSSEO)

POC: Dr. Mike Bienvenu
The MITRE Corp
7515 Colshire Dr., MS H305
McLean, VA 22102
voice: 703-883-6253
fax: 703-883-1379
bienvenu@mitre.org

Deleted: 20

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

"The DoD AF Views As Requirements Vehicles in an MDA Systems Development Process"

Authors: Michael P. Bienvenu (The MITRE Corp.), Keith A. Godwin (The MITRE Corp.)
Joint SIAP Systems Engineering Office (JSSEO)

POC: Dr. Mike Bienvenu
The MITRE Corp
7515 Colshire Dr., MS H305
McLean, VA 22102
voice: 703-883-6253
fax: 703-883-1379
bienvenu@mitre.org

Abstract:

The C4ISR Architecture Framework was developed in the 1990's and has been revised as the DoD Architecture Framework, Ver. 1.0 [DoDAF]. Their use has become required on many DoD projects, and similar frameworks have been developed for use in commercial and other government agencies. However, the integration of these architecture products into the systems engineering process has been debatable.

We present a (hopefully) new and unique approach to this problem, one that uses the architecture views as "vehicles" for capturing and providing context for the operational and system requirements of the system under development. In addition, through the use of an integrated suite of COTS tools, we demonstrate that requirements at the software level can be imported into the software development environment and thereby attached to software design elements.

The architecture views are not developed as stand-alone products, but are developed on demand from the databases maintained in the COTS toolsuite.

Requirements are developed and attached to the architecture views at any of a number of different levels of abstraction: the operational level, the system level, or (a special case for our project), the distributed system level (a cooperating ensemble of system instantiations). **Architecture views serve to place requirements in context.**

Traceability between levels of requirements is done directly between requirements at those levels. Rather than the current C4ISR AF/DoDAF SV-5 mapping product, we maintain a much richer and rigorous correlation between requirements at different levels of the architecture.

This overall process has several significant advantages:

Deleted: 20

(1) There exists a single, integrated database of both requirements and architecture view elements, and there exists traceable ties between the two.

(2) The architecture views are relevant to the systems engineering process, and become de facto living documents with the evolutionary system design.

(3) The process relies on only minimal modification of existing COTS products, so that its utility across the community is enhanced.

Examples of the requirements and their accompanying architectural views are presented. The traceability from operational requirements to software requirements will be demonstrated.

Deleted: 20

1. Introduction

Section 1 of the paper discusses the overall purpose (Sect. 1.1), the scope (Sect. 1.2), the background (Sect. 1.3), and the intended audience (Sect. 1.4). The overall problem statement is presented in Section 2, which addresses the role of both architectures and requirements traceability within our particular project. The Technical Approach is presented in Section 3, which presents both our particular design objectives, the underlying theory, and the tool adaptations which were performed to accomplish this work. Future work is discussed in Section 4, and the paper closes with Conclusions in Section 5.

1.1. Purpose of the Paper

As part of the systems engineering task force for the Single Integrated Air Picture (SIAP) effort, we realized that the technical and managerial aspects of the program would be best served by re-examining how requirements, architecture, and design were interrelated. This motivation led us to re-examine and broaden our considerations of how the architecture representation would be related to the requirements traceability issue and the software development iterations. This paper is a report on the initial version of the toolsuite and process which was developed as a result.

This paper presents a framework and methodology for linking the architecture views (based on the views from the DoD Architecture Framework v1.0 [DODAF]) and requirements within a toolsuite. This methodology is currently being used to support the Joint DoD software development project for an Integrated Architecture Behavior Model (IABM). Within this methodology, the DoD AF is used as a systems engineering tool, bridging the operational-to-system-to-software architecture gaps which are much of the source of interoperability difficulties today.

Specifically, this paper presents an approach for linking requirements to discrete elements within architecture views created according to the guidelines of the DoD AF. Both the architecture views and the requirements model the desired system – but from different perspectives. If either model is incorrect or incomplete, there will be unacceptable program risk and a future cost to fix the errors. The traceability approach described in this paper helps to mitigate the risk of these errors by creating a mechanism to have both the architecture views and the requirements correlate to define the system.

The motivation for this approach represents somewhat of a difference from what is commonly believed with regard to the DoD AF architectures – namely, that these architecture descriptions intentionally are not supposed to be tied to the system designs or software development portions of a program. Too often DoD AF views are created merely to satisfy administrative requirements and are not perceived as a critical part of the system engineering effort. They typically do not provide genuine insight into system design or required capabilities. This issue is discussed further in the Conclusions section of this paper.

1.2. Scope

The primary focus of the paper is on techniques for attaching requirements to AF view elements. We do not address the overall process for requirements validation or analysis. While

Deleted: 20

this is obviously a key and crucial part of any system/software development effort, requirements analysis and validation are not addressed here. Basically, we assume for the purposes of this paper that the requirements are either "draft" (not yet validated or approved) or "valid", i.e., they have been approved by project management. The point here is not on the validation process – it is acknowledged there must be one – but on the impact of that process on the architecture/requirements traceability toolsuite – the toolsuite will be required to track these differences, and the progress of individual requirements through the overall process.

The paper also describes a proof-of-principle implementation, using COTS products with their inherent flexibility, to support the architecture and requirements traceability. The process presented here has been implemented within the JSSEO using Popkin's System Architect [Popkin SA] and Telelogic's DOORS [DOORS Ref]. We refer to these two tools operating together (a file-exchange based interaction) as "the toolsuite". There is also a bi-directional file exchange linkage of requirements into the JSSEO software development environment, the Kennedy-Carter iUML tool [KC iUML].

These tools can report on requirements status, produce traceability trees, and can generate documents in a variety of formats (html, word, etc). The goal is to provide integrated management for the architecture (what we're building) and the requirements (how good).

This paper does not address requirements validation or analysis. The requirements are assumed to be correct and have supporting rationale independent of their relationship with the architecture views. Also, while this paper describes an approach for recording traceability of requirements with the architecture, it does not discuss the analysis necessary to identify this traceability. While the methodology does not prove that a specific requirement is appropriate or good, it does support identifying the status of each requirement in the overall "approval" process.

A much more detailed discussion of the scope of the JSSEO Integrated Architecture can be found in the JSSEO IA v2.0, AV-1 Overview and Summary Information [JSSEO IA].

1.3. Background

The SIAP development effort puts tremendous stress on the relationship between requirements and architecture. The program has adopted Agile software development and the Object Management Group's (OMG) Model Driven Architecture (MDA) [OMG MDA] approach to constructing an Integrated Architecture Behavior Model (IABM) in executable UML. One of the consequences of this approach is that there is concurrent development of requirements, architecture framework views, and the deliverable system. The overall development schedule is very ambitious, calling for successive IABM "Configurations" to be produced at 2 year intervals.

The following definition of SIAP is found in the TAMD CRD: "The SIAP (the air track portion of the Common Tactical Picture (CTP) consists of common, continual, and unambiguous tracks of airborne objects of interest in the surveillance area. SIAP is derived from real time and near real time data, and consists of correlated air object tracks and associated information (such as Combat Identification (CID) information). The SIAP uses

Deleted: 20

fused near real time and real time data, scaleable and filterable to support situation awareness, battle management, and target engagement"[1].

1.3.1. Requirement Sources

There is no single source for SIAP operational requirements. As noted in the JSSEO Integrated Architecture (IA) AV-1, there are a number of CRDs (Capstone Requirements Documents) which bear on the objectives of the SIAP, but there has been no prior work to actually design the SIAP as it is now conceived. All requirements are not known at beginning of development. Even the desired "capabilities" are somewhat vague and subject to modification. Also, there is no single system that the IABM is replacing. The envisioned mode of "deployment" will be to integrate the IABM into existing host systems [JSSEO Config05]. Since it will be distributed across many systems, there is no single system architecture that will implement the IABM. Once requirements are elucidated and validated, they are addressed in the architecture and in the software design,

1.3.2. System Development

The managerial structure of JSSEO – the task force itself – presents a variety of interesting differences from "traditional" DoD acquisition programs and from commercial software development projects as well. JSSEO is composed of government, contractor, FFRDC, and university personnel, all operating together in a co-located facility, working all phases of the IABM development. There are no prime contractors or integrating contractors as the terms are normally used. JSSEO itself is responsible for the requirements refinement and derivations, the architecture development, and the IABM development. Changes in requirements do not require changes in contract costs.

However, JSSEO is also responsible for interacting with the eventual host systems to determine the specific steps necessary for integration. Thus, JSSEO acts as its own prime contractor, and cooperates with the host system program offices to achieve the goals of the integration contractors.

SIAP will be developed in a sequence of iterative development, and multiple planned deployments of increasing capability. The major deployments are known as "Configurations", and are scheduled at 2 year intervals. Within the development timeline of a Configuration, the developers work within a sequence of Timeboxes, each of approximately 8 weeks. At any given moment, two Timeboxes are running concurrently, in a staggered fashion. Requirements allocated to an overall Configuration are divided and allocated to Timeboxes for sequential implementation.

1.3.3. System Characteristics

The technical Nature of the SIAP software product does drive the overall development process and the tools required to support that process. Basically, the SIAP software product is a Platform Independent Model (as per MDA) intended to be integrated (as the IABM Implementation) within multiple, heterogeneous existing and in-development combat systems.

Deleted: 20

[1] Theater Air and Missile Defense (TAMD) Capstone Requirements Document (CRD), March 2001.

The software element is referred to as the Integrated Architecture Behavior Model, or, more accurately, the IABM Implementation. The objective is not to simply develop the IABM, but to deploy a distributed system composed of a large number of these IABM Implementations that has desirable behavior, interfaces and operational/system requirements, and that should appear, in certain aspects, as transparent to the multiple users.

Note that this system-level objective, a distributed system – to be more accurate, a distributed system composed of replicated components embedded in heterogeneous hosts –requires a variety of "system views" in a way that do not map cleanly to the standard DoD AF System Views. Requirements and architecture views can be specified for the operational, distributed system, and system (software application) views. This requires two levels of system views for this project – one for the distributed system, and how it interacts with externals, and between nodes of itself, and another for the software, and its immediate interfaces and required behavior. The other combat systems, into which a version of the software element will be integrated, are referred to as host systems.

System views can extend to lower levels of detail, within the xUML development environment, showing package (domain) and class structure, however, this is not within scope of this paper.

1.4. Intended Audience

This paper is written to assist systems engineers involved in programs where both the DoD AF and requirements traceability are important. It may also be read by technical managers of these programs, who may see applicability of the capabilities described here.

2. Problem Statement

The Operational, System, and Technical Views contained in the DoD Architecture Framework seek to accurately reflect the relationships between the various elements of the system and between the system and other systems. However, as currently defined, the architecture views do not directly identify the specific system requirements addressed by features within the architecture. Since architectures are an artifact of design, there are requirements which drive the selection of particular architecture choices. In order for the architecture to correctly model the desired system, all requirements must be addressed. Without traceability into the architecture, there is no practical method for determining if the architecture completely addresses system requirements.

Within the context of the JSSEO IABM development there are several aspects to the problem that bear upon the viability and feasibility of the approach (i.e., requirements that the process must satisfy in order to support the project):

- A product (the IABM) designed to be implemented in numerous host systems, with different peripherals, yet with the goal of consistent behavior across the eventual distributed system
- Iterative nature of the development, using OMG's MDA. Within JSSEO this results in a monthly cycle of product development – monthly 'timeboxes'

Deleted: 20

- Unique nature of the IABM "system" means that operational and other high level system requirements have never been developed within DoD for a similar system. This requires that we develop these requirements concurrently with the initial configurations of the IABM.
- The IABM must be easily sharable with other organizations. There are essentially no issues of proprietary software, since this work is being performed entirely with JSSEO oversight by teams composed of government, industry, and academic personnel.

3. Technical Approach

The discussion of our technical approach is divided into the following subtopics: the objectives of our technical approach (Sect. 3.1), a summary of architecture development (Sect. 3.2), the approach selected for requirements management (Sect. 3.3), the adaptation of selected engineering tools (Sect. 3.4), the technique for linking requirements to the architecture (Sect 3.5), and the use of the tools to support analysis and produce reports.

3.1. Objectives of the Process

One of our primary cornerstone concepts is that the architecture views must serve as "vehicles" for the requirements. This is one of the key foundation concepts for this work: **There should be no element of the architecture for which there is not a requirement (either originating or derived), and all requirements should be attachable to some element in some view of the architecture.** To put it another way, the views present a context, sometimes graphical, sometimes not, for the various requirements of the system. Also, sometimes the graphical elements themselves should be interpreted as requirements, for instance, the simple existence of a data entity or class on the SV-11 Logical Data Model should be interpreted as a requirement that these classes must exist in the implemented system.

It is important to understand the programmatic and technical requirements that the process set out to satisfy:

- "One Fact – One Place" The project is an ambitious one, in terms of both scope and schedule. This requires an efficient engineering process, which includes the architecture, the requirements, and the documentation. Thus, efforts to maintain redundant documentation could not be afforded.
- DoD AF views would be required by program management. They would also be products expected by the program offices of the host systems.
- The SIAP system was expected to come up with operational alternatives that were modifications of current practices – thus, operational issues were involved, not just software interfaces. Yet, the linkage between operational procedures and the software requirements could be clearly seen – some procedures required alternate messages and software processes, so these aspects of design were not independent.

Deleted: 20

Thus, we set out to design a toolsuite and accompanying process that will handle the needs and challenges of this rather unique project:

- Requirements in development, refinement, and flux at essentially every level of design
- Need to track Timebox-based implementation of requirements at a consistent timescale (i.e., monthly)
- Link and monitor the volatile requirements from top to bottom, without excessive manual labor in database maintenance when changes occur

Our basic goal was to construct a toolsuite in which we would construct the architecture with requirements traceability, which would be able to generate a spectrum of documentation directly from the toolsuite, maximizing the power of current information technologies. The intended documentation include not only the "standard" architecture views, but design documents, requirements traceability analyses and reports, and requirements implementation reports.

3.2. Architecture Development

The DODAF structure for architecture views helps to maintain the integrity of the system's design. They serve as the framework for identifying system relationships and functions. As a result, they inherently address system requirements. However, given their graphical nature, it is not easy to discern which requirements are addressed by each element of the architecture.

The DoDAF identifies six steps in the successful construction of an architecture – in a sense, we are constructing an architecture here – an infrastructure and process that will be used to capture the SIAP operational and system architecture views and the associated requirements. So the six steps are also valid for this process.

To answer the 6 steps outlined in the DODAF:

"Step 1: Determine Intended Use of Architecture":

The architecture will be used for the following purposes:

- Capture and relate of the operational and system requirements to the OVs and SVs. (i.e., link requirements to diagram elements, and establish traceability between requirements)
- Operational, distributed system, and top level software architecture and requirements will be represented.

Architecture repository (diagrams, requirements) will be used to produce System Specification-like documents, as well as the products fulfilling the same purposes as the DoD AF views.

The toolsuite must also respond rapidly to the iterative cycles of the software development process (the monthly timeboxes). Implementation of requirements in a specified timebox may or may not be successful, can be deferred to later timeboxes, or could be completed via several timeboxes (not necessarily sequential).

Deleted: 20

"Step 2: Determine Architecture Scope":

- Subject Area – is the SIAP embedded computer software program, and the operational activities of the host systems involved in the aerospace picture
- Timeframe – spans near-term (2005), through 2010, out to 2020.
- Intended Users and Uses – include both the internal SIAP Systems Engineering Task Force (which includes the developers of the IA Behavioral Model), as well as external interested parties, including JTAMDO, JROC, and the host systems' program offices.
- Dimensionality – The architecture shall encompass the operational level to the software system specification level. This is acknowledged to be an usually wide dimensionality, however, we felt it was required by the nature of the SIAP program.

"Step 3: Determine Data Required to Support Architecture Development":

Interpreting this to mean the data required as inputs to the architecture development process, then we begin with the relevant documents that bear on the SIAP development process, including:

- Numerous Capstone Requirements Documents
- Existing host system ORDs, system requirements documents, and specifications
- Existing and planned standards that affect either the existing systems, near-term systems, or the SIAP product itself.

However, this step points out a difference between what may be an underlying assumption of the DoD AF and our process – the DoD AF seems to assume that the design of the system under consideration has already been accomplished, and that this design material is then used to develop the architecture views. However, our process is based on a different assumption – the architecture views ARE the design and the requirements in an integrated repository.

"**Step 4: Collect, Organize, Correlate, and Store Architecture Data":**

Again, the naming of this step indicates a difference in underlying development assumptions. Our architecture views, being that they are the operational and high level design of our system, are in flux, and continually under development. Many of the activities assumed here occur as part of the continual design process within our system engineering task force. The architecture views and the requirements are all contained in an integrated toolsuite repository by the JSSEO Engineering Architecture Branch.

"Step 5: Conduct Analyses to Support Architecture Objectives":

Analyses are conducted within JSSEO at a number of levels, to support design trade-offs, sizing analyses (communications bandwidth, dynamic memory, etc.). These are conducted essentially continually throughout the iterative development cycles. However, as stated earlier, these analyses are outside the scope of this paper.

Deleted: 20

"Step 6: Document Results in Accordance with the Architecture Framework":

Again, a difference between the assumed DoD AF procedures and the process within JSSEO. Our integrated toolsuite is designed to produce a varieties of reports based on user needs. Separate efforts to maintain the documentation are not conducted, rather, documentation is produced from the toolsuite when required. Maintenance of the architecture/requirements repository is in fact the same as maintenance of the documentation.

3.3. Requirements Management

Operational requirements describe how the system will satisfy users needs. These requirements are the justification for all subsequent requirements. System requirements specify the behavior of the system in context with other systems. Subsystem/Software requirements describe the characteristics of subsystems and processes that work together to satisfy system requirements. The requirements traceability process must achieve traceability between different levels of views: both from the operational to the system and also from the system to the subsystem. A requirement that does not trace upward indicates development of an unnecessary capability. A requirement that does not trace downward highlights an incomplete solution.

We employ a comprehensive, yet manageable requirements tracking system. Instead of tracking blocks of requirements in a (version controlled) specification, each requirement is individually managed. This aids in supporting the rapid development cycle of the JSSEO timeboxes. The requirements database supports more than requirements traceability. It is the source of data for monitoring progress of systems development from a requirements point of view. Requirements traceability is only concerned with the requirement statement. Additional attributes of each requirement are characterized to facilitate traceability to other requirements and traceability into architecture views. The criteria and rationale that support the requirement are managed in other documentation.

Although DOORS has internal requirements linkage features, there is a need to be able to trace requirements when using other tools (such as SA and iUML). This is accomplished by assigning a unique identifier to each requirement. To simplify the discrimination of requirements at the various levels (operational, system, domain) three identifier prefixes are used.

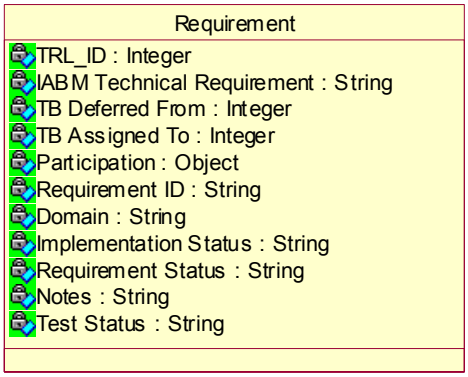

Figure 1: Detailed Attributes of a Requirement.

The detailed attributes of a requirement are illustrated in Figure 1. Several of these deserve additional discussion.

TRL_ID: the unique identifier assigned to the requirement. This is used when managing traceability across the various tools. In our management system three prefixes are used: ORL for operational, SRL for system, and TRL for the technical/domain requirements.

Deleted: 20

TB Deferred From: The timebox in which this requirement was initially planned to be implemented, but it was not completed. (Note: a timebox development cycle is currently four weeks)

TB Assigned To: The next timebox that will be used to implement capability against the requirement. May be the current timebox or a future one.

Participation: This is a list of the timeboxes in which work was done to implement this requirement.

Domain: The overall behavior model is subdivided internally into "domains", which are similar in concept to UML packages or software modules. The assignment of the requirement to the overall domain is recorded here. Domains directly correlate with domains in the KC- iUML tool.

Test Status: This attribute records the most recent test status of the requirement. This feature has not yet been fully exercised in the infrastructure.

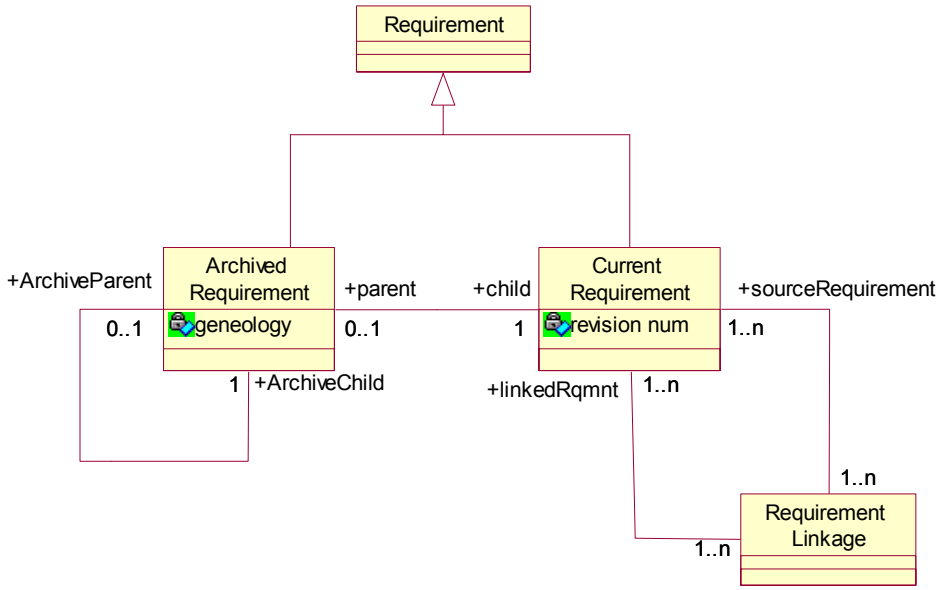

Figure 2. UML Representation of the internal metastructure of requirements.

Figure 2 presents a UML class model illustrating the relations between requirements and requirements linkages in our process. Messages that should be read from Figure 2:

Deleted: 20

- Requirements are of two kinds: Archived requirements and Current Requirements.

- Only Current Requirements are "linked" to each other, in the sense of requirements traceability. However, archived links are not removed from the database. They are made dormant once the requirement is no longer valid. There is no need to manually remove links for an inactive requirement.
- When requirements are superseded or cancelled, they become Archived Requirements. The "genealogy" of this history is maintained between the Current Requirement and its most recent ancestor Archived Requirement, and between all prior Archived Requirements.

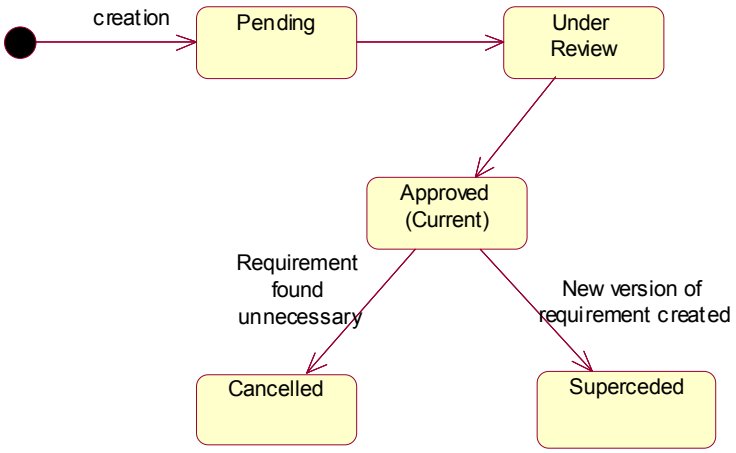

Figure 3: Notional Statechart for a requirement's lifecycle within the traceability system.

In Figure 3, a Statechart presenting the notional lifecycle of a requirement within the traceability infrastructure is presented. Note that Current Requirements (as referenced in Figure 2) must be in the "Approved (Current)" state. Requirements which have been archived (as per Figure 2) will be in either the "Cancelled" or "Superseded" states of Figure 3. It is intentional that there is no "end state" symbol on the diagram – a requirement, once created within the overall traceability system, is never destroyed. They can be cancelled or superseded, but they never leave the overall system.

Note that the assignment to an architecture diagram element is not a data field. This is due to the use of SA to graphically associate diagram elements to requirements. There is not a manual entry to make the association. It the architecture is modified, the requirement association is modified at the same time. This greatly reduces the risk of traceability errors in the mapping of requirements into the architecture views.

Deleted: 20

3.4. Tool Adaptation

Two readily available COTS tools were selected to support the methodology – Telelogic's DOORS for managing requirements and Popkin Software's System Architect (SA) for building the architecture framework views. In addition to having many standard features, both tools were easily modified to support our methodology.

DOORS is the main management tool for all our requirements. It serves as the repository for requirements at the operational, system, and domain levels. DOORS is used to maintain the formal traceability between requirements (operational to system and system to domain). Within our methodology SA is only used to show requirements traceability into the architecture views.

In addition to having features for requirements management and traceability, DOORS also has the ability to exchange requirements information directly with the IABM development tool. Requirements in DOORS can be assigned to domains within the The Kennedy-Carter iUML software and the KC tool can export requirements status to DOORS. In our methodology, the domain level requirements are exchanged between these tools.

System Architect has a default configuration for developing DODAF architecture view diagrams. This default configuration was modified to accommodate diagrams developed using UML notation. SA also has an extensible internal metadata structure. Although SA has a single default addressable for requirements, we customized it to have operational, system, and domain requirements. (This customization was made by modifying the SA usrprops.txt file.) The requirements exported from DOORS (in tabular Comma Separated Value – CSV format) are imported into the appropriate addressable in SA. The overall flow of requirements data between tools is illustrated in Figure 4.

Deleted: 20

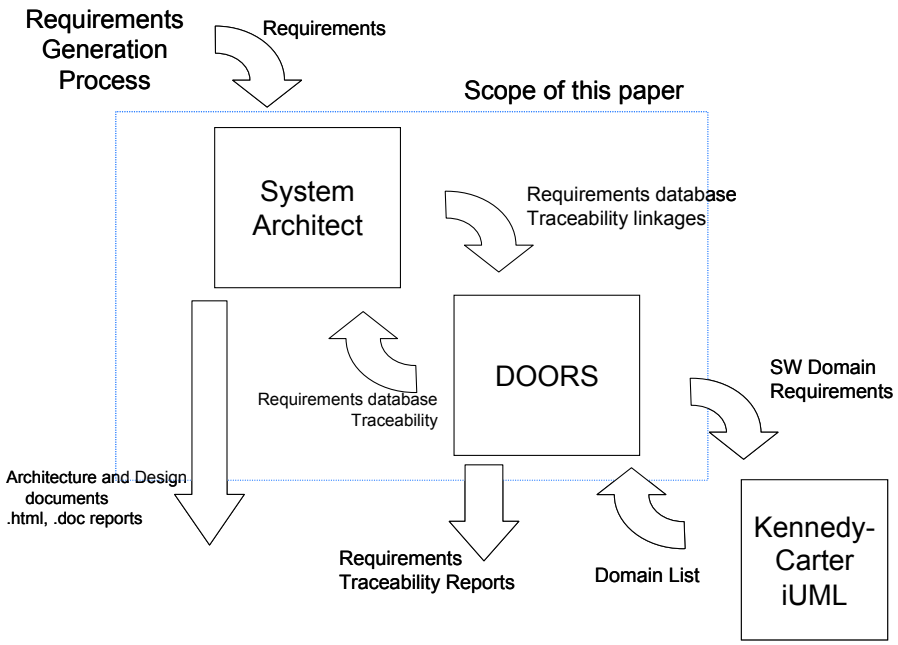

Figure 4 Schematic of the overall Architecture/Requirements Process

Additional modification were made to SA to enable the requirements to be associated with symbols on all the diagrams. As necessary, the SA diagrams, symbols, and definitions for each OV and SV were modified to accept requirements as addressables. These modifications were made in the SA usrprops.txt file. Corresponding to the three levels of requirements, three sets of requirements addressables were identified: Requirements-Operational, Requirements-System, and Requirements-T/IABM.

SA was also modified to use UML notation as part of the DODAF working template. By using UML we are able to directly correlate the system views with the IABM design architecture which is also in UML (within the K-C tool). SA was modified to add UML drawings to the default C4ISR framework working template.

All three tools have HTML reporting capabilities. Hypertext linkages were made between the reports using the unique identifier assigned to each requirement. This feature permits review and analysis of the architecture views and associated requirements without requiring expertise with each tool. It should be emphasized that the HTML reports can be constructed solely by the running of automated processes within the tools themselves and by Visual Basic scripts. No hand-coding or manual modifications of these files is required. Once the diagrams have been built in Systems Architect, and the requirements have been imported and linked (a manual process), the report generation is done solely through utilities and scripts.

Deleted: 20

3.5. Associating Requirements to Architecture Views

As is obvious from the IDEF1X representations of the various architecture views presented in the DoD AF Vol. II, there are a great number of different types of elements within the different views. We use the term "elements" to encompass all of the different graphical items which are used to compose the different diagrams. Elements can be symbols and the interconnecting lines between the symbols. Individual elements and the relationships between elements satisfy requirements and in some cases cause the generation of requirements. Multiple requirements can be allocated to a single diagram element. Also, a single requirement may be addressed by elements on multiple diagrams.

As the architecture matures and diagrams decompose to greater levels of detail, the requirements can be moved to more appropriate elements. The association of requirements to architecture views is dynamic. The intent is to always have the requirements address the current state of the architecture.

Figure 5 shows the basic concept. Requirements are attached to any diagram element. The set of diagrams to which we currently attach requirements, and the relevant diagram elements, are listed in Table 1.

Our methodology enables the development of architecture views and the derivation of requirements to occur in parallel. Neither has to be complete for the other to occur. The process of associating requirements to the architecture views assists in identifying the set of requirements necessary for each phase of IABM development.

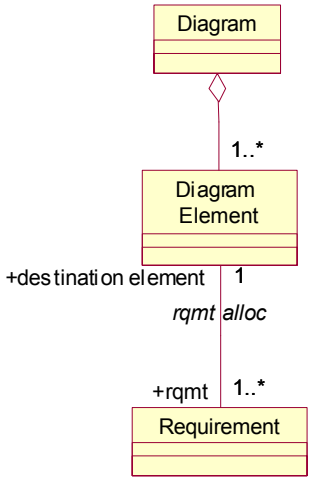

Figure 5. Requirements allocation to Diagram Elements.

Deleted: 20

Table 1: Current Views and Elements To Which Requirements Can Be Attached

DoD AF View	Diagram Elements
OV-2 Operational Node Connectivity Description	Operational Node Needline/IER
OV-5 Operational Activity Model	(IDEF0 Format) Operational Activity ICOM Line (Input, Control, Output, Mechanism)
OV-6b Operational State Transition Description	State Transition
OV-7 Logical Data Model	(UML Class Diagram Format) Class
SV-1 Systems Interface Description (UML Class/Instance Diagrams, systems level)	(UML Class Diagram Format – modified) Class/Instances Association/Interfaces
SV-10b Systems State Transition Description (UML Statechart Diagrams, for specific Object Classes)	(UML Statecharts – Kennedy-Carter iUML implementation)
SV-11 Physical Schema (UML Class Diagrams, detailed level)	(UML Class/Instance Diagrams -- Kennedy-Carter iUML implementation)

3.6. Analysis and Reports

The association of requirements to architecture views provides a mechanism for determining the completeness of both the architecture and the set of system requirements. The iterative assessment of requirements and the architecture aids in discovering requirements not yet addressed by the architecture and in identifying requirements that result from the architecture itself. These determinations are made by examining the requirements relationships contained in the reports generated by the tools.

Although both DOORS and System Architect have an internal report generation capability, we chose to use exported HTML files of requirements and architecture views for reviewing and assessing requirement traceability. Using HTML presented several advantages:

- It does not require expertise in the DOORS and SA tools to analyze requirements (all information is presented in a browser);
- Statistics can be collected by examining the unique identifier (ORL#, SRL#, TRL#) assigned to each requirement;
- Information can be hyper-linked to HTML files exported from other tools (such as MS Word, Excel, and KC iUML);,

Deleted: 20

- Data in the HTML files can be manipulated with standard scripting languages (such VBA).

The set of HTML reports exported from all tools is considered a single database from which traceability information can be extracted. Visual basic scripts are used to establish the hyperlinking between the individual files and to collect statistics concerning traceability. A quick review of the HTML reports highlights those architecture elements that are not yet traced to a specific requirements. If the architecture element is driving IABM design then a requirement should be identified.

Reports can be generated for subsets of requirements that address a single topical area.

4. Future Work

We plan to extend our use of UML as the notation for depicting all system views of the architecture. Work has begun on a concise statement of a revised set of products (views) needed for a project of this type, compared to the DoD AF standard SV products.

We also plan to extend the HTML links into documents that provide the rationale and other information supporting requirements. These documents include concept papers and formal requirement documents. The toolsuite is capable of being modified to include hyperlinks to these various documents at the appropriate points within the architecture or the requirements. Once this linking is added, the set of hyperlinked information will provide a comprehensive view of the system architecture and requirements that can be effectively used by those experienced with the program and by those new to IABM development.

We also plan to develop a capability to automatically generate requirements documents from requirements information maintained within the toolsuite. This feature will enable JSSEO to report on requirements in a manner similar to more traditional development efforts.

5. Conclusions

Building to the correct requirements is a key factor in the success of any program. Within our program both functional and non-functional requirements are needed to establish the success criteria for our product – the IABM. Traceability of requirements into the architecture views assists in determining the completeness of the requirements. This traceability has the additional benefit of aiding in determining the completeness of the architecture in addressing the requirements.

One of the benefits of this approach to requirements traceability is its adaptability to any program structure. The underlying premise is that Architecture Views should serve as the "vehicles" for requirements, whether originating or derived; operational, system or software. If there is no requirement for an element of the architecture, then it should not be included in the architecture.

Our process of requirements traceability into the architecture framework enhances our overall development process by making both the architecture and requirements more complete. Both

Deleted: 20

architecture views and requirements are needed for successful development. The architecture views simplify the understanding of the system while requirements make it easy to test and verify system performance. Together the architecture views and requirements give program managers, system engineers, and software developers insight into the objectives of the system.

Deleted: 20

References

[DODAF] DoD Architecture Framework Version 1.0, DoD Architecture Framework Working Group, www.c3i.osd.mil, Documents available at www.defenselink.mil/nii/doc. Volume I, Volume II, and the Deskbook, dated 15 August 2003.

[DOORS ref] DOORS product, version xxx, Telelogic Inc. www.telelogic.com,. Product website:

[JSSEO Config05] JSSEO Integrated Architecture Behavior Model (IABM) Configuration 05 Description Document, Version 1.0, 10 December 2003, JSSEO.

[JSSEO IA] JSSEO Integrated Architecure, v2.0, 27 Feb 2004. Joint SIAP Systems Engineering Organization.

[KC iUML]: *iUML* executable UML modeling tool, Version 2.2, Kennedy-Carter Ltd., http://www.kc.com,. Evaluation version available on the web at http://www.kc.com/download/index.html

[OMG MDA] Object Management Group's Model Driven Architecture http://www.omg.org/mda/. Description and specifications available at that website.

[Popkin SA] *System Architect*, version 9.1, Popkin Software . www.popkin.com

[TAMD CRD] Theater Air and Missile Defense (TAMD) Capstone Requirements Document (CRD), March 2001, Joint Theater Air and Missile Defense Organization.

Deleted: 20