Lessons Learned

in

Applying Architecture

to the

Acquisition of

Air Force

Command and Control Systems

Murray E. Daniels
AF Electronic Systems Center Chief Architect
The MITRE Corp.
Burlington Rd, Bedford MA 01730
781 377 9648
mdaniels@mitre.org

Ruth E. Sespaniak
Principle Information Systems Engineer
The MITRE Corp.
Burlington Rd, Bedford MA 01730
781 377 8269
res@mitre.org

Abstract

At the Air Force Materiel Command Electronic Systems Center, we are using architecture in multiple programs to support the acquisition of Air Force Command and Control systems. This use ranges from informally augmenting traditional requirements documents to actually delivering architecture products to the contractor as a formal representation of requirements. We also use architecture to ensure that programs better understand their operational and system context within the enterprise. The architecture activities described in this paper span the timeframe from 1997 to the present. During that time, we've had varying degrees of success. On the one hand, we've found the architecture to foster significant communication between the user, the acquirer, and the developer – mostly by having operational subject matter experts work directly with the acquirer and contractor in developing parts of the operational architecture views. In general, we have found that the use of a disciplined approach to architecture helps all stakeholders better understand the operational, system, and technical context in which they must operate. Conversely, we've found that additional attention is needed relative to working with contractors to incorporate the use of architectures into their system/software development processes and in applying architecture to support Enterprise Integration.

Introduction

Acquiring systems of any complexity, such as command and control (C2) systems, requires a disciplined approach to representing both the problem being addressed and the solution being acquired. The practice of information systems architecture, as it has evolved over the years and as codified in the DoD Architecture Framework (DoDAF) [reference 1], attempts to provide this discipline. There are at least two significant goals for doing this. First, we obviously want to share architecture best practices and lessons learned so that we don't have to start from scratch developing new ways to represent problems and solutions each time we acquire a new system. Second, we want to provide some architecture practice consistency across acquisition programs in the hope that this will help foster greater commonality and interoperability among the end products – the acquired systems.

At the Air Force Materiel Command's Electronic Systems Center (ESC), we are using architecture in multiple programs to represent requirements and to guide solution designs. We also use architecture to ensure that programs better understand their operational and system context within the enterprise. We've had varying degrees of success. On the positive side, we've found the architecture to foster significant communication between the user, the acquirer, and the developer. Conversely, we've found that additional attention is needed relative to working with contractors to incorporate the use of architectures into their system/software development processes. Finally, we believe that the use of architecture to support enterprise integration, while very promising, is still in its early stages of development.

This paper reports lessons learned on three large representative C2 programs. Two of the programs are updates to existing systems while the third is the acquisition of a significant new capability. The programs' identities remain anonymous so as to foster a more candid reporting of lessons learned.

Architecture as a Means to Represent Operational Requirements

There is no single, widely accepted definition of what is meant by "architecture-based acquisition." Some believe that it requires starting the acquisition from "architecture-based requirements" – although there is no agreed-upon definition of exactly what that means either. In this section, we report the experience of three ESC programs in using architecture to represent operational requirements.

Program A provides for the migration of multiple legacy systems to a more network-centric, enterprise-based system while utilizing evolutionary acquisition to support evolving requirements. The program also includes the sustainment of the legacy systems until they are migrated and eventually decommissioned. In this program, ESC and the operating MAJCOM jointly developed operational architecture views (OV's) using an object-oriented methodology and intended to use them as the requirements baseline for the program. The architecture development process also supported subsequent business process reengineering efforts by the operating MAJCOM. The OV's were included in the

RFP package and the bidders were required to submit system architecture views (SV's) as part of their proposal. The OV's have continued to evolve with the MAJCOM producing a validated major release on a yearly basis.

While the OV's were part of the RFP package, they were not put on the contract due to their need to continue to evolve separate from the acquisition effort. Instead, the contractor used the OV's to develop separate documentation detailing the functional requirements for each development increment. This more traditional development specification then went on contract along with separate documentation detailing performance-related requirements and legacy system sustainment requirements. It was intended that each increment's functional requirements documentation would be directly traceable to the OV's; however, this didn't work in practice as the contractual requirements documentation tended to be more general than the OV's making it difficult to provide the traceability from OVs to requirements to implementation.

Program B is also a major upgrade to a legacy system involving a move to a common infrastructure to provide a more distributed, net-centric capability. The initial increment in this program is more of an integration effort than the development of new functionality. In this effort, the ESC System Program Office (SPO) harmonized existing OV products developed by a separate government user organization with "as-is" SV products it had developed to represent the fielded legacy system. The SPO then developed and coordinated high-level "to-be" SV products with the users. The SPO presented these architecture products to a general industry forum prior to RFP release and then included them as part of the RFP package. These architecture products were not, however, validated in any formal sense.

Lacking a validated architecture heading into source selection, the ESC team developed a more traditional Technical Requirements Document (TRD) and a Statement of Objectives (SOO) that directed the contractor to develop specific OV and SV architecture products using the Unified Modeling Language (UML) specification [reference 2] and the Popkin System Architect tool [reference 3]. They also provided the contractor with government-generated OV products that had been coordinated with the users but were not formally validated. The contractor developed a fuller set of DoDAF views, to include operational views, that were again harmonized with the users. The contractor mapped their products to the government products thus providing a reasonable level of confidence that their OV products were accurate. The ESC team continues to work closely with the contractor as they develop their architecture products.

The Program B contractor also uses DOORS [reference 4] to manage requirements and this has allowed some requirements traceability to the architecture through the DOORS-Popkin System Architect interface capability. While the DOORS database is not a formal deliverable, the contractor has made it available for review by the Government. However, the requirements database is extremely large which has made it difficult for the Program Office to do an in-depth analysis to ensure that all requirements have been accurately mapped to the architecture.

Program C is developing a new capability requiring multiple contractors led by an overall integration contractor. In this program, architecture is viewed as a tool used in support of systems engineering. Program-related personnel develop specific architectural products in response to particular program needs. The focus is not on developing the definitive overall architecture to guide program development but instead on building a core set of architectural data incrementally as needed that can be reused as necessary to support key systems engineering analyses. There is no single overarching architecture, but rather multiple architectures. There are "driving" architectures, such as the enterprise-level C2 mission and infrastructure architectures as well as program-level architectures (such as an operational views developed by the SMO and system/technical views developed by the SPO and multiple contractors). These architectures vary in scope, level of detail and intended purposes.

Program C's SPO personnel also developed a separate TRD. The TRD references the enterprise-level architecture OVs, but there is no requirement on the contractor to develop specific architecture products or to utilize the DoDAF over any other architecture framework. However, it is anticipated that since the program and enterprise-level OVs do utilize the DoDAF and are referenced by "shall statements" in the TRD, the contractors will opt to generate DoDAF products.

The program has also required that the contractor use architecture-based processes. Again, while there is no widely accepted definition of exactly what this means, the SPO is trying to be proactive in determining how it will use architecture throughout the program and in communicating that to the contractors. The program has been using architecture since program conception to help understand the requirements, determine whether the requirements make sense, identify missing requirements, determine risk areas, and address interface disconnects. The program is currently using the architecture as a communication vehicle among the program office, the integration contractor, and the user to clarify requirements. Requirements documents "shall statements" are being traced to UML diagrams. The program expects to have good traceability between the requirements and the architecture but it is still early in the process.

Architecture as a Means to Drive System Design

It was anticipated that the use of architecture-based requirements would facilitate system design in much the same way that the architectural diagram of a building is used to drive engineering design plans for that building. While the use of architecture in building design is part of the accepted process within that domain, this is not yet the case in the development of C2 systems. The introduction of architecture (that is, rigorous architecture as defined by the DoDAF) into the acquisition of C2 systems represents a paradigm shift in the design of C2 systems that is still underway.

In Program A, the government developed technical architecture views based on the then-extant Joint Technical Architecture (JTA) [reference 5] and included them in the RFP package along with the operational views. The bidders then submitted a system architecture view that corresponded with the problem space (functional requirements)

described in the operational views and the technical constraints described in the technical views as part of their proposal. The government has not updated the technical views since contract award. The intent was that the contractor would recommend changes to the technical views as part of their evolution plan.

The hope was also that the contractor supplied system views would flow naturally in a single tool-based environment from the operational views all the way through to system implementation. For a variety of practical considerations (see issues below), this did not occur; however, the operational and technical views did serve as a significant communication mechanism between the government and the contractor as they developed their system design and eventual system implementations.

Similarly, in Program B the operational and system architecture products are serving to primarily facilitate communication both within and across contractor and government teams – but in this program this is one of the primary purposes of the architecture. The architecture is utilized throughout the acquisition process serving as the basis of discussion at every major review. The contractor developed specifications and systems-oriented use-cases from the architecture-derived requirements and developed a corresponding set of Use Cases. In the process of developing and refining the Use Cases, they discovered some common Uses Cases that could potentially be exploited to produce a more efficient system. Throughout this process, program personnel noted that having operational subject matter experts (SMEs) work directly with the contractor in developing the Use Cases was very effective in the requirements interpretation process and the developers ended up with a much better understanding of the operational requirements as a result. However, the contractor is not utilizing the architecture tool as a development tool, opting to use a different tool instead. This disconnects the design and implementation from the government- and contractor-developed architecture products and adds a degree of configuration management complexity.

In Program C, SPO personnel elected not to reference the evolving OSD and other net-centric guidance/direction documentation directly in the technical requirements but rather to distill this guidance into program-specific technical requirements. A primary rationale for this approach was the multiple, and sometimes conflicting, sources of net-centric guidance and its dynamically evolving nature. As new/modified guidance is developed, the technical requirements are updated as part of an ongoing technical exchange between the government and the contractors.

The program is requiring that executable models of the software be developed to demonstrate capability. The program office is looking to use these models to address schedule and performance aspects of the program. The program also constrains the contractors to use UML but leaves the choice of tool to the contractor. At this point in time, the program is still in the early phases and no actual design work has commenced.

Architecture as a Means to Support Enterprise Integration

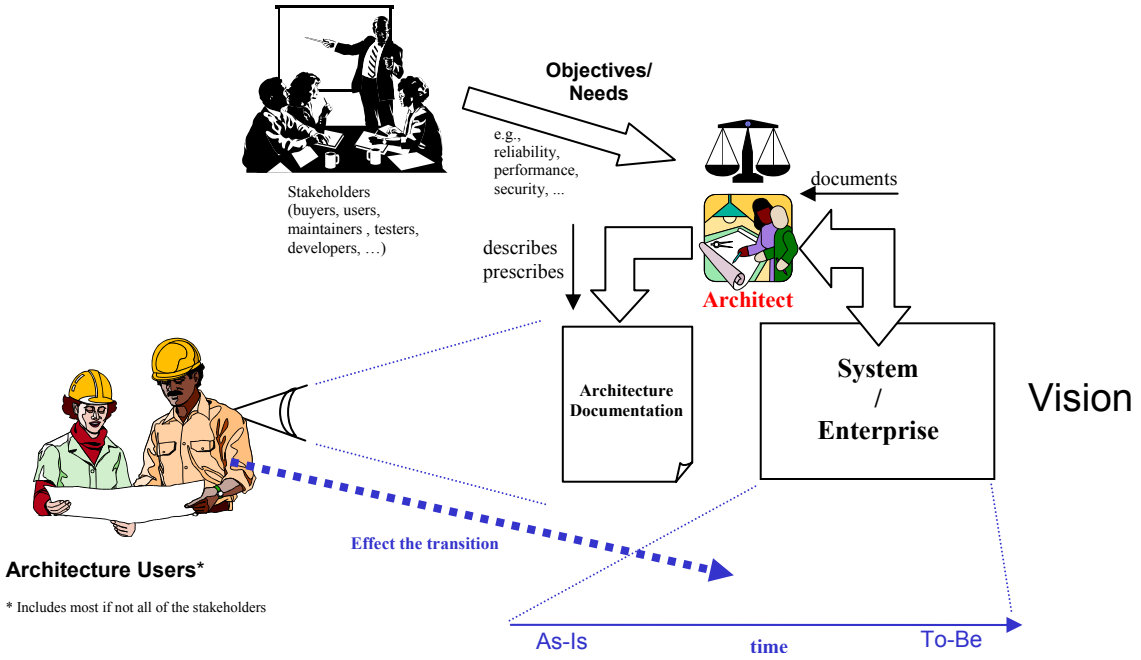

Figure 1 – The role of the Architect and the Architecture

Multiple stakeholders have a vested interest in the success of ESC's programs. This includes the PEO, users of various kinds and their representatives, program element managers, Air Staff and DoD policy makers, contractors doing development and maintenance, vendors providing COTS products, SPO personnel, testing agencies, system engineering support personnel, Congress, and the public at large. These stakeholders together contribute to defining the overall Vision for the programs as individual entities and as collections of entities (an "enterprise") that together provide integrated warfighter capabilities. This Vision is defined as some "optimal" balance of objectives and needs to include certain functionality and performance within a context of various (often competing) "-ilities" such as "out-of-the box" interoperability, reliability, flexibility, securability, testability, and affordability. It is the job of the architect to help the stakeholders come together to define this Vision (optimal balance) and to then lay out a technical strategy for how to achieve this Vision at both the enterprise- and program-level.

The architecture's operational and system views describe the current "as-is" state (to some degree) and potentially many future "to-be" states. These future states may be characterized as "planned" or "could-be." The architecture's technical views also prescribe how to move from the as-is to the to-be state. The architecture documentation represents the data underlying the architecture in a form that can be used by multiple interested parties for multiple purposes. A primary purpose is to effect the transition from the as-is to the to-be state.

The architecture represents multiple facets of the program or enterprise – notably much of the range of DOTMLP-F (doctrine, organization, training, materiel, leadership, personnel, and facilities). While ESC personnel generally acquire electronic systems, the architecture must include, and in fact be based upon, the operational context in which those systems are used. While it is possible to extract the "systems view" from the architecture, it is not proper to think of the "systems architecture" as a separate entity from the overall architecture.

While the development of program-level architectures is relatively well-understood, their development and use at the enterprise-level is not. At the enterprise-level, ESC's primary products conceptually include:

- An overall systems architecture view for the enterprise
- An overall technical architecture view for the enterprise
- Contributions to the Air Force Enterprise Architecture (AF-EA) Reference Models [reference 6]

The enterprise-level architecture products are more generalized (less detailed) than their program-level elements. The enterprise systems architecture view describes what systems make up the enterprise, what they do, and how they are inter-connected. The systems architecture view includes descriptions of alternative enterprise-level system deployments. The technical architecture view provides the common technical principles (building codes) for building the systems so that they all meet some set of desired characteristics.

The AF-EA Reference Models are an organized set of common reusable architecture entities (e.g., terms, profiles, standards, guidance, performance objectives and measures) that allow architectures at multiple levels to be interrelated. ESC has a key role in developing the AF System Function/Service Reference Model (SRM) and the AF Technical Reference Model (TRM).

As stated above, the systems architecture view can cover multiple timeframes to include an "as-is", "planned", and potentially several "could-be's." Ideally, the "planned" and the "could-be" architecture timeframe views are derived from time-stamped architectural information about the systems such that an observer could request a snapshot view for an arbitrary future time of their choosing. Barring this, there will be "planned" and "could-be" snapshots for pre-selected timeframes.

Imbedded in the multiple "could-be" system views is a certain amount of engineering and programmatic analysis leading to an assessment of the military value of these alternative architectures. This analysis could, for example, look at the various distribution strategies for a particular architecture based on intended operations and assuming various infrastructure lay-downs. The chosen architecture captures the results of the analyses in a form that facilitates enterprise engineering and integration.

These architecture products combined with program requirements become guidance and direction on our contracts with industry thereby supporting the integrated enterprise objectives of the AF and DOD.

In practice translating enterprise architecture into program requirements and guidance is still an evolving process. Program A was an early an adopter of architecture and predated much of the more recent efforts to develop enterprise level architectures. As a result it has become a cornerstone of the MAJCOM's enterprise architecture work with other program architectures being developed to align with the Program A architecture. On the other end of the spectrum, Program C was initiated in conjunction with the development of the AF-EA and incorporated a need to align with this enterprise architecture into the development of its program requirements. Many current C2 programs and enterprise level architectures fall somewhere in between these two cases with program and enterprise architecture products being developed in parallel. Thus enterprise guidance may necessitate a change in program requirements with potential impacts to cost and schedule.

Net-Centric Operations is a clear example of enterprise level direction coming down to the programs and all three programs are working to address this direction. A key challenge has been how to address net-centricity and Service Oriented Architectures (SOAs) with the DoDAF architecture products. The DoDAF products were originally developed with more of a point-to-point interface in mind and need to be tailored to address the concept of information services. Program C has elected to develop extensions to the SV-6 product to address information services by linking in meta-data such as information service name, access point, and the behavior and performance of the service. As part of their activity in this area, they recognized a need to introduce a higher level of abstraction into their depiction of interfacing elements by using high level node categorization to bring the amount of architecture data to a more manageable level. Program B has explored utilizing the OV-3 product to capture SOA-related data recognizing a similar need to somehow categorize receiving nodes to get the number of elements down to a manageable level. Program A has been approaching this area from more of an object-oriented focus with less emphasis on the DoDAF products. Their approach is centered on capturing transactions with systems as an interface class.

Issues

As implied above, there are several issues associated with the practical job of using architecture to deliver integrated warfighter capability. We mention a few of them here.

While the notion of architecture and the DoD Architecture Framework have been around for some time, this shift in culture and process is still oingoing. Contractors typically have existing non-architecture-based processes in place using traditional requirements and specifications documents. While the two are not incompatible, keeping two representations of essentially the same information synchronized takes extra work. This is true throughout all phases of the acquisition process. For example, while the architecture's Use Cases could be an excellent tool to drive testing (vs simply support it),

this is seldom the case in practice. However, we did find some success stories. In one case, the original contract called for final delivery of the architecture at the final design review. However, it was recognized early on that the architecture work needed to continue as the system continued to evolve in later phases and the government reprogrammed funding to continue the architecture effort to capture the "as built" system.

There are at least two significant methodological approaches to architecture – structured analysis and object-oriented – with advantages and disadvantages for each. They also have some degree of incompatibility. Some argue that the object-oriented approach is more suited at the program-level as it can lead potentially lead to direct software implementation and several of our programs have indeed made a move in this direction. The DoD and AF-level enterprise architectures, however, tend to be developed in structured analysis.

The existing DoDAF architecture products do not capture performance related requirements overly well. The only product specifically focused on performance requirements is the SV-7, Systems Performance Parameters Matrix., and it is focused on the solution set. What is specifically missing is an explicit characterization of the various "-ilities" requirements (stakeholder needs) associated with business processes to include temporal, reliability, and security performance.

The selection of architecture tools is frequently a hotly debated topic. Program A selected their tool based on what they felt was most likely to be useful to the contractor in the development of the system. Thus they utilized a development tool for their architecture development. While the contractor also adopted the same tool, their process did not address using the operational architecture products in their development effort. So despite using the same tool, this did not result in the close coupling of the system design to the operational architecture that was originally envisioned. Program B took the approach of specifying the use of a government-preferred tool for architecture development. However, the contractor did not utilize that tool in their coding efforts but rather elected to use a more appropriate code development tool that does not link to the architecture tool making traceability from the architecture to the code more difficult.

Over time it has become apparent that an important factor for incorporating architecture into system acquisition was the underlying architectural data and not the particular form or tool that is chosen. The architecture is the underlying data and not the products in which the data is presented. In addition, no single entity can develop all of the needed architectural data and therefore we need to coordinate and/or align multiple architecture development efforts by developing a common understanding of key architectural data. Program C has adopted an architectural data focus where there is no single overarching program architecture. The emphasis is on developing a common set of architectural data that can be used to support the various analyses that the program elects to undertake. This approach requires the development of a configuration management process for the various architectures that are developed to support the program to ensure that these activities are coordinated and that the common architectural data is maintained. Initial

results appear promising; however, it is too early in the process to determine whether this will lead to a system design that is truly architecture-based.

Finally, when inevitable program difficulties arise, the tendency is always to go back to the approach you know to meet the needs of the moment. Typically, both government and contractor are less willing to try to work with the new approach (architecture) and become more willing to sacrifice the perceived long-term benefits to address the near-term delivery needs.

Summary

Table 1 presents a summary of the architectural attributes of the three programs discussed in this paper. In general, architecture has proved useful in the acquisition of C2 systems but it has had the inevitable bumps in the road of any new approach. Architecture has proved to be a very useful communications tool in the requirements development and refinement/clarification process. It provides a common basis on which to discuss the requirements and achieve a common understanding between the user, the acquirer and the developer. However, to more effectively use architecture in this area requires corresponding changes in existing government and contractor acquisition and development processes.

	A	**B**	**C**
Intended Use	C2 requirements capture for system(s) acquisition	Description of "To Be" capabilities for system acquisition	Generation of architecture data to support key systems engineering decisions during system acquisition
Program Status	Initial increment operational, next increment under development	Initial increment entering testing	Integration contracts awarded, in requirements refinement phase

	A	B	C
Views/Products	AV-1/2, OV-1/2/3/4/5/6c/7 TV-1	AV-1/2 OV-1/2 SV-1/2/5 TV -????	Focus is on common core architectural data vice specific architecture products. Multiple sets of architecture products being developed to support key SE decisions. Specific products developed driven by underlying information needs of a particular decision.
Tools/File Formats	Rationale Rose MS Office Limited use of Popkin SA	Popkin SA	Rhapsody Popkin SA
Level of Detail	Detailed description of required system behavior	Detailed description of capabilities	Varies from high level to detailed description depending on specific area under investigation.
Methodology/ Strategy	UML-based approach organized along MAPE construct and 3-tier approach (role, operational node, physical node)	UML-based approach using 3 tier structure (A-level, B-level and C-level)	UML-based approach using 3-tier structure (A-level, B-level, C-level)
Fundamental Architecture Entities	Use cases, sequence/ collaboration diagrams, IER database, activity diagrams, class diagrams, operational trace sequences	Use cases, IER database?, activity diagrams?	Use cases, IER database?, activity diagrams?

Table 1 Program Architecture Attributes

The extension of architecture use into system design has been somewhat less successful – largely due to the fact the contractors' existing processes have not traditionally been architecture-based. Further study on how to incorporate architecture data into existing development processes will be needed to make improvements in this area.

Architecture usage at the enterprise level shows significant promise but again suffers from a lack of a common understanding on how architecture data will be used to support enterprise integration activities. There is much to be gained in this area to use architecture to show how programs can more effectively and efficiently fit together.

A common theme across all three levels of architecting is the need to understand how architecture fits into the business processes it is intended to support. Architectures are most useful when they are addressing specific concerns. A "one size fits all" approach to architecture is not achievable. A key to success in this area is to focus on the data underlying the architectures and to strive to align the data across multiple architectures at multiple levels where appropriate identifying potential reuse and interoperability opportunities along the way.

References

1. Department of Defense Architecture Framework, V1.0, 15 Aug 2003, DoD, http://www.defenselink.mil/nii/doc/DoDAF_v1_Volume_I.pdf
2. Unified Modeling Language, Object Management Group, http://www.uml.org/
3. Popkin System Architect modeling tool, http://www.popkin.com/
4. DOORS requirements management tool, http://www.telelogic.com/
5. Joint Technical Architecture, Defense Information Systems Agency, http://jta.disa.mil/jta/jta-vol-I.pdf
6. AFPD 33-4 Enterprise Architecting (Draft) and AFI 33-4xx Enterprise Architecting (Draft)