Architectural Frameworks: Defining the Contents of Architectural Descriptions

David E. Emery

The MITRE Corporation
1820 Dolley Madison Blvd, MS W538 McLean, VA 22102-3481 USA
emery@mitre.org, +1 703 883 7606 v, +1 703 883 6143 fax

Abstract. This paper describes experiences with several architectural frameworks. An "architectural framework" specifies what is included in the description of an architecture, independent of the specific system being described. The three frameworks are the U.S. DoD C4ISR Architecture Framework, the associated Core Architecture Data Model and the emerging IEEE Recommended Practice on Architecture Description. From these experiences, we speculate on the further evolution of architecture frameworks and architectural descriptions.

1 Introduction

The term "architecture" means many things to many people. Initially, there was no 'formal' defininition of the term. Users of the term relied instead on intuition and the analogy with other disciplines, such as structural architecture and landscape architecture. Within the last two years, though, there have been many attempts to add structure and rigor to the notion of 'architecture', resulting in several different approachs to defining what constitutes an architecture. These attempts have defined "architecture" by defining how to -describe- architectures.

These examples give us a notion of an abstract "architecture" with many possible "architectural descriptions", where the contents of an architectural description (the concrete representation of the architecture) is established by an "architectural framework". Thus an "architectural framework" is a specification of how to describe architectures, rather than the definition of a specific architecture.

This paper relates the author's experience with several different architectural frameworks. There are many architectural frameworks and approaches in the literature; we concentrate here on those that are used to describe the systems aspects of software-intensive systems, rather than the more specific notion of "software architectures." (A survey of architectural frameworks can be found in [9]). Architectural frameworks are important, since the value of architecture comes as a communications medium, and a common "language" (as defined by an architectural framework) is needed to compare architectures.

M. González Harbour and J.A. de la Puente (Eds.): Ada-Europe'99, LNCS 1622, pp. 64–75, 1999.

1.1 Characterizing Architecture Frameworks

An architectural framework tells the user how to describe an architecture. The framework may imply a methodology, but the intent of the framework is to establish the contents of a (conforming) description of an architecture. Generally, a framework consists of a definition of "architecture" and related terms and concepts, along with the definition of one or more "views", or representations. The architecture is described by a set of these views, where each view conforms to the requirements of the framework. Some frameworks may place additional requirements on the set of views (e.g. cross-view consistency rules).

2 DoD C4ISR Architecture Framework Document

This section discusses the U.S. Department of Defense Command, Control Communications and Computers, Intelligence, Surveillance and Reconaissance (C4ISR) Architecture Framework document. [16]

2.1 The C4ISR Architecture Framework

The C4ISR Architecture Framework was developed in response to U.S. Department of Defense need for a coordinated approach for developing, integrating and using Command, Control, Communications, Computers, Intelligence, Surveillance and Reconnaissance (C4ISR) systems. Operation Desert Storm showed both strengths and weaknesses in combining systems developed by each military service into an integrated system for the Commander in Chief. One of the most notable examples from Desert Storm is that the Navy and Air Force were reduced to exchanging air tasking information on floppy disks, rather than through any interactive or automated process.

The Framework document prescribes that architectural descriptions will be oriented around three views, with a series of products for each view. These views are

- Operational View: tasks and activities, operational elements and information flows required to accomplish or support a military operation. (Often abbreviated "OA")
- Systems View: descriptions, including graphics, of systems and interconnections providing for, or supporting, warfighting functions. (Often abbreviated "SA")
- Technical View: minimal set of rules governing the arrangement, interaction and interdependence of system parts or elements, whose purpose is to ensure that a conformant sytsem satisfies a specified set of requirements. (Often abbreviated "TA")

For each of the three views, the Framework document specifies a set of "products". A "product" is a document that contains information relating to some aspect of the architecture. Products are classified as "Essential" or "Supporting", with Essential products containing the minimum information needed to

understand a given architecture. Products are specified by examples in the Framework document, rather than through a rigorous Data Item Description. An architectural description is considered conformant with the Framework if it provides all of the Essential products and appropriate Supporting products, and if it uses terms and graphics consistent with the Framework document and normal DoD usage. Thus the C4ISR Framework approach sees architecture as a series of documents.

The Framework document has undergone a trial use and review cycle, producing Version 2.0, which has been mandated for use within the DoD. Each military service is responsible for determining how to implement the Framework document.

2.2 An Implementation of the C4ISR Framework

The U.S. Army has adopted the C4ISR Framework for describing how Army units and their command, control and communications systems are connected. This is described in the Army Enterprise Architecture (AEA) [14] document. The AEA document establishes responsibilities for the three views of the C4SIR Framework. The Operational View is the responsibility of the Army's Training and Doctrine Command (TRADOC), specifically the TRADOC Program Integration Office for Army Battle Command Systems at Ft. Leavenworth, KS. The Technical View is the responsibility of the Office of the Director, Information Systems, Command, Control, Communications, Computers (DISC4) at Army Headquarters in the Pentagon. Responsibility for the System view is spit between the Army Signal Center, Ft. Gordon, GA and the Program Executive Officer for Command, Control and Communications Systems (PEO C3S) at Ft. Monmouth, NJ. The integration responsibility for the Army Enterprise Architecture lies with DISC4 in the Pentagon.

The Army has adopted the notion that each C4ISR Framework View can be done relatively independently of the other views. This is clear by the fact that each view is done in a different location. Furthermore, the Army specifies a relative ordering for the various views. The Technical View, as defined by the Joint Technical Architecture-Army document [13] provides the core interoperability standards for Army systems. The Operational View is focused on the concept of "Information Exchange Requirements", which represent the flow of information from one element to another. For instance, one IER specifies that orders flow from the Battalion Commander to the Company Commanders, and another specifies that periodic spot reports flow from the Company Commanders back to the Battalion Commander. According to the Army's approach [14], both the Operational and Technical Views should be developed before work starts on the System View.

The Army has divided the Systems View into two parts. The first part, which is always completed first, is called the "Conceptual Systems Architecture" ('SA-C'). This part is developed by the Army Signal Center, and identifies organizations and equipment. So, from the example in the previous paragraph, the SA-C would specify that the Battalion Commander and each Company Commander

has a radio on the Battalion Command Net which is used to pass the orders from Battalion to Company, and the spot reports back from Company to Battalion. The second part of the Systems view is called the "Detailed Sstems Architecture" ('SA-D'). This work is performed by PEO C3S, and includes all of the specific pieces of equipment needed to implement the actual system architecture. For example, if the Battalion Command Net is implemented using a Packet Radio system such as EPLRS, then the Detailed Systems Architecture must ensure that there are sufficient packet nodes available to cover the battalion area.

With the "digitization" of Army tactical forces, much of the traffic formerly carried via voice radios is now carried as TCP/IP packets. So a Battalion may have both a Voice Command Net and a Digital Command Net. One of the main funtions of the Detailed Systems Architecture is to ensure that these digital nets have an appropriate network structure, including IP subnet definitions, routers, security firewalls, and related digital network gear.

Each architecture View is associated with one or more products. The Technical View is expressed by a single product, the Joint Technical Architecture-Army (JTA-A) document.[13] The JTA-A document extends the DoD Technical Architecture [15] definition by adding additional Army-specific requirements to the DoD baseline. (One notable Army addition is the Army's continuing use of Ada.) In this respect, the JTA-A looks like a profile of the JTA document. The Operational View is captured in a database of IERs and related requirements, along with publications that list the IERs, expected battlefield layouts and related doctrine. The Systems View products are captured using a network diagramming tool called "netViz" [10] that provides an interactive way to view Army organizations, the equipment assigned to organizations, and the networking of that equipment. There is also a database representation of this information with an on-line query capability, and a representation that shows only units and equipment (without network connections). This latter representation is widely used to make sure that the right equipment is being procured and delivered to the using organization. The Army implementation of the C4ISR Framework represents architecture as both documents and CASE tool databases.

3 Core Architecture Data Model

This section describes the C4ISR Core Architecture Data Model (CADM). The CADM captures the intent of the C4ISR Framework Document by describing the various data elements that can be used to describe architectures, and how these data elements interrelate.

3.1 IDEF-1X Data Modelling

Data Modelling is a formal process built on the Entity-Attribute-Relationship (ERA) model. A data model captures the data elements and relationships in a fully normalized form. The most common use for a data model is to capture the requirements for a database, but data models are also commonly used to capture data interchange requirements. [2] [4]

3.2 CADM Goal: Capture C4ISR Framework Architectures

The CADM attempts to capture the core data needed to represent the architecture products defined by the C4ISR Framework Document. The Framework document, as a text document, is subject to substantial interpretation by its users. The CADM represents the belief that ultimately architectural representations are really databases of information about architectures.

The C4ISR Framework document does not specify any particular methodology or approach for producing architectural representations. As previously described, the Army's Enterprise Architecture represents one approach. Other Services and projects can apply other approaches. The intent of the CADM is to capture the underlying data, regardless of approach. In particular, the belief is that two architectures, developed using completely independent approaches, can be combined and/or compared by representing each representation as an instance of a CADM database.

The CADM was developed by a group of experts who analyzed the Framework document, extracting data elements, and working those data elements into an IDEF1X format (using the software tool ERWin.) This produces a "rich" (or "complex") data model. This complexity is managed by having a separate database view for each C4ISR Framework product. Thus, it is possible to view the data model for the product entitiled "Node to Node Connectivity Matrix". This data model shows that nodes are related via one or more IER, and also that this relationship can be associated with one or more communications channels (used to pass the messages between the nodes.)

The CADM was also defined to be 'self-describing,' in that the CADM contains a data model for data models. This is not so difficult, since data models follow the Entity-Relationship-Attribute approach. But it allows an architecture description to contain information about itself, or about other architecture descriptions (described as a data model, or as some other form of document.

3.3 Applying the CADM to an Existing Architecture Framework

One of the motivations for the CADM was to serve as a basis for exchanging information among architecture description developers. The Army used the CADM as the basis for exchanging information between the Army Signal Center (SA-C) and PEO C3S (SA-D). This was accomplished by starting with CADM entities that matched data in the netViz representation of the Army System View. The Army then extended the CADM in two directions. One direction contained data that was tool-specific, such as the netViz icon for each organization type or equipment type. More importantly, additional entities were added that reflected Army-specific data requirements. This was done by preserving, as much as possible, the key structures in the CADM. So Army-specific extension to the CADM entity "Organization-Type" has the exact same keys as its parent CADM entity. This preserves both the CADM core data (by not modifying existing CADM structures), and the structural intent of the CADM, by following existing key structures as much as possible. The result is two entities in the data model.

A subsequent physical schema design reduced these two entities (the CADM "Organization-Type" and the Army-specific "Army-Organization-Type") into a single table, producing no performance penalty.

There were a few cases where the Army data model added new entities and keys. The Army System View is very concerned about fully describing the TCP/IP topology of the military Internet. Entities and attributes were added to fully capture the IP addresses for (appropriate) pieces of equipment, along with IP subnet routing data for organizations. These extensions were made in part to support network modelling and simulation activities that verify the behaviour of the resulting TCP/IP network under various loading conditions. It also supports the modelling of how this network will have to be reconfigured when a tactical unit reorganizes, moving organizations and equipment from one subnet to another.

4 IEEE 1471: Recommended Practice for Architectural Descriptions

IEEE Project P1471 for Architectural Descriptions is, at the time of this writing, in final ballot as an IEEE Recommended Practice. The focus of P1471 is specify how to produce an architectural description of a "software-intensive system." [6]

4.1 P1471 Basic Meta-model

P1471 describes an architectural description in terms of views, viewpoints and stakeholders. A viewpoint is the specification of a view, including the methods used to describe the view and the stakeholders and their concerns addressed by the view. The idea is that viewpoints can be shared across descriptions (e.g. a 'data viewpoint' or a 'performance modelling viewpoint'), providing for a common representation of architectural descriptions. In Ada terms, the Viewpoint is like a generic template that is instantiated for each architectural description. Another analogy is that a Viewpoint is a 'pattern' used to specify Views. See [6] more details.

The viewpoint-stakeholder-view approach in P1471 is very close to the method used by MITRE for several architectural products [3]. Thus we use P1471 as a codification of the MITRE method. The P1471 framework can be used to capture other architectural methods, such as ISO RM-ODP [7] and even C4ISR Architecture Framework. The key concept behind P1471 is that architecture is represented by sets of views and viewpoints that capture stakeholder concerns.

4.2 Applying Viewpoint-Stakeholder-View

In each of the MITRE projects, there was a multi-phased approach to developing the architecture description. The first phase determined the basic system requirements, both the functional requirements and the user preferences and trade-offs.

At the same time, all of the system stakeholders were identified, along with a set of concerns for each stakeholder.

The next phase selected viewpoints that met these stakeholder concerns. This started by iterating through previous viewpoint descriptions, selecting those viewpoints that would address stakeholder concerns. If all stakeholder concerns were not covered, we defined new viewpoints. For example, we applied this method to the Army's Distance Learning project. One key stakeholder was the training developer (often different from the instructor actually delivering the training.) Training developers were concerned about how their training materials would be distributed to the instructors, and also about gathering feedback from both students and instructors on the effectiveness of their training materials. For this project, we defined a new viewpoint that captured the Training Developer's concerns. This same project reused existing viewpoints for data, security, network/systems management and software development and maintenance.

Once we have the set of viewpoints defined, we then flesh out a view for each viewpoint. This view contains the actual architectural contents for the system. The separation of viewpoint from view allows us to figure out ahead of time the data requirements and framework. With this firm foundation, we concentrate on developing the system-specific contents of each view. Our experience to date is that about 75% of the viewpoints we select for each system are pre-existing viewpoints. But this means that each system has enough system-specific issues that some number of new viewpoints are needed to cover that system, its stakeholders, and their concerns.

5 What Do We Know about Representing Architectures?

We have learned a lot about the representation of architectures over the last several years. In particular, there exists a strong consensus on two points:

- Multiple views are required to capture an architecture.
- There exist definitions of the description ("viewpoints," in IEEE 1471 terms) that exist independent of any particular architectural instance.

These two points are consistent with practice in structural architecture, where floorplans, elevation drawings and architectural models are all used to represent different aspects of a building (multiple views). Each of these representations has a well-defined set of notations and conventions (viewpoints). [5]

5.1 It Takes Multiple Views to Dscribe an Architecture

There is substantial acceptance in other domains for representing a single description using multiple views. For instance, in structural architecture, floor plans, elevations and architectural models are all widely used representations of a single architecture.

Some representations of 'software architecture' [12] have concentrated on a single view of the software, consisting of the software 'structure'. Recent work

[1] has revealed that some essential system properties are not easily determined from a single structural view, including performance and security. In the Army implementation of the C4ISR Framework, security issues are spread across the three Views. Consolidating the security aspects from each view into a coherent discusson of system security is proving difficult. A separate Security view would certainly help capture all of the security issues into a single representation.

5.2 Viewpoints Can Exist Independent of the System Being Described

In each of our examples above, there is the notion of a 'viewpoint' that exists separate from any instance of an architectural description. The C4ISR Architecture Framework document provides a traditional textual presentation of 'viewpoints'. The CADM provides an alternate representation of the same viewpoints, described as an IDEF1X data model. IEEE P1471 codifies this notion of viewpoints, but does not specify the contents of viewpoints, instead leaving this decision to the architect.

Thus we can conceive of one step in an architecting process [11] as the selection of viewpoints to be used in a specific architectural description. There is an implied prior step, that of defining and archiving architectural descriptions. The C4ISR Framework starts with a static set of viewpoints, as do some other architectural methods, such as RM-ODP [7] and "4+1" [8]. The MITRE experience shows that we can define a library of viewpoints, supporting browsing and selection based on stakeholders and concerns.

5.3 Comparing and Analyzing Architecture

Once we have architectural representations, the next step is to compare or integrate these descriptions. Integration is a particular issue within the U.S. DoD, where each military service develops an architecture for its forces (e.g. Army Division, Air Force Wing, Navy Surface Action Group). When we build a Joint Task Force (JTF) with forces from each service, the JTF itself needs an architecture description that shows how each service works in the larger whole.

If two architectural descriptions are based on the same set of viewpoints, comparisons should be much easier. The views can be compared, "like to like", based on the underlying viewpoint. Without this level of consistency, much of the effort in comparing two architecture descriptions will occur in trying to reduce two dissimilar views to some sort of common denominator.

The CADM approach, describing viewpoints through a data model, may yield substantial results in comparing and integrating viewpoints. The underlying database notion makes it easy to extract common elements, or to join elements of two architecture descriptions into a new description. There are some efforts within the U.S. DoD that plan to use the CADM for integrating architectures.

Tools and techniques for comparing architectures are a very promising research area.

5.4 Architecture Descriptions as an Evolving Process

Each of the specification approaches described in this paper are relatively static, in that each defines a "complete set" of architectural description products, without providing any sort of ordering or process. Thus they specify the end-state of an architectural description, without providing any guidance on intermediate representations. The C4ISR Framework Document, as it has been commonly interpreted, generally implies that Operational and Technical view products preceed Systems view products, but this is not specified by the Framework Document itself.

Architecture specifications may be evolving from specifications of "paper products" such as reports and matrices, towards more interactive representations of architectures. Both the C4ISR Framework Document and the IEEE Recommended Practice define "paper products." The CADM, as a data model, lends itself to implementation via a database. The Army implementation of the C4ISR Framework Document uses both a database and a network diagraming tool.

As we gain more experience with architecture as a discipline, our description techniques will evolve to time/event sequences of products. The MITRE Architecture Approach [3] includes a set of preliminary products (Goals, Vision, Needs) that preceed those captured by the IEEE AWG document (viewpoints and views). Thus architectural descriptions will probably evolve towards more dynamic representations, including a process orientation that defines when and how products are produced, and a tool orientation that specifies architectural representations as databases and tool datasets, rather than paper products.

6 Where Are We Going with Architecture Descriptions?

There seems to be two evolving notions, that of "system architectures" as described in this paper, and the notion of "software architecture" as described in [12]. The primary focus of "software architecture" is on the (internal) structure of a software system, while "systems architecture" approaches concentrate more on the role of a given system in its environment. Thus it is common to read of "client-server software architectures" or "pipe and filter software architectures," but no similar terms describing common styles or 'patterns' have evolved in systems architecture representation.

However, the system architecture approaches, such as described in this paper, allow for a wider variety of topics than software architecture representations. There are many properties of systems, such as security, fault tolerance, maintainability, adaptability, etc, that are common in system architecture analyses. A definition of software architecture that focuses solely on structure is less suited to capture such non-structural properties of systems.

Both communities are investigating tools, particularly tools that capture the dynamic properties of systems and software. The tools used in the author's projects have concentrated on capturing the data behind the description, to allow various representations of the architectural description. One of the motivations

for a data-centric representation is that the database containing a specific architectural representation has been used as input to other efforts attempting to model or simulate the resulting architecture. These models have included classic network performance analysis, military simulations (wargames), and to a lesser degree, as inputs for testing systems that are components of the architecture.

Thus the future of architecture descriptions should include:

- Representations that cover non-structural aspects of systems
- Common sets of viewpoints.
- Increased tool support, particularly for interactive specification and analysis of architectural representations.
- "Interoperability" of architectural representations based on common viewpoints and underlying descriptions of the information in these viewpoints.

Both "system architecture" and "software architecture" approachs will contribute to this evolution.

7 Summary

The evolution of 'architecting' has reached the point where there is wide acceptance that 'architectures are important', but that previous ad-hoc methods for describing architectures are insufficient. Thus there have been several recent efforts to describe the contents of architectures, independent of any specific architecture.

This paper has described the author's experiences with several of these 'architecture description technologies', including the textual description of the US DoD's C4ISR Architecture Framework document, the associated Core Architecture Data Model and the IEEE Recommended Practice for Architectural Description. All of these approaches acknowledge the need for multiple views to describe a single architecture. They vary in how they define the contents of the views, and their relative focus on 'documents', 'databases' and 'viewpoints' as key architectural framework decisions.

Once we capture architectural representations, the obvious next step is to compare them. The current state-of-the-art in such comparisons, supporting a variety of methods that produce architectural descriptions, is based on data modelling. We require more experience with architectural analysis to know if architectural data models are sufficient to permit analysis and comparison of architectures.

The evolution of the architectural process will produce changes in our architectural specification technologies. In particular, current practice tends to imply paper-based static representations. With the evolution of tools and interactive representations (such as the World Wide Web), architectural meta-techologies will move towards process and behavior models. Methods for defining the contents and meaning of architectural descriptions should continue to evolve. We have captured here a snapshot of current efforts; stay tuned for further developments.

References

1. Allen, Robert J. "A Formal Approach to Software Architecture." PhD Thesis, Carnegie-Mellon University (CMU-CS-97-144), May 1997.
2. Bruce, Thomas A; "Designing Quality Databases with IDEF1X Information Models," Dorset House Publishing, 1992.
3. Emery, David E, Hilliard, Richard F II and Rice, Timothy B; Experiences Applying a Practical Software Architedture Method, in A. Strohmeier (ed), "Reliable Software Technologies - Ada Europe '96." Springer-Verlag: Lecture Notes in Computer Science 1088, 1996. *http : //thomas.pithecanthropus.com/ awg/CaseStudies.pdf*
4. Federal Information Processing Standards (FIPS) Publication 184, Integration Definition for Data Modeling (IDEF1X), 21 December 1993.
5. Hoke, John Ray (ed), "Architectural Graphic Standards," John Wiley & Sons, 1994
6. Institute of Electrical and Electronic Engineers; "IEEE Recommended Practice for Architecture Descriptions, Draft 4.1; IEEE, 1998. See *http : //www.pithecanthropus.com/ awg.*
7. Internationl Standards Organization; ISO/IEC 10746-3, Open Distributed Computing - Reference Model Part 3: Architecture. ISO, 1995. *http : //www.iso.ch : 8000/RM − ODP*
8. Kruchten, Philippe; ''The 4 + 1 View Model of Architecture", IEEE Software, 28 (11), 42-50, November 1995. *http : //www.rational.com/sitewide/support/whitepapers/dynamic.jtmpl?doc_key = 350*
9. Mowbray, Thomas J; "Will the Real Architecture Please Sit Down?" Component Strategies, December 1988.
10. "netViz 3.0." netViz Corporation. *http : //www.quyen.com*
11. Rechtin, Eberhard and Maier, Mark; "The Art of System Architecting" CRC Press, 1996.
12. Shaw, Mary A and Garlan, David; "Software Architecture: Perspectives on an Emerging Discipline" Prentice-Hall, 1996.
13. U.S. Department of the Army; Joint Technical Architecture - Army Version 5.0, Washington, DC 1999. *http : //www.usace.army.mil/inet/functions/im/lcmis/ata/ata.htm*
14. U.S. Department of the Army; Army Enterprise Architecture Guidance Document, Washington DC 1999. *http : //arch − odisc4.army.mil/aes/html/aeagd.htm*
15. U.S. Department of Defense; Joint Technical Architecture version 2.0, Washington, DC 1998. *http : //www − jta.itsi.disa.mil/jta/jtav2_dnld.html*
16. U.S. Department of Defense; C4ISR Architecture Framework, Version 2.0, Washington, DC. *http : //www.rl.af.mil/programs/jcaps/download.html#FRAME*
17. Walker, Robert (editor); C4ISR Core Architecture Data Model (CADM), Arlington, VA: Institute for Defense Analyses 1999. *http : //www.rl.af.mil/tech/programs/jcaps/cadm.html*

Acknowledgements: The Army System Architecture team was led by LTC Angel Colon and LTC Jim Travis. The CADM team was led by Dr. Robert Walker and Dr. Francisco Loaiza of IDA. Jim Perry, EER Corporation and Tim Anderson and Jack Garhart, BDM Corporation defined the Army System Architecture extensions to the CADM. Teams led by Steve Schwarm, Tim Rice and Kevin Heidemann produced the series of MITRE architecture projects. Basil Sherlund led the P1471 Architecture Working Group. Thanks to Karl Nyberg, Rich Hilliard and Olimpia Velez