16TH ICCRTS

Sixteenth International Command and Control Research and Technology Symposium

Collective C2 in Multinational Civil-Military Operations

June 21 – 23, 2011

Québec City

Québec Canada

Topic: Topic 8: Architectures, Technologies, and Tools

Title: Benefits and Challenges of Architecture Frameworks

Authors:

Name:	Daniel Ota
	Michael Gerz (Point of Contact)
Organization:	Fraunhofer FKIE
Address:	Neuenahrer Straße 20
	53343 Wachtberg-Werthhoven, Germany
Phone:	+49 228 9435 414
Fax:	+49 228 9435 685
E-Mail:	{daniel.ota|michael.gerz}@fkie.fraunhofer.de

Benefits and Challenges of Architecture Frameworks

Daniel Ota, Michael Gerz

Abstract

Architecture frameworks have become a popular means to cope with the complexity of today's enterprises. They support the specification of architectures by providing a method for designing and describing them. An architecture framework typically defines a common terminology, a set of views focusing on particular aspects of the architecture, a set of architecture types with varying levels of detail, and a methodology for the development and maintenance of an architecture and its views.

Despite their benefits, the adoption of an architecture framework is non-trivial in practice and does not always meet the expectations of the target audience. There are several reasons: First, the semantics of views – both in terms of what and how – leaves room for interpretation that must be filled by the architect(s). Second, the definition and maintenance of an architecture involves many different stakeholders and requires a modeling process. Finally, proper tool support is an important aspect when it comes to collaboration and promoting one's findings.

In this paper, we introduce the concepts of architectures and architecture frameworks. Next, we describe the core features of the NATO architecture framework. Thereafter, we present potential pitfalls when adopting a framework and conclude with a short summary.

1 Introduction

Architecture frameworks have become a popular means to cope with the complexity of today's enterprises by providing a method for designing and describing architectures.

Architectures The term *architecture* is applied in different fields of science, but due to this widespread use, there is no common definition. In computer science, several similar explanations are given. IEEE Standard 1471 [6], *Recommended Practice for Architectural Description of Software-Intensive Systems*, defines an architecture as "the fundamental organization of a system embodied in its components, their relationships to each other and to the environment and the principles guiding its design and evolution." The NATO Architecture Framework [8] uses this explanation, too.

An alternative definition that is often mentioned in the context of enterprise architectures is derived from ISO Standard 15704 [7], *Industrial automation systems – Requirements for*

enterprise-reference architectures and methodologies. According to ISO 15704, an architecture is "a description (model) of the basic arrangement and connectivity of parts of a system (either a physical or a conceptual object or entity)". While the latter definition focuses on the system or enterprise itself, the first one also takes the interactions with the environment and the *continuous* development of the system into account. ISO 15704 distinguishes between exactly two types of architectures: *type 1* architectures deal with the design of a system ("as-is architecture") whereas *type 2* architectures address the organization of the development and implementation of a project ("to-be architecture").

Architectures are used to describe parts and excerpts of the real world. These excerpts are considered from different perspectives and on varying levels of abstraction in *views*. IEEE Standard 1471 defines a view as "a description of the entire system from the perspective of a set of related concerns". Because of the complexity of modeling all relevant aspects in their entirety, architectures tend to be large. To be still informative, different modeling techniques (text, tabular, graphic) are required.

Architecture Frameworks Modeling architectures in detail requires some guidance, which is offered by architecture frameworks. TOGAF [9] defines them as follows: "An architecture framework is a tool which can be used for developing a broad range of different architectures." Architecture frameworks should support the specification of architectures by providing a method for designing and describing them. I.e., they should serve as templates for a variety of different architectures and describe the content, the structure, and the relationships of and between enterprise architectures.

Typically, architecture frameworks themselves are based on common concepts. In general, an architecture framework defines a set of views. Each of them focuses on a particular aspect of the architecture. These views present various details to different target audiences. Next, an architecture framework typically defines a common terminology that has to be applied to all architectures developed by the framework. This glossary enables a standardized wording and is often the semantic basis of a meta model. A meta model ideally captures the syntax and semantics of the different views. The meta model defines the valid and necessary elements of the different views and defines their relations. As stated earlier, an enterprise normally has a need for different kinds of architecture types. Therefore, an architecture framework defines types with varying levels of detail and different timeframes. To ensure development and maintenance of architectures, a corresponding methodology may be part of the framework. The methodology also provides procedures to determine and organize the data that is the basis for the architecture. It contributes to ensuring the consistency, accuracy, and completeness of the acquired data.

One of the first and most well-known enterprise architecture frameworks is the *Zachman Framework* that was published by John Zachman in 1987 [14]. Zachman suggests perspectives that should be taken into account to successfully set up the IT architecture of an enterprise. Thereby, he focuses on the involved persons, called *roles*, (e.g., owner or designer) and assigns to each of them different perspectives (when, what, ...) and objects that have to be considered by them.

Another architecture framework is *The Open Group Architecture Framework (TOGAF)*. The

first version of TOGAF was presented in 1995 and is still continuously maintained. In contrast to other architecture frameworks, TOGAF focuses on providing a methodology that allows a formalized creation of an architecture and provides less detailed information on the resulting architecture products.

The development of architecture frameworks in the last decades was often driven by military and governmental organizations. One representative is *DoDAF*, the *Department of Defense Architecture Framework*. The first version of DoDAF evolved out of the *C4ISR Architecture Framework*[1]. DoDAF is currently available as version 2.02 [5]. In response to it, the *Ministry of Defence Architecture Framework* (*MODAF*) was developed by the UK. Another well-known military architecture framework is the *NATO Architecture Framework* (*NAF*), which is used by many countries.

Despite their benefits, the adoption of an architecture framework is a non-trivial task and it does not always meet the expectations. There are several reasons: First, the semantics of views – both in terms of what and how – leaves room for interpretation that must be filled by the architect(s). Second, the definition and maintenance of an architecture involves many different stakeholders and requires a modeling process. Finally, proper tool support is an important aspect when it comes to collaboration and promoting one's findings.

In this paper, we discuss potential challenges and pitfalls when defining architectures based on an architecture framework.

Related Work Various architectures have been developed based on (military) architecture frameworks in recent years and the lessons learned have been documented. Due to the dominance and the long history of DoDAF, most of the articles cited below refer to it.

In 2004, Troche et al. [12] published their experience in developing architectural products using the C4ISR Architecture Framework. Their major conclusion was that architectures are developed to be used. This means that the development has to focus on user needs. Other issues like methods and tools are of secondary importance. Their report is divided into two parts: lessons learned related to the development process are followed by a discussion on development tools. The lessons learned primarily focus on preparatory aspects, while the development process itself is strongly influenced by the tools used.

Another well-documented process for creating an architecture is given by Richards et al. [10, 11]. They investigate the applicability of DoDAF to model satellite systems, in particular the Hubble Space Telescope. The authors criticize that there is only little guidance provided by DoDAF to construct the views and, therefore, it is hard to take advantage of the interconnected views. In [10], a methodology to define views by using a system engineering modeling tool is proposed and tested. Richards et al. have used it to create executable behavioral models that can be used for quantitative evaluations of the system behavior. In addition, lessons learned from using DoDAF are provided and improvements to its application are proposed. In their subsequent work [11], the authors deal with benefits of architecture frameworks and derive

[1] C4ISR = *Command, Control, Communications, Computers, and Intelligence, Surveillance, and Reconnaissance*

eight *measures of effectiveness* for architecture framework quality. These are purposefulness, applicability, internal consistency, external consistency, clarity, scalability, "execute-ability" and analytic extensibility. According to the authors, these measures should provide a basis for discussing best practices for architecture framework development.

Further architectural efforts, successes, and failures, are presented by Curts et al. [3]. The authors identified six simple recommendations. First, some high-level guidance and control has to be established. Next, a common lexicon as well as a standardized and well-defined architectural process is necessary. Additionally, interface and interoperability standards for architectures have to be identified and the cycles to produce meaningful results must be shortened. The last and most important requirement identified by Curts et al. is to employ automated architecture tools.

Another approach to determine the capabilities of architectures and architecture frameworks that differs from the ones above is described in the proceedings of a U.S. Army Workshop [1]. This workshop identified commonalities and differences of four different genres of architectures: enterprise, system of systems (SoS), system, and software architectures. The capabilities of DoDAF in supporting these genres are discussed. The workshop participants stated that DoDAF is helpful in some areas but is neither necessary nor sufficient for a high-quality representation of an architecture in any genre.

A common criticism on architecture frameworks, in particular military architecture frameworks, is missing guidance. On the one hand, they are praised for a detailed explanation of the purpose of their different views. On the other hand, they lack statements on the development and maintenance of architectures (enterprise architecture management). According to [13], the enterprise architecture management approaches differ significantly in a number of characteristics. Winter et al. state that there is neither a common understanding of the scope and the main activities that an enterprise architecture management function consists of, nor has a commonly accepted reference method been developed. Therefore, [13] have investigated the most prominent approaches. By doing so, they make several hypothesis about architecture management, which were reviewed in a survey to obtain the state-of-the-art in practice.

The primary focus of TOGAF is on an architecture methodology, called *Architecture Development Method* (*ADM*), without prescribing the latter architecture products. In contrast, DoDAF has its focus mainly on the description of a set of views. The potential synergy has been picked up by [2] and [4]. Both documents follow the same approach and differ only in the baselines of the documents used. While [4] is based on TOGAF 8.1 and DoDAF 1.0, [2] utilizes TOGAF 9 and DoDAF 2.0. Both papers outline, which views of the DoDAF architectural model are used in a dedicated phase of the TOGAF ADM and vice versa. Since both architecture frameworks complement each other, it is possible to use DoDAF in conjunction with the TOGAF 9 ADM. This allows to produce DoDAF architecture artifacts in a well-defined and repeatable process guided by the ADM.

Table of Contents The rest of this paper is structured as follows: In section 2, the NATO Architecture Framework is described in more detail. In practice, the adoption of an architecture framework in a large enterprise with distributed responsibilities is a non-trivial task. In

section 3, we first list some expectations of the target audience. In the following subsections, we will have a closer look at semantic issues regarding the framework itself (3.1), organizational aspects (3.2) and tool support (3.3). The paper concludes with a short summary in section 4.

2 NATO Architecture Framework

This section provides a brief overview of the NATO Architecture Framework (NAF). The core of the NAF version 3 [8] is nearly identical to the core of MODAF. The NAF is unclassified and freely available to all interested persons and organizations.

As mentioned in chapter 1, an architecture framework often defines multiple types of architectures. The NAF distinguishes between three kinds:

- An *overarching architecture* delivers an enterprise-wide description of the future situation with limited details. With a 10 to 12 years timeframe, it covers a long term and focuses on the *what?* question. What is the required functionality? What are desired capabilities and which objectives should be reached in the next decade?
- A *reference architecture* describes objectives of specific domains in an enterprise. It covers an entire planning cycle for a typical system development with an interval of 3 to 6 years. A reference architecture focuses on the description of decisions regarding system technologies, stakeholder issues, and product lines. It characterizes *how* specific functionalities can be met.
- A *target architecture* is limited to a single project or system on a very detailed level. Such a description is based on a reference architecture and answers *with what* the objectives of the reference architecture are reached. It describes in detail with what special product – or improvement of an existing system – an objective is satisfied.

An architecture of each type describes a particular matter by means of several views. The views vary in detail and time horizon but the elements that are used to model the views are the same. The NAF distinguishes between seven groups of views, where each group contains special subviews.

- The *NATO All View* (*NAV*) captures overarching aspects of the architecture that relate to all seven views. It sets conditions on the context and the scope of the architecture, which includes for example the conventions or time frames. The NAV also contains a dictionary for the architecture and documents changes on core meta data. It can be seen as an ”agreement” for the architecture.
- The *NATO Operational View* (*NOV*) describes tasks and activities of organizational elements. It also constitutes the resulting types of information flows and frequency of information exchanges as well as tasks and activities that are supported by these information exchanges.
- The *NATO System View* (*NSV*) supports the operational demands of the NOVs based on system functions. They describe systems, their components, their interfaces, and their interconnections. NSVs show how multiple systems can be connected to each other and can

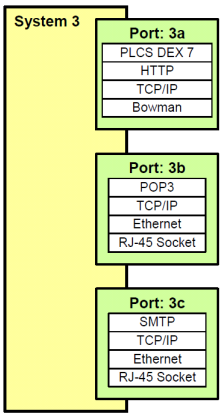

Figure 1: NAF System View 2a – Sample View [NAF v3, chapter 4, page 64]

work together. Especially it is possible to specify performance parameters and properties for these connections.

- The *NATO Technical View* (*NTV*) describes interfaces and standards of systems and their relations. NTVs provide guidelines for the technical implementation of systems and identify emerging and obsolete standards.
- The *NATO Capability View* (*NCV*) serves the analysis and optimization of military capabilities. NCVs show the dependencies between different capabilities and allow detecting gaps and overlaps of capabilities. NCVs deliver indirectly requirements for technical and organizational solutions and so they are the basis for the development of projects.
- The *NATO Service-Oriented View* (*NSOV*) describes cross-functional services that support operational tasks. NSOVs focus strictly on identifying and describing services. It has to be stated that services are understood in a broad sense and are not limited to IT services.
- The *NATO Program View* (*NPV*) is used to describe relationships between capabilities and projects. These views clarify the influence of acquisition decisions on other parts of the architecture. Also dependencies between the planning and the provision of skills are documented.

The first four views introduced above were already part of the NATO Architecture Framework version 2. The NAF version 3 introduces the three last views. All seven views are divided into a number of subviews. For each subview, an overview with its purpose and the corresponding definitions is given. The allowed objects and components of each view are determined and relationships within the view and to other views are shown.

An example of a subview, more precisely an example of NSV-2a, *system port specification*, is given in figure 1. The purpose of NSV-2a is to "define the ports on each system, and the protocol/hardware stack that is specified or implemented for each of those ports."

The architecture framework requires that the architect specifies for each port its name, the communications protocols used (e.g., the OSI stack), and the physical port specification (e.g.,

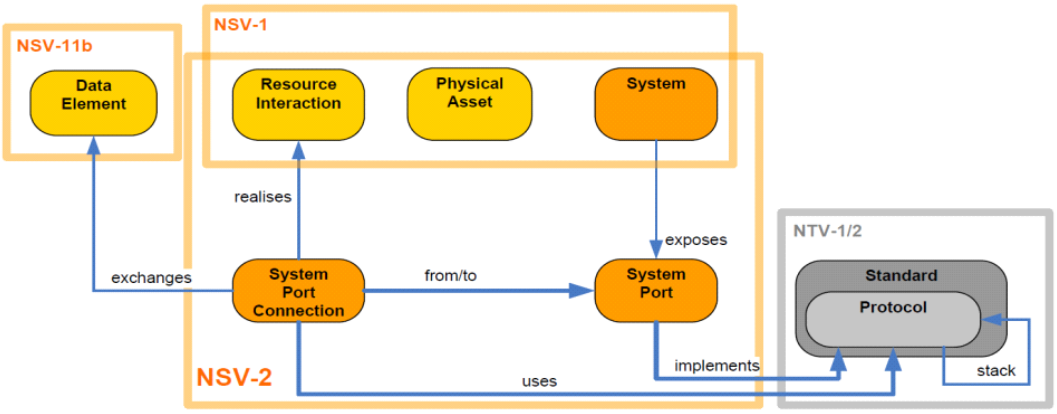

Figure 2: Relationship Between Individual NSVs [NAF v3, chapter 5, page 80]

the physical element of the stack).

As already stated, the NAF also defines the valid relationships within and between views. A simplified example for the system views is shown in figure 2. That description is also part of the *NAF Meta Model* (*NMM*). The meta model formally defines the syntax of each (sub-)view. It ensures the consistency of views and allows to link architectures and their components. In addition to the definition of the syntax, the meta model contains a glossary. This description of the semantics of each element of the meta model is given in order to achieve a common understanding. The NAF Meta Model is defined in the *Unified Modeling Language* (*UML*) and is based on IEEE Standard 1471 [6]. In NAF Version 3.1, the NMM is identical to the MODAF meta model v1.2.003.

3 Challenges of Architecture Frameworks

The user expectations to the application of architecture frameworks are many-fold. Some of the expected benefits from designing an architecture based on a framework are:

- *Interoperability:* An architecture-based approach ensures that systems will be truly interoperable.
- *Capability-Driven System Development:* Systems are no longer developed in an ad hoc manner (to satisfy urgent needs) but their requirements are derived systematically from high-level operational capabilities. With an architecture-based modeling approach, first the potential that is already available in the enterprise can be analyzed. Next, one can compare this to desired capabilities and identify critical gaps. By having a look on the changes of the capabilities when adjusting specific screws of one's system design, one is able to identify the points where adjustments offer the greatest benefit.
- *Collaboration across Project Boundaries:* An architecture-based approach supports/enforces collaboration across the boundaries of individual projects, preventing the development of stovepipe systems.

- *Semantically Unambiguous Descriptions:* The formal meta model of an architecture framework facilitates the semantically unambiguous and comprehensive description of operational activities, system properties etc.
- *Automatic Evaluation:* The meta model also makes it possible to query for information beyond what is given in the individual views. (e.g., "List all systems with a specific equipment or function"). Furthermore, it enables automated and repeatable evaluations of architectures and guarantees comparability, because the same algorithm can be applied to different architectures.
- *Comprehensive Specification:* A framework allows to describe all relevant aspects of an architecture.
- *Reuse of Architecture Views:* Architectural views, in particular system views, can be reused in other contexts/architectures. There is no need to remodel the same systems. Instead, views can be restored from an architecture library. That offers two advantages: Elder projects get an inexpensive quality assurance and additional work is decreased.

While architecture frameworks provide a valuable contribution to tackling the complexity of today's business, their application does not guarantee that all objectives are actually met. In the following sections, some challenges and pitfalls regarding semantics, organization, and tool support are discussed that we came across in our own projects. For illustration, the NATO Architecture Framework (NAF) is used as reference.

3.1 Semantic Issues

The availability of a formal meta model suggests that the elements and the structure of an architecture description are clearly prescribed. Therefore, in theory, there should be a straightforward approach to describe one's own architecture. Unfortunately, reading and understanding the architecture framework documentation is just half of the story...

3.1.1 Terminology

The first challenge to applying an architecture framework is to adapt its terminology. Due to the intended genericity of frameworks, common terms like *capability* have to be stated more precisely in a specific application context.

For instance, NAF 3 defines a *capability* as "a high level specification of the enterprise's ability." Depending on the objective, an ability can be a very different thing. For instance, possible capabilities of an army are intelligence, mobility, resistance, etc. On the level of individual formations, capabilities must be defined in much more detail. Then again, in the context of an international interoperability program, the term *capability* has a totally different meaning. Here, the capabilities of the interoperability solution must be described, such as the ability to exchange data in joint operations.

In addition, experience has shown that many people find it difficult to draw a clear line between different concepts such as *capability*, *service*, and *system function*. If capabilities are described in too much detail, they are very similar to services.

3.1.2 Architecture Views

Regarding individual architecture views, both the content (what?) and the design (how?) can become subject to discussions.

Content of Views The NAF defines a large set of views, each focusing on a specific aspect of the architecture. While most views are self-explaining, some others are not obvious. The purpose of NAF Operational View 7 (NOV-7) is to define an information model.[2] It is explained as follows:

> "An information model should not be confused with a data model. Although the distinction between the two is not clear, they at least serve different purpose. [...] Often, a high-level logical data model is presented as a conceptual data model, or even as an information model." [NAF, chapter 4, page 48]

Unless the reader already has a clear understanding of the terms used in the NAF documentation, this definition does not help to classify existing information/data models.

Design of Views The NATO architecture framework distinguishes between operational and system views. An operational node is "a logical entity that performs operational activities". A system is "a coherent combination of physical artifacts, energy and information, assembled for a purpose (software-intensive)". One of the fundamental questions that were raised in one of our projects was how to model the interaction between systems and human operators. For instance, take a vehicle and its crew. The crew members can be considered as

- operational nodes that make use of one or more systems
- systems
- parts of a system (e.g., a commander within a vehicle)

There is no definite answer to what is the right way of modeling. Nevertheless, the modeling approach has strong implications on the reuse of architecture views and the representation of specific aspects, such as swivel chair interfaces.

In practice, people do not make a clear distinction between operational nodes and systems. The reason is that specific operational nodes (military units) are characterized by their specific equipment. If a unit in a military scenario needs a specific system, e.g., an unmanned aircraft vehicle, this is likely to be highlighted in operational views, even though – strictly speaking – it belongs to the system views. (See figure 3)

3.1.3 Context

Architectures are described by a collection of views. Even individual systems are characterized by a series of views. All of these views are isolated "products", i.e., NAF does not provide a

[2] In contrast, NAF System Views 11a and 11b define logical and physical data models.

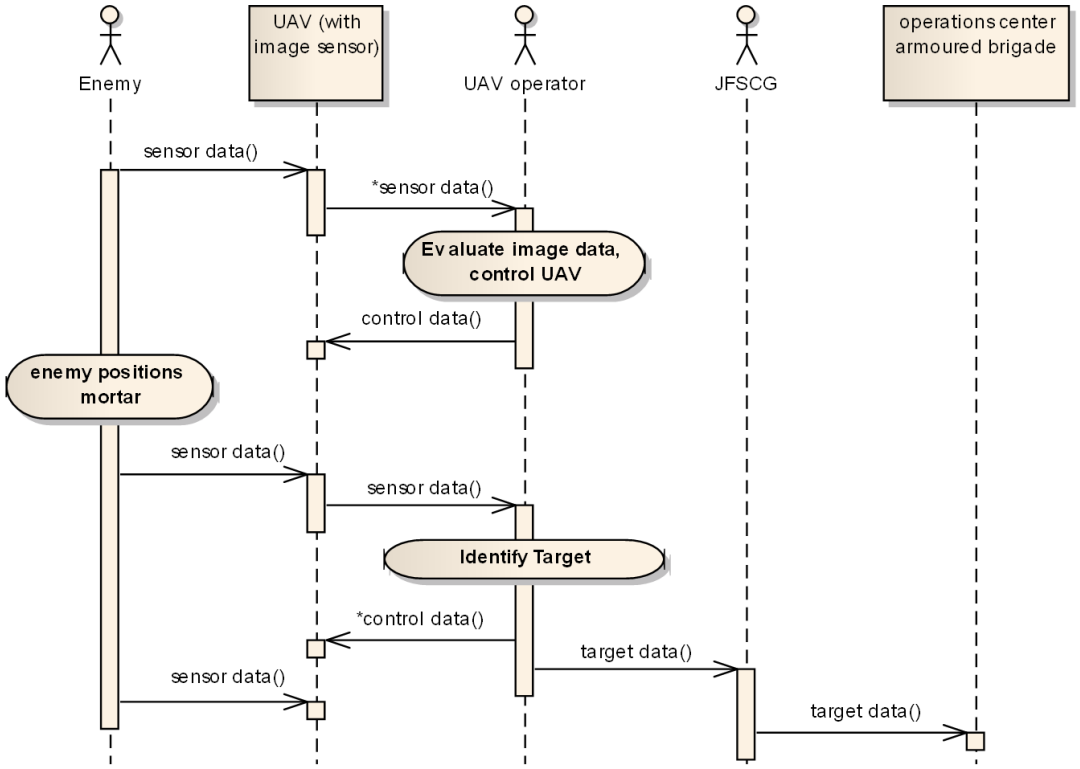

Figure 3: Lifting System to the Operational Level (NOV-6c)

mechanism to (a) group several views logically and (b) define their context. This makes it impossible to preserve the semantics of views when exchanging architecture views between projects or storing them in a central architecture repository. The lack of clear semantic boundaries is critical for systems that change over time or consist of modular components. For example, it is undefined whether a port connection in an NSV-1 between two systems is only valid in the context of this specific example or denotes a global connection.

A simple yet clumsy solution to overcome the problem is the strict adherence to naming conventions. In that case, the name of a view defines its context. A better approach is to use specific features of the modeling tool employed. For instance, UML tools allow to define (nested) packages to group diagrams, model elements etc. In both cases, the solution is outside the scope of NAF.

3.1.4 Semantics of Model Elements

Specific elements of views can also become subject to discussion. For instance, NAF allows to describe the internal structure of technical systems. In a concrete project, this had led to a number of subtle questions. Examples:

- Is a CAN bus a system, a port, or a network?

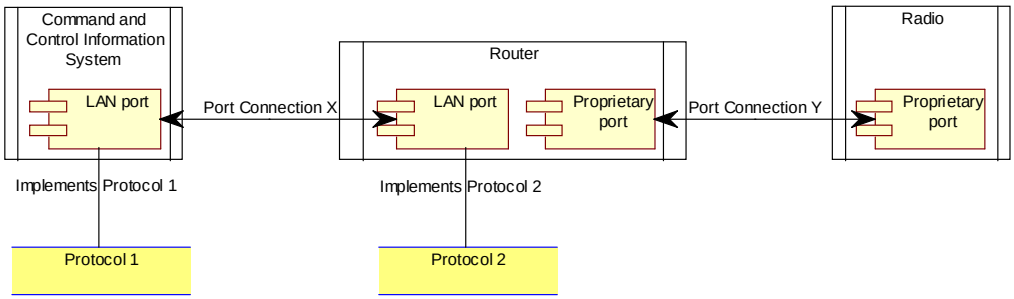

Figure 4: NAF System View 2b – Port Connections

  - To state that a specific system has an interface to connect to a CAN bus, one is tempted to model it as a port.

  - NAF only defines point-to-point port connections. In case of describing that various systems are connected to the same CAN bus, is it allowed to model the bus as a network or even a system?

- How can we express that a switch has eight identical ports?

- How can these ports be distinguished to express that some system S is always connected to port 2?

When it comes to port connections, the semantics is also not clearly defined as illustrated in figure 4. The view describes a system with three components: a computer with a C2 information system, a router, and a combat radio.

Which protocol stacks are implemented on each subcomponent? In case of the C2 application, one might want to specify, e.g., the messages that can be exchanged. On the contrary, the router may be agnostic to any specific application such that only the protocols for the lower OSI layers (1-4) have to be specified. Finally, the port of the combat radio may be vendor-specific.

A major concern of many architectures is system interoperability. NAF does not specify the semantics of ports and port connections in a formal manner. Therefore, it is impossible for a tool or an inexperienced user to determine whether two ports are compatible. However, if ports are specified as sketched above, it is very difficult to even determine the degree of syntactic interoperability, because the information flow through the internal subcomponents must be traced.

Another semantically ambiguous system specification is given in figure 1 on page 6. In this system view, it is impossible to determine the number of physical ports. Although the diagram lists three distinct ports, the system is likely to have only two physical instances, because ports 3b and 3c only differ with regard to the email protocol (POP3 vs. SMTP).

3.1.5 Complexity of Real-Life Systems

Systems can be composed of many subcomponents and can have multiple interfaces. NAF allows to describe such systems with the restrictions sketched above. However, typically we also have to deal with variants of systems:

- Vehicles are equipped differently depending on the specific mission
- C2 applications are tailored to the user's role

Depending on the objective, those variants can have a significant impact on the architecture. Since there is no formalism to deal with system variants, there are two options:

1. All system variants are modeled explicitly in independent views. This approach implies a significant modeling effort and holds the risk of inconsistencies.
2. A generic base system is modeled; variants are documented in an informal manner. This approach requires less work. On the other hand, relevant information on variants is no longer available in a structured manner and an automated evaluation is more difficult.

Another issue concerns the complexity of today's C2 information systems. A modern C2IS supports many interfaces to interoperate with a variety of external systems. Examples include:

- MIP Baseline 2 and 3 (in fact with a system-specific subset of all available data elements)
- ADatP-3 (selected message text formats only)
- Link-11/16/22
- Email (possibly with vendor-specific extensions/modifications)
- Proprietary, and possibly not well-documented, message formats
- Web services

Documenting those interfaces in a formal manner is beyond what is possible within an architecture framework. Therefore, the architect has to carefully decide what information on interfaces is relevant for the architectural considerations and what information can be generalized. Consequently, conclusions on the interoperability of heterogeneous C2IS cannot be derived from architecture views in general.

3.1.6 Genericity of the Meta Model

The meta model of an architecture framework defines the concepts and their relationships that can be used in architecture views. Due to the genericity of architecture frameworks, the meta model only includes core concepts such as *system*, *service,* and *operational node*. For some architecture, this level of detail is too coarse. For instance, one might want to highlight certain elements of an architecture and/or assign specific attributes to them. For instance, if the objective is to analyze sensor-to-shooter cycles, the generic concept *system resource* should be derived to *sensor* and *effector*. Extensions to the NAF are intended and can be documented in NAV-3. However, any change potentially means that reusing the architecture views is harder to achieve and proper tool support is needed to handle those meta model extensions.

3.2 Organizational Challenges

Beside the semantic aspects, there are also organizational issues that need to be considered to successfully introduce an architecture framework.

First, architects need to understand the concepts of a given framework. NAF 3 comes with a comprehensive documentation of 852 pages. (For comparison, the PDF export of the MODAF 1.2 website comprises 256 pages). In addition, the large number of architecture views (48 NAF subviews in total) has a significant deterrence potential. Therefore, one of the first steps is to figure out what is relevant for a particular task.

3.2.1 Cross-Organizational Modeling Process

If the definition of an architecture is put on several shoulders (rather than being the task of a small core team), a modeling process is needed. It shall clearly define who is providing which views at which stage and with what level of detail. This process and the associated user roles must then be mapped onto existing organizational units. In other words, the development of architectures requires an enterprise (meta) architecture on how to define architectures. It is important that all participants have a common understanding of this process. Experience has shown that uncoordinated modeling will quickly result in chaos.

In order to gain wide acceptance, all interest groups should be integrated into the modeling process from the very beginning. Otherwise, there is the inherent risk that the outcome will be subject to criticism (irrespective of the actual result). If the existing organization structure turns out to be inadequate to define an architecture, it should be checked whether current organizational processes can be improved. For instance, it may turn out that some collaboration is needed in cases where project groups work in isolation today or that some organizations have overlapping responsibilities.

3.2.2 Maintenance of Architectures

Architectures are not statically defined and fixed for eternity but need to be adjusted to changing operational requirements and constraints over time. Accordingly, architectural descriptions need to be maintained continually. Again, a modeling process is needed.

When it comes to the reuse of architectural elements, e.g., system descriptions, a central architecture repository is useful but it has to be maintained, as well.

Therefore, a central organization unit is needed that coordinates all architecture modeling work. Among others, its tasks are to:

- Provide methodological support
- Enforce the enterprise modeling process
- Adjust the enterprise modeling process
- Identify relationships between different architectures

- Avoid redundancies among different architectures
- Harmonize views with regard to the level of abstraction, terminology, and structure

3.3 Tool Aspects

When designing an architecture, the use of a proper tool set is essential. Some key factors are described in the following.

- *Licensing.* In order to establish and gain acceptance of the concepts of architectures and architecture frameworks within an enterprise, it is important to address all relevant people, even if they are not directly involved in modeling. In large enterprises (such the armed forces), licensing fees may turn out as a showstopper to a widespread use of the tool. Ideally, a viewer application is available that can be installed free of charge or with reduced licensing fee.
- *Export Functionality.* The presentation of results in a manner suitable to people not directly involved in architectural design is of importance. An inherent risk is that one does not see the full picture, because there are so many puzzle pieces. It is up to the beholder to put together the different views in his mind. In other cases, views may be overloaded. Sometimes, the graphical representation imposed by the tool adds to that complexity.

  In order to provide architecture views to people with no or little experience in modeling tools, sophisticated export functionalities are needed. The export should be customizable and represent information in different ways (graphics, lists, matrices, etc.). Information may not necessarily be grouped in a way that has a 1:1 correlation with official NAF views. Customer-specific views are a valid approach as long as they are derived from the framework's meta model.
- *Linking Formal and Informal Elements.* As outlined above, not all relevant information can be modeled formally. I.e., architecture views must be accompanied by free-text documents. The modeling tool must support the linkage between formal and informal elements and preserve them even during an export.
- *Distributed Modeling.* If an architecture is built by a group of people, mechanisms for distributed modeling are needed. A role-based approach is needed to enforce proper access control.
- *Support/Extensibility of the Meta Model.* One important aspect is the degree to which a tool enforces the framework's meta model and the way it supports meta model extensions. A modeling tool should ensure that only views can be created that fully comply with the meta model. To assist the user, a framework-specific user interface should offer the allowed (graphical) elements (context-based on a per-view basis). Ideally, it should also point out potential inconsistencies across individual views.

  Meta model extension should be possible on the level of individual architectures such that architecture models can be exchanged easily without having to touch the tool itself. (This is supported by, e.g., UML profiles)

4 Summary

Architectures are an essential means to deal with the complexity of today's operations and systems. Architecture frameworks provide a "template" to capture such architectures in a structured manner.

The primary purpose is to *document* an architecture. Our experience is that due to the weak semantics of the NAF meta model, an automated analysis of architectures (their views) is only possible in a very restricted way. For the same reason, NAF is not perfectly suited for detailed system specifications. Lack of guidance and ambiguities in the architecture framework (what? how?) tend to result in discussions on the meta level.

The reuse of architecture views is considered as problematic, because different teams – depending on their objectives – may opt for different modeling styles and different levels of abstraction. A permanent coordination team with expertise in framework methodology is needed throughout the entire modeling process to ensure that different artifacts will fit together. This organizational measure should be complemented by a central repository.

The development and maintenance of an architecture mandates a well-defined modeling approach. It needs to define who is going to provide which views. Since an architecture covers many aspects, different organizational branches must work together in a synchronized manner. The need to collaborate is not new; however, a framework and business process make roles and responsibilities more transparent than it may have been in the past.

References

[1] Bergey, John K.; Blanchette, Stephen Jr.; Clements, Paul C.; Gagliardi, Michael J.; Wojcik, Rob; Wood, William G.; and Klein, John: *U.S. Army Workshop on Exploring Enterprise, System of Systems, System, and Software Architectures.* Technical Report. CMU/SEI-2009-TR-008. 2009. 4

[2] Blevins, Terry; Dandashi, Fatma; and Tolbert, Mary: *The Open Group Architecture Framework (TOGAF 9) and the US Department of Defense Architecture Framework 2.0 (DoDAF 2.0).* The Open Group. 2010. 4

[3] Curts, Raymond J.; and Campbell, Douglas E.: *The Trouble With C2 Architectures.* Proceedings of the 12th ICCRTS Symposium. 2007. 4

[4] Dandashi, Fatma; Siegers, Rolf; Jones, Judith; and Blevins, Terry: *The Open Group Architecture Framework (TOGAF) and the US Department of Defense Architecture Framework (DoDAF).* The Open Group. 2006. 4

[5] Department of Defense: *DoD Architecture Framework 2.02.* 2010. 3

[6] IEEE: *Recommended Practice for Architectural Descriptions of Software-Intensive Systems. (IEEE 1471, Reference number: IEEE1471-2000).* 2000. 1, 7

[7] ISO: *Industrial automation systems – Requirements for enterprise-reference architectures and methodologies. (ISO 15704)*. 2000. 1

[8] NATO C3 Board: *NATO Architecture Framework Version 3*. 2007. 1, 5

[9] Opengroup: *TOGAF 8*. `http://www.opengroup.org/architecture/togaf8-doc/arch/`. 2006. [Last visited on 2011-05-01] 2

[10] Richards, Matthew; Shah, Nirav; Hastings, Daniel; and Rhodes, Donna H.: *Managing Complexity with the Department of Defense Architecture Framework: Development of a Dynamic System Architecture Model*. 4th Annual Conference on Systems Engineering Research (CSER). Los Angeles. 2006. 3

[11] Richards, Matthew; Shah, Nirav; Hastings, Daniel; and Rhodes, Donna H.: *Architecture Frameworks in System Design: Motivation, Theory, and Implementation*. INCOSE International Symposium. San Diego. 2007. 3

[12] Troche, Carlos; Eiden, Gerald F. Jr.; and Potts, Frederick C.: *Architecture Development Lessons-Learned: A Three-Year Retrospective*. MITRE Technical Report. MTR 04B0000036. 2004. 3

[13] Winter, Katharina; Buckl, Sabine; Matthes, Florian; and Schweda, Christian M.: *Investigating the state-of-the-art in enterprise architecture management method in literature and practice*. MCIS 2010 Proceedings. Paper 90. 2010. 4

[14] Zachman, John: *A framework for information systems architecture*. IBM Systems Journal, Vol. 26, No. 3, pp. 277–293, 1987. 2