Aligning TOGAF and NAF – Experiences from the Norwegian Armed Forces

Håvard D. Jørgensen[1], Tore Liland[2], Stein Skogvold[3]

[1] Commitment AS, PO Box 534, N-1327 Lysaker, Norway
[2] Norwegian Defence Logistics Organization, Rødskiferveien 20, N-1352 Bærum, Norway
[3] Acando AS, Lille Grensen 5, N-0159 Oslo, Norway
Havard.Jorgensen@Commitment.no, TLiland@mil.no, Stein.Skogvold@Acando.no

Abstract. This paper reports on experiences from establishing a reference architecture framework for the Norwegian Armed Forces. Like a number of other nations and NATO agencies, the armed forces chose TOGAF as their architecture development methodology (ADM), and the NATO Architecture Framework (NAF) for metamodel and content organization. In order to make TOGAF and NAF work together and address the particular requirements of the armed forces, significant adaptation was required. Previous work has analyzed the combination of TOGAF and military frameworks on the high level, but no detailed mapping between TOGAF 9 and the NAF, DoDAF, or MODAF architecture content frameworks were available. Such a mapping is presented here. The resulting framework has been implemented as a set of UML profiles, and as the content structure for the national military architecture repository. It has been applied by a number of initiatives, ranging from enterprise capability maps to technical interoperability between systems and acquisition projects.

Keywords: Architecture frameworks, TOGAF, NAF.

1. Introduction

This paper presents experiences from ongoing work at the Norwegian Armed Forces. The goal is to deliver a reference architecture for the networking and information infrastructure (NII), with the methodology, guidelines and competence needed to sustain it. The paper focuses on the adaptation of the architecture framework.

This is a case study, which focuses on the aspects of NAF and TOGAF that needed to be adapted to fit the needs of a particular organization. Previous analyses that has linked TOGAF ADM to military architecture frameworks [2,12,13] have taken a broader and more high level perspective, without drawing on experience from actual implementation of a combined framework. They have not dealt with implementation details such as metamodels and repository structures. In addition to practical relevance for other nations and agencies that seek to apply TOGAF with a

military architecture framework, our work sheds light on the different perspectives, strengths and weaknesses of TOGAF and NAF

The next section describes the background of the work, the use of enterprise architecture descriptions by the armed forces, and introduces NAF and TOGAF. Section 3 outlines the integration and adaptation that went into designing a customized architecture framework. Section 4 reports on implementation and usage experiences, while section 5 proposes directions for further development of practical architecture frameworks, as well as implications for future research.

2. Background

The key objectives for the architecture efforts of the Norwegian armed forces cover four levels: **Capability** planning and strategy development; **Project portfolio** management, migration planning, and investment decisions; **Project** management and inter-project coordination; **Solutions development** of secure, interoperable and flexible systems.

Up until now, the main focus has been on solutions development, and a major aim of the project reported here was to extend the use of architecture to cover also the higher levels. The armed forces have more than 10 years of experience with enterprise architecture (EA). It commissioned the development of a customized architecture framework called MACCIS [3, 8], and later participated in the development of NATO architecture standards, including NAF.

Top level management is committed to architecture. The strategic IT plan of the Chief of Defense describes architecture as a key enabler, and the architecture plan describes how to use "architecture as a methodology for describing complex relationships in a network-based defense and apply these descriptions as a foundation for management and decision making". The Department of Defense has developed a NII reference model, and identified core areas for improvement like service orientation, modularization, interoperability, standardization, and reduction of the number of system variants for different user communities and platforms.

In the armed forces, the architecture responsibility is distributed between the IT department (INI) and the logistics organization (FLO). INI is responsible for the functional architecture, while FLO is responsible for the technical architecture. A governance structure is in place. The Architecture Advisory Board has the whole Networking and Information Infrastructure (NII) as its area of responsibility, and holds regular meetings to assess the architectural implications of new projects. At FLO, the Architecture Forum plays a similar role for projects in the acquisition/development phase, and evaluates standards before ratification.

In addition to the formal architecture governance organization, there are local architecture initiatives. Most notably, the Norwegian Defense Research Establishment (FFI) uses architecture descriptions in the development and validation of new concepts, and the common administrative systems project (LOS) uses ARIS to design their SAP adaptations.

2.1. NAF

In NAF [8], NATO defines four kinds of architectures. The *overarching* architecture should look several years into the future and answer the questions of *what* the enterprise is doing, and *why*. A *reference* architecture typically covers a span of a few years, describing *how* the enterprise functions, leading to a set of different *target architectures* for solutions development, which covers the technical aspects (*with what?*). A *baseline* architecture describes the technical aspects of the current enterprise. The core of NAF is a set of views that describe different aspects of an architecture [8]:

- **All view** (NAV) sets the scope and context of the architecture, including the subject area and timeframe, doctrines, tactics, techniques, procedures, relevant goals and vision statements, concepts of operations, scenarios, and environmental conditions.
- **Capability view** (NCV) supports the process of analyzing and optimizing the delivery of military capabilities in line with strategic intent. It contains a capability taxonomy and dependencies between capabilities, augmented with schedule data and measures of effectiveness to enable the analysis of gaps, overlaps, and trade-offs.
- **Operational view** (NOV) is a description of the tasks and activities, operational elements, and information exchanges required to accomplish missions and realize the capabilities expressed in NCV.
- **Service-Oriented view** (NSOV) supports the development of a Service-Oriented Architecture (SOA). NSOV describes the services needed to support the operations described in NOV. A service is understood in its broadest sense, as a unit of work through which a provider provides a useful result to a consumer.
- **Systems view** (NSV) describes the technical systems and system interconnections, their structure, functionality, behavior, and quality. Organizational, material, hardware and software resources are covered in order to define the physical architecture that implements the logical views.
- **Technical view** (NTV) provides the technical systems implementation guidelines upon which engineering specifications are based. NTV includes a collection of standards, implementation conventions, rules, and criteria.
- **Programme view** (NPV) describes the relationships between capability requirements and the ongoing development projects. This information can be leveraged to show the impact of acquisition decisions on the architecture.

Each of these seven views is further decomposed into subviews, which are diagram types for the enterprise architecture models. NAF derives this core structure of views and subviews from the US Department of Defense Architecture Framework (DoDAF) [1]. It also includes additional views from the UK Ministry of Defence Architecture Framework (MODAF) [5], and NAF's metamodel is aligned with that of MODAF. NAF does not prescribe a detailed methodology, though users are advised to follow the guidelines of DoDAF.

2.2. TOGAF ADM

TOGAF ADM has matured over more than a decade of industrial experience. Until version 9, it was agnostic of architecture framework and metamodels. It has been widely used with frameworks from Zachman and various modeling tool vendors, and with customized frameworks developed by different industries and organizations.

TOGAF ADM consists of nine phases. The *preliminary* phase outlines vision, objectives and scope, and mobilizes resources for the main architecture development cycle, which covers the phases A to H. Though the phases are represented as sequential, activities within different phases are often performed concurrently. The ADM is iterative, over the whole process, between phases, and within phases.

The central activity of requirements management collects, organizes and feeds architecture requirements into the phases of the cycle. Phase A continues the preliminary work of defining the vision, objectives, principles, and scope of the architecture. Phases B, C, and D collect information and populate the architecture model with business, information systems and technology descriptions respectively, while phases E and F utilize the architecture to select and govern development projects. Phases G and H deal with the long term governance and change management of the architecture, respectively.

2.3. Related work

Previous analyses have explored the use of TOGAF 8 ADM with DoDAF [12] and MoDAF [2]. These analyses form the foundation of our work in integrating the two frameworks. However, in order to define a fully functioning methodology, we also explored the new architecture content framework (ACF) developed for TOGAF 9:

- How the architecture products of this framework maps to NAF subviews. TOGAF connects its architecture products to the phases of the ADM.
- How the metamodel of ACF maps to that of NAF. This provides insights into e.g. how a service oriented approach is best realized.

We also looked at the revisions that NAF v.3 and 3.1 makes to previous DoDAF, MODAF and NAF versions. In total, this provides a more up to date and detailed reference than previous work [2,12,13].

3. Adapting and Integrating TOGAF ADM with NAF

At the start of our project, NAF had been selected as the standard architecture content framework, in order to interoperate with coalition partners. TOGAF was the chosen architecture development methodology. These approaches had however not been customized to the needs of the armed forces. Enterprise Architect from Sparxsystems, a UML tool, was selected as the standard modeling tool for the whole enterprise, and a NATO Architecture Repository (NAR) had been set up, storing XMI files in a version control system.

3.1. Approach

Standard frameworks like TOGAF and NAF can be used in a wide variety of organizations. However, before they can be effectively used together within an architecture project, tailoring at three levels is necessary.

1. **Framework**: Align the TOGAF ADM phases and activities with the content framework of NAF.
2. **Enterprise**: Tailor the frameworks for integration into the enterprise of the armed forces. This includes integration with project and process management frameworks, customization of terminology, development of presentational styles, selection, configuration, and deployment of architecture tools, etc.
3. **Project**: Adapt the framework for the stakeholders of each particular architecture project. Tailoring at this level will select appropriate deliverables and model views to meet stakeholders' concerns.

The scope of our project is the overall reference architectures for the armed forces, so we did not customize to any specific project. We followed this approach:

Framework Adaptation

- Resolve the differences in approach between the two frameworks.
- Adapt the detailed steps in each TOGAF phase to the content structure reflected in the NAF subviews.
- Establish a minimal set of principled mappings from TOGAF elements to NAF elements, one-to-many and many-to-many where necessary. This should remove any ambiguities uncovered above.

Adaptation to the Enterprise

- Define the purpose, scope and role of the reference architecture in the landscape of other architectures in the military sector.
- Define clearly the stakeholders and user roles for the reference architecture, their concerns and objectives.

Implementing the architecture framework

- Define metamodels for the modeling languages, in our case as UML profiles in Enterprise Architect,
- Establish template architecture content and navigation structures, in our case as package structures in an Enterprise Architect model,
- Establish the architecture repository, and structure it according to the content framework,
- Establish customized frames of reference for different diagrams, like the NNEC Services Framework [8] for service taxonomies,
- Provide example models of each diagram type, for training and support,
- Define a template project plan with the work breakdown structure of TOGAF ADM, in our case in Microsoft Project.

This section describes the framework adaptation results, while the next section deals with adaptation to the enterprise and implementation experiences.

3.2. NAF and TOGAF Approaches

As a starting point for adapting the ADM to NAF, Figure 1 helps us to understand the use of ADM in the landscape of different architecture descriptions in the armed forces. The same figure is found in NAF, which substitutes Architecture Vision with Overarching Architecture, Architecture Definition with Reference Architecture, and Transition Architectures with Target Architectures. This means that what TOGAF sees as an integrated architecture description constructed by a single ADM cycle, NAF envisions as a set of interrelated descriptions, each developed by different people for different purposes.

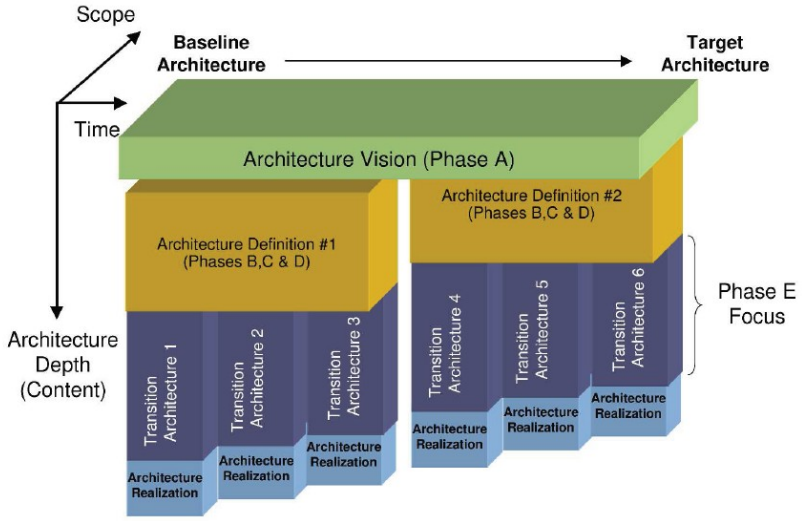

Figure 1. TOGAF ADM phases and architecture content [11].

3.3. NAF and TOGAF Content Frameworks

Figure 2 below shown the views of NAF organized in the content framework of TOGAF. At this level the frameworks are well aligned. The only minor deviation is the conceptual information model, which TOGAF places in the data architecture, and NAF regards as an operational view. DoDAF v.2 [1] is better aligned with TOGAF in this area, through its Data and Information viewpoint. The motivation part of TOGAF's business architecture corresponds to NAF capability views, while operational views cover the organizational and functional aspects. NAF system views define most of TOGAF's IS and technology architectures, though NSOV should probably be used for high level services. At the bottom, technical views define implementation governance, while program views may be used for migration planning.

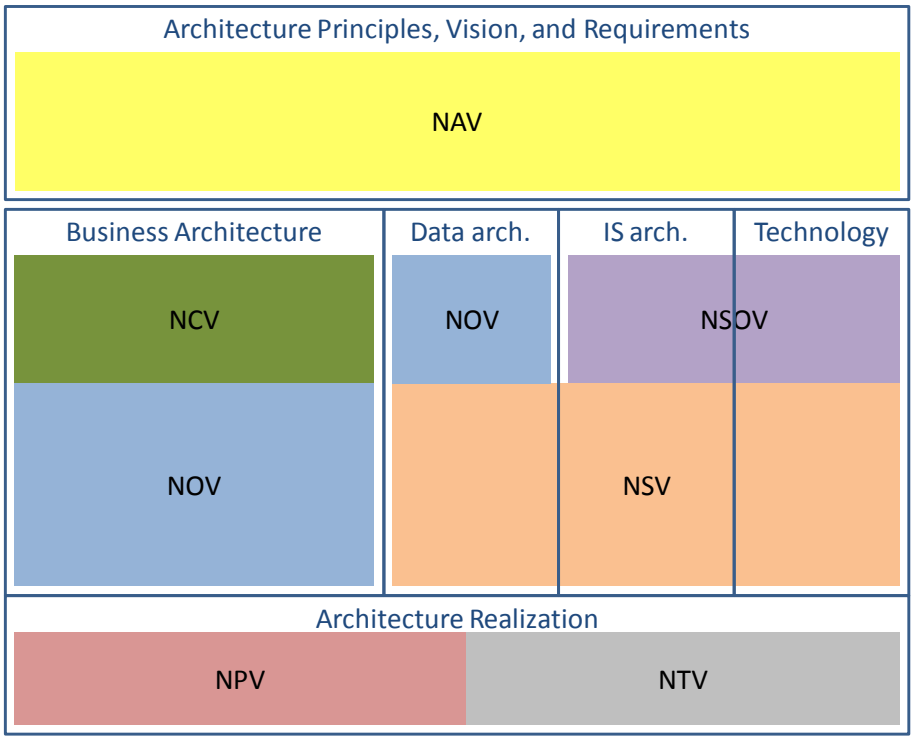

Figure 2. Alignment of NAF and TOGAF Content Frameworks.

3.4. NAF Subviews and TOGAF Architecture Products

In order to implement an architecture framework, however, we need to define precisely which architecture products to use. The devil is in the details, and when we approach the level of NAF subviews and TOGAF architecture products, the alignment of the two standards is no longer so straight forward.

As mentioned above, previous analysis [2,12,13] have mapped MODAF and DoDAF subviews to the phases of TOGAF ADM. With TOGAF 9, we have an additional resource for this mapping that these analysis did not, the TOGAF content framework (ACF) and metamodel. We therefore explored every TOGAF architecture product and identified suitable NAF subviews for each, using the metamodel types listed as corollary. The table below summarizes our mappings (v), and compares it to previous proposals (x) [2,12,13].

The differences between these mappings illustrate that TOGAF and NAF stem from different traditions, information systems and systems engineering respectively, and take different perspectives. Until you look into the detailed metamodels, these differences may not be so evident. Another important issue is that the high level mapping is mainly based on DoDAF, which compared to MODAF and NAF offers better support for an information systems perspective. The most important differences between the two mappings are:

			A. Architecture Vision	B. Business Architecture	C. Information Systems Arch.	D. Technology Architecture	E. Opportunities and solutions	F. Migration Planning	G. Implementation Governance	H. Architecture Change Mgmt.
NAV	1	Overview and summary	xv						x	
NAV	2	Integrated Dictionary		x	x					
NAV	3	Architecture metadata	x							
NCV	1	Capability vision	xv	xv					x	x
NCV	2	Capability taxonomy	x	xv					x	
NCV	3	Capability phasing				xv	x		x	
NCV	4	Capability dependencies		v		x	x		x	
NCV	5	Capability to organisational deployment	v	xv	x	x		x		
NCV	6	Operational activity to capability mapping		x						
NOV	1	High level operational concept description	xv	x					x	
NOV	2	Operational node relationship description		xv	v	v				
NOV	3	Operational information Exchange matrix		xv	v					
NOV	4	Organisational relationships chart	v	xv						
NOV	5	Operational activity model		xv	x			x	x	
NOV	6	Operational behaviour	v	xv	v					
NOV	7	Information model			xv					
NSOV	1	Service taxonomy		xv	xv					
NSOV	2	Service definitions		x	x	x			x	
NSOV	3	Capability to service mapping		xv	x					
NSOV	4	Service behavior		x	x	x				
NSOV	5	Service functionality		x	x					
NSOV	6	Service composition		v	v	x	x	x	x	
NSV	1	System interface description		v	v	xv				
NSV	2	System communications description			v	xv				
NSV	3	Resource Interaction Matrix				x				
NSV	4	Systems functionality description			xv					
NSV	5	System function to operational activity			xv	x				
NSV	6	Systems data exchange matrix			xv	v				
NSV	7	System quality requirements description				xv				
NSV	8	Systems configuration management			v	v	x	x		
NSV	9	Technology and skills forecast					x			x
NSV	10	Resource behaviour		v	xv	x				
NSV	11	System data model			xv	x				
NSV	12	Service provision		x	x		x			
NTV	1	Standards profile				xv				
NTV	2	Standards forecast				xv	x			x
NTV	3	Standard configurations				v		x	x	
NPV	1	Programme Portfolio relationships					x	x	x	
NPV	2	Programme to capability mapping				v	x	x	x	

Figure 3. Assignment of NAF subviews to TOGAF phases.

- Our detailed mapping does not find direct usage of All views in TOGAF, except NAV-1. This is thus an extension that NAF introduces.
- The TOGAF content framework does not provide much detail for the later phases (E-H), so here the earlier mapping is valuable.
- Our mapping finds more use for the operational views in the information systems architecture phase. This has to do with an emphasis on the logical, implementation-independent models of the applications and data, which are important for portfolio planning.
- Our mapping also finds more use for system views in the information systems architecture, in order to represent physical application components as well as the logical ones.
- Detailed service oriented views, defining functions, composition and behavior, do not have a clear counterpart in TOGAFs content framework, where operational and system diagrams seem sufficient for representing these aspects on the logical and physical layer, respectively.

The detailed mapping also illustrates some important features of NAF. First and foremost, some NAF subviews fill several different purposes, according to TOGAF ADM. The most important case is NSV-1, which can be used for at least 12 different architecture products: Application Portfolio Catalog, Interface Catalog, System/Organization Matrix, Role/System Matrix, Application and User Location Diagram, Software Engineering Diagram, Software Distribution Diagram, Technology Standards Catalog, Technology Portfolio Catalog, System/Technology Matrix, Environments and Locations Diagram, and Platform Decomposition Diagram. These products should be distinguished in the architecture models as different diagrams.

Vice versa, there are several TOGAF architecture products that require modeling of constructs from multiple NAF subviews. These products are candidates for customized views as extensions to the NAF content framework. The typical examples are diagrams that show connections from the logical architecture down to the physical, or from overarching capability views down to operational models. NAF supplies some of these, but not everything that TOGAF requires.

3.5. NAF and TOGAF Metamodels

An underlying issue in the architecture products mapping presented above, is the mapping between language constructs in NAF and TOGAF. Their metamodels are quite different. TOGAF presents a simple conceptual definition of a modeling language, with a core set of elements and five extensions. NAF defines a much larger metamodel as a UML profile. It is a technical implementation, fragmented into separate metamodel diagrams for each subview, and lacks a conceptual core that connects similar constructs across the views into a unified type hierarchy.

One critical issue that we had to resolve, was the situation where a single TOGAF construct could be mapped to a number of NAF constructs within different views. An example is given below, for Organization Unit. When defining scope and objectives, NAF models organizations as Enterprises, and links them to the phases or time spans

that the architecture descriptions address. On the operational level, Nodes represent organizational actors, and you can also model actual organization units. In the physical systems architecture, organization types interact with other kinds of resources, and capability configurations represent the set of human and physical resources that together realize a capability, implement a node, or provide a service.

In addition to these direct representations of Organization Unit, NAF Capability can also be used for defining the functional composition and dependencies of the organization, at the logical level without relating it to concrete organizations.

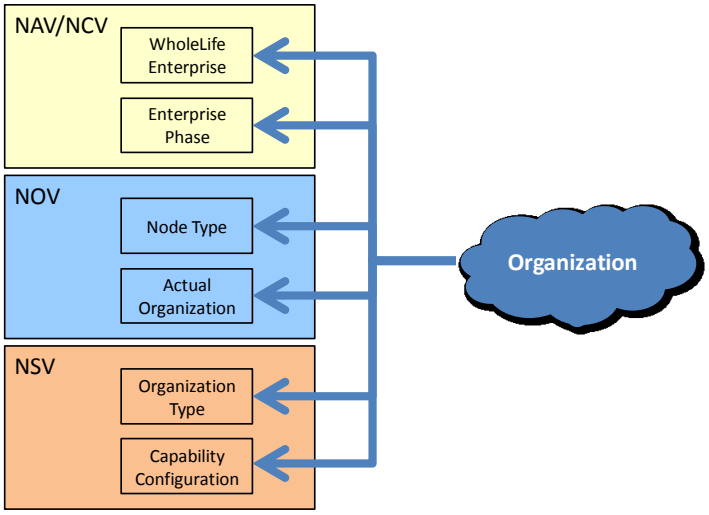

Figure 4. Example of metamodel mapping problem.

Similarly, a TOGAF Process can be mapped to a Mission, Enduring Task, Standard Operational Activity, Operational Activity, Service Function, and Function in NAF.

In addition to making it difficult to apply TOGAF ADM guidelines directly to a NAF architecture model, these metamodel mismatches causes a fragmentation of the architecture. Where TOGAF connects both scoping and objectives, logical operations and physical realization to a single Organization Unit element, NAF forces you to represent the organization as different elements in different views, and the framework does not even include all the links needed for linking these different representation together. These issues had to be resolved in our architecture framework implementation.

The most important metamodel mapping challenge that we encountered, however, was due to the more detailed and physical perspective that NAF takes, compared to TOGAF. Where TOGAF provides direct links for simple mapping between the core elements of the business, application, data, and technology architectures, it often takes several steps of more detailed indirect links to connect the same elements in NAF. For instance, rather than simply stating that a Project contributes to a Goal, in NAF you always have to state *when* (CapabilityIncrement, ProjectMilestone) the contribution happens, and *what* (CapabilityConfiguration) it consists of. If you model goals as EnterpriseGoals rather than Capabilities, you have

two additional steps to go, via EnterpriseVision. Another example is the link between a Service and the data it uses. In NAF you must go from Service via ServiceInterface, ServiceInterfaceDefinition, ServiceOperation, and ServiceParameter before you arrive at the Entity in the information model. There are several examples like these, where we often have decided to add the direct relationships from TOGAF to our metamodel, in order to create a more high level, cost-effective, and sustainable model.

4. Implementation and Usage Experiences

The initial analysis of NAF and TOGAF, as reported above, identified a difference in their top level frameworks, focusing on

- Architecture layers (business, application, data, technology) in TOGAF,
- Depth of detail (overarching, reference, baseline, target) in NAF.

In order to integrate the frameworks, the Norwegian Armed Forces Architecture Framework in Figure 5 proposes a combination of these two dimensions, with NAF going down and TOGAF across. We have decomposed the business architecture of TOGAF into the NAF categories of capabilities, processes and organizations, which are similar to the three layers of business architecture in TOGAF. Similar frameworks are found in NATO documents [6], and in the Danish government's OIO "shelf system" [9], where the NAF layers are called conceptual, logical, and physical.

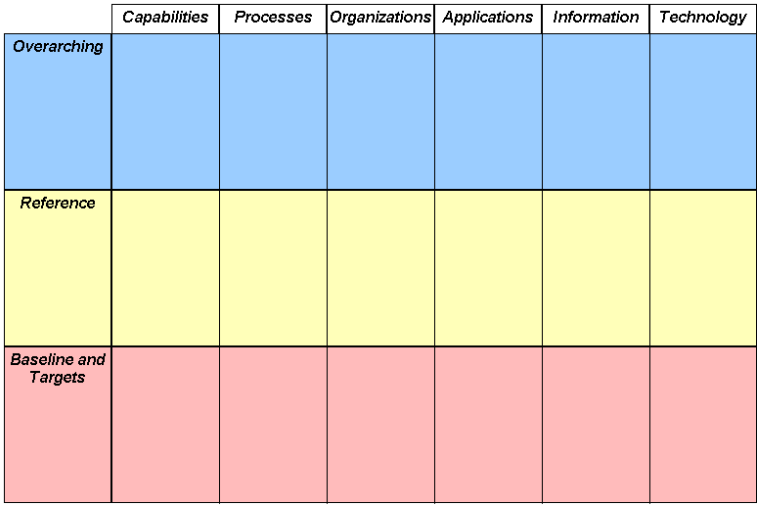

Figure 5. The architecture content framework of the Norwegian Armed Forces.

The organization principles of this two-dimensional framework are:

- *Vertical traceability* from one level to the next through specialization, decomposition, instantiation,

- *Horizontal traceability* on each level through various associations and dependencies, e.g. “uses”, “supports”, “is provided by”,
- *Dependencies* between similar elements within each cell, e.g. communication relationships, information flow, and some decomposition and specialization hierarchies.

In our implementation, this matrix guides the organization of content for asset management in the governance framework. The upper level primarily contains NAF all and capability views, while the middle level deals with operational, service oriented, and program views. The lower level consists mainly of system and technical views.

The figure below places core concepts from the NAF metamodel in the architecture content framework. It provides high level guidance about how to model on the different levels. Some concepts are applied on more than one level, e.g. Capability, Service, and Entity. A common set of types are used for organization, application and technology resources on each level. In the overarching architecture, we focus on the services that these resources offer without bothering about their structure, while the reference architecture captures more detailed services and the nodes that perform them. Finally, on the physical level, ResourceType specializations are defined for organizations (organization, role, post), applications (software), and technology (artefact, physical architecture).

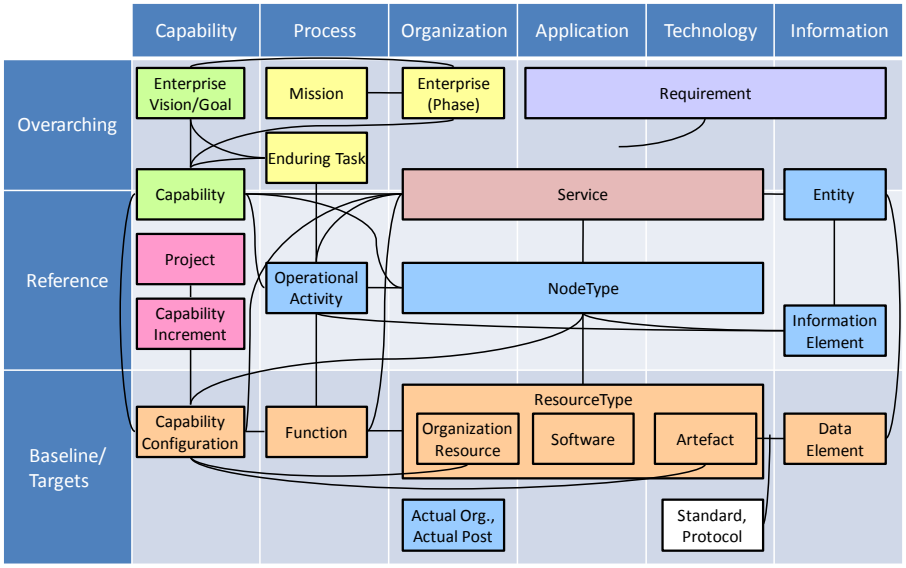

Figure 6. Core NAF concepts in the architecture content framework.

The architecture framework further defines which NAF constructs to use at the overarching, reference, target and baseline levels respectively, as well as which views and subviews should be emphasized. This solution combines framework adaptation with enterprise level adaptation, because we found the different layers of NAF to correspond well with the concerns of the core stakeholder groups in the armed forces.

The overarching architecture is needed by top level management for planning long term capability development, while the reference architecture is used by the IT departments for business and IT alignment, project portfolio management, and managing system ownership. Target architectures are mainly applied by the IT departments and their suppliers in acquisition and development projects.

4.1. Implementation Experiences

The amount of work required for creating a fully functional implementation of a metamodel that is not supported out of the box by your tool vendor, should not be underestimated. In our case, we've had to make close to 150 changes to the NAF metamodel provided as a Sparxsystems model by the UK MOD [5], and several of the changes had to be replicated in a number of diagrams. Just a few of these changes were of a conceptual nature, many simply had to do with fixing omissions in the metamodel diagrams of each subview, so that the produced profile would be complete.

As it turned out, some of the ways in which MODAF and NAF uses UML constructs are theoretically valid, but practically not very useful. In particular, we changed the way several relationships are modeled. In the specification, NAF represents relationships in many different ways. Almost none of them use UML Associations, though some use Dependency. The constructs that created problems for us, however, modeled relationships between elements as Property (22 cases), TaggedValue (33 cases), Attribute (5 cases) or Slot (2 cases). In addition to the added complexity of representing relationships in so many different ways, these solutions were difficult to visualize and track in the tool, cumbersome to model because they could not be dragged directly from a profile toolbox, and not as interoperable with non-UML tools that support NAF. Rather than representing relationships the way you would if you were programming, we thus decided to switch to a more high-level modeling approach, using two way links rather than one way attribute type references.

The representation of concrete instances was another area where we chose to extend the standard NAF metamodel. NAF out of the box supports the modeling of actual organization elements and projects as instances. We however saw the need to model other actual elements in the same way, e.g. locations, IT resources, military vessels. We thus allow the modeling of instances of all types, but see no need for defining separate stereotypes for each of these instance types, when the type is already given by the stereotype of the classifier of the instance. In order to ensure a uniform and simple framework, we also decided against a common practice in the past of using instances that stand for their classifier in a given diagram, as pure symbolic instances.

A final important simplification was to remove stereotypes for behavior modeling. These constructs are already built into the standard UML diagrams, and adding several stereotypes just to say which kind of NAF elements the behavior diagram elements stand for, was unnecessary when the models already contain these links. There was also a problem in many cases that these elements could not easily be dragged from the toolboxes, and the standard UML elements catered for more convenient ways of modeling. Finally, stereotyping some elements, like Ports,

cluttered their visual presentation in the diagrams, e.g. by making their minimum size much larger than we wanted.

In addition to adapting the metamodel and UML profile, our implementation involved customizing analysis reports, XSLT transformations and scripts for exchange of model data with other tools. A particularly challenging issue was the need for extracting portions of the overall architecture database into smaller models for parallel development, e.g. by the suppliers of a given project. Again UML made things difficult. The roles of a relationship are modeled as properties of the elements that participate in the relationship. This means that adding a relationship involves altering the packages where both endpoints reside. When everything in the entire architecture is connected indirectly to everything else, this makes it difficult to modularize the architecture into sub-models that can be worked on independently. We finally arrived at a solution to this problem that involves strictly controlled use of the compare and merge functionality of the tool. This is coupled with a custom script that separates out a package from the rest of the architecture, putting all of the elements that the package refers to but does not own, in a separate Context package that the user has to handle the right way during the merging process.

4.2. Usage Experiences

The established architecture framework and development methodology has been in place a few months at the time of this writing. Major aspects of 5 development projects have been modeled, including project definitions with milestones and objectives, requirements, operational nodes and their information exchange, and systems with components, interfaces and standards. At the time of this writing, the architecture repository contains roughly 17K elements and 29K relationships.

It soon became evident that a generic architecture development methodology was insufficient to motivate usage, so a number of customized methodologies have been developed, for requirements management at the project and portfolio level, and for integrated solutions delivery across projects. These methodologies apply a small subset of TOGAF ADM tasks and NAF architecture views, extended with custom views and tasks.

Finally, the implementation has proven capable of insourcing a number of previously developed architecture descriptions, from national as well as international activities. This also includes models developed in other tools, like ARIS and ERWin, as well as business architecture models developed in a locally defined metamodel by the Norwegian Defense Research Establishment.

5. Conclusions and Further Work

As the experiences reported in this paper illustrate, there are fundamental differences between NAF and TOGAF that you should take into account when bringing the two together. The physical systems engineering perspective of NAF conflicts with the information systems approach of TOGAF, and the enterprise wide portfolio management scope of TOGAF does not always fit the acquisition project focus of

NAF. These differences are natural consequences of the differences between military and business environments. In most businesses, hardware is a commodity, and most of the IT complexity lies in the application software. Consequently, this is the primary focus of TOGAF. Military hardware, on the other hand, is more custom made, with diverse and dynamic communication backbones. This implies that the cost, uncertainty, and complexity of the IT architecture to a much larger extent reside on the physical level. NAF consequently pays more attention to these aspects than TOGAF does.

Where TOGAF proposes an elaborate methodology and a simple content framework, NAF contains a simple methodology and an elaborate content framework. The two approaches are thus complementary. Ideas for simplifying the rather complex methodology of TOGAF or the content framework of NAF, can be derived from the simpler solutions chosen by the other standard. So far, this has mainly resulted in a simpler metamodel in our work, while the architecture development methodology to a greater extent has been adapted in order to fit with the local organizational practices and procedures. The work has also resulted in a number of change requests for NAF, put forward to NATO. As many stakeholders still see the metamodel as too complex, further simplification is ongoing.

Compared to previous work [2,12,13], our experiences shows that going all the way to implementation uncovers a lot more challenges than a high level conceptual analysis. Future practice-oriented enterprise modeling research should similarly focus on real world implementations to understand which differences actually make a difference when you compare modeling frameworks.

References

1. Department of Defense, *Department of Defense Architecture Framework version 2.0*, 2009.
2. Hi-Q Systems: *TOGAF to MODAF mapping*, 2008.
3. Jørgensen, H.D., Ohren, O.P. (2004): *Achieving Enterprise Interoperability through the Model-based Architecture Framework for Enterprises*, Enterprise modelling and ontologies for interoperability workshop (EMOI), CAiSE, Riga, Latvia.
4. Jørgensen, H.D.: *Enterprise modeling – What We Have Learned, and What We Have Not*, Practice of Enterprise Modeling (PoEM), Stockholm, Sweden, Springer, 2009.
5. Ministry of Defence, *Ministry of Defence Architecture Framework version 1.2.004*, 2010.
6. NATO NC3 Board: *Compendium of NNEC-Related Architectures*, 2006.
7. NATO NC3 Board: *RFCP Regarding NAF v.3.1, Chapter 5*, 1 March 2010.
8. NATO NC3 Board: *NATO Architecture Framework (NAF) v.3*, appendix 1 to annex 1 to AC/322-D (2007) 0048, 2009.
9. OIO Architecture Guide, http://ea.oio.dk/
10. Object Management Group: *Unified Profile for the Department of Defense Architecture Framework (DoDAF) and the Ministry of Defence Architecture Framework (MODAF)*, v. 2.0, 2010.
11. The Open Group: *TOGAF Version 9*, standard, 2009.
12. The Open Group: *The Open Group Architecture Framework (TOGAF) and the US Department of Defense Architecture Framework (DoDAF)*, white paper, 2006.
13. The Open Group: *The Open Group Architecture Framework (TOGAF 9) and the US Department of Defense Architecture Framework (DoDAF)*, white paper, rev. 2010.