Towards a Comprehensive DND/CF Enterprise Architecture Methodology

- A Critical Review DNDAF for an Integrated C2 Capability Development

R. Farahbod
A. Guitouni
É. Bossé
(DRDC Valcartier)

Defence Research and Development Canada – Valcartier

Technical Report
DRDC Valcartier TR 2011-022
June 2013

Towards a Comprehensive DND/CF Enterprise Architecture Methodology

- A Critical Review DNDAF for an Integrated C2 Capability Development

R. Farahbod
A. Guitouni
É. Bossé
(DRDC Valcartier)

Defence Research and Development Canada – Valcartier

Technical Report
DRDC Valcartier TR 2011-022
June 2013

IMPORTANT INFORMATIVE STATEMENTS

Abstract

The objective of this report is to present a critical review of the DNDAF as a core enterprise architecture (EA) framework to support the development of Integrated Command and Control Capability. In this document, we focus on systems engineering. In systems engineering, architectures facilitate the understanding and communication of different aspects of a system by providing a structured approach to document requirements, design decisions, and technical details of the implementation. This report presents a study of the concept of EA and a number of top EA frameworks in order to identify different aspects of EA that each framework covers and the various components each. The differentiation between the notions of methodology and framework in the context of EA leads to a critical review of the DNDAF framework. This report proposes a review of DNDAF, its purpose, and its components, and puts forward a set of recommendations for improvement in order to achieve a comprehensive DND/CF EA methodology and framework. This methodology is required for developing an Integrated Command and Control Capability. Future work includes determining the critical components (e.g., tools, methods, standards) that should be developed in order to support the implementation of a comprehensive EA for DND/CF.

Résumé

L’objectif de ce rapport est de présenter une analyse critique de la DNDAF comme élément central d’un cadre d’architecture d’entreprise (AE) pour soutenir le développement de la capacité intégrée de commandement et contrôle (IC2C). Dans ce document, nous nous concentrons sur l’ingénierie de système de systèmes. En ingénierie de systèmes, les architectures facilitent la compréhension et la communication des différents aspects d’un système en fournissant une démarche structurée pour documenter les exigences, les décisions de conception et les détails techniques de la mise en œuvre. Ce rapport présente une étude de la notion d’AE et un certain nombre de cadres d’AE afin de dégager les différents aspects d’une AE et ses divers composants. La différenciation entre les notions de méthodologie et de cadre dans le contexte d’AE a conduit à un examen critique DNDAF. Ce rapport propose une revue de DNDAF, son but et ses composantes et met en avant une série de recommandations pour son amélioration afin de mettre en place une méthodologie d’AE pour le MDN/FC. Cette méthodologie est nécessaire au développement d’une IC2C. Les travaux futurs comprendront la détermination des composantes critiques (p. ex., outils, méthodes, normes) qui devraient être développées afin d’appuyer la mise en œuvre d’une AE pour le MDN/CF.

This page intentionally left blank.

Executive summary

Towards a Comprehensive DND/CF Enterprise Architecture Methodology

R. Farahbod, A. Guitouni, É. Bossé; DRDC Valcartier TR 2011-022; Defence R&D Canada – Valcartier; June 2013.

Introduction or background: The objective of this report is to present a critical review of the DNDAF as a core enterprise architecture framework to support the development of Integrated Command and Control Capability. To develop a future Integrated Command and Control (C2) capability there is a need for a comprehensive system engineering methodology across the Department of National Defence. C2 occurs at many levels of an organization. C2, at the enterprise level, shapes the force (or the enterprise) determining the purpose of the organization, its priorities, and ultimately the capabilities it has. Thus, C2 at the enterprise level determines what is possible. C2 at the mission level is about employing the assets of an organization, its people, systems, materiel, and its relationships with others in the pursuit of mission-specific goals and objectives (intent).

In this document, we focus on a system of systems engineering. In systems engineering, architectures facilitate understanding and communication of different aspects of a system by providing a structured approach to document requirements, design decisions, and technical details of the implementation. Especially in dealing with development and maintenance of large complex systems, architectures help in managing the complexity, through techniques such as modularization and abstraction at various levels, and facilitate change by documenting of the components, relationships, and constraints of the systems or processes. Modern enterprises, especially in the public sector, are perhaps the most complex systems we have ever created. With different organizations and a vast number of stakeholders, each with different interests and concerns, a proper mechanism has to be in place to control complexity, facilitate change, and align different aspects of the enterprise towards achieving common goals and objectives. In light of this observation, Enterprise Architecture (EA) promises management with insight and overview of the enterprise to control complexity and support decision making by providing a continuous process of cre-ation, maintenance, and dissemination of information about the enterprise.

In 2001, the Department of National Defence and the Canadian Forces (DND/CF) called for the use of enterprise architecture. In response to this, the Directorate of Enterprise Architecture (DEA) has the goal of establishing enterprise architecture as a practice within DND/CF. The expectation has been that the DND/CF enterprise architecture practice, managed through the DND/CF Enterprise Architecture Program (DND EAP), will improve the effectiveness and efficiency of DND/CF. The program aims to define the framework, the methodologies and the tools needed to establish an EA practice within

DND/CF. All architecture efforts in DND/CF will follow the framework and methodologies defined in DNDAF to ensure interoperability and integration into an overall EA repository. DNDAF is an important and necessary step forward towards establishing the practice of EA within DND/CF. At the time when DNDAF version 1.5 was released (March 2008), it was one of the most advanced military EA frameworks, compared to DoDAF, MODAF, and others. However, to fulfill its purpose and to be effective in practice, DNDAF has to be more comprehensive and needs to address a number of shortcomings and limitations. To begin with, it has to be clear (and clearly stated) what DNDAF is supposed to be: is it expected to be a framework to provide guidance to DND/CF architects [1, Vol. 1, Sec. 1.2], or is it supposed to represent the DND/CF enterprise [1, Vol. 2, Sec. 2]?

Enterprise architecture is a continuous practice. In our opinion, the current version of DNDAF lacks two important pieces required when adopting a practice of EA within an enterprise: a) a set of fundamental guiding principles that states how the practice of EA is expected to support the enterprise business direction and processes, and b) a proper methodology (or at least a process) that guides the development and maintenance of architectural artefacts. In this report, we propose a set of potential solutions that might extend DNDAF and create a DND/CF enterprise architecture.

Approach: In order to provide a meaningful analysis of DNDAF, we structured our approach around the following steps:

a) studying the concept of Enterprise Architecture in order to better understand its goal and its requirements;

b) studying a number of top EA frameworks in order to identify different aspects of EA that each framework covers and the various components each one consists of;

c) differentiating between the notions of *methodology* and *framework* in the context of EA, and identifying the main components of an EA framework and an EA methodology based on the available literature;

d) critically analyzing the DNDAF framework in light of these studies considering the purpose of DNDAF and the goal of DND/CF EAP, as stated in DNDAF documentation;

e) finally, providing a discussion and recommendations for future work and some of the challenges that we may want to face in order to improve DNDAF.

Results: This report proposes a critical review of DNDAF its purpose and its components, and puts forward a set of recommendations for improvement in order to achieve a comprehensive DND/CF EA methodology and framework.

Significance: In the conclusion, we propose a set of recommendation to evolve DNDAF towards a comprehensive EA methodology for DND. In particular, this methodology is required for developing an Integrated Command and Control Capability.

Future plans: Future work includes determining the critical components (e.g., tools, methods, standards) that should be developed in order to support the implementation of a comprehensive EA for DND/CF.

Sommaire

Towards a Comprehensive DND/CF Enterprise Architecture Methodology

R. Farahbod, A. Guitouni, É. Bossé ; DRDC Valcartier TR 2011-022 ; R & D pour la défense Canada – Valcartier ; juin 2013.

Introduction ou contexte : L'objectif de ce rapport est de présenter une analyse critique de DNDAF comme cadre central d'une architecture d'entreprise pour soutenir le développement de la capacité de commandement et contrôle intégrée (IC2C). Pour développer une capacité future de commandement et contrôle intégrée, il est nécessaire d'avoir une méthodologie d'ingénierie de systèmes au sein du ministère de la Défense nationale. C2 se produit à plusieurs niveaux d'une organisation. C2, au niveau de l'entreprise, façonne la force (ou l'entreprise) pour déterminer le but de l'organisation, ses priorités et ultimement ses capacités. Ainsi, C2 au niveau de l'entreprise détermine ce qui est possible. C2 au niveau de la mission se limite à employer les ressources de l'organisation, ses capacités humaines, systèmes et matérielles, et leurs relations avec d'autres pour la réalisation de but et objectifs (intention) spécifiques à la mission.

Dans ce document, nous concentrons l'ingénierie de système de systèmes. En ingénierie des systèmes, les architectures facilitent la compréhension et la communication des différents aspects d'un système en fournissant une démarche structurée pour la documentation des besoins, des décisions de conception et les détails techniques de la mise en œuvre. Surtout en matière de développement et la maintenance de grands systèmes complexes, les architectures aident à gérer la complexité, grâce à des techniques telles que la modularité et l'abstraction à différents niveaux et à faciliter le changement en prévoyant des mécanismes de documentation des composants, des relations et des contraintes des systèmes ou des processus. Les entreprises modernes, notamment dans le secteur public, sont peut-ˆetre les syst`emes plus complexes que nous avions jamais créés. Avec différentes organisations et un grand nombre d'intervenants avec des intérêts et des préoccupations diverses, un mécanisme approprié doit être mis en place afin de contrôler la complexité, faciliter le changement et aligner les différents aspects de l'entreprise vers l'atteinte des objectifs et des buts communs. À la lumière de cette observation, l'architecture d'entreprise (AE) promet la gestion, avec perspicacité et vue d'ensemble, de l'entreprise afin de contrôler la complexité et soutenir la prise de décision en fournissant un processus continu de création, l'entretien et diffusion d'informations sur l'entreprise.

En 2001, le ministère de la Défense nationale et les Forces canadiennes (MDN / FC) ont demandé l'utilisation de l'architecture d'entreprise (AE). En réponse à cela, la Direction de l'architecture d'entreprise (DAE) a été crée dans le but d'établir l'AE comme une pratique au sein du MDN/FC. L'attente a été que la pratique de l'architecture d'entreprise du MDN/FC, gérée par le MDN/FC Enterprise Architecture programme EAP (MDN), va améliorer l'efficacité et l'efficience du MDN/FC. Ce programme vise à définir le cadre, les méthodologies et les outils nécessaires pour établir une pratique d'AE au sein du MDN/FC. Tous les efforts d'architecture de système du MDN/FC sont supposés suivre le cadre et les méthodes définies dans DNDAF pour assurer l'interopérabilité et l'intégration dans un référentiel d'AE globale. DNDAF est un pas important et nécessaire à l'établissement d'une pratique d'AE au sein du MDN/FC. Au moment où DNDAF version 1.5 est sorti (mars 2008), il fut l'un des cadres d'AE militaires les plus avancés comparés à DoDAF, MODAF et d'autres. Cependant, pour remplir son objet et être pratiquement efficace, DNDAF doit être davantage amélioré et certaines lacunes doivent être abordées. Dans un premier temps, il est nécessaire de clarifier ce que DNDAF est censé être : s'agit-il d'un cadre pour guider les architectes du MDN/CF [1, Vol. 1, article 1.2] ou est-il censé représenter l'entreprise du MDN/FC [1, Vol. 2, article 2] ?

L'architecture d'entreprise est une pratique continue. À notre avis, il manque à la version actuelle de DNDAF deux composantes importantes et requises pour l'adoption d'une pratique d'AE au sein d'une entreprise : a) un ensemble de principes fondamentaux et directeurs qui stipulent comment la pratique d'AE est-elle supposée soutenir la direction des affaires de l'entreprise et ses processus, et b) une méthodologie appropriée (ou au moins un processus) qui guide le développement et l'entretien des artefacts architecturaux. Dans le cadre de ce rapport, nous proposons un ensemble de solutions potentielles susceptibles d'étendre DNDAF et de créer une architecture d'entreprise pour le MDN/FC.

Approche : Notre approche est structurée selon les étapes suivantes :

a) étudier le concept d'architecture d'entreprise afin de mieux comprendre son objectif et ses exigences ;
b) étudier un certain nombre de cadres d'AE afin d'identifier les différents aspects de l'architecture d'entreprise que couvre chaque cadre et les divers composantes de chacun ;
c) différencier les notions de méthodologie et de cadre dans le contexte d'AE et d'identifier les principaux composants d'un cadre d'architecture d'entreprise et une méthodologie d'architecture d'entreprise basée sur la documentation disponible ;
d) mener une analyse critique du cadre DNDAF à la lumière de ces études, compte tenu de l'objet de DNDAF et de l'objectif du MDN/FCF EAP comme indiqué dans la documentation de DNDAF ;
e) enfin, discuter et faire des recommandations pour des travaux futurs et certains des défis afin d'améliorer les DNDAF.

Résultats : Ce rapport présente une revue critique de DNDAF, son but et ses composantes, et met de l'avant une série de recommandations pour son amélioration en vue d'AE complète pour la réalisation d'une capacité intégrée de Commandement et Contrôle pour le MDN/FC.

Importance : Nous proposons un ensemble de recommandations afin de faire évoluer DNDAF vers une méthodologie d'EA globale pour le MDN. En particulier, cette méthodologie est nécessaire pour le développement d'une capacité intégrée de Commandement et Contrôle.

Perspectives : Les travaux futurs comprendront la détermination des composants critiques (p. ex., outils, méthodes, normes) qui devraient être développés afin d'appuyer la mise en oeuvre d'AE complète pour le MDN/FC.

Table of contents

List of figures

1 Introduction

1.1 Background

To develop a future Integrated Command and Control (C2) capability there is a need for a comprehensive system engineering methodology across the Department of National Defence. C2 occurs at many levels of an organisation. **C2, at the enterprise level**, shapes the force (or the enterprise) determining the purpose of the organisation, its priorities, and ultimately the capabilities it has. Thus, C2 at the enterprise level determines what is possible. **C2 at the mission level** is about employing the assets of an organisation its people, systems, materiel, and its relationships with others in the pursuit of mission-specific goals and objectives (intent).

NATO C2 reference model [7] and its evolutions [1] [2], are described in SAS-085 [3]. According to SAS-065, [4] two key realities dominate thinking about *command and control* (C2) in the 21st century. The first is the nature of the 21st century military mission space. This space is characteriszed by its extreme uncertainty. In addition to the high-intensity combat operations that are traditionally associated with military operations, the 21st century mission space has expanded to include a wide spectrum of mission challenges, ranging from providing support to multi-agency disaster relief operations to complex coalition efforts within a political-military environment involving a large variety of military and non-military actors; which we describe as *Complex Endeavours*.

The second reality is the ongoing transformation of 21st century militaries, and for that matter, other 21st century institutions and actors from the Industrial Age to the Information

1. NATO SAS-065 Research Task Group. *NATO NEC C2 Maturity Model Overview*. Draft for Peer Review, 2008. Available at `www.dodccrp.org`
2. NATO SAS-085 Research Task Group, NATO
3. NATO SAS-085 *'C2 agility and requisite maturity'* has three objectives:
   - To understand and validate the implications of C2 Agility (or a lack of C2 Agility) for NATO missions by improving the breadth and depth of our understanding of C2 Agility;
   - Match the characteristics of alternative C2 Approaches to situational attributes (e.g. complexity, dynamics) so that Requisite C2 Maturity and its encompassing C2 Agility can be recognized for complex endeavours;
   - Support the dissemination of this improved understanding through applications involving appropriate military, research and educational institutions.
4. SAS-065 is a NATO research task group operating under the auspices of the SAS Panel. It was formed in 2006 for the purpose of developing a C2 Maturity Model for network-enabled operations. SAS-065's principal products include a detailed description of a NATO NEC Command and Control Maturity Model (N2C2M2) with a User Guide (see section entitled *Applying the NATO NEC C2 Maturity Model*) and a revised C2 Conceptual Reference Model (originally developed by SAS-050). SAS-065 builds on the work of a series of research task groups dating back to 1995 that have explored issues in command and control. These have included RSG-19 and SAS-026, which produced the NATO Code of Best Practice for C2 Assessment, and SAS-050, which produced the C2 Conceptual Reference Model. The members of SAS-065 and the countries and organizations they represent can be found in the *Acknowledgments* section.

Age. With this transformation comes the ability to leverage new information technologies. This has had, and will continue to have, a profound effect on how institutions manage themselves and how they can work with coalition partners.

The Canadian Forces have identified the requirement for an integrated C2 Information Capability (IC2C) [Ref. CDS Directive to CFD June 06, VCDS Direction CF IC2C, and Integrated C2 Capability Strategy]. DRDC is undertaking an applied research initiative to support CFD and DCCI/DGIMO/ADM(IM) creating a vision for the CF's Integrated Com-mand and Control Capability (IC2C), and establish an implementation strategy for achieving that vision. It is therefore important to define what "integrated C2 Capability" for the CF and document its key enablers. The integrated C2 capability requirements include but not limited to:

- System of Systems networking people, data and capabilities,
- Open System Architecture for "plug-and-play" and interoperability
- Service-Oriented Architecture interoperability,
- Getting OGDs to think & work in secure environment,
- JIMP Secure Environments Whole of Enterprise Data Knowledge Defence Enterprise Resource Planning (ERP) Capability: ERP systems with integrated information for DND/CF data and information management, transaction processing, and decision making requirements.

In this document, we focus on a system of systems engineering. In systems engineering, *architectures* facilitates understanding and communication of different aspects of a system by providing a structured approach to document requirements, design decisions and technical details of the implementation. Especially in dealing with development and maintenance of large complex systems, architectures helps in managing the complexity, through techniques such as modularization and abstraction at various levels, and facilitate change by providing a documentation of the components, relationships, and constraints of the systems or processes.

Modern *enterprises*, especially in the public sector, are perhaps the most complex systems we have ever created. With different organizations and a vast number of stakeholders each with different interests and concerns, a proper mechanism has to be in place to control complexity, facilitate change, and align different aspects of the enterprise towards achieving common goals and objectives. In light of this observation, *Enterprise Architecture (EA)* promises management with insight and overview of the enterprise to control complexity and support decision making by providing a continuous process of creation, maintenance, and dissemination of information about the enterprise. [8, 9]

In 2001, as one example of a large enterprise in public sector, the Department of National Defence and the Canadian Forces (DND/CF) called for the use of enterprise architecture. [5] In response to this, the Directorate of Enterprise Architecture (DEA) was tasked with the goal of establishing enterprise architecture as a practice within DND/CF.

5. Canadian Defence Planning Guide (DPG) 2001

The expectation has been that the DND/CF enterprise architecture practice, managed through the DND/CF Enterprise Architecture Program (DND EAP), will improve the effectiveness and efficiency of DND/CF. [1]

Explicit details of the DND EAP are provided through the DND/CF Architecture Framework (DNDAF) [1]. It aims at defining the framework, the methodologies, and the tools needed to establish an EA practice within DND/CF. All architecture efforts in DND/CF will follow the framework and methodologies defined in DNDAF to ensure interoperability and inte-gration into an overall EA repository. The latest version of the DNDAF[6] comes in four volumes:

1. *Overview and Definitions* provides an overview of the main concepts of EA, its purpose and its role within DND/CF.
2. *DND/CF Views and Sub-views* provides a structure for organizing DND/CF enterprise information into six architectural views and corresponding sub-views each documenting different aspects of the architecture (see Section 5.2).
3. *DND/CF Architecture Data Model (DADM)* defines a standard set of architecture data entities and their relationships showing how the underlying information is organized in DNDAF.
4. *User Guide* provides interim guidance and advice for the effective use of DNDAF in creation of architecture information and products until an EA toolset is procured and available for use across DND/CF.

A DNDAF is therefore expected to become:

- The standard framework to be tailored for all DND Communication and information systems projects and system, thereby ensuring that capability production and exploitation are synchronized with and a contributor to the overarching Integrated Command and Control Capability. As a result, there will be a common approach for describing architectures across all projects, and allow for easy analysis re-use by each team.
- The framework that provides the necessary documentation to allow the reader to "wade in" gradually to an understanding of DND Integrated Command and Control Capability.
- The framework that provides the flexibility to select the data representation required to meet IC2 needs.
- The framework can be modified to document a "capability" instead of just a "system", which is essential to the IC2 level of analysis. As a result, structured systems engineering techniques may be used on the IC2 initiative to extend to the capability level.

The above types of benefits have been recognized in US programs, as all DoD projects must now have DoDAF views for approvals within the Joint Capability Development and Integration System (JCIDS). DoDAF provides a wide selection of "views" that are developed and used to fully document the current state, or the proposed future state, of a system or capability. It is not necessary to use all of these views at all times, however the existence of

6. DNDAF Version 1.6, released May 2009 [1]

the framework provides a useful logic for the addition and integration of views as required throughout the life cycle of a project.

In this document, we examine DNDAF based on the set of expectations (listed above) in order to determine if all enterprise architectural requirements are met by DNDAF.

1.2 Structure of the Document

The rest of this document is structured as follows: Section 2 discusses the problem domain. Section 3 provides an overview of Enterprise Architecture and surveys a number of top EA frameworks and methodologies. Section 4 looks into the modeling aspect of systems engineering and discusses the role of modeling and formal methods in engineering of complex systems. An overview of DNDAF version 1.6 and its strengths and shortcomings is provided in Section 5. This overview is complemented by an analysis and a set of suggestions and recommendations for improvement in Section 6. Finally, Section 7 concludes the report, identifies the challenges ahead and discusses the future work.

2 Problem Description

The span of C2 is too broad and plural. In the N2C2M2 report (from cite NATO-SAS065- 08), the authors stated that “The term command and control is clearly a product of the Industrial age. The first use of the term as we understand it appears to be by Jomini in *The Art of War*, when he entitles a section of the book, “The Command of Armies and the Supreme Control of Operations.” It emerged as a term of art around the middle of the last century. Prior to this, command was always associated with a commander (an individual) and a headquarters (a management team). Even the idea of a formal staff does not emerge before Gustavus Adophus (1594-1632), and modern staff structures not until Napoleon Bonaparte. Modern command and control organizations trace their origins to Napoleon who is credited with the development of the first modern military headquarters and the associated creation of a “modern” command staff. At this point, the functioning of a command staff became a subject of analysis. Different militaries had different approaches to headquarters organization and correspondingly different approaches to the way in which intent was expressed and control was exercised.

For some, Command referred to a commander's intent and decisions while control was associated with how the “intent” of the commander became translated into instructions and promulgated throughout forces by the command staff. This view parses the term "the art and science of command and control" with command being the art while control is the science. This resulted in, until very recently, a bifurcation of inquiry where the study of commanders and their behaviours continued to be a subject for military historians and the study of control became fair game for a variety of scientific disciplines.

Command and Control (C2) can be interpreted in its broadest sense to include acquiring, managing, sharing, and exploiting information, and supporting individual and collective decision-making. The NATO NEC C2 Maturity Model (N2C2M2) was developed by the RTO SAS-065 Research Task Group over a period of about three years. The model starts by defining a number of C2 approaches, ranging from *Conflicted C2 to Edge C2*, that correspond to different regions within the C2 Approach Space shown in Figure 1.

In Figure 1, there is a gap between Conflicted and De-Conflicted C2 and a gap between Collaborative and Edge C2. De-Conflicted, Coordinated, and Collaborative C2 are shown without gaps between them. This is because the exact boundaries between De-Conflicted and Coordinated and between Coordinated and Collaborative are difficult to define precisely. Figure 2 gives a brief description of each of these C2 approaches, in terms of the region they occupy on the C2 approach space (described by the three variables across the top). In Figure 2, the relationships among the approaches are depicted by gaps between Conflicted and De-Conflicted, and Collaborative and Edge C2, and dashed lines between De-Conflicted, Coordinated, and Collaborative.

These different approaches to collective C2 are key considerations in determining C2 ma-

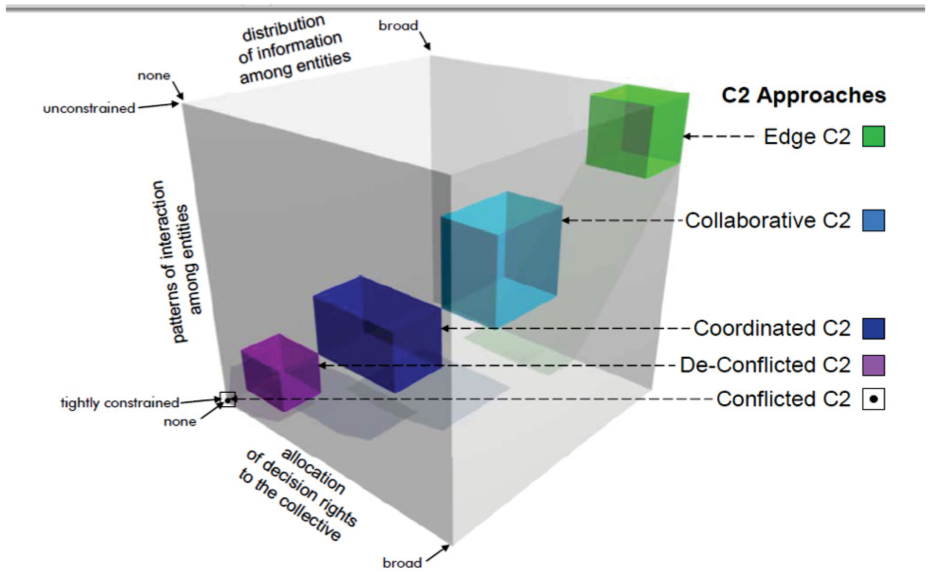
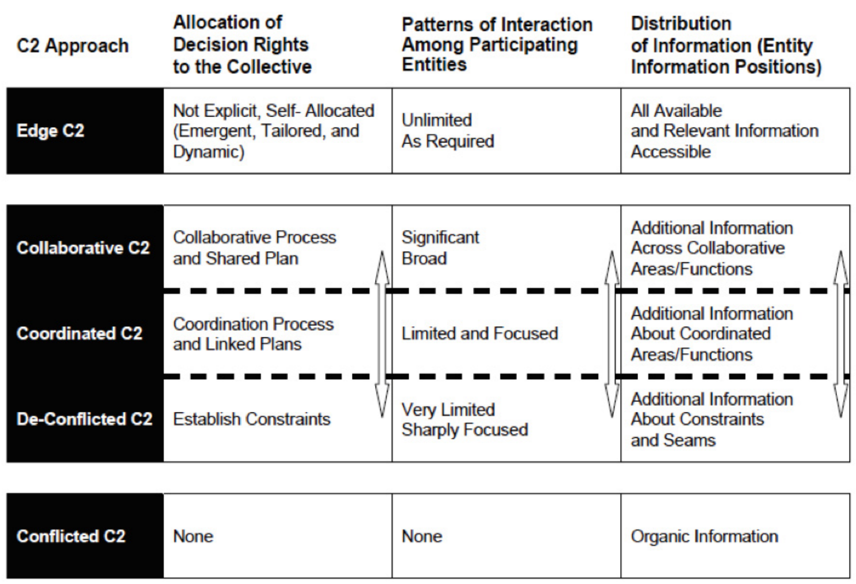


Figure 1: *C2 Approaches as regions in the C2 Approach Space (from [2])*



C2 Approach	Allocation of Decision Rights to the Collective	Patterns of Interaction Among Participating Entities	Distribution of Information (Entity Information Positions)
Edge C2	Not Explicit, Self- Allocated (Emergent, Tailored, and Dynamic)	Unlimited As Required	All Available and Relevant Information Accessible
Collaborative C2	Collaborative Process and Shared Plan	Significant Broad	Additional Information Across Collaborative Areas/Functions
Coordinated C2	Coordination Process and Linked Plans	Limited and Focused	Additional Information About Coordinated Areas/Functions
De-Conflicted C2	Establish Constraints	Very Limited Sharply Focused	Additional Information About Constraints and Seams
Conflicted C2	None	None	Organic Information

Figure 2: *Different C2 Approaches (from [2])*

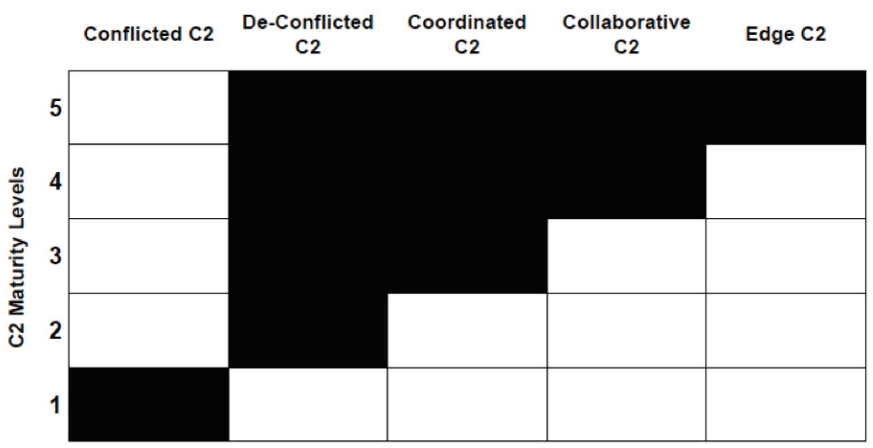


Figure 3: *Collective C2 Maturity Levels 1 to 5, defined in terms of the different possible C2 Approaches (from [2])*

turity. C2 maturity levels are defined in terms of the specific approaches to C2 that an entity or collection of entities can implement and the ability to recognize which approach is appropriate, and adopt the most appropriate associated with a specific set of C2 capabilities. Furthermore, each higher level of C2 maturity subsumes the capabilities associated with the lower levels. From the collective or coalition perspective there are thus five possible levels of C2 maturity, as shown in Figure 3.

In related work, DRDC has identified that IC2C should be considered from the two following perspectives:
- Mission imperatives: as stated in the different strategic documents, the role of the CF is to the government of Canada Canada First Strategy and be a core element of its security strategy in Canada and abroad,
- Enterprise prospect: In a networked world and during the information/knowledge age, Command and Control spans across all organisations in the National Department and even reach out to other government departments, agencies, and stakeholder.

It is important to establish a common understanding of the CF C2 requirements and identify the future S&T challenges. Process, Organization and Technology (POT) are key elements of C2. The effective networking of people, organization and technologies to conduct collaborative working is considered key to achieving an integrated C2. An understanding of organizational and individual C2 dimensions is essential to the development of requirements for an integrated C2 Capability for the Canadian Forces. The challenges of an Integrated C2 Capability are multiple and beyond one single organization or directorate. In this report, we focus on the evaluation of System of system engineering (architectures and integration) methodology. We recognize that the Integrated C2 Capabilities challenges include but not limited to:
- Characterization of the future CF force employment contexts based on the CF Force

Planning Scenarios (e.g., Cyber Operations, Urban Ops, Coalition Ops, Counter-Insurgency Ops, JIMP context-comprehensive approach, Dispersed Ops, Critical Infrastructure Protection),

- Identification of the integration requirements (e.g., between mission and enterprise C2IS, among the different services C2IS, with OGDs),
- Agility of IC2: the ability to rapidly assemble an as-needed mission-specific C2 capability from an available set of standard and relatively low-cost elements,
- Data exchange standards and interfaces: data and information modelling, data exchange requirements for interoperability across forces, with allies and OGDs (JIMP),
- Information infrastructure for sharing information: develop interoperability requirements and performance models for CF strategic networks, operational and tactical networks,
- Common Information Model for command, logistics and ISR,
- Machine-understandable information technologies: An efficient, secure, resilient, and agile method and architecture supporting federation of information services between nations (for coalitions),
- Technologies that facilitate information sharing (down to the platform level) and enable the integration of unclassified and classified systems for joint and coalition operations,
- Information assurance and security: Provide intra-, cross-, and inter-domain authentication, encryption, and information assurance/integrity services and Information quality assessment,
- Assess options for future C2 architectures in current and future platforms, and for coalition operations including distributed heterogeneous architecture, System Openness, Human-System integration,
- Methods and tools to assess software security, trustworthiness, reliability, robustness, and performance, and
- Business process modelling and enterprise architecture frameworks (e.g. DNDAF)

DNDAF is an important and necessary step forward towards establishing the practice of EA within DND/CF. At the time when DNDAF version 1.5 was released (March 2008), it was one of the most advanced military EA frameworks, compared to DoDAF, MODAF, and others (see Section 3.2). However, to fulfill its purpose and to be practically effective, DNDAF has to be more comprehensive and needs to address a number of shortcomings and limitations. To begin with, it has to be clear (and clearly stated) what DNDAF is supposed to be: is it expected to be a framework to provide guidance to DND/CF architects [1, Vol. 1, Sec. 1.2] or is it supposed to represent the DND/CF enterprise [1, Vol. 2, Sec. 2]?

Enterprise architecture is a continuous practice. The current version of DNDAF lacks two important pieces required in adopting a practice of EA within an enterprise: *a)* a set fundamental guiding principles that state how the practice of EA is expected to support the enterprise's business direction and processes, and *b)* a proper methodology (or at least a process) that guides the development and maintenance of architectural artifacts.

In this report, we look at the concept of enterprise architecture, its definition and its purpose, and provide an overview of a number of top EA frameworks with the goal of identifying the main components of a comprehensive EA framework. The EA framework is analyzed in the context of DND enterprise core processes as shown in Figure 4. In light of this study, we then critically look into DNDAF, its purpose and its components, and put forward a set of recommendations for improvement in order to achieve a comprehensive DND/CF EA methodology and framework.

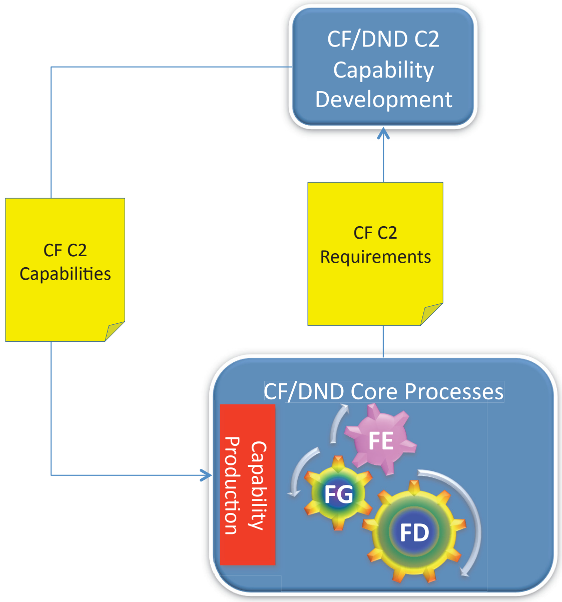


Figure 4: *Context of Integrated C2*

3 Enterprise Architecture

"The reasons one would 'do' architecture for anything *are two: a) complexity and b) change. If the object you are trying to create is simple... that is, if, at a glance, you can see it in its entirety, at the level of definition that is required to create it... and if it is not likely to change appreciably over the period of its existence... then, you don't need architecture. You need a tool (a machete)... some material (some grass)... some time... and then chop down grass... build a grass shack.*

"If, on the other hand, the object you are trying to create [is] so complex that you cannot see it in its entirety at the level of definition required to create it... like an Airbus 380... then forget the machete and the grass... and it doesn't make any difference what tools you have, what material you have and how much time you have, you are not going to be able to create an Airbus 380. In this case, you NEED Architecture..." – John A. Zachman [8]

An *enterprise* is a goal-oriented cooperative (a collection of organizations) comprised of business (processes), people, information, and technology (Figure 5) that transform over time. The stakeholders of an enterprise[7] use the information to make decisions about the enterprise that are in line with its business objectives. In making decisions about an enterprise, stakeholders need to acquire a clear understanding of the enterprise (its processes, components, and their relationships) in order to understand the effects and the risks of their decisions. This process of decision-making is often complicated by the complexity and constant change of the enterprise and its products [1, 10].

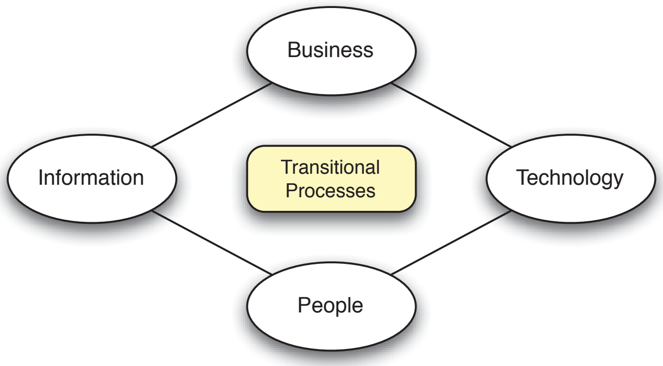


Figure 5: *Four Aspects of an Enterprise*

In order to control complexity and facilitate the process of decision-making and transformation of the enterprise, *Enterprise Architecture* (EA) provides insight and overview of

7. A stakeholder is an individual, team, or organization with interest in, or concerns relative to the enterprise. [10]

the enterprise. According to Defence Terminology Databank (DTB) EA is a "*collection of strategic information that defines a business, the information and technologies necessary to operate the business, and the transitional processes necessary for implementing new technologies in response to the changing needs of the business. It is represented through a set of integrated blueprint.*" [1]

EA is a corporate asset that is both a practice and a tool. The practice of EA is a continuous process of creation, modification, application, and dissemination of information about an enterprise. The architecture identifies the main components of the enterprise, their relationship, and how these function together to achieve business objectives [9]. The architecture can be used to create a roadmap to achieve business objectives by providing a common communication platform to systematically and comprehensively define the current state of the enterprise (*as-is*) and its desired future state (*to-be*). In fact, a good enterprise architecture acts as an analysis tool: it guides the development of new capabilities, addresses the organization's business and information needs, supports the business objectives of reducing costs and improving the operational service, and facilitates the migration from the current state to the future state. [1, 9, 11]

In the practice of EA, it is important to understand what one is trying to achieve and not to start by doing everything at the same time. The best practices suggest that an organization should start with emphasizing on certain areas and step-by-step pick up different objectives and apply a broader emphasis. At any stage of maturity of EA, it is important for the architecture to be *a)* technically feasible and implementable at a reasonable cost within a reasonable timeframe, and *b)* traceable from business requirements to technical implementation [12, 9].

In the following sections, we first clarify our understanding of the terms *methodology* and *framework*, then provide a brief overview of top EA frameworks that serve as a basis for identifying the main elements of an EA framework. Finally, we conclude this section by a discussion of the importance of having a well defined EA methodology in carrying out an EA practice within an enterprise.

3.1 Framework vs. Methodology

The literature does not clearly differentiate between the two terms *methodology* and *framework* in the context of EA, and the terms are at times used interchangeably [13]. [8] In the realm of software development, a *framework* does not dictate what to do but serves as a generic model that guides the development of artifacts [14]. The American Heritage Dictionary, defines a framework as:

> "*A structure for supporting or enclosing something else, especially a skeletal support used as the basis for something being constructed; An external work*

8. The same is true about *methodology* and *process*.

> *platform; a scaffold; A fundamental structure, as for a written work; A set of assumptions, concepts, values, and practices that constitutes a way of viewing reality.*"

On the other hand, a *methodology* is about the practice: the *what* to be done and *how*. The American Heritage Dictionary defines methodology as:

> "*A body of practices, procedures, and rules used by those who work in a discipline or engage in an inquiry; a set of working methods.*"

In the technology industry, methodology is vaguely used to describe the processes and procedures used during the delivery of a technology effort [9]. In many books, *methodology* refers to a few techniques and drawing notations for a few roles. There is another notion of *methodology*, referred to by Alistar Cockburn as the *Big-M Methodology* [6], that is more than just the process of developing or delivering of a product. A Big-M methodology encodes as much as possible about how an organization or an enterprise works, including processes, techniques, and standards. It includes people, roles, skills, teams, tools, techniques, processes, activities, milestones, work products, standards, quality measures, team values [6] and other elements depending on the context and the organization it applies to.

For the rest of this report, we differentiate between *framework* and *methodology* (unless they are part of a name) such that: by *framework* we denote a generic structure, including set of assumptions, principles, concepts, processes and practices, that supports achieving an objective such as development of an enterprise architecture; and by *methodology* we denote an instantiation of a framework tailored to a specific application context in an organization (or enterprise) that encodes how the organization works and repeatedly achieves its objectives.[9]

3.2 EA Frameworks

Enterprise architectures are documented through graphics, tables, and narratives that describe various aspects of the enterprise. In order to provide a coherent and consistent set of descriptions, *Enterprise Architectures Frameworks* are introduced to provide a set of methods, practices and procedures for developing architectural descriptions.

In this section we look at a number of Enterprise Architecture Frameworks[10] in order to better understand the main requirements and components of a comprehensive EA framework.

3.2.1 The Zachman Framework

The concept of Enterprise Architecture essentially started by a paper by J.A. Zachman on "A Framework for Information Systems Architecture" [15] that laid out the challenge

9. As we will see later in this section, FEA is an example of a methodology.
10. We intentionally leave DNDAF out of this section and discuss it in more details in Section 5.

and vision of EA as to manage the complexity of large distributed systems. His view was that in order to achieve business value and agility, the enterprise needs a holistic approach to system architecture explicitly looking at every important issue from every important perspective [15, 13].

The Zachman framework [3] basically tells us how to categorize architectural artifacts and it can better be described as an ontology. It provides a structured way of defining an enterprise (see Figure 6). Zachman originally used the building industry analogy to explain his framework. In that industry there are various "players" (stakeholders), such as the owner or the builder, each demanding a "complete" description of the artifacts. However, what "complete" means is different between these stakeholders. In Zachman framework, architectural artifacts are organized in a two-dimensional matrix. In the vertical dimension are six stakeholders groups (strategists, executive leaders, architects, engineers, technicians, and workers) and the horizontal dimension are six descriptive focus of the artifact (what, how, where, who, when, and why). This decomposition allows for several presentations of the same information to be developed for the same architecture, each focusing on certain aspects of the artifact and targeting the needs of a certain group of stakeholders.

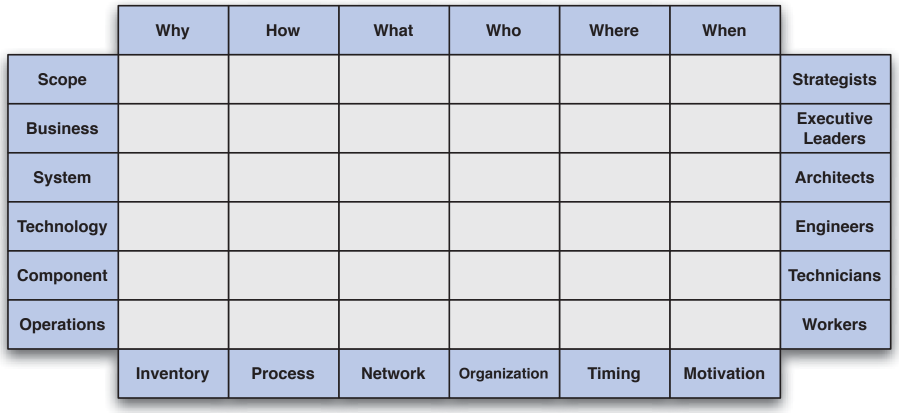


Figure 6: *The Zachman Grid of Architectural Artifacts [3]*

The Zachman framework is not a complete EA framework. It offers a neat structure to document the architecture but it does not provide a process for creating such an architecture, nor does it help with showing the need for change toward a future architecture.

3.2.2 DoDAF: Department of Defense Architecture Framework

The Department of Defense Architecture Framework (DoDAF) is introduced to provide a basic framework for developing and representing architecture descriptions. It provides "*the guidance needed to establish a common vocabulary for architecture development, for the exchange of architecture information, and for facilitating interoperability between*

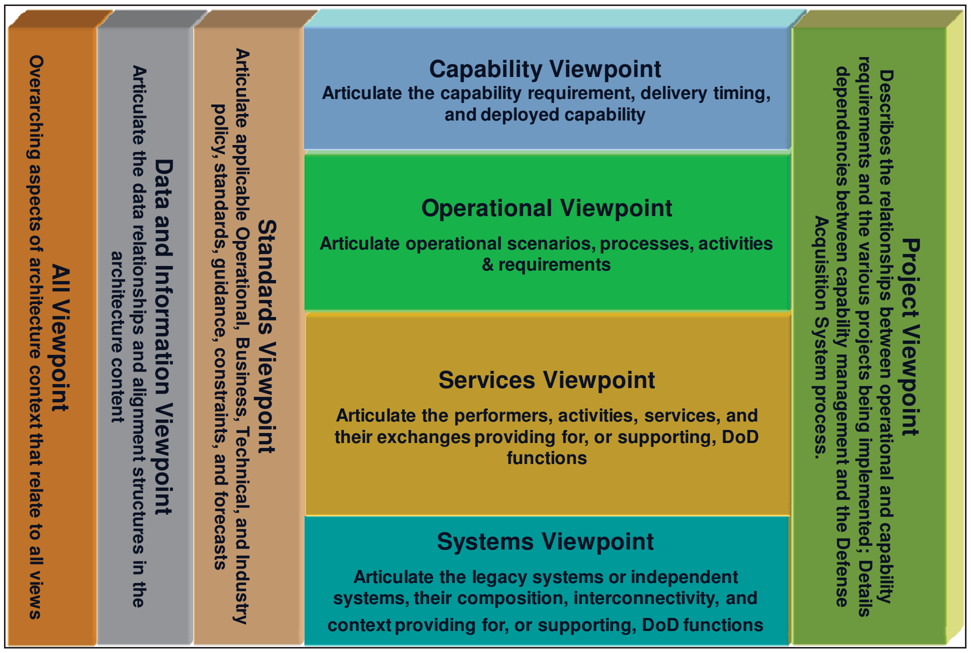

All Viewpoint
Overarching aspects of architecture context that relate to all views
Data and Information Viewpoint
Articulate the data relationships and alignment structures in the architecture content
Standards Viewpoint
Articulate applicable Operational, Business, Technical, and Industry policy, standards, guidance, constraints, and forecasts
Capability Viewpoint
Articulate the capability requirement, delivery timing, and deployed capability
Operational Viewpoint
Articulate operational scenarios, processes, activities & requirements
Services Viewpoint
Articulate the performers, activities, services, and their exchanges providing for, or supporting, DoD functions
Systems Viewpoint
Articulate the legacy systems or independent systems, their composition, interconnectivity, and context providing for, or supporting, DoD functions
Project Viewpoint
Describes the relationships between operational and capability requirements and the various projects being implemented; Details dependencies between capability management and the Defense Acquisition System process.

Figure 7: *DoDAF 2.0 Architectural Viewpoints (from [4, Vol. 1])*

architectural descriptions.” [4] DoDAF is prescribed for the use and development of architectural descriptions in the US Department of Defense.

DoDAF is organized around data, models, and views. At its core, DoDAF 2.0 is a data-centric approach, focusing more on collection, storage and maintenance of data as the necessary ingredient for architecture development that leads to efficient and effective decisions. It defines a DoDAF Meta-model (DM2) (replaces the Core Architecture Data Model in the previous versions) that provides a high-level view of the data maintained in architectural descriptions. To achieve DoDAF conformance, it is required that the data in a described architecture be defined according to the DM2 concepts, associations and attributes. [4]

Models in DoDAF are templates for collecting specific data to form views. *Views* are representation of data in an understandable format that enable the architectural information to be communicated to stakeholders in diverse functional organizations. *Viewpoints* describes the information that should appear in individual views, how to create and use such views, what modeling techniques should be used to express and analyze the information and a rationale for these choices. [4] DoDAF 2.0 defines 8 viewpoints (see Figure 7).

The following lists some of the interesting features of DoDAF 2.0:

Guiding Principles DoDAF comes with a set of high-level guiding principles which provide a general roadmap for success in developing architectural descriptions.

A Methodology-based Approach Although DoDAF does not define a specific methodology, it recognizes the role of methodologies in developing architectures and provides a 6-step architecture development process (see [4, Vol. 1, Sec. 7.1.1]). It recognizes that several methodologies with supporting tools, techniques, and notations exist for developing architectural descriptions and it aims to be open to these methodologies by providing the basic rules, standard entities, and relationships for developing architectural descriptions in a consistent and interoperable fashion.

Towards a Comprehensive Framework In addition to the definition of the data model, the viewpoints, and architecture development process, DoDAF also provides an overview of other aspects of enterprise architecture such as architecture presentation techniques, architecture analysis and guidance on configuration management (CM). [11]

3.2.3 MODAF: Ministry of Defence Architectural Framework

The U.K. Ministry of Defence has developed the Ministry of Defence Architectural Framework (MODAF) [16] with the purpose of supporting Defence planning and change management activities. One of the original purposes of MODAF was to provide rigour and structure in definition of MOD equipment capabilities; however, recently MODAF is more focused on establishing the practice of EA to capture the information about MOD business processes and resources needed to deliver the MOD vision.

MODAF is originally based on DoDAF V1.0. It defines a number of views, similar to DoDAF, to capture architectural information and introduces two additional views of *strategic* and *acquisition* to support MOD business processes, procedures and capabilities required to achieve the desired business outcome. MODAF also provides a MODAF Meta Model (M3) that defines the structure of the underlying architectural information in MODAF views.

Although MODAF does not provide a "MODAF Method" for architecting and creating MODAF views, it offers a 6-step example of such a process [17]. MODAF also offers various examples and supporting documents on the use of MODAF in different aspects of the practice of EA such as user and system requirements definition a nd dependency analysis. [12]

There are a number of commercially available tools that support the use of MODAF.

11. According to [4, Vol. 1] configuration management is required "*to assure an orderly and stable evolution of any Architectural Description and also to ensure that the DoDAF remains current in the face of evolving methods and techniques of Architectural Description creation and management.*"
12. For more details see "Use and examples of MODAF" under `http://www.mod.uk/modaf`.

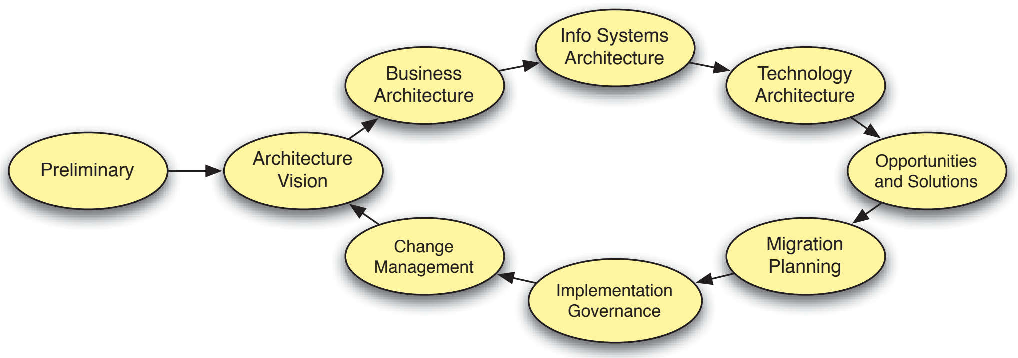


Figure 8: *Phases of TOGAF Architecture Development Method*

3.2.4 TOGAF: The Open Group Architecture Framework

The Open Group Architecture Framework (TOGAF) [18] is an EA framework [13] that is mainly focused on the process of developing architectural artifacts. TOGAF divides an enterprise architecture into four categories of *Business Architecture*, *Application Architecture*, *Data Architecture*, and *Technical Architecture*. The most important part of TOGAF is the Architecture Development Method (ADM), which provides a repeatable process for developing architectures. It includes establishing an organization-specific architecture framework, developing architecture artifacts, transitioning, and guiding the realization of architectures. Figure 8 depicts the phases of ADM.

TOGAF complements ADM with guidelines, techniques, and frameworks that facilitate the application of TOGAF in adopting an EA practice within an organization:

– Through sets of guidelines, TOGAF supports adoption of ADM to different usage scenarios and specific architectures.
– TOGAF offers a description of 10 techniques in developing enterprise architectures such as development of architecture principles, stakeholder management, business scenarios, and risk management [18, Part 3].
– TOGAF defines an Architecture Capability Framework to guide the establishment of an architecture capability within an enterprise through organization structures, processes, roles, responsibilities, and skills. The framework provides a process for establishing architecture capability in the context of ADM in addition to sets of guidelines on architecture governance, establishing an architecture board, defining and using architecture contracts, etc. [18, Part 4]

Since TOGAF is focused on the process of architecture development, it can be used to complement other frameworks that are weak on providing a process and stronger on the ontology aspect, such as the Zachman framework.

13. TOGAF is owned by The Open Group, www.opengroup.org.

3.2.5 NAF: NATO Architecture Framework

The NATO Architecture Framework (NAF) [11] offers the rules, guidance, and product descriptions for developing, presenting and communicating enterprise architectures across NATO. It provides an organizing structure for the information contained in an architecture.

The previous version of NAF, released in 2004, was tightly coupled to DoDAF, while the latest version of NAF (version 3) appears to be more coupled with MODAF. The latest version provides the NAF Metamodel (NMM), an extension of the UK MODAF Metamodel, that defines the relationships between the different components of the framework. The NMM is expected to be implemented in a NATO Architecture Repository, which will be the basic storage for all NATO architecture descriptions.

Similar to its sister frameworks, NAF defines a number of views and sub-views to categorise architecture artifacts: NATO All View (NAV), NATO Capability View (NCV), NATO Programme View (NPV), NATO Operational View (NOV), NATO Systems View (NSV), NATO Service-Oriented View (NSOV) and NATO Technical View (NTV). The Capability, Service-Oriented and Programme views are new to NAF version 3.0. The NATO Capability View captures essential elements of NATO's strategic vision and concepts and NATO's capability planning process. The purpose of the Programme Views is to describe the relationships between NATO capability requirements and the various programs and projects being implemented. The Service-Oriented View supports development of SOA-based architectures [14] and complements the NATO Operational View by providing a description of services needed to support the operational domain. [11]

NAF version 3.0 also offers guidelines for management of NAF architectures, [15] in addition to supporting Spectrum and Bandwidth Management for utilizing limited resources in the most efficient way.

3.2.6 FEA: Federal Enterprise Architecture

The US Federal Enterprise Architecture (FEA) program focuses on building a comprehensive business-driven bluepring of the entire Federal government [20]. The purpose of FEA is to provide a sustainable mechanism for identifying, developing, and documenting architecture descriptions built on common business areas and designs that cross organizational boundaries [21]. The main component of FEA is a set of five reference models that are designed to "*facilitate cross-agency analysis and the identification of duplicative investments, gaps and opportunities for collaboration within and across agencies*." [22] The five reference models (Performance, Business, Service Component, Technical, and Data)

14. Service-Oriented Architecture (SOA) is an architectural style and a set of design principles that supports design and development of systems in terms of services and service-based development. [19]

15. See [11, Ch. 6]

collectively form a framework for a coherent and consistent description of the important elements of FEA.

FEA introduced some innovative ideas for development of enterprise architectures. According to FEA, an enterprise is built of *segments* where every segments is a major line-of-business functionality. The segment approach of FEA allows different parts of the overall enterprise to be developed individually and later be integrated into a larger enterprise architecture (see Figure 9). A single Federal agency has *core mission area* segments that are central to the purpose of the agency (such as 'health' for the Health and Human Services agency) and *business services* segments that are common or shared services supporting the core mission areas (such as 'financial management'). FEA also introduced the notion of *enterprise service* as a shared IT service that spans political boundaries (e.g., records management) [5, 13]. FEA also defines an Enterprise Architecture Assessment Framework (EAAF) [23] to support agencies in measuring their success in using enterprise architec-ture. The framework defines 12 assessment criteria grouped into three capability areas of Completion, Use, and Results, to evaluate the performance and effectiveness of agency enterprise architecture programs.

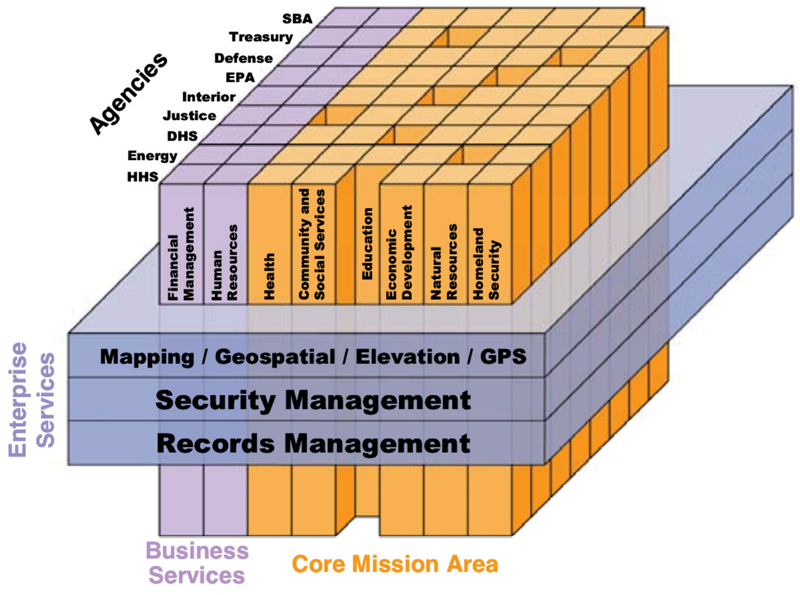


Figure 9: *FEA Segments and Services* (courtesy of [5])

The original version of FEA as a framework was first released in 1999 under the name of Federal Enterprise Architecture Framework (FEAF) (version 1.1) for developing an EA within any Federal Agency or for a system that transcends multiple inter-agency boundaries [12]. The word "framework" was later removed from the name in 2002 as FEA focused on the development of a US Government enterprise architecture methodology. However, we believe that FEA can still be adopted by other organizations and can serve as a general framework and methodology for developing enterprise architectures [13]. FEA in its current form offers a complete taxonomy, a process to develop architectures, a transitional process for migrating from pre-EA to a post-EA model, and a set of reference models that provide standard terms and definitions for describing different aspects of the enterprise architecture [5, 13].

The original FEA Framework has been one of the three architectural frameworks sponsored by the US Federal Government. The other two are DoDAF (see Section 3.2.2) and the Treasury Enterprise Architecture Framework (TEAF) [24], which is developed by the US Department of Treasury based on the Zachman framework. TEAF version 1.0 is published in July 2000 to provide guidance to Treasury bureaus concerning the development and evolution of information systems architecture and serve as a template for the development of a Treasury EA [24, 12].

The link between FEA, TEAF, and DoDAF is not very clear. According to [4], the DoDAF leverages the FEA construct and core principles while ensuring that upward reporting and review can be accomplished against the FEA. The TEAF documentation aligns TEAF architectural views and supporting work products with the original FEAF views and models [24]. However, TEAF is not updated in the past 10 years since its original version.

3.3 Components of an EA Framework

Based on an analytical review of the EA frameworks we presented in this section and taking into account the purpose and the promise of Enterprise Architecture, we identify the main components of a generic comprehensive EA framework as follows:

1. ***Architecture Taxonomy*** as a method for organizing (classifying) architectural artifacts. For example, the Zachman framework is strong on providing an enterprise architecture taxonomy. In general, taxonomy is defined as "*a collection of controlled vocabulary terms organized into a hierarchical structure.*" and the term is sometimes used in specific contexts (e.g., "DND/CF standards taxonomy" or "planning taxonomy") and should not be confused with *architecture taxonomy*.
2. ***Reference Model Guidance*** that facilitates building a relevant set of reference models. Reference models provide standard terms and definitions for describing different aspects of EA and facilitate collaboration and sharing between organizations. Reference models are about establishing a common language. For example, TOGAF, as a framework, supports the development of reference models, and the FEA, more as an EA methodology, is basically composed of a set of five reference models (see Section 3.2.6).
3. ***Architecture Development Process*** is an essential asset in adopting an EA practice. A good architectural process guides the enterprise through a step-by-step process in developing architectural artifacts. The importance of an EA process and methodology is discussed later in this section.

4. ***Practice Guidance*** in the form of documentation, examples, and best practices, facilitates the use of the framework to absorb the concepts and adopt the process of EA in the organization.
5. ***Architecture Analysis and Maturity Guidance*** is important in facilitating the assessment of effectiveness and success of using enterprise architecture in organizations within the enterprise. A good example is the Enterprise Architecture Assessment Framework (EAAF) provided by FEA.

3.4 EA Methodology

"*Architecture is a verb, not a noun.*" This symbolic and grammatically incorrect statement points out the fact that static architectural artifacts that do not evolve in time are useless and it is the ongoing process of creating, maintaining, and leveraging an enterprise architecture that makes the architecture a valuable asset and gives it vitality. Effective EA programs are process-driven and methodology-based. [13, 25] An *EA process* prescribes what must be done when, and how the steps are linked to one another [6, 26]. An established EA process is necessary in achieving a coherent set of architectural artifacts.

An EA Framework, as we discussed earlier, defines a generic structure that guides the practice of EA within an enterprise. The framework defines the patterns, abstract elements, process templates, and the guidelines. An effective and practical use of an EA framework within an enterprise requires the development of an EA methodology (a Big-M methodology) tailored for the enterprise. We believe that a Big-M methodology in the context of Enterprise Architecture, should include additional EA-specific elements such as the objectives of the enterprise, EA principles, architectural modeling techniques, EA processes, guidelines, taxonomy, and reference models.

Figure 10 extends the Big-M methodology diagram of [6] to illustrate the elements of a Big-M methodology in the context of EA. Following an EA methodology, different teams, organizations, or agencies within an enterprise fill different roles that engage in activities defined by EA processes. They use techniques to develop architectural artifacts with respect to an architectural taxonomy. The artifacts are based on certain standards and meet required quality criteria. The tools help enforce the standards, and the reference models facilitate producing a coherent set of architectural products across the enterprise. All these elements operate with respect to a set of enterprise values, EA principles and guidelines that are aligned with the objectives of the enterprise.

EA principles

EA Principles are an important element in the practice of EA within an enterprise. They provide fundamental statements on how an organization will utilize its resources in adopting an EA practice and how the practice will support the directions and the operational

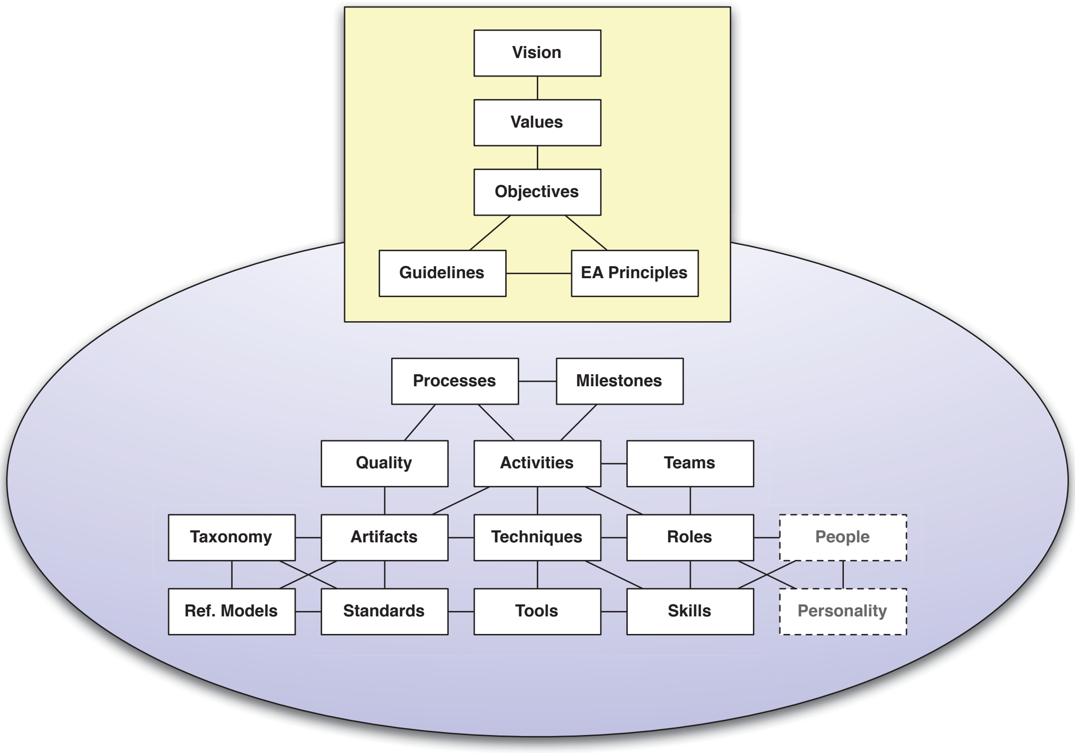


Figure 10: *Elements of a Big-M Methodology for EA (based on [6])*

processes of the enterprise. The principles guide the development of the architecture and its continued evolution taking into account the needs and objectives of the enterprise [27]. Although the actual principles differ from enterprise to enterprise, they can be developed based on generic EA principles, such as the EA principles of TOGAF [18].

Engineering and Formal Methods

At the core of EA is the process of developing architectural artifacts with the purpose of providing a clear documentation of the enterprise information that would act as a platform for understanding, analysis, and communication of this information in order to facilitate the processes of decision making and change management. As such, the resulting artifacts has to provide precise—not necessarily detailed, unambiguous, and correct models of the enterprise. Development of such models, in a given context, requires the use of *systems engineering methods* that guide capturing the current model, requirements and design of the future architecture, and *formal approaches* that bring rigour and precision serving as a basis for validation and verification of the models. We discuss these aspects in more detail in Section 6.3.

Tools and Standards

Standards are an integral part of an enterprise architecture [28]. They enable interoperability of systems, technology independence, and above all, sharing and exchange of information, which is one of the most valuable assets of an enterprise. Tools facilitate the proper use of standards and provide machine assistance in development and analysis of architectural artifacts and validation of design ideas. EA frameworks can recommend the use of standards or identify a certain set of standards to be used. An EA methodology, designed for a certain enterprise, can be more explicit and indicate the set of standards that must be used within the enterprise.

4 Models and Formal Methods

With the increasing complexity of computer-based systems, efficient design and development of high-quality computational systems that faithfully conform to their requirements are extremely challenging, and the costs of design flaws and system failures are high. Proper understanding of the requirements, precisely documenting design decisions, and effectively communicating such decisions with the domain experts as early as possible play important roles in the design of complex systems. These challenges call for adoption of proper engineering methods and tools and have motivated the use of *formal methods* in software engineering.

Abstraction and formalization provide effective instruments for establishing critical system requirements by precisely modeling systems prior to construction so that one can analyze and reason about specification and design choices and better understand their implications [29]. In addition, machine assistance plays an increasingly important role in making design and development of complex systems feasible. Abstract executable specifications serve as a basis for design exploration and experimental validation through simulation and testing. Model-checking tools based on formal verification techniques help with proving critical properties of systems and ensuring "correctness" before deployment.

The rest of this section briefly discusses the concept of formal modeling and explores various formal languages and techniques for modeling software-intensive computer-based systems. We mainly focus on state-based formal modeling techniques, and introduce the Abstract State Machines method as an agile formal technique suitable for high-level modeling and documentation of systems.

4.1 Formal Modeling

A *model* of a system is an abstract representation of that system so that one can view, manipulate, and reason about it [30]. Such a representation also helps in understanding the complexity that is inherent in the system under study. We build models to increase productivity because it is often cheaper to explore and to manipulate the model than the real system. A "good" model omits unnecessary information but accurately reflects the essential aspects of the subject matter in order to help the viewer to clearly see the subject, and focus on those essential aspects. A model that is easily understandable can also serve as a means of communication by clearly illustrating the subject and its main concepts and ideas.

There are many modeling languages available to express computational models, each one focusing on certain aspects or targeting certain types of systems. A popular example of a widely used modeling language is the *Unified Modeling Language*, or UML[16] for short.

16. http://www.uml.org

UML is a visual (graphical) language and is one of the most common industrial modeling languages in the area of software engineering. However, UML is an informal language [17] as its semantics is not formally defined. [18]

In this section, we focus on *formal modeling* approaches. According to Daniel M. Berry [29], a formal method is any attempt to use mathematics in the development of a software-intensive computer-based system in order to improve the quality of the resulting system. We define a *formal modeling language* as a modeling language that has a formally defined (read "mathematically defined") syntax and formally defined semantics for that syntax. There are many formal languages and notations for modeling and specification [19] of systems, such as: the *Vienna Development Method (VDM)* [34], one of the longest-established formal methods for the functional modeling of computer-based systems; the family of *Algebraic Specification* languages, currently subsumed under the *Common Algebraic Specification Language*, or *CASL* [35], which are all based on first-order logic with induction, viewing states of systems as first-order many-sorted structures; the family of *process calculi* or *process algebras* languages and approaches to formal modeling of concurrent systems (such as π-*calculus* [36] or *Communicating Sequential Processes (CSP)* [37]), supporting high-level description of interactions between a collection of independent processes; *Specification and Description Language (SDL)* [32], a standard formal language [38] for specification and description of reactive and distributed systems, which provides both graphical and textual representations; the *Petri nets* [20] [39] graphical language for description of distributed systems in the form of bipartite graphs [21]; the *B method* [40], an abstract machine modeling approach mostly used in the development of software with a rich set of com-mercially available tools for specification, design, proof, and code generation; the *Z notation* [41], a formal specification language for modeling computing systems and formulation of proofs about the intended program behavior based on axiomatic set theory, lambda calculus, and first-order predicate logic; the *Alloy* specification language [42], a light-weight formal specification language (inspired by the Z notation) together with a tool designed for providing fully automatic analysis of software specifications; and last but not least, the *Abstract State Machines (ASM)* method [43], a versatile semantic framework for compu-tational modeling of virtually all kinds of discrete dynamic systems, combining common abstraction principles from computational logic and discrete mathematics.

Each formal modeling approach focuses on a certain view towards systems, being declar-

17. Some people claim that UML is a semi-formal modeling language since its (abstract) syntax is precisely defined [31].

18. There have been attempts to formally define the semantics of UML (see [31] for example).

19. There is a slight difference between a 'model' and a 'specification' of a system. Strictly speaking, a specification of a system tends to view the system as a black box, focusing on the behavior of the system as a whole and its interface to its environment [32, 33]; i.e., focusing on *what* the system does. A model of a system, on the other hand, can include both a specification and a description of the system; i.e., describing *what* the system does and *how* it does it.

20. http://www.petrinets.info

21. A *bipartite* graph is a graph that does not contain any odd-length cycle.

ative, functional, or operational. Some languages are particularly good in modeling data structures and the state of systems but are less supportive of the operational aspects. Some are low-level, staying closer to code and the final implementation of systems and some are more formal and stay on the mathematical level. Among these formal methods, abstract state machines, while being primarily operational in nature, provide a good compromise between declarative, functional, and operational views towards modeling distributed discrete dynamic systems. The emphasis on *freedom of abstraction* [43] in ASMs leads to intuitive yet accurate descriptions of systems which, thanks to the pseudo-code style of its language, are easily understandable by both domain experts and system designers. Since ASMs are in principle executable, the resulting models are validatable and possibly falsifiable by experiment. Finally, the well-defined notion of *step-wise refinement* in ASMs [44] bridges the gap between the abstract model and its final implementation.

4.2 State-based Formal Methods

Abstract state machines, among many others, fall into the category of state-based formal methods that view the states of a system in terms of mathematical structures. This view towards modeling of systems is particularly helpful in capturing models in DNDAF Operational Views and System Views that are primarily focused on viewing operations and systems as a sequence of transitions between states (or actions). In this category, one can point to methods such as Alloy, B, CASL, and the Z notation as four of the most popular approaches that share many similar concepts such as offering abstract notations, supporting declarative descriptions of system behavior in terms of constraints, and relying on tool support for analysis of specifications.

The Common Algebraic Specification Language, or CASL, is a general purpose specification language based on first-order logic with induction. Different extensions of the language have been designed for specification and development of various kinds of systems such as reactive or concurrent [35]. The language is supported by a number of tools for checking the correctness of specifications, proving certain properties of models, and managing the formal software development process. Currently, The *Heterogeneous Tool Set*, [22] or *Hets* for short, seem to be the mainstream central toolset for CASL. Its software is free, with a license similar to the GNU General Public License [45], and offers parsing, analysis, and prover integration for CASL and its extensions.

The Z notation [41] is a formal specification language designed with proofs in mind; it is based on axiomatic set theory, lambda calculus, and first-order predicate logic. There are quite a number of tools available for Z, most of them focused on theorem proving such as *ProofPower* [23], a suite of tools supporting specification and proof in the Z notation,

22. `http://www.dfki.de/sks/hets`
23. Registered trademark of Lemma 1 Ltd., `http://www.lemma-one.com/ProofPower`

Z/Eves [46], a front-end for the *Eves* theorem prover, and *HOL-Z*,[24] a proof environment for Z specifications based on the generic theorem prover Isabelle/HOL[25]. A free and open source animator for Z specifications, called *Jaza*,[26] is also available for evaluation, testing and (for some specifications) also execution of Z specifications.

Inspired by Z, the Alloy specification language [42] is designed as a light-weight formal specification language with the goal of providing fully automatic analysis of software specifications. However, unlike Z, Alloy's data structures are strictly first order. Alloy comes with *AlloyAnalyzer*, a model-checker that checks certain properties of specifications by exploring the states of the system and looking for execution instances that satisfy the properties (*simulation*) or by finding counterexamples that violate them (*checking*).

Among these approaches, the B method [40] has a more operational flavor and is the most similar approach to ASMs. The B method is essentially an abstract machine notation with a well-defined notion of refinement that facilitates transformation of abstract models into implementation. B comes with a rich set of both commercial and open-source tools. Commercial tools such as *Atelier-B*[27] and the *B-Toolkit*[28] are available providing syntax analysis, theorem proving, and automatic refinement of B specifications down to implementation. A single-user free version of Atelier-B, called *B4Free*, is also available for the academic environment. There are also model checkers available for B; for example, *ProB*[29] offers fully automatic animation of many B specifications and can be used to systematically check a specification for errors.

4.3 Abstract State Machines

Abstract state machines [43, 47] are well known for their versatility in computational and mathematical modeling of complex distributed systems with an orientation toward practical applications. The ASM framework offers a universal model of computation and serves as an effective instrument for analyzing and reasoning about complex semantic properties of discrete dynamic systems. For almost two decades, abstract state machines have been studied, practiced, and applied in modeling and specification of systems to bridge the gap between formal and pragmatic approaches. Combining common abstraction principles from computational logic, discrete mathematics, and the concept of transition systems, ASMs have become a well-known method, and have assumed a major role in providing a solid and flexible mathematical framework for specification and modeling of virtually all kinds of discrete dynamic systems.

24. `http://www.brucker.ch/projects/hol-z`
25. `http://www.cl.cam.ac.uk/research/hvg/Isabelle`
26. `http://www.cs.waikato.ac.nz/~marku/jaza/`
27. `http://www.atelierb.eu`
28. `http://www.b-core.com/btoolkit.html`
29. `http://users.ecs.soton.ac.uk/mal/systems/prob.html`

Egon Börger [43] further developed ASMs into a *systems engineering* method that guides the development of software and embedded hardware-software systems from requirements capture to their implementation.

Widely recognized applications of ASMs include semantic foundations of industrial system design languages like the ITU-T standard for SDL [48], the IEEE language VHDL [49] and its successor SystemC [50], programming languages like JAVA [51], C# [52] and Prolog [53], Web service description languages [54], communication architectures[55, 56], embedded control systems [57, 58, 59], et cetera. [30]

4.3.1 ASM Systems Engineering Method

The ASM method for systems design and analysis builds on the concept of abstract state machines and brings together two tasks of requirement capture and system design. The goal is to improve industrial system design and development by integrating precise high-level, problem-domain-oriented modeling into the development cycle, and systematically linking abstract models down to executable code.

The method consists of three essential elements: *a)* capturing the requirements into a precise yet abstract operational model, called a *ground model* ASM, *b)* systematic and incremental refinement of the ground model down to the implementation, and *c)* experimental model validation through simulation or testing at each level of abstraction. This process emphasizes freedom of abstraction as a guiding principle, meaning that original ideas behind the design of a system can be expressed in a direct and intuitive way so as to enable system designers to stress on the essential aspects of design rather than encoding insignificant details. The operational characteristics of ground models combined with the freedom of abstraction supports design space exploration and experimental validation of design ideas at early stages of development before the expensive task of coding begins.

Starting from a ground model and applying the process of step-wise refinement, a hierarchy of intermediate models can be created that are systematically linked down to the implementation (see Figure 11). At each step, the refined model can be validated and verified to be a correct implementation of the abstract model. The resulting hierarchy serves as a design documentation and allows one to trace requirements down to the implementation.

4.3.2 The CoreASM Modeling Suite

With machine assistance playing an ever increasing role in making systems design more feasible, the ASM method [43] takes advantage of the fact that ASM models are executable in principle [60]. Abstract executable specifications serve as a basis for exploration design decisions and validation by means of simulation, testing and model checking. Over the

30. See also the ASM website at `www.asmcenter.org` and the overview in [43].

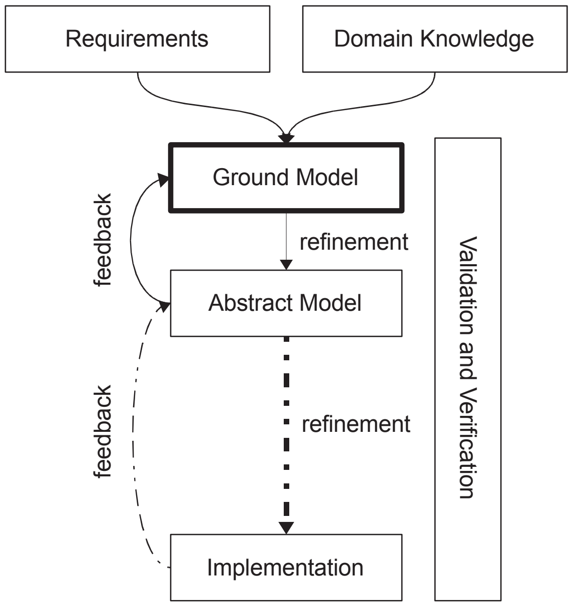


Figure 11: *The ASM method of systems engineering*

years, a variety of ASM tools and executable languages have been developed, each coming with their own strengths and limitations [61]. All of these languages build on predefined type concepts rather than the untyped language underlying the theoretical ASM model. Most of them do not provide a run-time system that supports the execution of distributed ASM models [61, 62]. Conversely, CoreASM preserves the very idea of ASM modeling—building accurate abstract models (*ground models* [60]) at a level of abstraction directly reflecting the application view—and facilitates rapid prototyping of such models for requirements elicitation, design space exploration, experimental validation and conformance testing.

CoreASM [62, 61] is an environment for writing, running, and validation of executable specifications according to the ASM method. It has been designed with an extensible plugin-based architecture that offers a great deal of flexibility for customizing its language definition and execution engine depending on the application context. The CoreASM environ-ment consists of a platform-independent engine for executing CoreASM specifications and a GUI for interactive visualization and control of simulation runs. The engine comes with a sophisticated interface enabling future development and integration of complementary tools, e.g., for symbolic model checking and automated test generation [62].

The latest version of the CoreASM modeling framework offers *a)* an extensible specification language, *b)* an extensible multi-agent ASM simulation engine, faithful to the mathematical definition of ASMs, that animates CoreASM specifications in addition to providing other services, such as parsing, through a rich API, *c)* a library of optional plugins offering additional features and language constructs not originally part of the ASM dialect, and *d)* an Eclipse front-end together with a command line user interface.

Over several years, CoreASM has been put to the test in a range of applications in the private and public sectors, spanning computational criminology, coastal surveillance, situation analysis, decision support systems, and Web services. The diversity of application fields has been invaluable to examine the practicability of using CoreASM for requirements analysis, design specification and rapid prototyping of abstract executable models [63, 64, 65].

CoreASM has been recognized by the ASM community and has been used by various research groups in Europe, Asia, and North America [66, 67, 68, 69, 70, 71].[31] Currently, the CoreASM project is publicly available on Sourceforge.net,[32] one of the most popular repositories of open source software offering online resources for open source software development and content creation. Since its first beta release in September 2006, CoreASM has gone through a number of revisions and its latest version (under testing at the time of writing this document) offers substantial improvements over its previous version in terms of both features and performance.

31. To name a few, CoreASM has been applied in a number of research projects at SAP Research in Germany [72, 73], the Computer Science Department of the University of Pisa in Italy, the Embedded Software Laboratory at the RWTH Aachen University in Germany, the Open Systems Development Group at the University of Agder in Norway, and the Department of Computer Science and Engineering at the Anna University in India.

32. http://www.sourceforge.net

5 The DND/CF Architecture Framework

The Canadian Defence Planning Guide (DPG) 2001 called for the use of enterprise architecture within the Department of National Defence and the Canadian Forces (DND/CF). DND Architecture Framework (DNDAF) [74, 75] provides details of the DND Enterprise Architecture program. It defines the framework, the methodologies, and the tools. The idea is to have all architectures in DND/CF follow the framework and methodologies (see Section 6) defined in DNDAF to ensure interoperability and integration into an overall EA repository.

The DNDAF facilitates representing an enterprise (in particular DND/CF) in terms of its components, their role, their relationship to each other and to the environment, as well as the rules and constraints governing them. Within the DNDAF, the DND/CF architectures are described in terms of views and sub-views. [1, Vol. 2, P. 9]

5.1 Purpose

According to [1], the main purpose of Enterprise Architecture is to "*inform, guide, and constrain the decisions made by an enterprise, especially those related to investments.*" The role of EA in DND/CF and the benefits of establishing an enterprise architecture practice is very well discussed in [1, Vol. 1, Sec. 3.2]. Within DND/CF, the role of enterprise architecture is to contribute to improving the effectiveness and efficiency of DND/CF by facilitating change management and reducing risk through providing products that explicitly show the implications of proposed changes to the enterprise. In summary, EA helps in

- reducing the complexity of requirements and facilitating requirement traceability,
- options analysis in order to reduce risks and increase efficiency in decision making and planning,
- linking requirements to capabilities and measuring improvements for the purpose of aligning and prioritizing investments.

"*The purpose of the DNDAF is to provide guidance to all DND/CF organizations in initiating, developing, using, and maintaining architectural products during the life cycle of their programmes, projects, initiatives, and business transformation efforts.*" [1, Vol. 2, Sec. 1.2] The goal of DNDAF is to support decentralized creation of architecture products (documentation) with common characteristics that enable understanding, comparison, and integration between architectures. The goal facilitates organization, analysis, and communication of information regarding the enterprise (DND/CF) by grouping this information into a set of views. The following section provides an overview of DNDAF views and sub-views.

5.2 Views

The DNDAF uses six views to represent information about the enterprise: Common View (CV), Operational View (OV), System View (SV), Technical View (TV), Information View (IV), and Security View (SV). [75] Each view represents a group of sub-views that are graphical, textual, or tabular items documenting different aspects of the architecture (see Figure 12). Sub-views are interrelated within and across views.

– The *Common View* captures the over-arching aspects of the architecture. It defines the scope, context, definitions and taxonomies of the architecture.
– The *Operational View* describes the tasks, activities, business and operational elements, and the information flow within the architecture.
– The *System View* focuses on the description of systems (components or systems of systems) and their interconnections providing or supporting DND/CF functions.
– The *Technical View* describes the minimal set of rules and protocols governing the implementation, interaction, and interdependence of system parts.
– The *Information View* defines the overall pattern or structure that is imposed on information design. This view exists in all other views.
– The *Security View* deals with security and information assurance architecture of DND. This view has two main streams: the first stream is integrated in the other views (Common, Operation, System and Technology) that have security related attributes (e.g., SV-2) and the second stream is represented by specific sub-views under Security View that complement the security functions of other views.

5.3 Data Model

DNDAF views and sub-views represent different aspect of the enterprise and, given that DNDAF supports a decentralized creation of these views, there needs to be a mechanism ensuring that all these views are consistent and can be assembled in a collective set. In order to achieve this consistency, DNDAF defines the basic common building blocks upon which these views and sub-views are constructed in the DND/CF Architecture Data Model (DADM). The DADM defines a standard set of architecture data entities and their relationships, and provides a common approach for capturing common data requirements and portraying the structure of architecture information. [1] DADM is a logical data model that shows how the information is organized in DNDAF.

The DADM will be extended iteratively as DNDAF evolves in order to capture its new requirements.

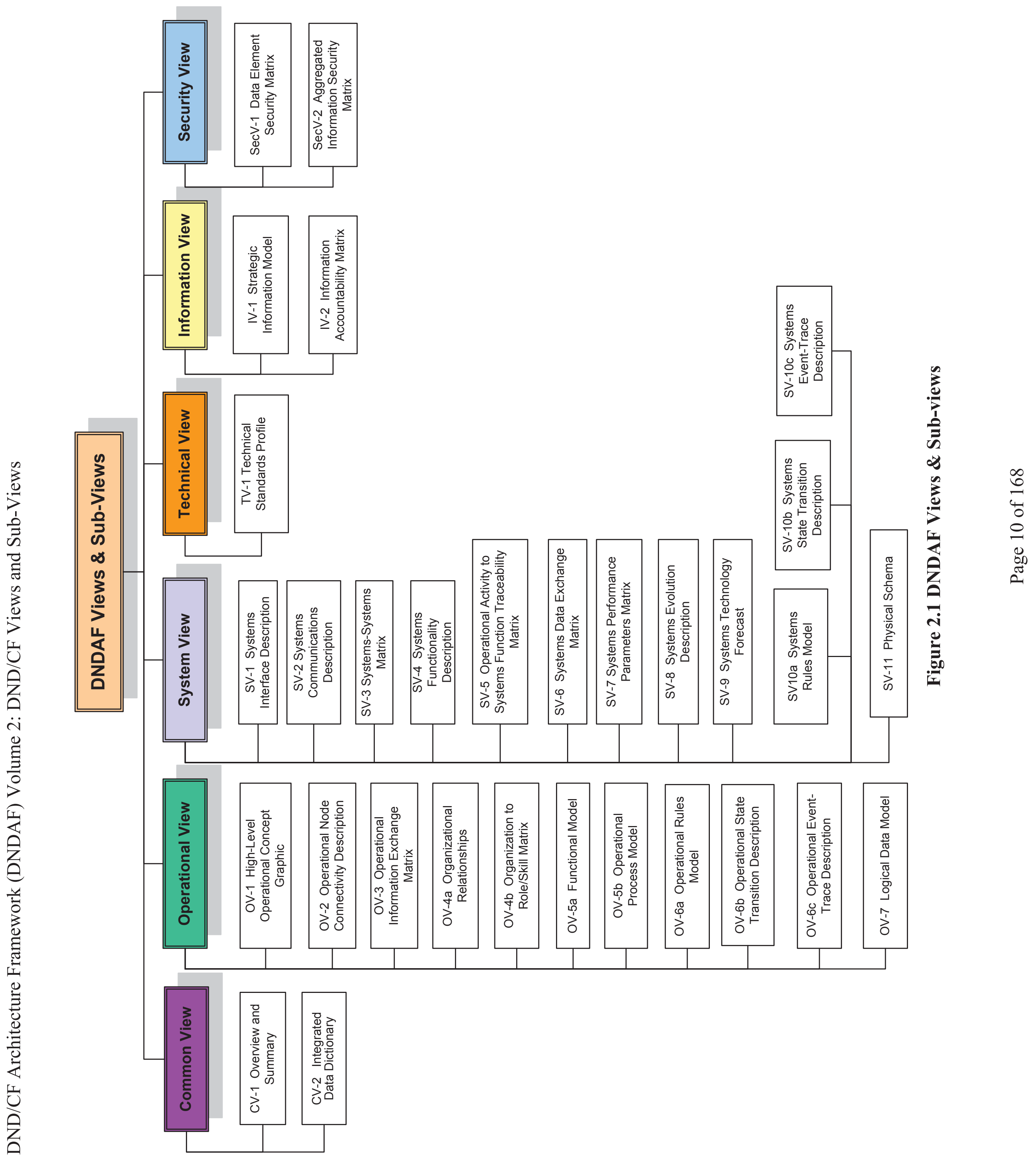

DND/CF Architecture Framework (DNDAF) Volume 2: DND/CF Views and Sub-Views
DNDAF Views & Sub-Views
Common View
CV-1 Overview and Summary
CV-2 Integrated Data Dictionary
Operational View
OV-1 High-Level Operational Concept Graphic
OV-2 Operational Node Connectivity Description
OV-3 Operational Information Exchange Matrix
OV-4a Organizational Relationships
OV-4b Organization to Role/Skill Matrix
OV-5a Functional Model
OV-5b Operational Process Model
OV-6a Operational Rules Model
OV-6b Operational State Transition Description
OV-6c Operational Event-Trace Description
OV-7 Logical Data Model
System View
SV-1 Systems Interface Description
SV-2 Systems Communications Description
SV-3 Systems-Systems Matrix
SV-4 Systems Functionality Description
SV-5 Operational Activity to Systems Function Traceability Matrix
SV-6 Systems Data Exchange Matrix
SV-7 Systems Performance Parameters Matrix
SV-8 Systems Evolution Description
SV-9 Systems Technology Forecast
SV10a Systems Rules Model
SV-10b Systems State Transition Description
SV-10c Systems Event-Trace Description
SV-11 Physical Schema
Technical View
TV-1 Technical Standards Profile
Information View
IV-1 Strategic Information Model
IV-2 Information Accountability Matrix
Security View
SecV-1 Data Element Security Matrix
SecV-2 Aggregated Information Security Matrix
Figure 2.1 DNDAF Views & Sub-views
Page 10 of 168

Figure 12: *DNDAF Views and Sub-views (from [1])*

5.4 Reference Models

In order to facilitate the creation of coherent architecture views, DEA together with the DND/CF user communities are developing a series of DND/CF Enterprise Architecture Reference Models to provide lists of standard reference values to be used in development of sub-views.

In the context of Enterprise Architecture, Reference Models (RM) serve as the architecture taxonomy, abstractly describing and classifying the basic concepts and entities and their relationship to one another. Currently, the DND/CF Technical Standards List (DTSL) is considered a DND/CF RM of current and emerging IM/IT standards. DTSL represents the DND/CF standards taxonomy and is the official DND/CF repository from which the applicable standards in Technical View 1 (TV-1) are chosen.[33]

33. Unfortunately, as of writing this document, there is no publicly accessible reference to the technical standards list (DTSL) in DNDAF documentation.

6 Towards a DND/CF EA Methodology

In this section we critically look at the DND/CF Architecture Framework (as defined in [1]) and, with respect to our discussions and reviews of EA frameworks in Section 3, endeavor to identify main limitations and shortcomings of the framework.

6.1 Is DNDAF an EA Methodology?

We start by contrasting what DNDAF currently is and what it is expected to be. DNDAF is part of an effort to establish enterprise architecture as a practice in DND/CF. The DNDAF documentation (officially referred to as the 'DNDAF') intends to provide EA information and guidance to architects across DND/CF. It is supposed to provide "*explicit details of the DND EAP [Enterprise Architecture Programme] including: the framework, the methodologies and tools used to build, analyze, modify, manage and maintain the DND/CF Enterprise Architecture.*" [1, Vol. 1, Sec. 1.2]

In addition, Volume 2 of the same document states that the DNDAF "*represents the enterprise (DND/CF) in terms of its constituent parts, what those parts do, how they relate to each other and to the environment, as well as the rules and constraints governing them.*"

These two statements describe two different entities. According to the first description in Volume 1, DNDAF is supposed to be a comprehensive framework and an enterprise-wide methodology for the practice of EA within DND/CF. However, the second description in Volume 2, illustrates DNDAF as an already developed *architecture* that documents the DND/CF enterprise as a whole.

It is interesting to observe that DNDAF as its current state, is neither of the above. The documentation only provides the definition of the Data Model, the views and sub-views together with a guide on how each sub-view has to be created. Despite the above statements, DNDAF as it is currently defined does not represent the enterprise, nor does it offer a clear methodology or set of principles or guidelines to build, analyze, and maintain the DND/CF Enterprise Architecture. DNDAF only offers a light-weight framework within which different aspects of an enterprise architecture can be documented. It neither defines the enterprise itself, nor the processes of creating such documentations (the architecture).

In our opinion, the lack of a clear understanding on what DNDAF is and what it is expected to be defies the main purpose of DNDAF, which is to provide guidance to all DND/CF organizations in initiating, developing, using, and maintaining architectural products and to support the decentralized creation of such products with common characteristics.

6.2 What Is Needed?

Considering the goal of EAP and the purpose of DNDAF mentioned in the previous section, we believe that a limited framework such as the current version of DNDAF is not sufficient to establish the use of EA within DND/CF. An EA framework is useful but not sufficient to create a successful and sustainable EA practice [25]. Effective EA programs are process-driven and methodology based. An EA framework, in general, provides guidance for adopting an EA practice within any enterprise. However, effectively establishing such a practice in a specific enterprise such as DND/CF requires a detailed and comprehensive framework together with an enterprise-wide methodology that clearly defines and provides specific EA processes, techniques, reference models, standards, and guidance for the assessment of architectural efforts. The methodology should be designed to take into account the specific requirements of DND/CF, its established processes, organizations, capabilities, and objectives, and to offer flexibility for organizations to customize the methodology according to their programs, projects, and processes. Such a comprehensive framework and methodology is required to provide practical guidance to different organizations within the enterprise in developing, analyzing, and maintaining their segment of the architecture in the overall DND/CF enterprise architecture.

What DNDAF mainly provides is a well-defined architectural taxonomy through a set of views and sub-views that categorize architectural artifacts to be produced. It is our opinion that what is needed is a comprehensive methodology (see Figure 10) for the practice of EA within DND/CF that provides:

1. a set of EA principles designed with the specific needs of DND/CF in mind,
2. a proper set of reference models,
3. an architecture development process,
4. systems engineering and formal techniques,
5. architecture analysis methods and maturity guidance,
6. standards and guidance on tools,
7. practice guidance.

In the following we further discuss some of these elements (1, 2, 3, 6, and 7) in the context of a DND/CF EA practice. The role of systems engineering and formal methods as well as architecture analysis and maturity guidance in DNDAF require more elaborate discussions. These are presented in separate sections.

6.2.1 EA Principles

We discussed the importance of defining EA principles in Section 3.4, which is a set of guiding principles providing fundamental statements of how DND/CF enterprise architecture will support DND/CF directions and operational processes. Best practices suggest that every principle should have to be documented with a *name*, a concise *statement*, a *motivation* (or *rationale*) describing why the principle is in place, and a set of *implications* that describes its impact on the enterprise. More discussion on enterprise architecture principles and various examples can be found in [28, 18, 27].

6.2.2 Reference Models

Reference models provide standard terms and definitions for the enterprise architecture and facilitate collaboration across the enterprise. Currently, DTSL (see Section 5.4) is the DND/CF technical reference model of current and emerging IM/IT standards. However, additional reference models for describing other EA areas (such as business functions and components) would be required to support coherent and consistent artchitectural products across DND/CF. The five reference models of FEA can serve as a good example.

6.2.3 An Architecture Development Process

The need to have a properly defined methodolgoy and process in an EA program is discussed earlier in Section 3.1. An EA process (as part of an EA methodology) prescribes what must be done when and how the steps are linked to one another [6, 26]. An established EA process is necessary in achieving a coherent set of architectural artifacts.

The DNDAF claims that it offers a “methodology” for developing architectural artifacts. According to the overview section of DNDAF, the four volumes of DNDAF are expected to specify a detailed method and a set of supporting tools for developing and maintaining the DND enterprise architecture. However, the current version of DNDAF does not provide such a detailed method nor does it define a high-level process for developing an enterprise architecture. Volume 4 of the documentation (the “The User Guide”) offers interim advice and guidance on how to use DNDAF to document architectural artifacts. The guide currently focuses on using the MS Office suite of tools until an EA toolset is available for use by DND/CD projects. In comparison, DoDAF defines a 6-step process for the development of architectural descriptions and is composed of an Architecture Development Process and a set of guidlines and recommendations on development and analysis of architectural artifacts (see Section 3.2.2).

There are a number of processes defined in the context of enterprise architecture which can be used as a starting point in defining a DNDAF architecture development process:
– the DoDAF process for architecture development [4, Vol. 1, Sec. 7],
– the TOGAF Architecture Development Method (ADM) [18],
– the Gartner Enterprise Architecture Process [25], which proposes a process model identifying the activities involved in the development of an enterprise architecture,
– the Methodology for AGency ENTerprise Architecture (MAGENTA) [26] defining a successfully applied methodology for development of enterprise architectures for the Singapore Government.

6.2.4 Standards and Tools

DNDAF provides DADM (see Section 5.3) as a standard set of architectural entities to be used in capturing common data requirements. This is an important first step supporting interoperability of systems and information sharing within the enterprise. However, there is more that needs to be done:

a) architectural artifacts produced under each sub-view need to be based on established standards or a common set of formats;

b) a set of recommended tools based on Open Standards need to be identified so that the products are not bond to propriotory technologies.

According to DNDAF Volume 2, a set of technical standards has been compiled within the Defence Technical Standards List (DTSL) that is expected to be used in the creation of the technical sub views [1, Vol. 2, Sec. 2.6]. However, there is no pointer to DTSL throughout the DNDAF documentation and the mentioned list is not publicly available online.

6.2.5 Practice Guidance

Best practices, "user's guides", and examples that are tailored to the needs, and available capabilities of DND/CF, facilitate the adoption of an EA practice within the enterprise and play an important role in the development of a culture in which EA is valued and used. The importance of such practice guidance for a DND/CF EA methodology is evident and does not require further discussions.

6.3 Systems Engineering: Modeling and Analysis

Given the complexity of DND/CF enterprise architecture, in capturing the current and developing the future architecture, the use of established system design concepts are inevitable to manage complexity through systematic approaches to modularization, refinement, validation and verification of high-level models.

One of the main purposes of employing an enterprise architecture practice is to be able to manage change and foresee the implication of changes in the organization. In order to do so, the architects must have the ability to validate the architectural artifacts against the current state of the affairs (for the as-is architecture) and the future requirements (for the to-be architecture) and, in certain aspects (such as security), verify key properties of the architecture.

In the practice of enterprise architecture changes are modeled prior to application to the organization so that one can (rigorously) inspect and reason about the key attributes and implications of the proposed changes to ensure the they are properly understood. A precise semantic foundation is a prerequisite for analysis, validation and verification of the models. Mathematical precision is vital, not only for analyzing critical properties of the design in order to uncover and eliminate design flaws and weaknesses through validation and verification, but also as a prerequisite for using computational methods and tools [76]. Sensible use of formal methods can arguably make system design more reliable, system development more predictable, and overall improve quality of the final product through analysis, validation and verification of system requirements [29].

As its current version, the DNDAF together with DADM provide the logical basis for capturing the architectures into structured architecture data that can be stored in architecture data repositories and manipulated with automated tools. Although, this is a step forward in adding structure and rigour to the framework, the use of established and more precise (formal) methods in capturing the architectural artifacts is still missing in DNDAF.

6.3.1 Formal Modeling

There are various approaches to formal modeling of complex systems as briefly discussed in Section 4.1. Considering the diversity of the stakeholders and the multi-disciplinary nature of the DND/CF enterprise architecture program, we need a formal method that enables the architects to create formal models that

- are concise and easily readable, both for domain experts and system designers,
- are described at appropriate levels of abstraction (not requiring unnecessary details),
- are intuitively expressed in application-domain terms,
- and can be empirically validated using machine assisted simulation and testing.

In light of these requirements, we recommend using an agile formal modeling approach such as Abstract State Machines (ASMs) [43]. ASMs capture system structures and behaviour in form of executable pseudo-code-like models that come with a precise semantics. In modeling, ASM offers the freedom to choose an appropriate level of abstraction in order to focus on the essential aspects of the design and avoid including unnecessary details. In addition, ASM syntax and constructs can be easily extended in practice to introduce application-domain terms and structures into the framework.

The ASM formalism and abstraction principles are known for their practical side of formal modeling in high-level specification and design. This formalism has been used extensively over many years in modeling of architectures, languages, and protocols [51, 54, 48, 58, 55], resulting in solid methodological foundations. Thus, ASM refinement and modularization techniques [44] are well established and have been used successfully in various real-life (industrial) applications. In addition to IT systems, ASMs have been used more recently in computational criminology [77, 78] and for modeling and validation of aviation security [79, 80]. A driving factor in many of the above applications is the desire to systematically reveal abstract architectural and behavioural concepts so that the underlying *blueprint* of the functional system requirements becomes clearly visible and can be inspected by analytical means (verification) and empirical techniques (simulation) using machine assistance as appropriate. In summary, the following features of the ASM formalism are desirable to have in the development of an enterprise architecture:

a) simple and concise specifications that are understandable both for domain experts and system designers, and facilitate reasoning about design concepts and communicate the design decisions to the various stakeholders;

b) a precise semantic foundation which is a prerequisite for analysis, validation and verification of models;

c) freedom of abstraction that supports writing of abstract and minimal specifications that express the original idea behind a system design at essentially the same levels of abstraction and complexity, thus enabling designers to stress on essential aspects of their design rather than encoding insignificant details;

d) a well-defined refinement technique (to be paired with abstraction) that allows the designer to cross levels of abstraction and link models at different levels through incremental steps all the way down to a concrete model;

e) executability of abstract and even incomplete models which is often a desirable feature for experimental validation of design ideas in early stages to obtain feedback from stakeholders prior to producing concrete systems.

Abstract Opertational Models in DND/CF EA

Abstract operational models, such as ASMs, can be used in various architectural artifacts in modeling and description of processes, functionalities, data structures, and state transition patterns. In the context of DNDAF, we can identify a number of DNDAF sub-views in which ASMs can be utilized to formally define the dynamic aspects of an enterprise architecture. However,

Process Models: OV-5b, OV-6b, and SV-10b

The Operational Process Model (OV-5b) describes the operational and business processes within the architecture projects. DNDAF suggest that UML activity diagrams can be used in this sub-view. The Operational State Transition Descriptions (OV-6b) view describe the detailed sequencing of activities or work flow in a business process, adding more details to the process model of OV-5b. The Systems State Transition Descriptions (SV-10b) represents state transitions from a systems perspective. DNDAF recommends using UML state chart diagrams for the description of state transitions (both OV-6b and SV-10b).

For all these views, Control State ASMs are a perfect fit by formally capturing the operational process model (in abstract form) in terms of control state changes and further capturing the state transitions in both operational and systems views. The *freedom of abstraction* in ASMs enables one to choose the level of abstraction as one deems suitable at any stage of the design. This also allows a seamless transition form the abstract view of OV-5b to the more detailed view of OV-6b and further to the systems level view of SV-10b.

Functionality Models: OV-5a and SV-4

The Functional Model (OV-5a) describes the functions and their relationships, focusing on "what must be done" without the details of "how it is to be done". The Systems Functionality Description (SV-4) documents systems functional hierarchies and systems functions, and the data flows between them.

High-level ASM programs can be used in both of these views, especially as part of the composite view SV-4, to formally model functional decompositions and provide abstract descriptions of system functionalities at desired levels of abstraction.

Rules Models: OV-6a and SV-10a

The Operational Rules Model (OV-6a) describes operational (mission-oriented) statements defining constraints or other aspects of the architecture project. The Systems Rules Model (SV-10a) builds on the rules of OV-6a to describe the physical implementation by defining rules that control, constrain, or otherwise guide the actions within a system.

The current recommendation of DNDAF is to use any of the following options to formally define these statements: decision trees and tables, structured English [34], and mathematical logic. Because computational mathematical logic is an integral part of ASMs, rules of OV-6a and SV-10a can be formulated in ASMs and later be applied to and be analyzed, in combination with other views, as part of the overall ASM model of the architecture.

Other views

Although the above views represent the main DNDAF views in which ASMs can be most affective, application of ASMs in DNDAF is not limited to those views. ASMs rules and state descriptions can be used to formally describe the Operational Event-Trace Description (OV-6c), Logical Data Model (OV-7), and Systems Event-Trace Description (SV-10c).

6.3.2 Validation and Verification

In modeling an enterprise, whether capturing the architecture of the enterprise as-is or designing the future architecture to-be, a notorious question is: *how do we know if we have created the right model?* That is, how can we *validate* that the as-is model reflects the reality and the to-be model reflects our future requirements?

There are two approaches to validation: *a)* validation by analytical means, and *b)* validation through simulation and testing. The first approach requires a formal (precise) model the properties of which can be carefully analyzed and validated against the reality or intended requirements. The second approach, requires models that are executable and can be tested

34. Structured English can hardly be considered a *formal* approach to describe constraints and rules.

against various scenarios. An executable model is interesting in two aspects: *a)* it helps finding ambiguities, missing pieces and loose-ends of the model and forces the system analyst/modeler to think clearly about the main concepts and their definitions, and *b)* it supports experimental validation through execution (simulation).

Once we have established and validated the requirement specifications and a formal model of a system, the next question to face is: *how do we know if we are building the system right?* That is, how can we *verify* that the implemented system accurately represents and satisfies the requirements. Formal verification techniques aim to increase quality of systems by providing a consistent logical framework in which key properties of a system can actually be proven using model checking techniques. A proper formal specification is a prerequisite for any meaningful approach to formal verification.

In capturing the current DND/CF enterprise architecture and design and modeling of its tar-get architecture, it is critical to prove correctness of certain properties, such as those related to security and warzone operations. By augmenting DNDAF with agile practical formal methods (such as ASM), operational and functional aspects of the developed architecture can be validated and formally verified against the reality and future requirements.

ASM models are executable in principle and, when combined with the CoreASM tool support,[35] enable experimental validation of highly abstract models and design specifications through both symbolic execution (model checking) and simulation and testing, and facilitate interoperability with other tools and analytical techniques in an open tool environment (see www.coreasm.org).

ASM together with CoreASM has a track record of application in high-level modeling and validation of complex systems. A number of recent applications are as follows:

- Dynamic Resource Configuration & Management (DRCM) [63] is a highly adaptive and auto-configurable, multi-layer network architecture for distributed information fusion. The ASM model of the underlying design provides a reliable basis for reasoning about key system attributes at an intuitive level of understanding, supporting requirements specification, design analysis and validation of dynamic properties. Building an abstract yet executable DRCM model in CoreASM enabled advanced experiments to validate consequential design decisions.
- Integrating ASM modeling with Interpreted Systems for situation analysis decision support system design [64], exemplifies the benefits of using ASM and CoreASM in combination with the Interpreted Systems approach of [82] in modeling multiagent systems for situation analysis. Refinement of the abstract model into an executable CoreASM model serves two purposes: *a)* it helps finding ambiguities, missing pieces and loose ends of the model and forces the system analyst/modeler to think clearly about the main con-

35. The only machine-assisted verification supported by the current implementation of CoreASM is in the form of rudimentary model checking [81]. More sophisticated interfaces to existing model-checking tools are needed to fully exploit the potential they provide.

cepts and their definitions, and *b)* it supports experimental validation through execution (simulation).

- The Mastermind project [65] is a pioneering interdisciplinary project in computational criminology that focuses on modeling and simulation in the study of spatiotemporal patterns of offender behavior in urban environments. At the heart of Mastermind is a robust ASM ground model, developed over many iterations, for checking the validity of the model with respect to established crime theories. The process of establishing the key properties and ensuring the validity of the model was greatly facilitated by running experiments on abstract models using CoreASM. In this project, CoreASM has played an important role in facing the challenges of two major phases of the Mastermind project, namely *formalization* and *validation*. [65]
- Altenhofen and Börger [72] analyze a given cluster protocol implementation using an abstract ASM model, which they refine into an executable CoreASM model for running scenarios.
- Lemcke and Friesen at SAP Germany [66] propose a Web services composition algorithm for collaborative business processes defined in terms of a distributed ASM, using CoreASM for executing their ASM model to show that the generated orchestration steers the execution of the business processes as intended.
- Beckers et al. [68] use the simulation capabilities of CoreASM in their approach to model checking ASMs without the need for translation of the ASM specification into the modeling language of an existing model checker.

6.3.3 An ASM Example

Figure 13 presents an example used in DNDAF documentation [1] for an operational state transition diagram as part of OV-6b. In Figure 14 we provide a control state ASM version of that diagram in which ASM rules (activities) marked in blue and conditions (events and guards) marked in yellow. The resulting control state diagram differentiates between conditions and actions that affects the operational state. The control state diagram of Figure 14 can be precisely translated into an ASM model. Each transition action (an ASM rule denoted by a blue rectangle) can itself be represented as a more detailed control state ASM revealing further details of the activity and its potential inner-states. ASM conditions (yellow diamonds) can be precisely described using a boolean formula based on the current state of the system. Alternatively, more details of the activities can also be revealed using abstract ASM programs that capture the essential idea behind the activities in a readable fashion.

By simply producing a more precise diagram, like the Control State ASM of Figure 14, we can already detect some ambiguities in the artifact. As an example, there is no guard on the transitions from the state *In Conflict* to either of *Controlled* or *Maneuvering*. Could there be a conflict resolution *and* a clearance revision occuring at the same time? If so, what would be the future state of the airplane?

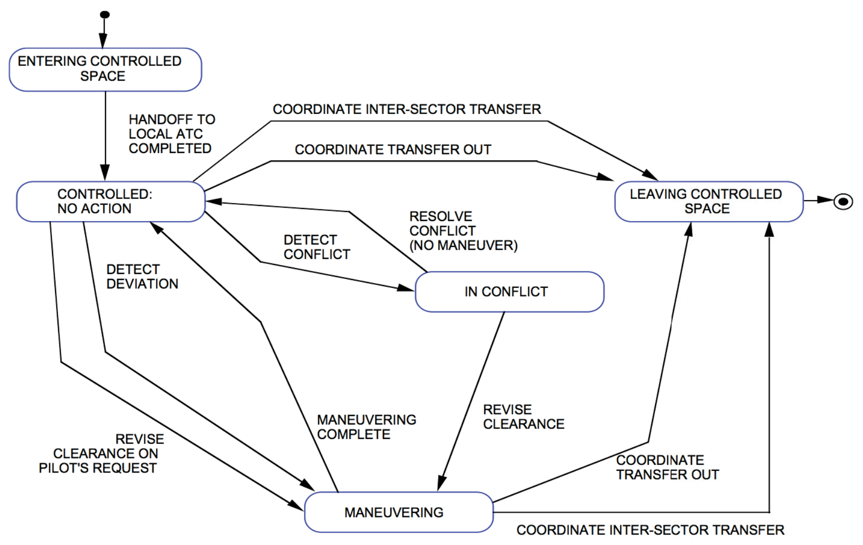

ENTERING CONTROLLED SPACE
HANDOFF TO LOCAL ATC COMPLETED
COORDINATE INTER-SECTOR TRANSFER
COORDINATE TRANSFER OUT
CONTROLLED: NO ACTION
LEAVING CONTROLLED SPACE
RESOLVE CONFLICT (NO MANEUVER)
DETECT CONFLICT
DETECT DEVIATION
IN CONFLICT
MANEUVERING COMPLETE
REVISE CLEARANCE
REVISE CLEARANCE ON PILOT'S REQUEST
COORDINATE TRANSFER OUT
MANEUVERING
COORDINATE INTER-SECTOR TRANSFER

Figure 13: *The Air Traffic Operations example of [1, Fig. 3.12.2]*

Such problems can not only be noticed by analytical means once a rigorous approach of documenting is employed, but they can also be discovered by model analysis tools. Figure 15 shows the same operational view model produced in *CSDe*, a Control State Diagram Editor udner development as part of the CoreASM project. CSDe enables modeling of Control State ASMs using a graphical interface. The tool can analyze the diagram for potential control flow problems and translate correct diagrams into executable ASM models, which (in principle) can be simulated using the CoreASM engine. In case of our example, CSDe finds a number of ambiguities in state transitions in our operational view example of Figure 15.

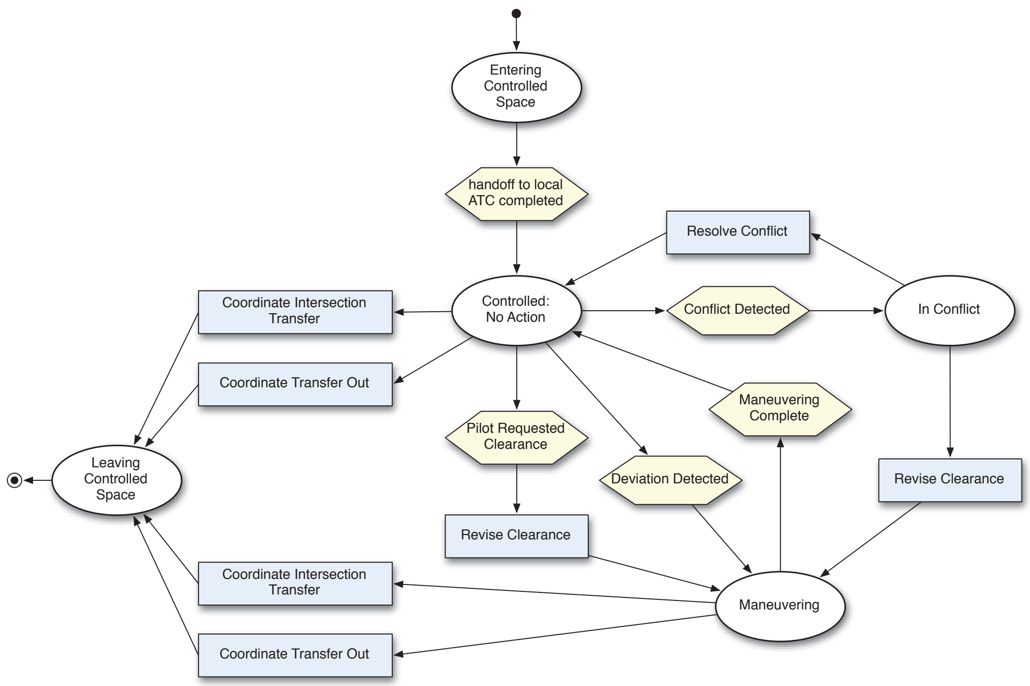


Figure 14: *Control State ASM model of Figure 13*

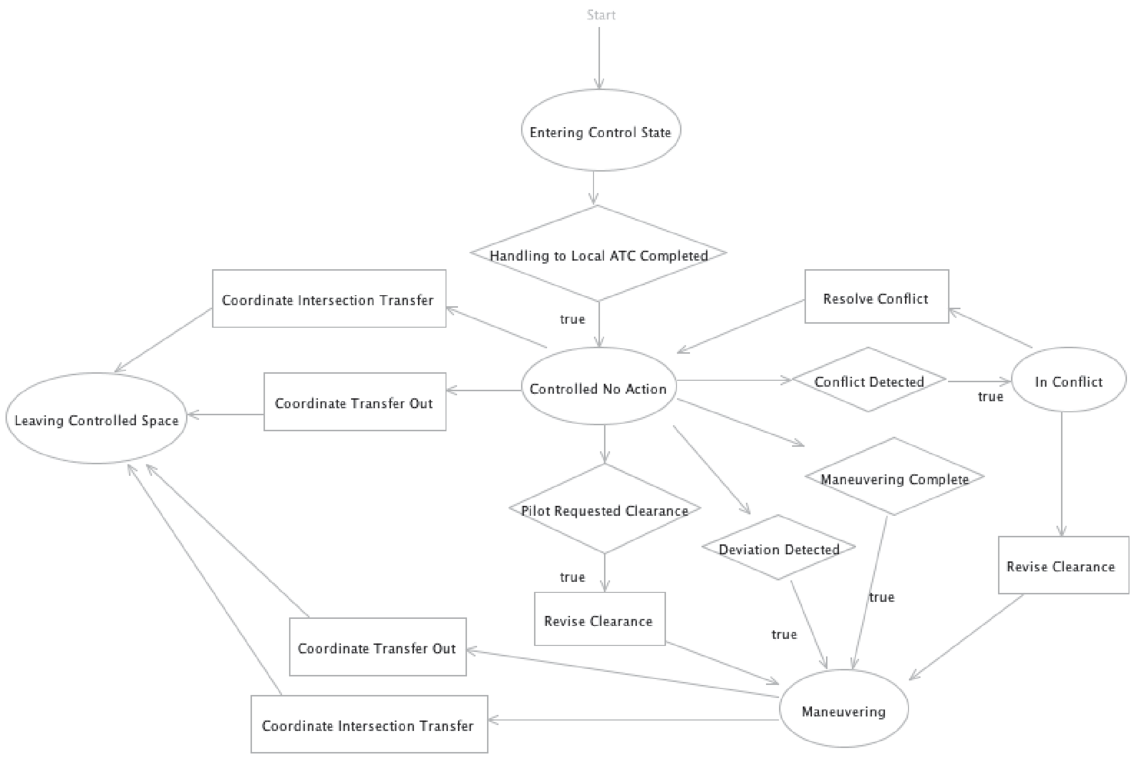


Figure 15: *Control State Diagram of 13 in CoreASM CSDe Editor*

7 Concluding Remarks and Future Work

The Department of National Defence and Canadian Forces is a large complex enterprise with a wide range of stakeholders that, with different interests and concerns, work together towards a common set of goals and objectives. In 2001, the department recognized the importance of adopting an Enterprise Architecture practice in controlling complexity and managing the evolution of the enterprise. In order to guide the process of establishing an EA practice within DND/CF, the details of the DND Enterprise Architecture Framework has been documented in form of a DND/CF Architecture Framework. All architecture efforts in DND/CF are expected to follow the framework to ensure interoperability and integration into an overall DND/CF enterprise architecture repository.

In order to achieve a DND/CF Integrated Command and Control (IC2) Capability, it is essential to have a comprehensive EA methodology that supports understanding of the current state of the DND/CF enterprise and facilitates managing the transition from what it currently is (as-is) to what it is desired to be in the future (to-be). Such a transition requires an enterprise architecture that helps in managing the complexity and facilitates change by providing a documentation of the components, relationships, constraints, and capabilities of the systems or processes.

In this report we critically reviewed DNDAF in light of EA concepts, definitions, and objectives. We discussed the concept of Enterprise Architecture and the notions of EA framework and EA methodology and we surveyed a number of top EA frameworks with the goal of identifying the main components of a comprehensive EA framework and methodology. Adopting the practice of EA and maintaining an enterprise architecture for DND/CF requires a detailed and comprehensive framework together with an enterprise-wide methodology that clearly identifies specific EA processes, techniques, reference models, standards, and guidance for assessment of architectural efforts. As its current state, DNDAF does not provide neither such a comprehensive framework nor a methodology for adopting EA within DND/CF. What DNDAF mainly provides is a well-defined architectural taxonomy through a set of views and sub-views that categorize architectural artifacts to be produced.

As a result of this analysis, we pointed out the main shortcomings and limitations of DNDAF and provide a set of recommendations for extending and improving DNDAF towards a comprehensive EA methodology that can be effectively applied across DND/CF in order to guide and facilitate assimilation of EA concepts and adoption of an EA practice within the enterprise. DNDAF is a constructive starting point towards establishing the practice of EA within DND/CF. However, we have argued in this report that in order to adopt an EA practice, DND/CF requires a comprehensive EA methodology that provides the following components (among many others listed in Section 6.2):

– a systems engineering and architecture development process that guides the development of architectural artifacts,

– formal engineering techniques that provide rigour and enable validation and verification of architectural artifacts, and
– open standards and tools that support the development and analysis of the architecture and facilitate interoperability between organizations.

We believe that the future direction of DNDAF should be shifting the framework towards a comprehensive EA methodology designed to address the specific requirements of DND/CF. One of the major aspects of this transition will be the introduction of high-level systems engineering methods and agile formal modeling techniques together with the adoption of open standards and tools that guide and facilitate the development, validation, and verification of architectural artifacts. In developing architectural artifacts, one should have a proper understanding of the real world situation, precisely document the current state and future design decisions, and effectively communicate such decisions with the domain experts as early as possible. These challenges call for adoption of proper engineering methods and tools.

Abstraction and formalization paired with tool support for producing executable models that can be validated through simulation and testing provide effective instruments for capturing models of the systems as-is and the requirements for future improvements. One of the promising methods for high-level system design and analysis that builds on abstraction and formalization is the Abstract State Machines method. The ASM method aims at industrial system design and development by integrating precise high-level, problem-domain oriented modeling into the design and development cycle, and by systematically linking abstract models down to executable models that can be validated and tested. For almost two decades now, abstract state machines have been studied, practiced, and applied in modeling and specification of systems to bridge the gap between formal and pragmatic approaches.

One of the most prominent tools based on the ASM method is the open source CoreASM toolset (see Section 4.3.2). It has been put to the test in a range of applications in the private and public sectors, spanning computational criminology, coastal surveillance, decision support, and Web services architectures [83]. The diversity of application fields confirms the practicability of using CoreASM for requirements analysis, design specification and rapid prototyping of abstract executable models.

CoreASM has been built with an innovative extensible architecture that allows seamless extensions of the toolset, addition of new features and capabilities, in order to adapt CoreASM to various application domains. Developed based on a solid mathematical foundation and an established systems engineering method, we consider CoreASM as a promising candidate to be adopted to enable development of precise executable architectural artifacts that can be validated and verified as needed.

DND/CF needs to develop and evolve a formal methodology and a set of open tools to support DND/CF EAP. Abstract State Machines and CoreASM have the potential to be further improved and customized for the particular needs of the DND/CF EAP as needed.

References

[1] DND/CF Director Enterprise Architecture (2009), DND/CF Architecture Framework (DNDAF) Version 1.6, Department of National Defence and Canadian Forces. http://www.img-ggi.forces.gc.ca/pub/af-ca/index-eng.asp.

[2] NATO SAS-065 Research Task Group (2008), NATO NEC C2 Maturity Model Overview, Draft for peer review ed. Available at www.dodccrp.org.

[3] Zachman, John A., The Zachman Framework: The Official Concise Definition. http://www.zachmanframeworkassociates.com.

[4] U.S. Department of Defense (2009), DoD Architecture Framework Version 2.0. http://cio-nii.defense.gov/sites/dodaf20.

[5] Federal Enterprise Architecture Program Management Office, Office of Management and Budget (2007), FEA Practice Guidance.

[6] Cockburn, A. (2000), Selecting a project's methodology, *IEEE Software*, 17, 64–71.

[7] NATO SAS-050 Research Task Group (2006), Exploring New Command and Control Concepts and Capabilities, Final report ed, Prepared for NATO. http://www.dodccrp.org/files/SAS-050FinalReport.pdf.

[8] Saha, Pallab (2008), Advances in Government Enterprise Architecture, Hershey, PA: Information Science Reference - Imprint of: IGI Publishing.

[9] McGovern, James et al. (2003), The Practical Guide to Enterprise Architecture, Upper Saddle River, NJ, USA: Prentice Hall PTR.

[10] Land, M. O., Proper, E., Waage, M., Cloo, J., and Steghuis, C. (2008), Enterprise Architecture: Creating Value by Informed Governance, Vol. 1st Edition of *The Enterprise Engineering Series*, Springer.

[11] NATO C3 Board, NATO Architecture Framework (NAF) Version 3. http://www.nhqc3s.nato.int/Architecture/default.asp.

[12] U.S. Federal Chief Information Officer Council (2001), A Practical Guide to Federal Enterprise Architecture (Version 1.0). http://www.cio-index.com/nm/articlefiles/42125-bpeaguide.pdf.

[13] Sessions, Roger (2007), A Comparison of the Top Four Enterprise-Architecture Methodologies. http://msdn.microsoft.com/en-us/library/bb466232.aspx.

[14] Draffin, Anthony (2010), Methodology vs framework: why waterfall and agile are not methodologies. http://anthonydraffin.posterous.com/methodology-vs-framework-why-waterfall-and-ag.

[15] Zachman, John A. (1987), A Framework for Information Systems Architecture, *IBM Systems Journal*, 26(3), 276–292.

[16] UK Ministry of Defence, MOD Architectural Framework (MODAF). http://www.mod.uk/modaf.

[17] UK Ministry of Defence, MODAF Architecting Process. http://www.mod.uk/NR/rdonlyres/6A737771-4BD3-41D9-A308-D5F377B39316/0/20090210_MODAF_Architecting_Process_V1_0_U.pdf.

[18] (2009), The Open Group Architecture Framework (TOGAF) Version 9, The Open Group. http://www.opengroup.org/architecture/togaf9-doc/arch.

[19] The Open Group, Definition of SOA. http://opengroup.org/projects/soa/doc.tpl?gdid=10632.

[20] The White House, Federal Enterprise Architecture Website. http://www.whitehouse.gov/omb/e-gov/fea.

[21] The Chief Information Officers Council (1999), Federal Enterprise Architecture Framework.

[22] Office of Management and Budget (2007), FEA Consolidated Reference Model Document Version 2.3. http://www.whitehouse.gov/omb/assets/fea_docs/FEA_CRM_v23_Final_Oct_2007_Revised.pdf.

[23] The Office of Management and Budget (OMB), Enterprise Architecture Assessment Framework (EAAF). http://www.whitehouse.gov/omb/e-gov/eaaf.

[24] US Department of the Treasury, Chief Information Officer Council (2000), Treasury Enterprise Architecture Framework Version 1. http://www.eaframeworks.com/TEAF/teaf.doc.

[25] Bittler, R. Scott and Kreizman, Gregg (2005), Gartner Enterprise Architecture Process: Evolution 2005, (Technical Report G00130849) Gartner.

[26] Saha, Pallab (2008), A Methodology for Government Transformation with Enterprise Architecture, In Saha, Pallab, (Ed.), *Advances in Government Enterprise Architecture*, Information Science Reference - Imprint of: IGI Publishing.

[27] Schultz, Mark (2007), Architecture principles: Creating the foundation for robust architecture. http://www.ibm.com/developerworks/library/ar-archprinc. IBM Software Group.

[28] Ostergaard, Dean (2009), Enterprise Architecture Principles. http://www.architectureinpractice.com/sites/default/files/EA-Principles.doc.

[29] Berry, Daniel M. (2002), Formal Methods: the very idea—Some thoughts about why they work when they work, *Science of Computer Programming*, 42(1), 11–27.

[30] Mellor, Stephen J., Scott, Kendall, Uhl, Axel, and Weise, Dirk (2004), MDA Distilled: Principles of Model-Driven Architecture, Addison-Wesley.

[31] France, R., Evans, A., Lano, K., and Rumpe, B. (1998), The UML as a formal modeling notation, *Comput. Stand. Interfaces*, 19(7), 325–334.

[32] Ellsberger, Jan, Hogrefe, Dieter, and Sarma, Amardeo (1997), SDL : Formal Object-oriented Language for Communicating Systems, Prentice Hall.

[33] Huggins, J. and Wallace, C. (2002), An Abstract State Machine Primer, (Technical Report CS-TR-02-04) Computer Science Department, Michigan Technological University.

[34] Bjørner, Dines and Jones, Cliff B., (Eds.) (1978), The Vienna Development Method: The Meta-Language, Vol. 61 of *Lecture Notes in Computer Science*, Springer.

[35] Bidoit, M. and Mosses, Peter (2004), Casl User Manual: Introduction to Using the Common Algebraic Specification Language Casl, SpringerVerlag.

[36] Milner, Robin, Parrow, Joachim, and Walker, David (1992), A Calculus of Mobile Processes, *Information and Computation*, 100, 1–40.

[37] Hoare, C. A. R. (2002), Communicating sequential processes, pp. 413–443.

[38] ITU-T Recommendation Z.100 Annex F (11/00) (2001), SDL Formal Semantics Definition, International Telecommunication Union.

[39] Peterson, J. L. (1981), Petri Net Theory and the Modeling of Systems, Prentice-Hall.

[40] Abrial, J.R. (1996), The B-Book: Assigning Programs to Meanings, Cambridge University Press.

[41] Spivey, J. Michael (1992), The Z Notation: a reference manual, 2 ed, Prentice Hall International Series in Computer Science.

[42] Jackson, Daniel (2006), Software Abstractions: Logic, Language, and Analysis, MIT Press.

[43] Börger, E. and Stärk, R. (2003), Abstract State Machines: A Method for High-Level System Design and Analysis, Springer-Verlag.

[44] Börger, E. (2003), The ASM Refinement Method, *Formal Aspects of Computing*, 15, 237–257.

[45] Foundation, Free Software (2007), GNU General Public License. Available electronically at `http://www.gnu.org/copyleft/gpl.html` (Last visited in March 2009).

[46] Canada, Ora (1998), Z/EVES Version 1.5: An Overview, In *FM-Trends*, pp. 367–376.

[47] Gurevich, Y. (1995), Evolving Algebras 1993: Lipari Guide, In Börger, E., (Ed.), *Specification and Validation Methods*, pp. 9–36, Oxford University Press.

[48] Glässer, U., Gotzhein, R., and Prinz, A. (2003), The Formal Semantics of SDL-2000: Status and Perspectives, *Computer Networks*, 42(3), 343–358.

[49] Börger, E., Glässer, U., and Müller, W. (1995), Formal Definition of an Abstract VHDL'93 Simulator by EA-Machines, In Delgado Kloos, C. and Breuer, P. T., (Eds.), *Formal Semantics for VHDL*, pp. 107–139, Kluwer Academic Publishers.

[50] Müller, W., Ruf, J., and Rosenstiel, W. (2003), An ASM Based SystemC Simulation Semantics, In Müller, W. et al., (Eds.), *SystemC - Methodologies and Applications*, Kluwer Academic Publishers.

[51] Stärk, R., Schmid, J., and Börger, E. (2001), Java and the Java Virtual Machine: Definition, Verification, Validation, Springer-Verlag.

[52] Börger, E., Fruja, N. G., Gervasi, V., and Stärk, R. F. (2005), A High-level Modular Definition of the Semantics of C#, *Theoretical Computer Science*, 336(2/3), 235–284.

[53] Börger, E. (1990), A Logical Operational Semantics for Full Prolog. Part I: Selection Core and Control, In Börger, E., Kleine Büning, H., Richter, M. M., and Schönfeld, W., (Eds.), *CSL'89. 3rd Workshop on Computer Science Logic*, Vol. 440 of *LNCS*, pp. 36–64, Springer.

[54] Farahbod, R., Glässer, U., and Vajihollahi, M. (2007), An Abstract Machine Architecture for Web Service Based Business Process Management, *International Journal of Business Process Integration and Management*, 1, 279–291.

[55] Glässer, U. and Gu, Q.-P. (2005), Formal Description and Analysis of a Distributed Location Service for Mobile Ad Hoc Networks, *Theoretical Comp. Sci.*, 336, 285–309.

[56] Glässer, U., Gurevich, Y., and Veanes, M. (2004), Abstract Communication Model for Distributed Systems, *IEEE Trans. on Soft. Eng.*, 30(7), 458–472.

[57] Börger, E., Riccobene, E., and Schmid, J. (2000), Capturing Requirements by Abstract State Machines: The Light Control Case Study, *Journal of Universal Computer Science*, 6(7), 597–620.

[58] Beierle, C., Börger, E., Durdanovic, I., Glässer, U., and Riccobene, E. (1996), Refining Abstract Machine Specifications of the Steam Boiler Control to Well Documented Executable Code, In Abrial, J.-R., Börger, E., and Langmaack, H., (Eds.), *Formal Methods for Industrial Applications. Specifying and Programming the Steam-Boiler Control*, Number 1165 in LNCS, pp. 62–78, Springer.

[59] Börger, E., Päppinghaus, P., and Schmid, J. (2000), Report on a Practical Application of ASMs in Software Design, In Y. Gurevich and P. Kutter and M. Odersky and L. Thiele, (Ed.), *Abstract State Machines: Theory and Applications*, Vol. 1912 of *LNCS*, pp. 361–366, Springer-Verlag.

[60] Börger, Egon (2007), Construction and Analysis of Ground Models and their Refinements as a Foundation for Validating Computer Based Systems, *Formal Aspects of Computing*, 19(2), 225–241.

[61] Farahbod, Roozbeh (2009), CoreASM: An Extensible Modeling Framework & Tool Environment for High-level Design and Analysis of Distributed Systems, Ph.D. thesis, Simon Fraser University, Burnaby, Canada.

[62] Farahbod, R., Gervasi, V., and Glässer, U. (2007), CoreASM: An Extensible ASM Execution Engine, *Fundamenta Informaticae*, pp. 71–103.

[63] Farahbod, R., Glässer, U., and Khalili, A. (2009), A Multi-Layer Network Architecture for Dynamic Resource Configuration & Management of Multiple Mobile Resources in Maritime Surveillance, In *Proc. of SPIE Defense & Security Symposium*, Orlando, Florida, USA.

[64] Farahbod, R., Glässer, U., Bossé, É., and Guitouni, A. (2008), Integrating Abstract State Machines and Interpreted Systems for Situation Analysis Decision Support Design, In *Proc. of the 11th Intl Conf. on Information Fusion (Fusion 2008)*.

[65] Brantingham, P. L., Glässer, U., Jackson, P., and Vajihollahi, M. (2009), Modeling Criminal Activity in Urban Landscapes, In Memon, N., Farley, J. D., Hicks, D. L., and Rosenoørn, T., (Eds.), *Mathematical Methods in Counterterrorism*, pp. 9–31, Springer.

[66] Lemcke, Jens and Friesen, Andreas (2007), Composing Web-service-like Abstract State Machines (ASMs), *Services, IEEE Congress on*, pp. 262–269.

[67] Altenhofen, M., Friesen, A., and Lemcke, J. (2008), ASMs in Service Oriented Architectures, *Journal of Universal Computer Science*, 14(12), 2034–2058.

[68] Beckers, Jörg, Klünder, Daniel, Kowalewski, Stefan, and Schlich, Bastian (2008), Direct Support for Model Checking Abstract State Machines by Utilizing Simulation, In *ABZ '08: Proceedings of the 1st international conference on Abstract State Machines, B and Z*, pp. 112–124, London, UK.

[69] Jensen, Olav, Koteng, Raymond, Monge, Kjetil, and Prinz, Andreas (2007), Abstraction using ASM Tools, In Prinz, A., (Ed.), *Proceedings of the 14th International ASM Workshop (ASM'07)*.

[70] Mazzei, Daniele, Vozzi, Federico, Cisternino, Antonio, Vozzi, Giovanni, and Ahluwalia, Arti (2008), A High-Throughput Bioreactor System For Simulating Physiological Environment, *IEEE Transactions on Industrial Electronics*, 55(9), 3273–3280.

[71] Demuru, Matteo (2008), Modeling Cell Methabolic Mechanisms Through Abstract State Machines, Master's thesis, University of Pisa, Italy.

[72] Altenhofen, M. and Börger, E. (2009), Concurrent Abstract State Machines and +CAL Programs, *Recent Trends in Algebraic Development Techniques: 19th International Workshop, WADT 2008, Pisa, Italy, June 13-16, 2008, Revised Selected Papers*, pp. 1–17.

[73] Altenhofen, M. and Farahbod, R. (2010), Bârun: A Scripting Language for CoreASM, In *ABZ '10: Proceedings of the 2nd International Conference on Abstract State Machines, B and Z*, Orford, Canada.

[74] DND/CF Director Enterprise Architecture (2008), DND/CF Architecture Framework (DNDAF) Volume 1: Overview and Definitions, (Technical Report RDIMS OTT_ TUNN-# 298730) Department of National Defence and Canadian Forces. Version 1.5.

[75] DND/CF Director Enterprise Architecture (2008), DND/CF Architecture Framework (DNDAF) Volume 2: DND/CF Views and Sub-views, (Technical Report RDIMS OTT_ TUNN-# 463363) Department of National Defence and Canadian Forces. Version 1.5.

[76] Farahbod, R. and Glässer, U. (2010), The CoreASM Modeling Framework, *Software: Practice and Experience (to be published)*.

[77] Brantingham, P. L., Kinney, B., Glässer, U., Jackson, P., and Vajihollahi, M. (2008), Mastermind: Computational Modeling and Simulation of Spatiotemporal Aspects of Crime in Urban Environments, In Liu, L. and Eck, J., (Eds.), *Artificial Crime Analysis Systems: Using Computer Simulations and Geographic Information Systems*, Information Science Reference.

[78] Brantingham, P. L., Glässer, U., Kinney, B., Singh, K., and Vajihollahi, M. (2005), A Computational Model for Simulating Spatial Aspects of Crime in Urban Environments, In Jamshidi, M., (Ed.), *Proceedings of the 2005 IEEE International Conference on Systems, Man and Cybernetics*, pp. 3667–74.

[79] Glässer, U., Rastkar, S., and Vajihollahi, M. (2006), Computational Modeling and Experimental Validation of Aviation Security Procedures, In Mehrotra, Sharad, Zeng, Daniel Dajun, Chen, Hsinchun, Thuraisingham, Bhavani M., and Wang, Fei-Yue, (Eds.), *Intelligence and Security Informatics, IEEE International Conference on Intelligence and Security Informatics, ISI 2006, San Diego, CA, USA, May 23-24, 2006, Proceedings*, Vol. 3975 of *Lecture Notes in Computer Science*, pp. 420–431, Springer.

[80] Glässer, U., Rastkar, S., and Vajihollahi, M. (2008), Modeling and Validation of Aviation Security, In Chen, H. and Yang, C.C., (Eds.), *Intelligence and Security Informatics: Techniques and Applications*, Vol. 135 of *Studies in Computational Intelligence*, pp. 337–355, Springer.

[81] Farahbod, R., Glässer, Uwe, and Ma, G. (2007), Model Checking CoreASM Specifications, In Prinz, A., (Ed.), *Proceedings of the 14th International ASM Workshop (ASM'07)*.

[82] Maupin, Patrick and Jousselme, Anne-Laure (2007), Interpreted Systems for Situation Analysis, In *Proc. of the 10th Intl. Conf. on Information Fusion*, Quebec city, Canada.

[83] Farahbod, R. and Glässer, U. (2011), The CoreASM Modeling Framework, *Software: Practice and Experience*. (in print) http://www.roozbeh.ca/pubs/2010-SPE-CoreASM.pdf.

[84] Pagan, Frank G. (1981), Formal Specification of Programming Languages: A Panoramic Primer, Prentice Hall.

[85] Stoy, Joseph E. (1981), Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory, The MIT Press.

Annex A: Requirements & Applications of EA

According to [10], using an Enterprise Architecture, *"it should be possible to:*

a) *Gain insight into the current state of the enterprise at a suitable abstraction level to understand and to analyze issues that hamper the execution of the strategy of the enterprise;*

b) *Gain insight into the current state of the enterprise to assess its compliance to (external) regulations;*

c) *Deal with social complexity of stakeholders involved in enterprise transformations;*

d) *Develop a business case for the chosen strategic direction;*

e) *Explore strategic alternatives for the future direction of the enterprise, while considering issues, challenges, feasibility and impacts, and eventually making for an alternative of choice;*

f) *Express/depict a coherent, comprehensive and concrete image of the desired future state(s) of the enterprise;*

g) *Design a roadmap for the transformation;*

h) *Distinguish between short-term solutions and long-term (structural) solutions;*

i) *Give a clear context and direction limiting design freedom of individual projects that contribute to the transformation;*

j) *Select available solutions and/or packages that are to remain or to be come apart of the solution, whether in-house or sourced by a business partner;*

k) *Guard the proper execution of any transformation project to be in line with the strategic direction (or to be knowingly informed that it deviates) and with external regulations;*

l) *Provide a common language to a portfolio of changes/transformations of an enterprise;*

m) *Enable traceability of design decisions from the strategic level via programs to specific projects.*

There are seven key applications for enterprise architecture:

a) *Investigate problems/shortcomings in a preexisting situation, including the creation of a shared (among stakeholders) understanding of the existing situation;*

b) *Express (and motivate) the future direction of an enterprise, as well as investigate (and evaluate) different alternatives. This also involves the creation of a shared (among stakeholders) conceptualisation of the (possible) future directions, and shared agreement for the selected alternative;*

c) *Identify key problems, challenges, issues, impediments, chances, etc., as well as make well-motivated design decisions that enable a move from the existing situation into the desired strategic direction;*

d) *Provide boundaries and identify plateaus (intermediary steps) for the transformation of the enterprise toward the articulated strategic direction. In this context, enterprise architecture is used as a planning tool, making the realization of a strategy more tangible;*

e) *Give a clear context and direction for a portfolio of projects working toward the realization of the first plateau as defined at the tactical planning level;*

f) *Select one or more standard solutions and/or packages that are to become part of the solution and/or decide to outsource an entire business process/service to another enterprise;*

g) *Create the high level design of an actual step in the enterprise transformation as it will be realized (and implemented) in the context of a specific project."* — [10]

Annex B: Formal Language Semantics

Modeling languages are used to create a formulation of a system, based on one's understanding of that system or its requirements, so that it can be documented, communicated with peers and domain experts, and better yet, empirically validated if possible. Such a formulation needs to be clear, precise, and comprehensive at a given level of abstraction. In order to achieve this, one needs to have a good understanding of the underlying modeling language used, which in turn requires a "good" description of the language.

A complete description of a modeling language covers three aspects of the language: *syntax*, *semantics*, and *pragmatics* [84]. The syntax is about the superficial form of the language constructs. It answers questions like, "is *X* a proper statement in this language?" The semantics is about the interpretation and the meaning of statements of the language. It answers questions like, "what does '`x := y + 1`' mean and what are its effects?" Finally, the pragmatics is about the use of such statements.

Language descriptions used to be more informal (i.e., expressed in a narrative form using a natural language), since formal descriptions using rigorous notations are not easily understandable without special training. However, it is often difficult to precisely and clearly describe the semantics of languages using an informal language. Informal descriptions rely on a common understanding of the underlying informal language and are amenable to different interpretations which in case of modelling languages defies the purpose of having a clear, precise and comprehensive formulation. If we want completeness, consistency, precision, absence of ambiguity, and understandability, we have to look into formal descriptions.

Formal semantic specification of a language can serve many purposes [85, 84]:

a) *Reference for users:* A formal specification can serve as a reference for users of the language, providing a detailed and accurate description of the language, its meaning and its effects.
b) *Reference for implementations.* Those who implement tools for a language such as compilers, interpreters or debuggers, need to precisely know the details of the language and its semantics. Also, such specifications are needed if one wants to prove the correctness of language compilers or interpreters.
c) *Improved language design.* Formal specifications can expose irregularities and inconsistencies in language design and can guide language designers towards the design of better and cleaner languages.
d) *Standardization.* It is now generally accepted that formal specifications are necessary to have a successful language standardization process.
e) *Program/model verification.* To mathematically prove the correctness of models and programs, the properties of the underlying language constructs must be formally defined.

This page intentionally left blank.

Annex C: Abstract State Machines

Here we briefly outline the basic concepts of ASMs—more precisely Distributed ASMs or DASMs—and Control State ASMs in formal modeling of distributed systems.

C.1 Distributed ASMs

The original notion of *basic ASMs* was defined to formalize simultaneous parallel actions of a single computational agent. A basic ASM M is defined as a tuple of the form $(\Sigma,\ I,\ \mathcal{R},\ P_M)$ where Σ is a finite set of function names and symbols, I is a set of initial states for signature Σ, $\mathcal{R}$ is a set of transition rule declarations, and $P_M \in \mathcal{R}$ is a distinguished rule, called the *main rule* or the Program of machine M.

A state $\mathfrak{A}$ for Σ is a non-empty set X together with an interpretation $f^{\mathfrak{A}} : X^n \mapsto X$ for each function name f in Σ. Functions can be *static* or *dynamic*. Values of dynamic functions can change from state to state. The evaluation of a transition rule in a given state produces a set of *updates* of the form $\langle (f, \langle a_1, \ldots, a_n \rangle), v \rangle$ where f is an n-ary function name in Σ and $a_1, \ldots, a_n, v \in X$. An update $(f, args, v)$ prescribes a change to the content of location $f(args)$ in the next state.

A distributed ASM (DASM) M_D is defined by a dynamic set AGENT of autonomously operating computational *agents*, each executing a basic ASM. This set may change dynamically over runs of M_D, as required to model a varying number of computational resources. Agents of M_D interact with one another, and typically also with the operational environment of M_D, by reading and writing shared locations of a global machine state.The underlying semantic model resolves potential conflicts according to the definition of *partially ordered runs* [56].

M_D interacts with its operational environment—the part of the external world visible to M_D—through actions/events observable at external interfaces, formally represented by controlled and monitored functions. Of particular interest are *monitored functions*, read-only functions controlled by the environment. A typical example is the abstract representation of global system time in terms of a monitored function *now* taking values in a linearly ordered domain TIME. Values of *now* increase monotonically over runs of M_D.

C.2 Control State ASMs

In this section we briefly look into *control state ASMs*, a frequently used class of ASMs that represents a normal form of synchronous UML activity diagrams. This particular class of ASMs is expressive enough to model many classical automata such as various extensions of finite state machines, timed automata, push-down automata, etc. It extends finite state machines by synchronous parallelism and by the possibility to also manipulate data [43].

A control state ASM is an ASM whose rules are all of the form presented in Figure C.1.[36] Such a control state ASM can be formulated in textual form by a parallel composition of Finite State

36. See [43, Sec. 2.2.6]

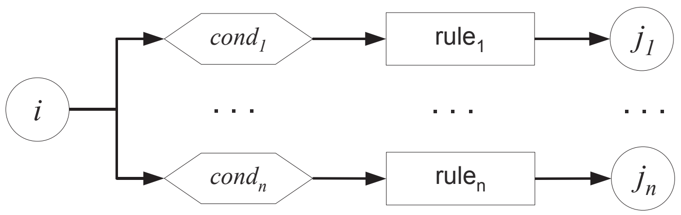


Figure C.1: *Control State ASMs*

Machine (FSM) rules, where each FSM rule is defined as:

```
$\textbf{FSM}(i, \textbf{if } cond \textbf{ then } rule, j) \equiv$
  $\textbf{if } ctl\_state = i \textbf{ and } cond \textbf{ then}$
    $rule$
    $ctl\_state := j$
```

Thus, the control state ASM of Figure C.1 can be formulated as a parallel composition of the following FSM rules:

```
$\mathsf{FSM}(i, \textbf{if } cond_1 \textbf{ then } rule_1, j_1)$
$\mathsf{FSM}(i, \textbf{if } cond_2 \textbf{ then } rule_2, j_2)$
$\ldots$
$\mathsf{FSM}(i, \textbf{if } cond_n \textbf{ then } rule_n, j_n)$
```

Since control state ASMs can be presented in graphical form with a precise semantics, they are a good candidate for documenting functional requirements and modeling of functional aspects of systems at the early stages of design and development when proper communication of the requirements and the abstract model plays a key role.

Index

This page intentionally left blank.

DOCUMENT CONTROL DATA

(Security markings for the title, abstract and indexing annotation must be entered when the document is Classified or Designated)

1. ORIGINATOR (The name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Centre sponsoring a contractor's report, or tasking agency, are entered in section 8.) Defence Research and Development Canada – Valcartier 2459 Pie-XI Blvd North Québec (Québec) G3J 1X5 Canada	2a. SECURITY MARKING (Overall security marking of the document including special supplemental markings if applicable.) UNCLASSIFIED 2b. CONTROLLED GOODS (NON-CONTROLLED GOODS) DMC A REVIEW: GCEC JUNE 2010

3. TITLE

Towards a Comprehensive DND/CF Enterprise Architecture Methodology - A Critical Review DNDAF for an Integrated C2 Capability Development

4. AUTHORS (last name, followed by initials – ranks, titles, etc. not to be used)

Farahbod, R., Guitouni, A., Bossé, É.

5. DATE OF PUBLICATION (Month and year of publication of document.) June 2013	6a. NO. OF PAGES (Total containing information, including Annexes, Appendices, etc.) 80	6b. NO. OF REFS (Total cited in document.) 85

7. DESCRIPTIVE NOTES (The category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Technical Report

8. SPONSORING ACTIVITY (The name of the department project office or laboratory sponsoring the research and development – include address.)

Defence Research and Development Canada – Valcartier
2459 Pie-XI Blvd North
Québec (Québec)
G3J 1X5 Canada

9a. PROJECT OR GRANT NO. (If appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant.) 42zz78	9b. CONTRACT NO. (If appropriate, the applicable number under which the document was written.)
10a. ORIGINATOR'S DOCUMENT NUMBER (The official document number by which the document is identified by the originating activity. This number must be unique to this document.) DRDC Valcartier TR 2011-022	10b. OTHER DOCUMENT NO(s). (Any other numbers which may be assigned this document either by the originator or by the sponsor.)

11. DOCUMENT AVAILABILITY (Any limitations on further dissemination of the document, other than those imposed by security classification.)

Unlimited

12. DOCUMENT ANNOUNCEMENT (Any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in (11) is possible, a wider announcement audience may be selected.))

Unlimited

13. ABSTRACT (A brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual.)

The objective of this report is to present a critical review of the DNDAF as a core en-terprise architecture (EA) framework to support the development of Integrated Command and Control Capability. In this document, we focus on systems engineering. In systems engineering, architectures facilitate the understanding and communication of different aspects of a system by providing a structured approach to document requirements, design decisions, and technical details of the implementation. This report presents a study of the concept of EA and a number of top EA frameworks in order to identify different aspects of EA that each framework covers and the various components each. The differentiation between the notions of methodology and framework in the context of EA leads to a critical review of the DNDAF framework. This report proposes a review of DNDAF, its purpose, and its components, and puts forward a set of recommendations for improvement in order to achieve a comprehensive DND/CF EA methodology and framework. This methodology is required for developing an Integrated Command and Control Capability. Future work includes determining the critical components (e.g., tools, methods, standards) that should be developed in order to support the implementation of a comprehensive EA for DND/CF.

L’objectif de ce rapport est de présenter une analyse critique de la DNDAF comme élément central d’un cadre d’architecture d’entreprise (AE) pour soutenir le développement de la capacité intégrée de commandement et contrôle (IC2C). Dans ce document, nous nous concentrons sur l’ingénierie de système de systèmes. En ingénierie de systèmes, les architectures facilitent la compréhension et la communication des différents aspects d’un système en fournissant une démarche structurée pour documenter les exigences, les décisions de conception et les détails techniques de la mise en œuvre. Ce rapport présente une étude de la notion d’AE et un certain nombre de cadres d’AE afin de dégager les différents aspects d’une AE et ses divers composants. La différenciation entre les notions de méthodologie et de cadre dans le contexte d’AE a conduit à un examen critique DNDAF. Ce rapport propose une revue de DNDAF, son but et ses composantes et met en avant une série de recommandations pour son amélioration afin de mettre en place une méthodologie d’AE pour le MDN / FC. Cette méthodologie est nécessaire au développement d’une IC2C. Les travaux futurs comprendront la détermination des composantes critiques (p. ex., outils, méthodes, normes) qui devraient être développées afin d’appuyer la mise en œuvre d’une EA pour le MDN / CF.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (Technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus, e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

enterprise architecture; EA; frameworks; Zachman; MODAF; TOGAF; NATO architectural framework; Federal Enterprise Architecture (FEA); data model; validation; verification; abstract state machines (ASM); N2C2M2; maturity level; methodology; DoDAF; architecture taxonomy; formal systems methods; formal modeling; abstract modeling; DNDAF; integrated command and control capability