SANDIA REPORT

SAND2015-7850
Unlimited Release
Printed August, 2014

Enterprise and System of Systems Capability Development Life-Cycle Processes

David F. Beck

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@osti.gov
Online ordering: http://www.osti.gov/scitech

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5301 Shawnee Rd
Alexandria, VA 22312

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.gov
Online order: http://www.ntis.gov/search

SAND2015-7850
Unlimited Release
Printed August, 2014

Enterprise and System of Systems Capability Development Life-Cycle Processes

David F. Beck
Fuzing & Instrumentation Technologies
Sandia National Laboratories
P.O. Box 5800
Albuquerque, New Mexico 87185-0661

Abstract

This report and set of appendices are a collection of memoranda originally drafted circa 2007-2009 for the purpose of describing and detailing a models-based systems engineering approach for satisfying enterprise and system-of-systems life cycle process requirements. At the time there was interest and support to move from Capability Maturity Model Integration (CMMI) Level One (ad hoc processes) to Level Three. The main thrust of the material presents a rational exposé of a structured enterprise development life cycle that uses the scientific method as a framework, with further rigor added from adapting relevant portions of standard systems engineering processes. While the approach described invokes application of the Department of Defense Architectural Framework (DoDAF), it is suitable for use with other architectural description frameworks.

CONTENTS

FIGURES

TABLES

NOMENCLATURE

act. *Activity diagram*
AD. *Architectural Description*
ANSI. *American National Standards Institute*
AoA. *Analysis of Alternatives*
AV-1. *Overview and Summary Information*
AV-2. *Integrated Dictionary*

BDA. Battle Damage Assessment
bdd. *Block Definition Diagram*

CDR. *Contractor Requirements Document*
CJCSI. *Chairman of the Joint Chiefs of Staff Instruction*
CM. *Configuration Management*
COA. *Course Of Action*
COAL. *Consolidated Operational Activities List*
CONOPS. *Concept of Operations*
CV-1. *Vision*
CV-2. *Capability Taxonomy*
CV-4. *Capabilities Dependencies*
CV-6. *Capability to Operational Activities Mapping*
CV-7. *Capability to Services Mapping*

DFD. *Data Flow Diagrams*
DIME. *Diplomatic, Informational, Military, and Economic*
DIV-1. *Conceptual Data Model*
DIV-2. *Logical Data Model*
DIV-3. *Physical Data Model*
DoD. *Department of Defense*
DoDAF. *Department of Defense Architecture Framework*
DOE. *Department of Energy*
DOTMLPF. *Doctrine, Organization, Training, Materiel, Leadership and education, Personnel, and Facilities*
DOTMLPF-P. *Doctrine, Organization, Training, Materiel, Leadership and education, Personnel, Facilities, and Policy*

ECD. *Elaborated Context Diagram*
ECE. *Enterprise Capability Engineering*
EIA. *Electronics Industries Alliance*
EPISTEL. Environment, Political, Informatic, Social, Technological, Economic and Legal
E-R. *Entity-Relationship*

FEAF. *Federal Enterprise Architecture Framework*
FMI. *Functional Mock-up Interface*

G&O. *Goals and Objectives*
GERAM. *Generalized Enterprise-Reference Architecture and Methodologies*
GPA. *Generic Physical Architecture*
GUI. *Graphical User Interface*

HLA. *High-Level Architecture*
HSI. Human Systems Integration

I/O. *Input/Output*
ibd. *Internal Block Diagram*
IDEF. *Integration Definition*
IEEE. *Institute of Electrical and Electronics Engineers*
ISO. *International Organization for Standardization*

JCA. *Joint Capability Area*
JCSFL. *Joint Common System Function List*
JMETL. *Joint Mission Essential Task List*
JMT. *Joint Mission Thread*

KPP. *Key Performance Parameter*, *Key Performance Parameter*

M&S. *Modeling and Simulation*
MA. *Morphological Analysis*
MBE. *Models-based Engineering*
MBSE. *Models-Based Systems Engineering*
METL. *Mission Essential Task List*
MNS. *Mission Need Statement*
MODAF. *Ministry Of Defense Architecture Framework*
MOE. *Measure of Effectiveness*
MOP. *Measure Of Performance*
MT. *Mission Task*

OMG. *Object Management Group*
OOA/D. *Object-Oriented Analysis and Design*
OODA. Observe, Orient, Decide, and Act
OV. *Operational Viewpoint*
OV-1. *High Level Operational Concept Graphic*
OV-2. *Operational Resource Flow Description*
OV-4. *Organizational Relationships Chart*
OV-5a. *Operational Activity Decomposition Tree*
OV-5b. *Operational Activity Model*
OV-6a. *Operational Rules Model*
OV-6b. *State Transition Description*
OV-6c. *Operational Event-Trace Description*

par. *Parametric Diagram*
PEST. Political, Economic, Social, and Technological
PM. *Project Management*
PMBOK. *Project Management Body of Knowledge*
PMESII. Political, Military, Economic, Social, Information, and Infrastructure
PMP. *Project Management Plan*

QAP. *Quality Assurance Program*
QFD. *Quality Function Deployment*

req. *Requirements Diagram*

S/W. *Software*
SAM. *Surface-to-Air Missile*
SBS. *System Breakdown Structure*

SE. *Systems Engineering*
SEMP. Systems Engineering Management Plan
SEP. Systems Engineering Plan, *Systems Engineering Process*
seq. *Sequence diagram*
SOA. Service-Oriented Architecture
SoS. *System of Systems*
SRD. *System Requirements Document*
Std. *Standard*
StdV-1. *Standards Profile*
STEER. Socio-cultural, Technological, Economic, Ecological, and Regulatory factors
stm. *State Machine diagram*, *State Machine diagram*
SV-10a. *Systems Rules Model*
SV-7. *Systems Measures Matrix*
SvcV-1. *Services Context Description*, *Services Interface Description*
SvcV-10a. *Services Rule Model*
SvcV-10b. *Services State Transition Description*
SvcV-10c. *Services Event-Trace Description*
SvcV-2. *Services Resource Flow Description*, *Services Resource Flow Internal Description*
SvcV-4. *Services Functionality Description*
SvcV-7. *Services Measures Matrix*
SWOT. Strengths, Weaknesses, Opportunities, and Threats
SysML. *Systems Modeling Language*

TDD. *Technical Description Document*
TOGAF. *The Open Group Architecture Framework*
TPM. *Technical Performance Measure*
TPP. *Technical Performance Parameter*
TRM. *Technical Requirements Model*

uc. *use case*, *use case*
UJTL. *Universal Joint Task List*
UML. *Unified Modeling Language*
UPDM. *Unified Profile for DoDAF and MODAF*

V&V. *Verification and Validation*

1 INTRODUCTION

1.1 Background

Continued human endeavors, be they enterprises of an individual, or of a programmatic or corporate sort, generally benefit by pursuing a policy of continual introspective examination in order to identify areas for improvement. At the present time, this idea is formally embodied in quality management standards such as ISO 9001:2008,[1] which are frequently invoked in contractual or regulatory requirements.[2] The quality approaches set forth in all such current standards portray a clear heritage that, in the technical literature base, traces through the ideas of W. Edwards Deming and Joseph Juran—the so-called founders of the quality improvement movement of the 20th Century—to Walter Shewhart, who, circa 1924, followed the scientific method in developing recommendations for how Western Electric Company engineers could improve the quality of telephone hardware at the Hawthorne manufacturing plant. [3]

It is asserted herein that the type of introspection that leads to real, useful improvement is, by definition, concerned with mission, and with the management of the portfolio of resources and capabilities that are available to fulfill it. Questions to ask include those following below:

- Is the mission the right one?
- Are the near-, mid- and long-term goals and objectives being worked on aligned with the mission?
- Are the necessary and sufficient resources and capabilities available to effectively fulfill the goals and objectives, or is the mission being plagued with insufficient supplies?

---

[1] *Quality management systems—Requirements*, ISO 9001:2008(E), 4th Ed., International Organization for Standardization, Switzerland, 2008-11-15.

[2] For example, U.S. Department of Energy (DOE) contractors are required to comply with "Contractor Requirements Document (CRD)," *Quality Assurance*, DOE O 414.1C, Attachment 2, 4-25-2011; the CDR, in turn, specifies that contractors shall have a Quality Assurance Program (QAP) that is to use "the appropriate national or international consensus standard where practicable …" and specifically identifies ISO 9001 as "appropriate" for nonnuclear activities.

[3] See, for example, Best, M., and D. Neuhauser, "Walter A Shewhart, 1924, and the Hawthorne factory," *Quality and Safety in Health Care,* 15(2), April 2006, pp. 142-143.

- Is "excess baggage" (resource and capability overhead) being carried along, unnecessarily consuming otherwise limited resources?
- Is there a need to develop one or more competencies in order to effectively deploy the available resources in successfully accomplishing the mission?

If by such introspection it is determined that change is needed—that mission is not being met in a satisfactory manner—it is further asserted that, in order to have any degree of assurance at all that resource or capability development efforts will actually lead to the desired improvement, significant up-front investment in rational (reasoned) thought is generally necessary. For this is the only way to clearly set forth the requirements that top-down intentional development can proceed against. This is not to say that it is not possible for "leaps" in human intuition to jump directly to creative, innovative solutions to problems (e.g., technology push), only that such leaps are not the normative means to success.

1.2 Overview

The material presented in this report is intended to provide a rational exposé of a process framework and associated methodology that can be used to guide the formal development, assessment, and maintenance of an enterprise-level or system-of-systems capability portfolio. The necessity to specify a process for development activities is rooted in risk management. Depending generally upon the desired degree to which project risk (e.g., the impact of failure) will be managed—depending upon the desired or required level of design surety (assurance), and the level of management risks willing to be tolerated—different development approaches and levels of rigor are available. While in reality a continuum of approaches exist, for the purpose of illustrating this point, the spectrum of choices from a systems engineering perspective can be quantized into three paradigms: (1) *alchemy* (aka *fly-fix-fly*, *trial-and-error*, *build-test, black art*, *hunt and try,* an *Edisonian approach,* or even *expert opinion)*; (2) the *scientific method*; and (3), *science-based, systems engineering*. Roughly speaking, these three paradigms can be thought of as processes that support, in order, development of low, medium and high levels of design surety. Similar paradigms exist for program and project perspectives.

The process framework outlined in the following sections was developed in order to follow the "high road." To meet such a goal, the process adapted relevant elements of various systems and software engineering life cycle process standards such as ANSI/EIA-632,[4] IEEE-1220,[5] IEEE/EIA-12207,[6] and IEEE-15288.[7] The process, in turn, is overlain by a methodology that supports implicitly, if not explicitly, models-based engineering (MBE) practices (a boon to quality);[8] in particular, object-oriented (i.e., combined process and data) models-based systems engineering (MBSE) methods are followed throughout as significant benefits can accrue to a project from the expressiveness and rigor of the design artifacts so produced. The enterprise-level or system-of-systems architecture descriptions that are developed to contain and portray the modeling objects follow the recommendations of IEEE-1471.[9]

In general, architectural products developed by the process methods employed are described by DoDAF "Viewpoints" and "Models"[10] using the OMG *Unified Model for DoDAF and MODAF*[11] (UPDM) modeling language specification, although at times, due to DoDAF and UPDM limitations, the OMG *Systems Modeling Language*[12] (SysML) and *Unified Modeling Language*[13] (UML) are invoked. (There are even a few places where the object-oriented languages fall short of process needs, and "old-school" SE methods have to be called upon.) It should also be noted that other architectural frameworks or

---

[4] *Processes for Engineering a System*. ANSI/EIA-632-1999, January 1999.
[5] *IEEE Standard for Application and Management of the Systems Engineering Process*. IEEE Std 1220-2005, September 9, 2005.
[6] *Standard for Information Technology--Software Life Cycle Processes*. IEEE/EIA 12207.0, March 1998.
[7] *Systems Engineering--System Life Cycle Processes*. IEEE Std 15288-2004, June 8, 2005.
[8] The term MBE refers to the use of in integrated engineering infrastructure where engineering information is a hierarchy of models, and **not** simply to the use of engineering models. Effective MBE infrastructures: use a single model-based product definition within a unified information management structure (a particular item is only represented once); enable engineers to manage product information so that models contain the appropriate level of fidelity; and are computer based.
[9] *Systems and Software Engineering--Recommended Practice for Architectural Description of Software-Intensive Systems*. IEEE Std 1471-2000, July 15, 2007.
[10] *DoD Architecture Framework Version 2.0*. DoDAF V2.0, May 28, 2009.
[11] *Unified Profile for DoDAF and MODAF (UPDM)*. Version 2.0, January 2012.
[12] *OMG Systems Modeling Language (OMG SysML™)*, Version 1.3, June 2012.
[13] *OMG Unified Modeling Language™ (OMG UML)*, Version 2.4.1, August 2011.

modeling languages are suitable for use within this process as well,[14] and the illustrations contained herein will often be free form or based on other standards in order to communicate the desired content output of each process step. Finally, the treatment of capabilities within the supporting methods is consistent with the DoD process used in identifying, assessing, validating, and prioritizing joint military capability requirements.[15]

Collectively, for the purposes herein, the approach followed will be referred to as the *enterprise capability engineering* process, or simply ECE. It is asserted that proper application of the ECE process will benefit an enterprise by providing a means to effectively evaluate organic resources in terms of mission outcomes. It will do so by accomplishing the following high-level goal:

> *Provide an analytical assessment of the mission performance potential of the capability portfolio of an enterprise (real, planned or prospective) within the specified environment.*

An ECE process cycle is typically initiated when one of the following situations arises:

(1) a strategic development results in changes to a strategic plan;

(2) one or more strategic objectives are not being met due to shortcomings of current capabilities as identified, e.g., through a perceived need or actual operational failure;

(3) for "what if" purposes, such as, e.g., when a technology-push activity proposes use of a new product; or,

(4) when a new capability needs to be developed and deployed to meet a new mission.

ECE process inputs, to put it simply, include relevant strategic guidance for, and descriptions of, the enterprise in question. For those organizations that practice formal planning and quality management processes, it would be expected that most, if not all, of the necessary ECE input would be available at the start of the assessment. Where ad-hoc management processes are the norm, it is likely that a significant fraction—if not the majority—of the effort required to conduct the ECE would actually be devoted to

---

[14] For the interested reader, such other frameworks include MODAF, TOGAF, HLA, FEAF, GERAM, and Zachman, and modeling languages include DFD, IDEF, E-R diagrams.

[15] *Joint Capabilities Integration and Development System,* CJCSI 3170.01H, 10 January 2012.

developing and structuring the required input, and not on performing the ECE itself. ECE process outputs are based on need- or purpose-focused assessment objectives.

1.3 Objectives

ECE process output will depend, at least in part, on both the type and scope of the assessment it is being used for (e.g., is it intended to identify capability gaps alone, or to evaluate potential resource changes as well). An assessment conducted for situational case (1) in §1.2 above would likely focus on addressing existing capability performance in light of the new strategic plan (e.g., is the portfolio able to provide acceptable mission performance under the new conditions?). An assessment conducted for case (2), on the other hand, would at least be scoped to assess performance within the problem area(s), but may also include identification and evaluation of proposed solutions. A case (3) "new product" assessment would provide a capability performance comparison between, say, the existing baseline and the proposed change. Case (4) is conceptually different from the other cases in that a baseline does not exist, and thus it represents a "clean sheet" development activity that should, nevertheless, involve mission performance comparisons between multiple options.

To put it succinctly, the ECE process can be used to meet these differing needs by satisfying the following measurable, high-level technical objectives: [16]

- Identify the capabilities and associated operational performance criteria required to successfully execute specified missions.
- Identify the shortfalls in resources to deliver the required capabilities and the associated operational risks.
- Identify the possible solution space for the capability shortfalls.

1.4 Organization

The discussions that follow below are divided into two main sections. The first (§2) provides a review of several important contextual definitions, an understanding of which

---

[16] Adapted from *Joint Capabilities Integration and Development System,* CJCSI 3170.01F, §4c, 1 May 2007, p.2.

is required to fully decipher the material that follows. This is followed by a discussion of strategic planning and an introduction to the ECE process. The second main section (§3) delves into a detailed discussion of the ECE process and supporting methods.

2 ENTERPRISE BY INTENT

2.1 Definitions

Like any field of endeavor, an ECE has its own set of jargon or specialized terms (technobabble if you must). While some of these terms will be introduced in context later in this report, there are several, commonly-used but overloaded terms that represent important concepts related to an ECE for which particular (but not unique) definitions are given here below in order to avoid misunderstandings of the material that follows. Note that the order of presentation of the definitions follows more of a logical than alphabetic ordering scheme, in order to build on the concepts.

2.1.1 System

The word *system* (from Latin *systēma*, in turn from the Greek *σύστημα*) in its meaning as used herein has had a long history. The root of the word can be traced back to a verb (συνιστούν) used in ancient Greek that meant *constitute* (combine to form a whole). The root has been used by writers such as Plato (*Philebus*), Aristotle (*Politics*) and Euclid (*Elements*) as referring to concepts such as "total", "crowd" or "union."

However, there is something more to a system that just a combination of elements. As Aristotle is often quoted in saying: "... the whole is not, as it were, a mere heap, but the totality is something besides the parts ... ."[17] That is, it is the way in which properties and interactions of parts work together that produce system behavior. It is important to understand that the parts acting in isolation may not even provide or demonstrate the behavior that is observed at the system level. The way complex patterns can arise out of a multiplicity of relatively simple interactions is referred to as *emergence*. Note that such emerging behaviors are one reason why bottoms-up design approaches can fail.

A system is scoped by defining its boundary. Those entities inside the boundary are part of the system, while those outside are considered (treated) as part of the environment.

---

[17] Aristotle, *The Works of Aristotle*, *Volume VIII*, *Metaphysica*, Book H [eta or 8, VIII], Chapter VI, Smith, J.A., and W.D. Ross, ed., At the Clarendon Press, Oxford, 1908, p. 1045^{a}, lines 9-10. This translation follows the Greek text of W. Christ (*Areistotelis Metaphysica*, Leipzig, 1895, p. 178).

Most systems are "open" in that they exchange matter and energy with their environment (as opposed to "closed" or "isolated"); thus they can be understood to be a bounded transformation process, that is, a process or collection of processes that transforms inputs into outputs (inputs are consumed and outputs are produced). Of particular interest herein are man-made systems—in contrast to natural—that can be deliberately designed with some intended purpose in mind (input-to-output transformation), fulfillment of which is the primary source of its "returns" (e.g., profit, gain, revenue, or other rewards, tangible or intangible). See *Enterprise*.

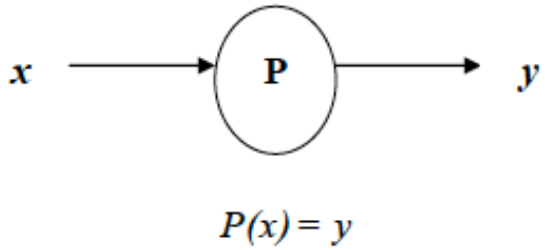


Figure 1. *System* represented as a function transforming inputs to outputs.

2.1.2 System of Systems

No scale or scope is implied by the concept of *system*; that is, systems occur at all scales and levels of complexity. To elaborate further, given a system, each and every one of the parts, members, elements or entities of which it is composed can be treated as a system in its own right (at least until you get down to the level of the simplest sub-atomic particle). That is, most systems are formed of other systems and, in this sense, can be thought of as forming a system of systems (SoS). However, it should be noted that, as commonly used, the term *system of systems* has a more restricted sense as a reference to interacting man-made systems that are operationally and managerially independent, or geographically dispersed.[18]

---

[18] DeLaurentis, D., "Understanding Transportation as a System of Systems Design Problem," *43rd AIAA Aerospace Sciences Meeting*, Reno, Nevada, January 10-13, 2005, AIAA-2005-0123. See also Sage, A.P., and C.D. Cuppan. "On the Systems Engineering and Management of Systems of Systems and Federations of Systems," *Information, Knowledge, Systems Management, Vol. 2*, No. 4, 2001, pp. 325-345.

2.1.3 Architecture

An *architecture* is the "fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, … ." (IEEE Std 1471 §3.5)

2.1.4 Architectural description

An *architectural description* is a collection of products (model[19]) intended to document a system, including, e.g., definitions of all components (e.g., subsystems), and their connectivity and interactions. Often widely, if imprecisely, referred to as the system architecture (aka systems architecture, system model, architectural framework).

2.1.5 Enterprise

Following after ISO 15704, an *enterprise* is defined to be "one or more organisations sharing a definite mission, goals, and objectives to offer an output such as a product or service."[20]

An enterprise is a class or specialized type of system in that it is more than just a collection of hardware or other inanimate objects; it is a system of which people form a part and who are working together collectively to meet a mission. An enterprise may also represent a system of systems. Each enterprise is formed from a collection of resource and capability portfolios (definitions follow below).

Enterprises are responsible for transforming inputs (consumable resources) into outputs (products or services) by virtue of their capabilities to successfully deploy available

---

[19] A general classification scheme for different model taxonomies would include the following three types (any or all of which may be used as appropriate to describe a system): (1) Iconic models—represent reality and look like the real thing, but may employ, e.g., a change in scale or materials; includes sketches or drawings, 3-D constructs, and virtual reality; (2) Analogue models—a simplification of reality (limited detail) focused on key elements that make no pretense of looking like the real thing; includes schematic models (graph theory), 2-D contour maps, and functional relationships displayed on graphs; (3) Symbolic models—an abstraction of reality that represents ideas by means of a code and is used for analyzing performance and predicting events; includes numbers and mathematical (deterministic and stochastic) models, words (verbal description), and musical notation.

[20] ISO 15704:2000, *Industrial automation systems —Requirements for enterprise-reference architectures and methodologies*, 1999-08-20, §3.6.

assets. These portfolios—especially those resources (assets) and capabilities that are deemed to be "core," "strategic," "key," "critical," "unique," or "distinctive"—should be managed to reflect the long-term or overall (strategic) aims and interests (viability) of the enterprise.

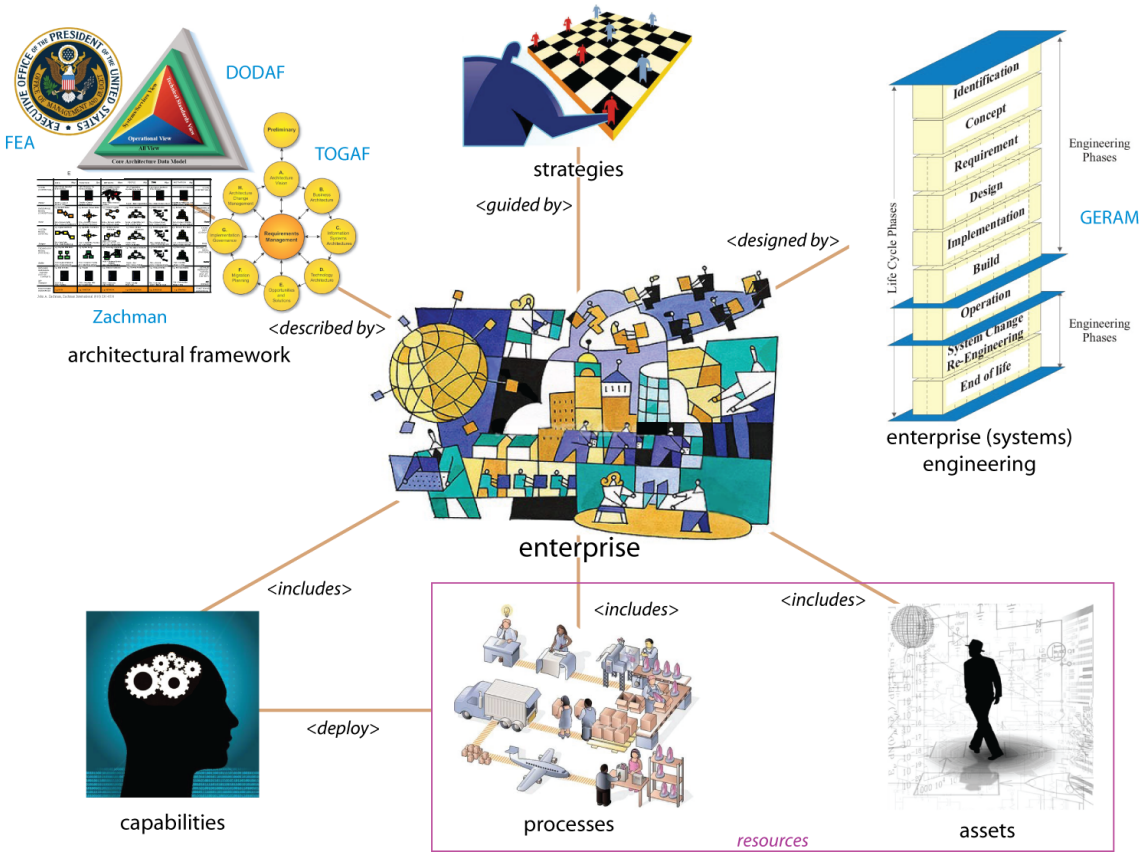


Figure 2. Enterprise.

2.1.6 Mission

That activity in which an enterprise engages to fulfill the customer product or service function for which it was established, as defined by major goals and performance objectives. See *Strategic Plan(s)*.

2.1.7 Resource

A stock or supply of, e.g., money, materials, or staff, that can be used or consumed by an enterprise in order to perform some operational function (produce an output) is referred to herein as a resource. Those resources that are part of the "value chain" of an enterprise

(i.e., involved in transforming inputs into outputs) are said to be assets; those that are not are overhead. Resources may be tangible or intangible. Tangible resources include financial assets, physical assets (e.g., specialized equipment, geographic location), processes (know-how that can be traded), certain types of intellectual property (e.g., patents), or any other assets whose value can be reflected on a balance sheet. Intangible resources could include certain types of information and knowledge, the right people (e.g., skills or expertise in engineering), culture and working relationships, and reputation. For purposes herein—and from a simplistic standpoint—resources can be divided into two broad categories: assets and processes.[21] Collectively, they represent a "bundle" of *potential* services; that is, resources are not productive in and of themselves (i.e., they are inert).

2.1.8 Capability

If an enterprise can successfully deploy (assemble, integrate, and manage) available resources to meet some end (e.g., perform some task or activity, produce some desired output), it is said to possess an organic capability in that regard.[22] In some sense, capabilities are rooted in *system-level* skills (vice individual) that are formed and held by the practices and routines of an enterprise that serve analogously as a kind of memory. Like individual skills, the capabilities of an enterprise can change—evolve, develop, or degrade—over time in terms of level of performance and in type, intentionally or not. That is, capabilities can be intentionally designed and developed (e.g., through deliberate, enterprise-specific investments) to meet the strategic needs of an enterprise in an ever-changing environment.

---

[21] As another example, in a DoD planning context, resources are categorized by one of seven types: doctrine, organization, training, materiel, leadership and education, personnel and facilities (DOTMLPF); some authors include a "-P" suffix to represent policy as distinct from doctrine.
[22] CJCSI 3170.01G, PART II, defines *capability* to be "The ability to achieve a desired effect under specified standards and conditions through combinations of means and ways across the doctrine, organization, training, materiel, leadership and education, personnel, and facilities (DOTMLPF) to perform a set of tasks to execute a specified course of action. It is defined by an operational user and expressed in broad operational terms …"

Capabilities can be categorized along operational or functional lines.[23] Operational capability taxonomies have a mission-centric view, with a focus on the mission tasks or activities to be performed (e.g., major theater war, nuclear war, special operations; cf. the list of examples provided in §3.2.1.3). In contrast, functional capability categories indicate how mission activities are enabled (e.g., force application or logistics; cf. the list of high-level examples provided in §3.2.2.1.3). Functional capabilities cross-cut mission space (e.g., a particular functional capability could be used to support a variety of missions). Required functional capabilities are derived from operational capabilities through mission scenario modeling. Functional capabilities also offer clearer boundaries for system allocation purposes.

If an enterprise is both successful and efficient in using a capability it holds, it is often referred to as a *competency*;[24] other frequently used synonyms for an enterprise capability include ability, capacity, potential, and power. In general it is desirable—and even necessary—to develop performance measures for a capability (e.g., in terms of efficiency or value added).

2.1.9 Strategic Plan(s)

An enterprise-level document (or set of documents) that outlines its overall purpose (vision), philosophy (values), and direction (near-, mid-, and long-term). "Direction" is concerned with the development and application of capabilities, and availability and use of resources; it is documented in a mission statement along with supporting, measurable and verifiable strategic goals and objectives.

---

[23] As captured, e.g., in the definition for *capable* found in MCRP 5-12C: "The ability to accomplish a mission, task, function, or subfunction."

[24] See, for example, Prahalad, C.K., and Gary Hamel, *The Core Competence of the Corporation*, 1990.

2.2 The "50,000 ft." View

If an enterprise is to be mission oriented, some form of explicit traceability is required to exist between mission needs and operations. It is asserted here that, if it is desired to have, a priori, any assurance at all in the validity of this link, it must be forged through the application of a top-down design strategy. The issue then becomes one of how to take a top-down approach.

While most discussions concerning top-down design center on the decomposition and subsequent specification of hardware or software systems, it would be a mistake to restrict such an approach to the product domain. The approach works equally as well when considering an enterprise—or a particular capability thereof—as a system that is built of various types of components (resources) that include material (hardware), but also "soft" wares or assets (personnel with particular skills—education & training—who are organized and lead to execute particular doctrine—plans, processes and procedures—in a particular place and time—facilities). Presumably these wares are being deployed to provide one or more capabilities to enable an enterprise to meet some end—the mission. Given measures of mission accomplishment and resource costs, it becomes possible to discuss the success and efficiency of a particular enterprise or capability, and so the deliberate design and development thereof.

The top-down design (or assessment), development and deployment process for the capabilities of an enterprise takes the following basic approach: the purpose of an enterprise is linked with strategic goals and objectives (mission) by strategic planners, which are linked to capability performance—and future development—needs by enterprise architects, which are linked, in turn, to resource development requirements. Resources are then developed as necessary, and are then integrated as planned with the enterprise system in order to enhance (or standup) a capability and so improve returns (in whatever way they are measured). Note that development requirements *may* call for a new material solution ("widget"), but they may also call for further development of "soft" wares as well. There is, perhaps, an unstated assumption behind all of this discussion: the enterprise is being managed (directed and controlled in ISO 9000 terms) with intentional

or deliberate intent to do well vis-à-vis the established measure(s) of performance over some specified period—perhaps into the foreseeable future. There is little or no place for top-down design in an ad-hoc enterprise. An example application of enterprise-level top-down design can be found within DoD as a set of interlocking Instructions and Directives, as illustrated in Figure 3; here, "Capabilities-based Assessment" fulfills the role of an ECE process.

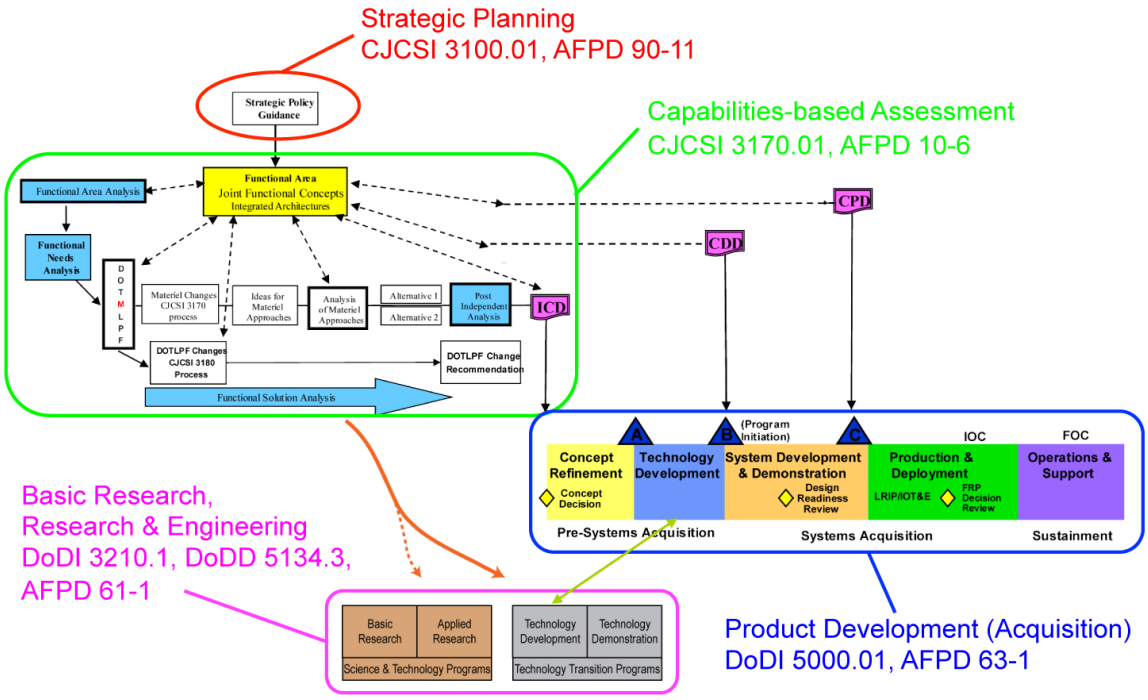


Figure 3. Top-down enterprise design in DoD.

For those unfamiliar with the workings of DoD, a simplified view of this process is provided in Figure 4 below.

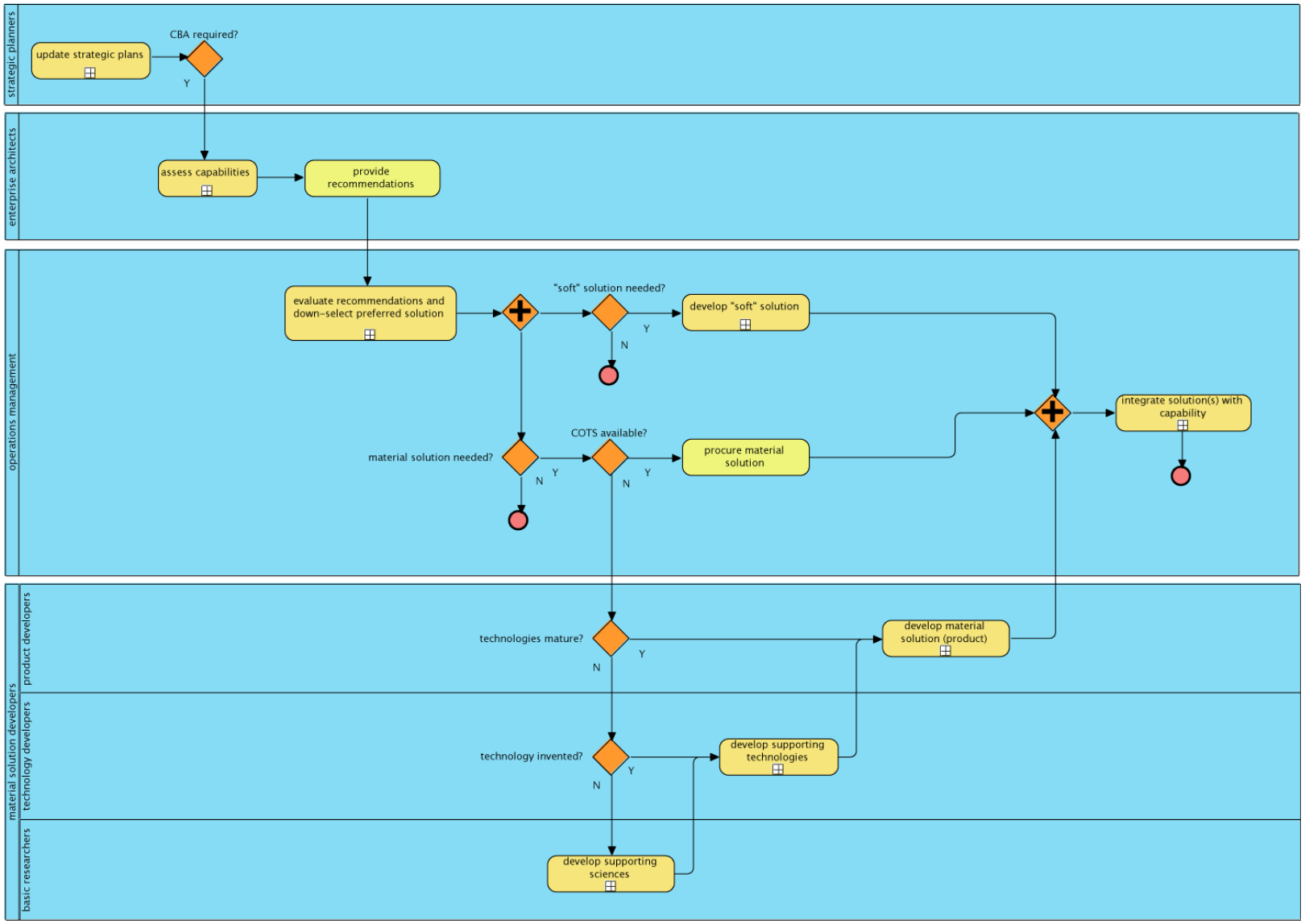


Figure 4. Linking capability development to strategic guidance.

Intentionally blank.

2.3 Strategic planning

Done right, strategic management of resource and capability portfolios necessitates strategic plans, which are predicated on some form of strategic planning process. The associated strategic planning analysis techniques or frameworks found in business go by various monikers such as SWOT (Strengths, Weaknesses, Opportunities, and Threats), PEST (Political, Economic, Social, and Technological), STEER (Socio-cultural, Technological, Economic, Ecological, and Regulatory factors), EPISTEL (Environment, Political, Informatic, Social, Technological, Economic and Legal), and OODA (Observe, Orient, Decide, and Act). In DoD interagency planning communities, the construct found most often is called PMESII (Political, Military, Economic, Social, Information, and Infrastructure), although others (like OODA) originated within DoD. All of these approaches are essentially similar, and begin with a basic underlying—if unstated—tenet: in the real world differing interests exist which tend to upset the status quo, i.e., alter the capabilities of an enterprise and so impact the desired return. A related axiom is that competition, as well as cooperation, among enterprises is allowable within some set of applicable—e.g., legal—boundaries. An illustration of this idea in a DoD context is provided in Figure 5.

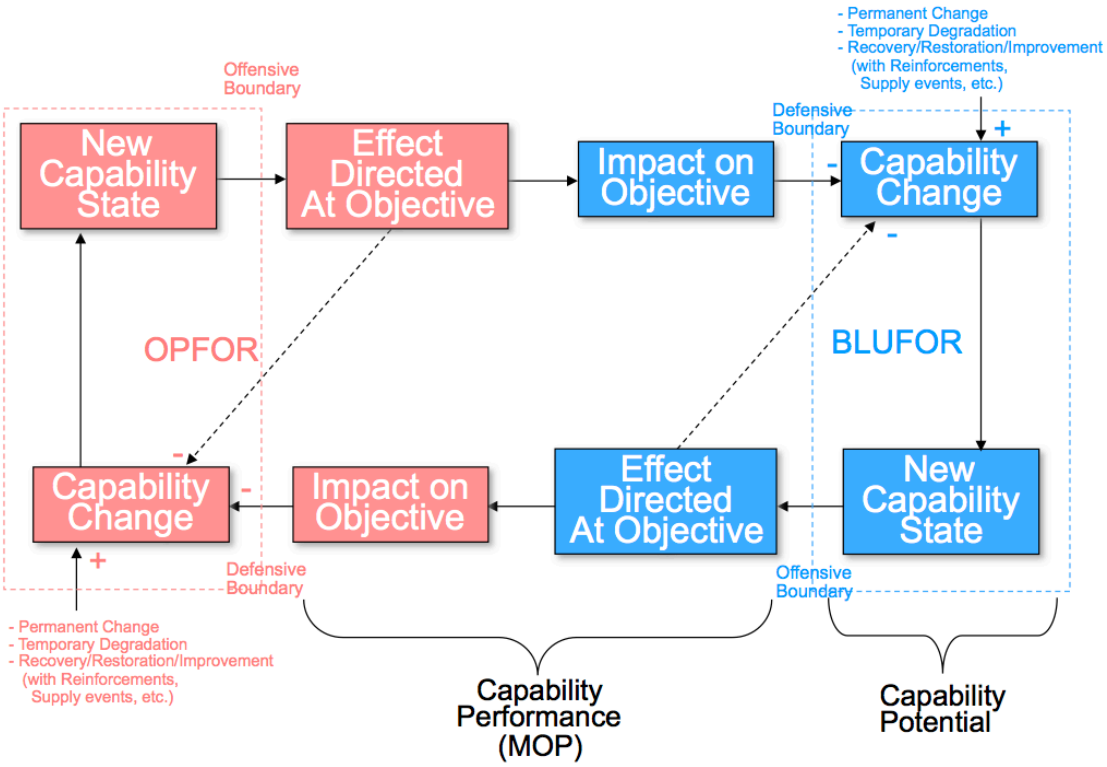


Figure 5. Capabilities-effects-impacts cycle.[25]

Although this figure is useful in developing a basic understanding of the strategic problem in the use and development of capabilities, it is too simple for strategic planning purposes (being more of a view of a tree and not the needed view of the forest). Strategic analysis generally involves an investigation into what can be described as a multiple system of systems problem. The multiplicity to be considered is established by the PEST, STEER, EPISTEL, OODA, or PMESII paradigm selected for use. Each system of systems, in turn, can be understood as forming a network of nodes that represent enterprises (or elements of enterprises, including tangible resources such as people, facilities, individual systems, forces, information, and material), and where the links represent the behavioral, physical, or functional relationships that exist between the nodes that enable them to function as a system (e.g., consumer-supplier relationships). Of course not all nodes and links in such networks are of equal importance to an enterprise,

[25] May, David, "Consolidated Operational Activities List (COAL) Taxonomy Proposal," presentation to J7 JCS, 12 March 04, slide 5, http://www.dtic.mil/futurejointwarfare/ideas_concepts/dm_coal.ppt last accessed 14SEP10.

which gives rise to the notion of a strategic "center of gravity" that is defined by the set of key or critical nodes and links that a plan should address, as illustrated in Figure 6. A particular instance of the model can be described as a "state."

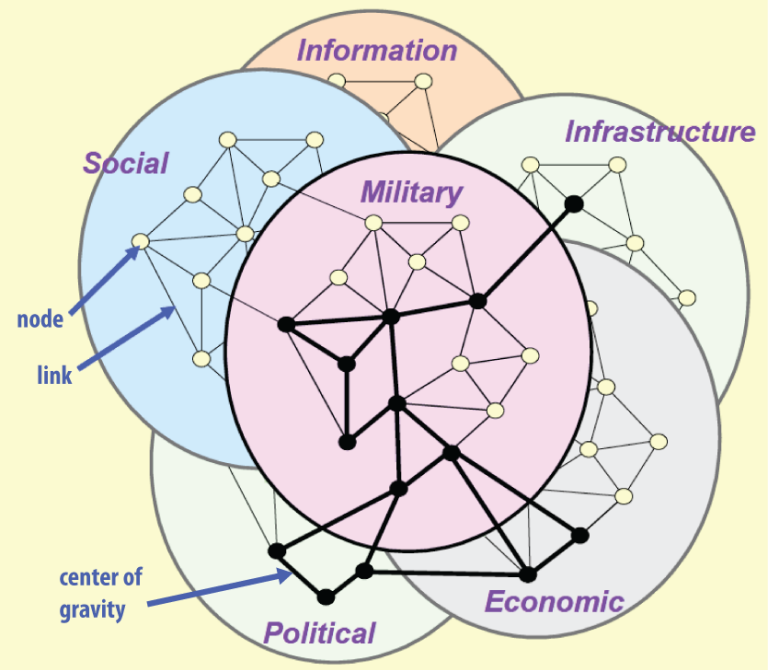


Figure 6. Envisioning a strategic environment.[26]

Strategic planning, then, attempts to endlessly adapt to the changing world—as modeled, e.g., by the networks described above—and identify achievable "end" states (really future states associated with some time scale) that represent the enterprise in a more desirable position vis-à-vis its competition, consistent with its core purpose and values (both of which should generally be fixed). An instance of a strategic plan can thus be viewed as a documented intent to change state from an existing or less-desired state to particular desired end state (and, of course, each update to the plan must be successfully executed to do any good for the enterprise), such as illustrated in Figure 7 below.[25]

[26] *Joint Intelligence, Joint Publication 2-0* (JP 2-0), 22 June 2007, p. IV-2.

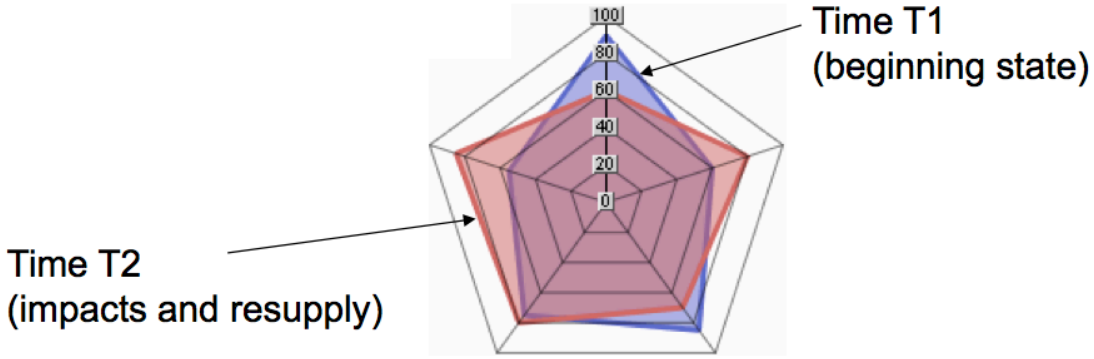


Figure 7. Envisioning the multi-dimensional and temporal nature of capabilities.

Intended changes to nodes, or to connections between nodes, that are associated with the desired state change are reflected in strategic goals and supporting objectives (G&O). Strategic plans should also identify: (1) the effects[27] that will (or are expected to) cause the desired state transitions to occur, as well as unintended or unwanted collateral effects that are to be avoided; and (2), the high-level capabilities that are expected to be deployed in order to produce said effects (this may call for application of existing resources, require further development of existing capabilities, or, conceptually, could include the establishment of an entirely new capability). Said another way, a strategic plan is intended to strengthen the capability portfolio of an enterprise in order to increase return, and (at least in a relative sense) weaken a competitor's capabilities that otherwise would have the potential to negatively impact said return.

[27] Categories of effects include: strategic, operational, or tactical; direct or indirect; desired or intended vs. collateral; 1st, 2nd, and 3rd order; systemic, functional, and physical; physical or psychological; Diplomatic (political), Informational, Material (e.g., military), and Economic, or *DIME*. Characteristics of effects include: decomposable, if required, to enable clear identification of specific changes that are required; traceable to higher order effects and eventually to a particular strategic objective and goal; measurable as described by an associated Measure of Effectiveness (MOE).

3 ASSESSING THE ENTERPRISE

If management is taking a proactive stance vis-à-vis its responsibilities for an enterprise, an assessment of the capability portfolio in terms of mission performance should be initiated under the following situations (these are, in effect, ECE process use cases):[28]

(1) a strategic development (e.g., a change in the environment—PMESII network—threatens the enterprise) results in changes to a strategic plan (a goal-driven investigation);

(2) one or more strategic objectives are not being met due to shortcomings of current capabilities as identified, e.g., through a perceived need or actual operational failure (a problem-driven investigation);

(3) for "what if" purposes, such as, e.g., when a technology-push activity proposes use of a new product (a solution-driven investigation); or,

(4) when a new capability needs to be developed and deployed to meet a new mission (a variant type of goal-driven investigation).

If management has stated its intent is to shift to proactive management of an enterprise, another ECE process use case can be identified:

(5) a baseline performance-based assessment has to be created "from scratch;" the process is one of evaluation research that is focused on describing implemented solutions, identifying impacts in terms of properties of the implemented solutions, identifying relevant stakeholder goals and associated metrics (criteria), and evaluating the impacts in terms of the criteria so established (i.e., an impact-driven investigation).

To be of practical use, such investigations must satisfy the following goal:

> *Provide an analytical assessment of the mission performance potential of the capability portfolio of an enterprise (real, planned or prospective) within the specified environment.*

This goal would be met by completing one or more of the following technical objectives that represent the solution of both "knowledge" and "practical" problems, depending upon the specific need (use case):

- *The capabilities and associated operational performance criteria required to successfully execute specified enterprise missions are identified prior to investigating real, perceived, or potential gaps.*

[28] Cf. Wieringa, Roel, "Design Science as Nested Problem Solving," DESRIST'09, May 7-8, Malvern, PA.

- *The shortfalls in resources needed to deliver the required enterprise capabilities are identified, along with the associated operational risks, prior to investigating potential solutions.*
- *The possible solution space for the enterprise capability shortfalls is identified prior to identifying a preferred option.*

In keeping with the technical management processes of, e.g., EIA-632, these technical objectives can be "wrapped" by various project and quality management objectives, such as these two high-level ones:

> *The enterprise capability assessment is successfully completed.*
>
> *The assessment results are used to initiate improvements to the enterprise capability portfolio vis-à-vis mission.*

However, as the intent of this paper is not to regurgitate best practices regarding project management (for which see, e.g., the PMBOK[29]) or quality management (cf. ISO 9000), for the purposes herein, these two high-level "wrappers" will be reduced in scope to a level commensurate with the direct interface to the ECE process technical objectives:

- *The capability-assessment is planned before technical work commences.*
- *Recommended solutions for enterprise capability shortfalls are provided prior to initiating changes.*

It should go without saying that the high-level goal and the combined set of technical- and project/quality management-objectives are intended to be measurable and, for a specific instantiation, measured, in order to provide assurance that the intended benefit is accrued to the enterprise. The objectives are achieved or realized through the satisfactory completion of the deliverables that the capability assessment (architecture study) team is expected to provide. Here "satisfactory" refers to a deliverable-specific measure or standard against which the deliverable is compared. Provision of a specific deliverable can also be tied to a particular process "mission" task and activity (function), as well as to the project resources (capability) required to perform the same.

---

[29] Cf. *A Guide to the Project Management Body of Knowledge (PMBOK Guide)*, ANSI/PMI 99-001-2008.

Such relationships have been illustrated in Figure 8 using a DoDAF *CV-1 Vision* diagram. In turn, the activities and associated information flows (data item deliverables) provide an outline of the ECE Process itself, and which can be portrayed, e.g., in a "waterfall" diagram, such as shown in the DoDAF *SvcV-4 Services Functionality Description* of Figure 9. This diagram has also been used to identify externally-sourced information flows that are required to support the process, and, for illustration purposes, a materiel acquisition function has been added as a "customer" for the final information product (deliverable). Suggested qualitative measures for the top-level data items are provided in Figure 10 as a DoDAF *DIV-1 Conceptual Data Model.*

The sub-processes and methods of the ECE Process as set forth herein are based on, among other things, the **principle of decomposition**: a complicated problem is recursively broken up into smaller problems until a tractable level is reached; available resources are then focused on systematically developing problem-by-problem solutions that are then integrated (composed) into a solution for the original, complicated problem. This approach has a long history in politics,[30] military art,[31] and scientific thought,[32] and was embedded early in current engineering design methods.[33] Formalization of modern design principles—including decomposition—in the 1960s was quickly embodied in standards of the day, such as with the issuance of MIL-STD-499, *System Engineering Management*, in 1969. Decomposition is reflected in current standards (all of which ultimately trace from this military standard), such as found in, e.g.: (1) the "Solution Definition Process" of EIA-632; (2) the "Detailed design stage" of IEEE Std 1220, where it is explicitly stated that the "project applies the SEP [systems engineering process] ... as many times as needed to decompose identified component functions into lower-level functions ... and design architectures;" and (3), the "Architectural Design Process" of

---

[30] E.g., the *divide et impera* (divide and rule) motto of Julius Cæsar in conquering Gaul (Holmes, Thomas Rice, *Caesar's Conquest of Gaul: an Historical Narrative*, Macmillan and Co., 1903, p. 79.).
[31] The *defeat in detail* principle widely propounded in military strategy and tactics references.
[32] E.g., that of the ancient Greek "philosophers" Leucippus, Democritus, and Epicurus.
[33] E.g., Alexander, Christopher, *Notes on the Synthesis of Form*, Harvard University Press, Cambridge, Mass., 1967, and Archer, Bruce, "The structure of the design process," in *Design Methods in Architecture*, Broadbent and Ward, eds., Lund Humphries, 1969, pp. 76-102.

ISO/IEC 15288 that states "[t]his process encapsulates and defines areas of solution expressed as a set of separate problems of manageable, conceptual and, ultimately, realizable proportions." In other words, the conduct of a dynamic- or performance-based assessment[34] at the enterprise-capability level, as defined by the ECE Process described herein, follows a sequence that aligns directly with a basic systems engineering (SE) process, [35] although at times the terminology used may mask this fact. The decomposition artifacts produced by the ECE Process are illustrated in Figure 11.

It should be noted that, when taken to the logical conclusion of specifying implementable changes at the resource level (e.g., people or materiel), the application of the ECE process—along with the associated predecessor and successor processes—to complex enterprise or system-of-systems strategic problems generally produces design artifacts that can be described as an integrated, multi-tier (or -level), hierarchical architecture (or inverted "tree"). Strategic planning drives capability portfolio development by producing a hierarchy of data products stemming from the enterprise strategic vision that include goals, objectives, effects and missions. Enterprise (or system-of-systems) capability development begins by considering a particular mission for which it defines operational capabilities, services (logical elements), and capability configurations (including, e.g., specifications of the desired materiel—hardware and software systems—solutions) that would allow its successful execution.[36] Finally, the acquisition process will develop a given system specification by defining and detailing a system breakdown structure that can include products, subsystems, assemblies, components, subassemblies, and subcomponents, as well as software objects.[37] During development, each tier within the strategic plans, capability architectures, and system architectures, contains multiple

---

[34] Other types of portfolio management tools (e.g., catalogues or capabilities-tasks-functions-systems mappings) exist that can be used to perform other types of assessments (e.g., identify possible redundancies and gaps in system coverage of desired capabilities). Such tools do not, however, model system behavior per se. For example, see Marlow, Kevin, et al., "Practical Applications of the Capability Mapping Framework," 77th MORS Symposium, 15 June 2009.
[35] Cf. Levis, Alexander H., and Lee W. Wagenhals, "C4ISR Architectures: I. Developing a Process for C4ISR Architecture Design," *Syst Eng 3*, John Wiley & Sons, 2000, pp. 225-247.
[36] The definition of complex system-of-systems may actually require repeated decomposition at the capability or service levels rather than just the single pass through the ECE process invoked here.
[37] This is the generic system breakdown structure (SBS) of IEEE Std 1220 §4.10.

options for implementing the nodes in the next-higher tier from which it stems. Collectively the set of options describes the identified solution space (which is likely only a subset of the possible solution space). Best engineering practices will delay "pruning" of this tree (via trade-off studies) until as late as reasonably possible in the processes involved in order to provide a broader range of designs.

Intentionally blank.

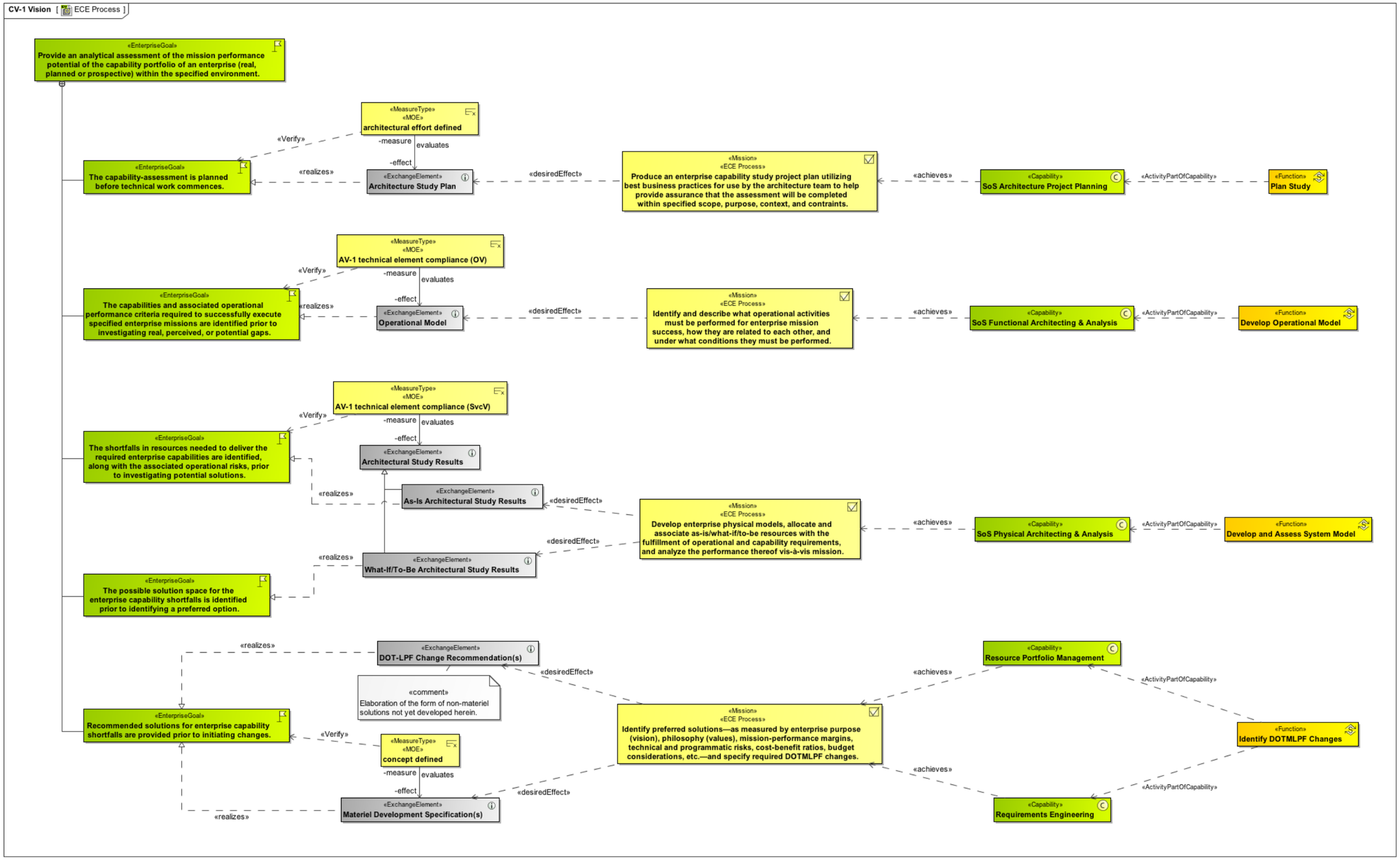


Figure 8. ECE Process high-level goal, objectives, deliverables, measures, mission tasks, activities (functions), and capabilities.

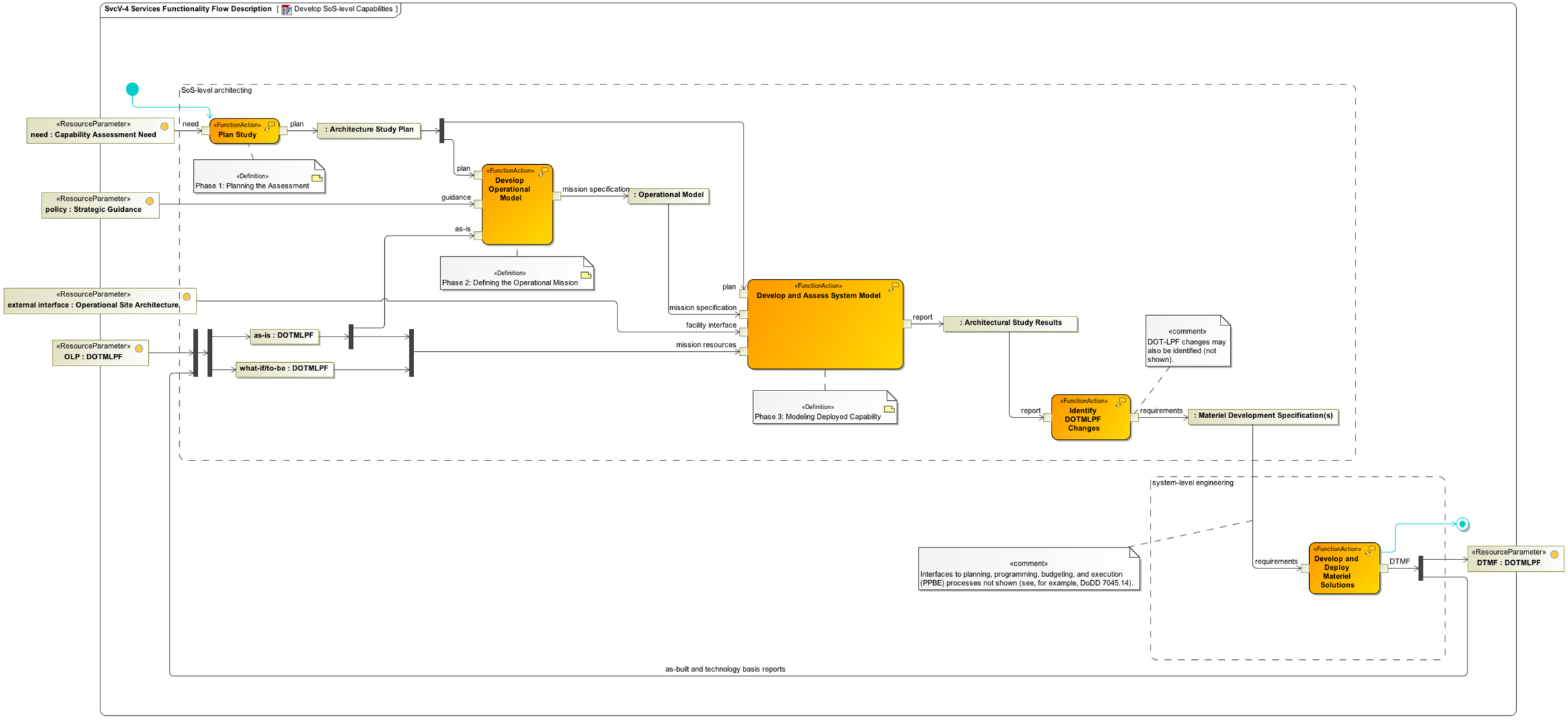


Figure 9. Enterprise Capability Engineering (ECE) Process diagram.

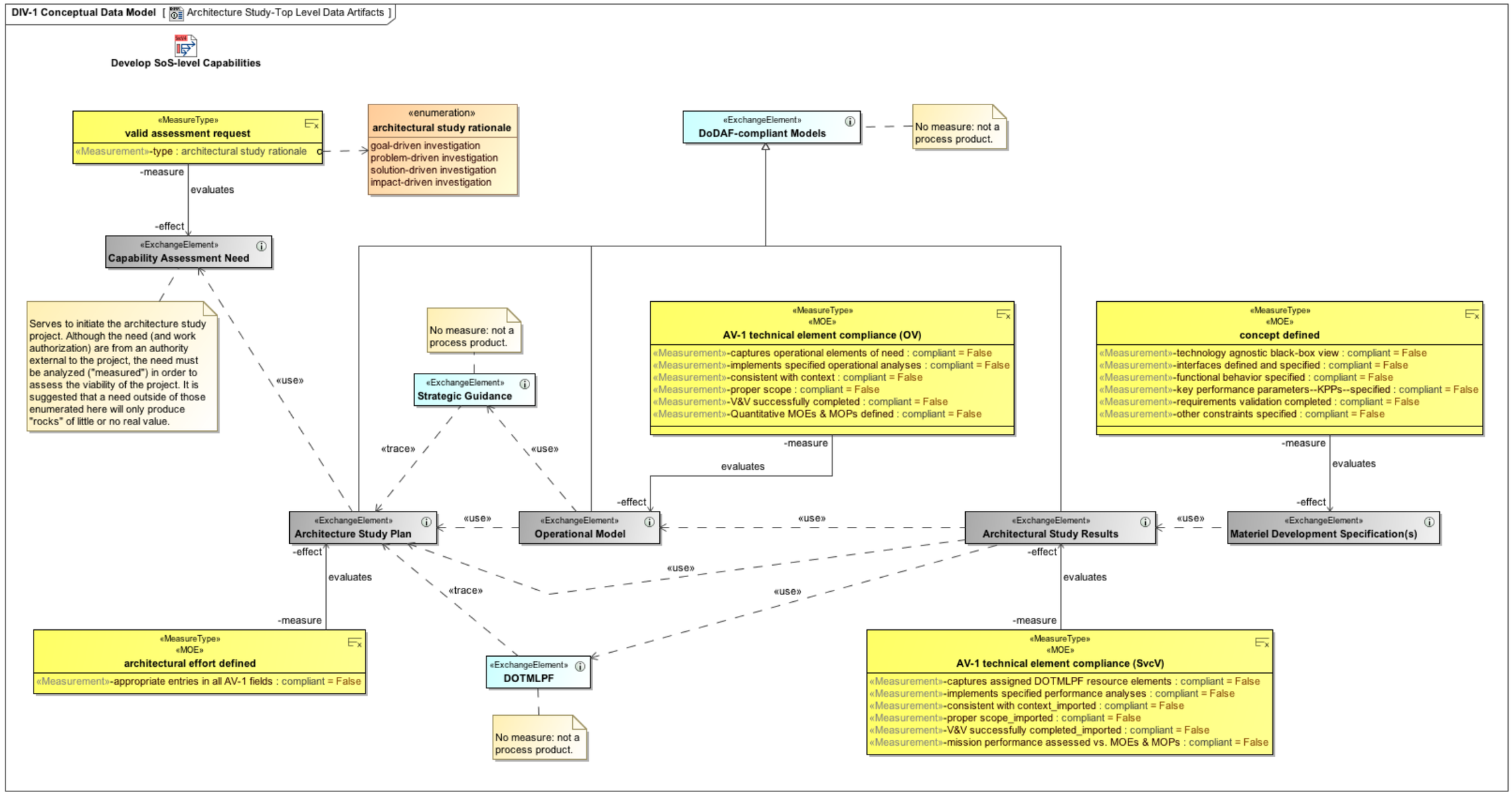


Figure 10. Top-level ECE Process data artifacts (deliverables) and some suggested measures.

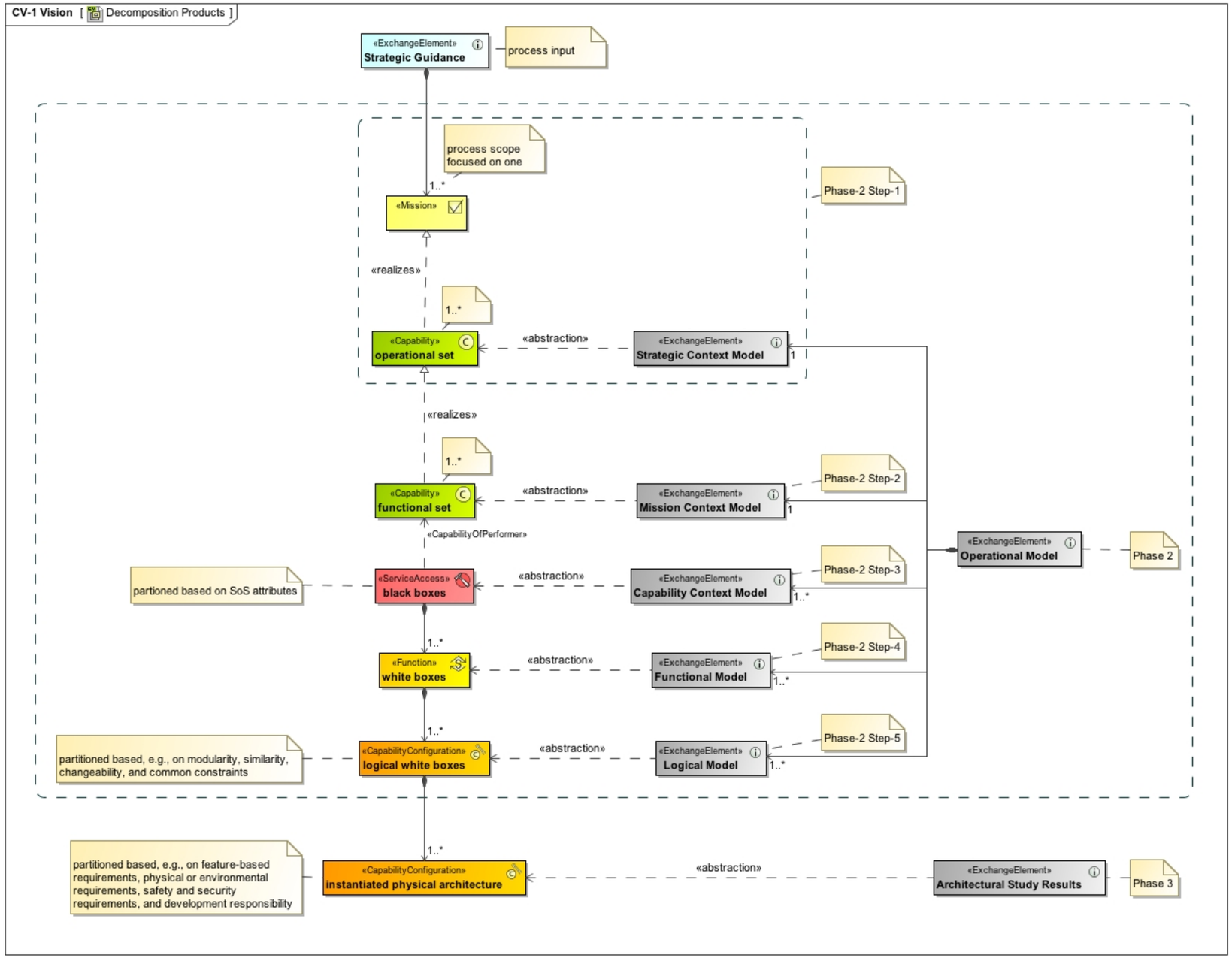


Figure 11. Illustration mapping the mission-to-technology problem decomposition sequence followed by the ECE Process.

3.1 ECE Process Phase-1. Planning the Assessment

In keeping with project management (PM) best practices, an ECE process cycle should begin by developing an architecture study plan that captures the appropriate PM initiating and planning process artifacts, including, e.g., the content found in a project charter (e.g., high-level objectives, requirements, schedule and budget) and PM plan (PMP, e.g., scope), and, not to forget, the appropriate quality management artifacts like management and technical reviews. From a DoDAF-compliant, models-based architecture perspective, such information is conveyed to the study team via an *AV-1 Overview and Summary Information* report. It is important to note, however, that an *AV-1* is used not only to initiate an architectural study, it is intended to provide a view to external stakeholders of the current state of a study; therefore, provision for frequent update is required throughout execution of a project. Another item of note is that it is expected that—prior to the start of an architectural study, a priori—information such as the models, analyses, analysts, and modeling tools to be used is a required entry in an *AV-1*. This often necessitates that a very limited scope, quick-look (pilot) study be performed in order to refine the study plan (e.g., to identify what type of analyses are likely to produce useful results). It should be understood that production of an *AV-1* represents a minimalist, DoDAF-compliant approach to planning an architectural study project. Consideration should also be given to developing a complete *Systems Engineering Management Plan* (SEMP, aka *Systems Engineering Plan*, or SEP), and even a PMP, in a models-based framework like DoDAF[38] (although that is beyond the scope of the present paper).

A simple process model of Phase-1 activities from the perspective of the architecture team is provided in Figure 12. The high-level content of an *AV-1*—principally a text-based product—is given in the *DIV-1* of Figure 13, along with optional DoDAF diagrams that may provide necessary, *graphical* information important for conveying the intent of

[38] For example, documenting planning information in DoDAF diagrams that conform to the *DIV-1* (for deliverables), *OV-2* (for information flows between performers), *OV-4* (for organization charts, required skills, and personnel assignments), *SvcV-10b* (for project processes), and *SvcV-4* (for task definitions and allocations to performers) viewpoint models as defined elsewhere in this paper.

the plan. Further data element definition of *AV-1* content is provided in Figure 14, a *DIV-2 Logical Data Model*, and in Table 1, an *AV-2 Integrated Dictionary* that provides a text definition for each data element of the *AV-1* as detailed in the *DIV-2*.

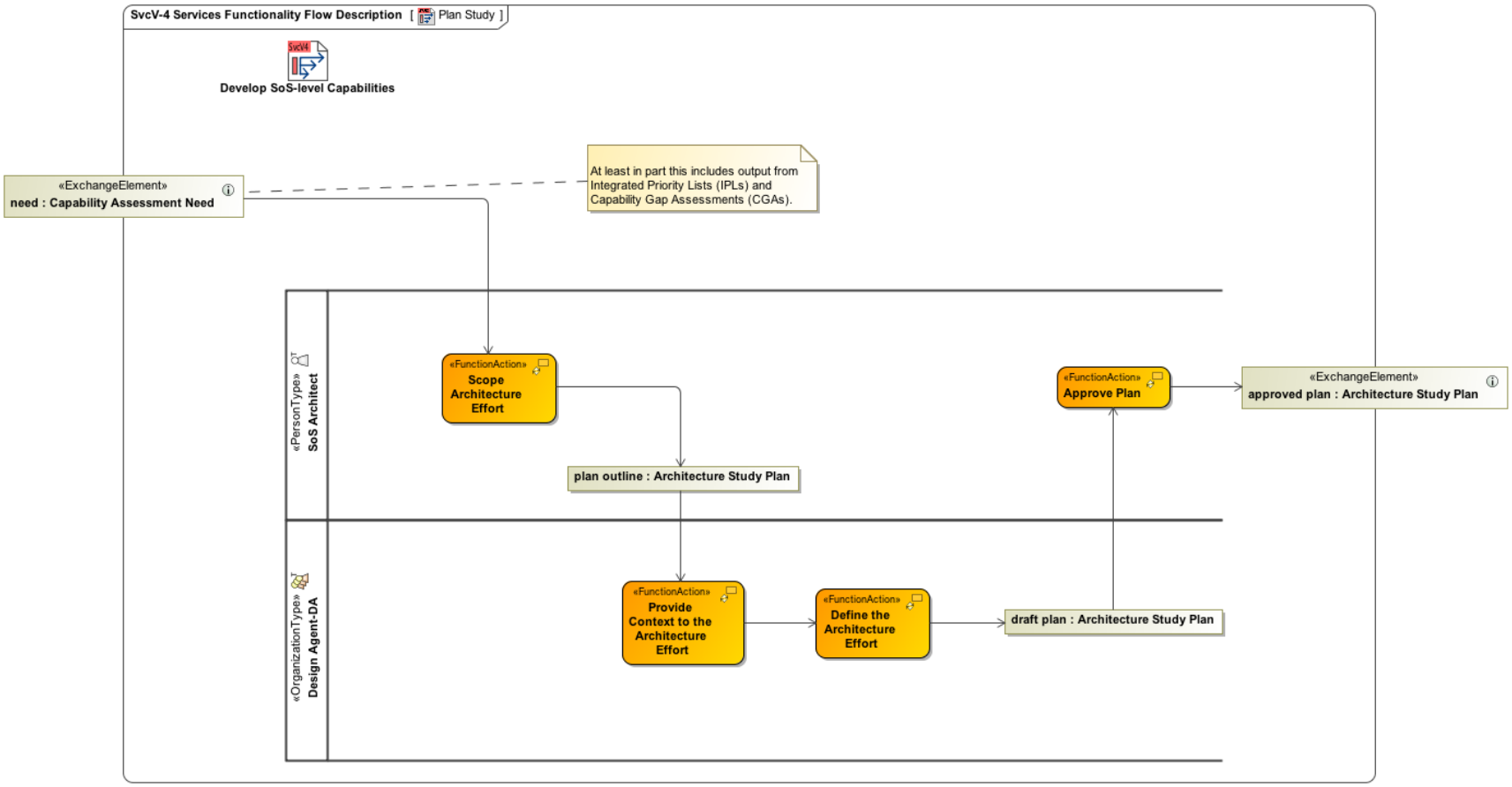


Figure 12. A notional process for developing an approved Architecture Study Plan.

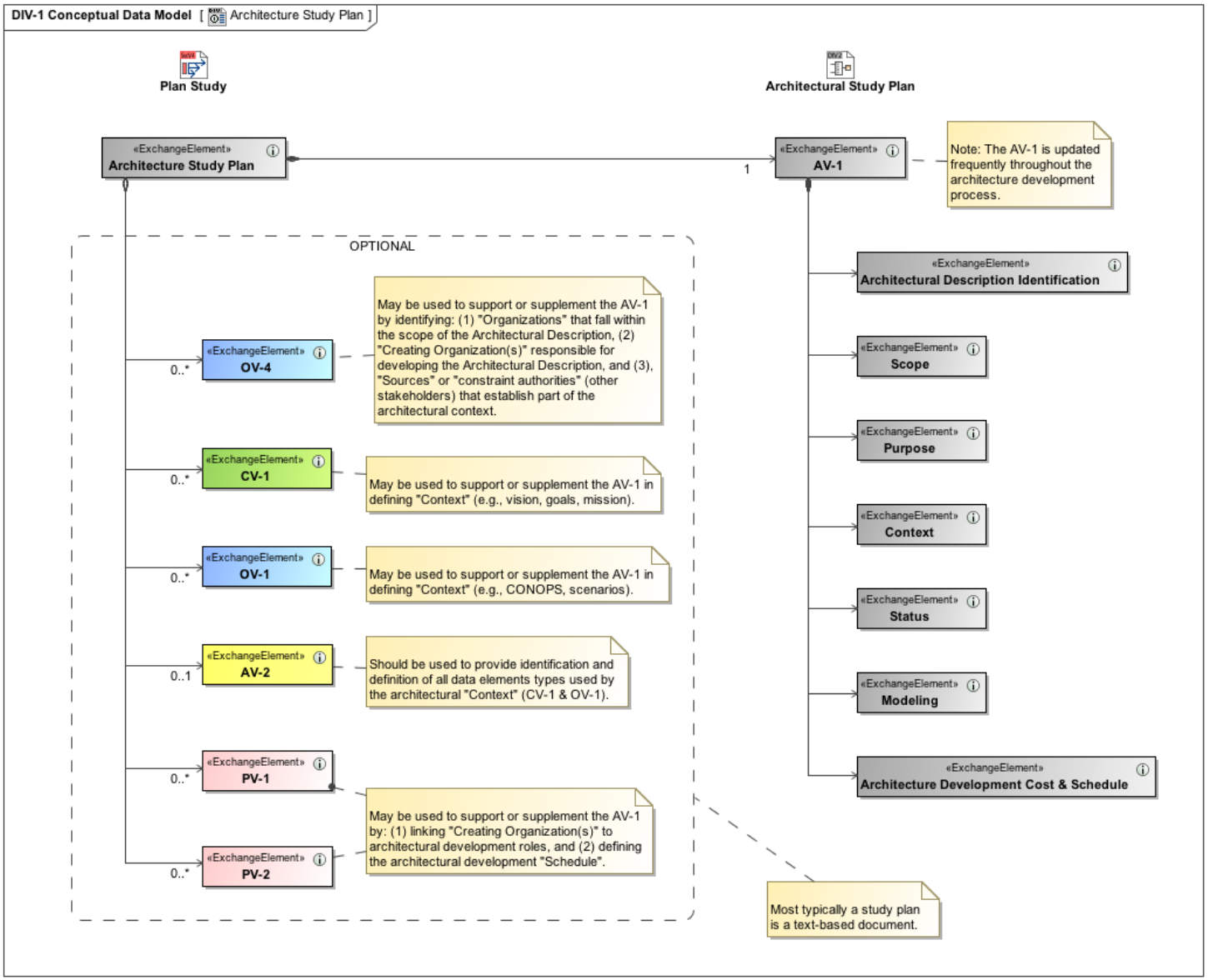


Figure 13. Required and optional data elements in an Architectural Study Plan.

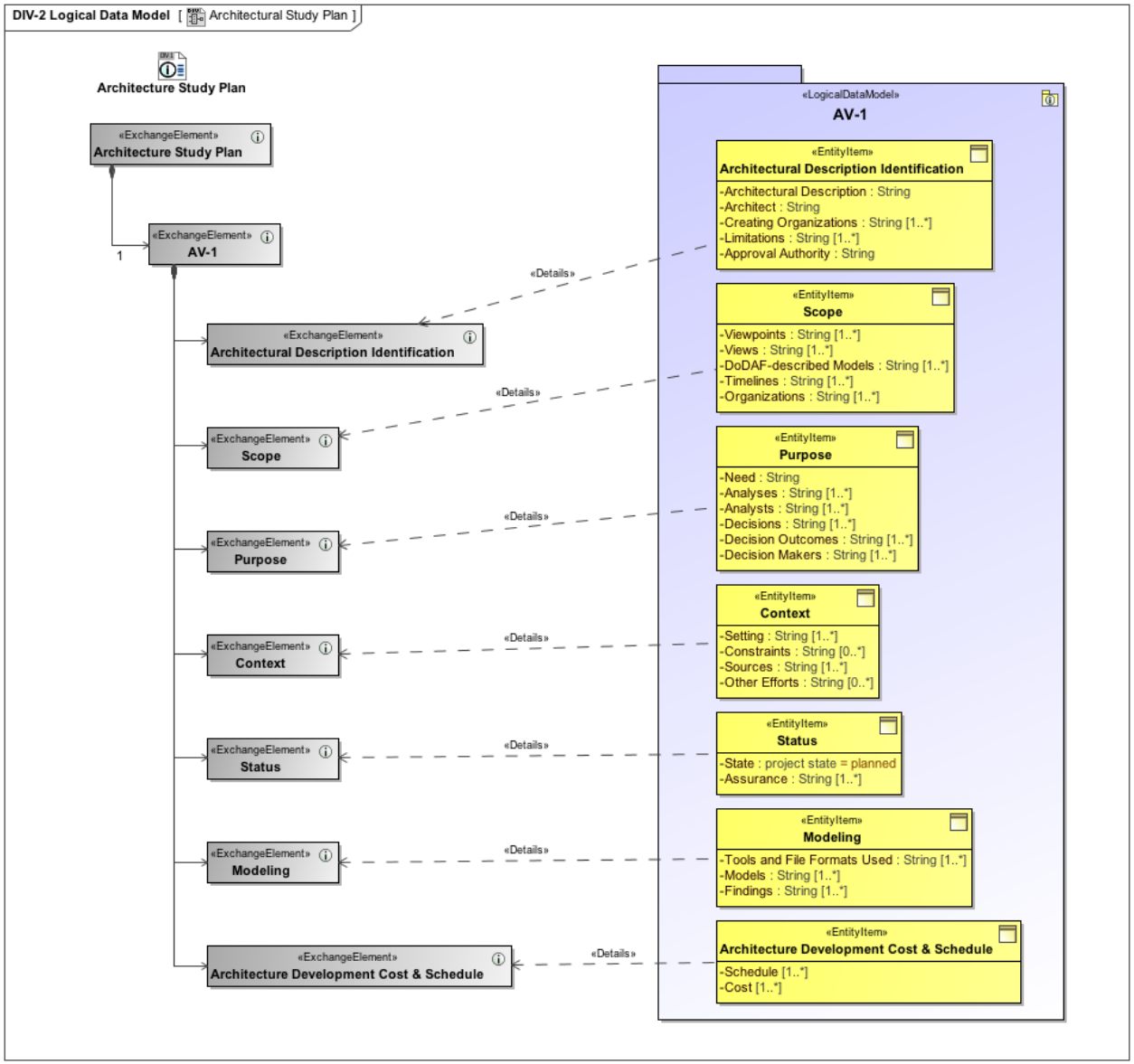


Figure 14. Detailed element descriptions of an *AV-1* report.

Intentionally blank.

Table 1. *AV-2 Integrated Dictionary* of *AV-1* Data Elements[39]

Identifier	Name	Definition
ADI 1	Architectural Description	The project name of the architectural description effort.
ADI 2	Architect	The name of the architect.
ADI 3	Creating Organizations	The name(s) of the organizations creating the architectural description.
ADI 4	Limitations	Assumptions and constraints on the architectural description effort.
ADI 5	Approval Authority	The approving authority for the architectural description.
Scope 1	Viewpoints	Viewpoints that have been selected and developed.
Scope 2	Views	Views (including Fit-for-Purpose) that have been selected and developed.
Scope 3	DoDAF-described Models	DoDAF-described Models that have been selected and developed.
Scope 4	Timelines	The temporal nature of the Architectural Description, such as the time frame covered, whether by specific years or by designations such as "current", "target", or transitional [or long range, mid term, or near term].
Scope 5	Organizations	The organizational entities and [their] timelines that fall within the scope of the Architectural Description. ["Users" in an IEEE Std 1471 sense.]
Purpose 1	Need	The need for the Architectural Description, what it will demonstrate.
Purpose 2	Analyses	The types of analyses that will be applied to the Architectural Description.
Purpose 3	Analysts	Who is expected to perform the analyses on the Architectural Description.
Purpose 4	Decisions	What decisions are expected to be made based on each form of analysis.
Purpose 5	Decision Outcomes	What actions are expected to result from decisions made on the basis of each form of analysis.
Purpose 6	Decision Makers	Who is expected to make decisions based on each form of analysis.
Context 1	Setting	Mission, doctrine, relevant goals and vision statements, concepts of operation, scenarios. These are executive-level summary statements with traceability to authoritative sources.
Context 2	Constraints	Information assurance context (e.g., types of system or service data to be protected, such as classified or sensitive but unclassified, and expected information threat environment), other threats and environmental conditions, and geographical areas addressed, where applicable. Also use to capture any constraints or expectations concerning the DOTMLPF level to be modeled as determined by the level of the problem being studied (tactical, operational, or strategic; e.g., soldier, squad, platoon, company, regiment, battalion or army level, or ship, task force, or fleet level). These are executive-level summary statements with traceability to authoritative sources.
Context 3	Sources	Authoritative sources for the standards, rules, criteria, and conventions that are used in the architecture.
Context 4	Other Efforts	Linkages to parallel architecture efforts.
Status 1	State	The status of the architectural description development effort [the creation status such as planned, in work, or complete] at the time of publication or development of the AV-1.
Status 2	Assurance	Record of architectural validation or assurance activities.
Modeling 1	Tools and File Formats Used	Identifies the tool suite used to develop the architectural description and the file names and formats used for the architectural models.
Modeling 2	Models	A list of each individual model and brief commentary including, e.g., methods applied in their creation, rationale used for grouping models, viewing assumptions, community of practice, and focus of work.
Modeling 3	Findings	The findings and recommendations that have been developed based on the architectural effort, such as identification of shortfalls, recommended system implementations, and opportunities for technology insertion.
C&S 1	Schedule	The architectural description development schedule including start date, development milestones, date completed, and other key dates.
C&S 2	Cost	The architectural description development budget, cost projections, or actual costs that have been incurred in developing the architectural description or supporting analyses.

[39] Adapted from DoDAF 2.0 vol. 2 §3.1.1.2.1

Intentionally blank.

3.2 ECE Process Phase-2. Defining the Operational Mission

The second phase of the ECE Process is focused on defining (bounding, scoping) the problem to be studied (capability to be assessed), and describing it from an operational perspective. Broadly speaking, this involves the disciplines of requirements engineering and architecting as applied to the development of analogue and symbolic mission models as derived from specified strategic guidance within the defined project scope. The content of the *Operational Model* so developed is comprised of information in the following categories: strategic, mission, and capability context models, and functional and logical architectures. The engineering activities that must be completed in order to produce these data items can be described with a simple process model, as in Figure 16 below.

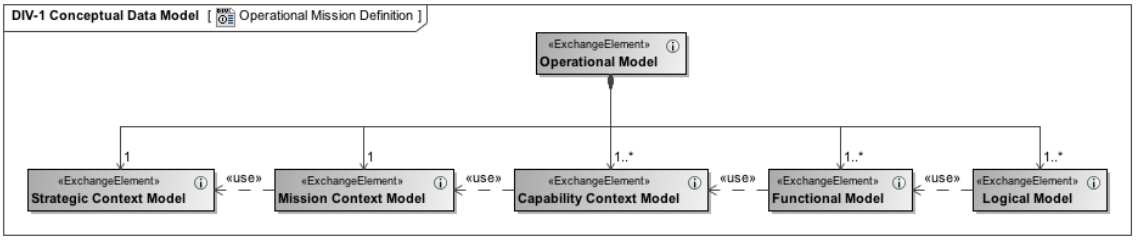


Figure 15. ECE Process Phase-2 conceptual data model.

Table 2. *AV-2 Integrated Dictionary* of Phase-2 Process Artifacts

Name	Definition
Strategic Context Model	Integrated set of enterprise-level stakeholder needs (goals and objectives), and other relevant concerns, requirements, and constraints.
Mission Context Model	Self-consistent set of mission-level requirements, possible courses of action, and performance evaluation plan.
Capability Context Model	Set of mission-critical capability specifications.
Functional Model	Set of mission-critical capability functional architectures.
Logical Model	Set of mission-critical capability logical architectures.

Further description of the methods and other considerations for developing these data follows in the subsections below.

Intentionally blank.

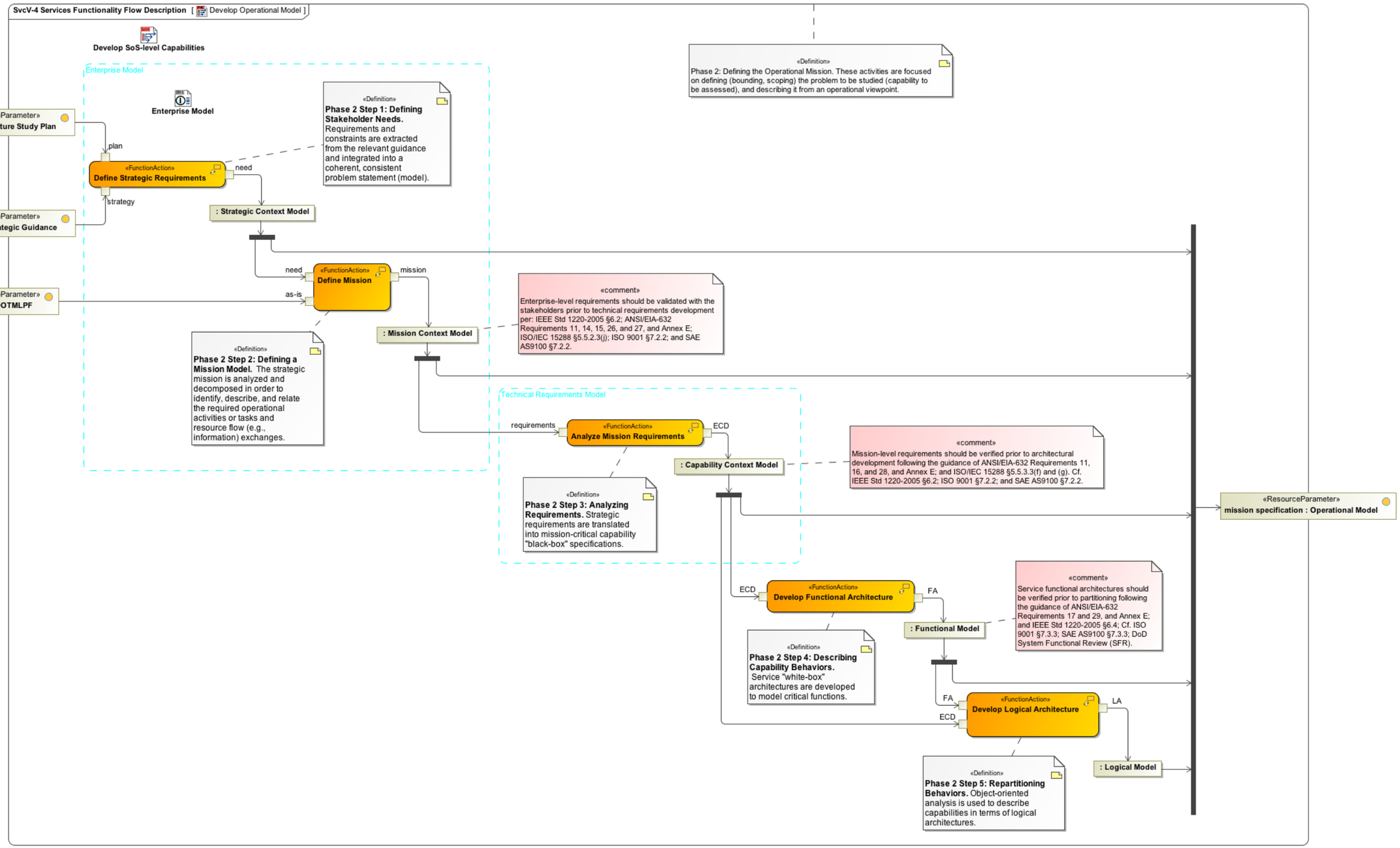


Figure 16. ECE Process Phase-2 steps.

Intentionally blank.

3.2.1 Defining Stakeholder Needs (Phase-2 Step-1)

This step is concerned with describing the strategic context behind a mission, and defining the information to be produced by the architecting effort necessary to satisfy mission stakeholders. First, a preliminary contextual analysis of the strategic[40] problem is performed on the basis of specified guidance. Requirements and constraints are extracted and integrated into a coherent, consistent problem statement (model), the required mission-level capability is defined, and high-level operational depictions are developed. Stakeholder concerns are identified and documented, *viewpoints* are selected to provide coverage of all said concerns, and one *view* is defined for every *viewpoint*.

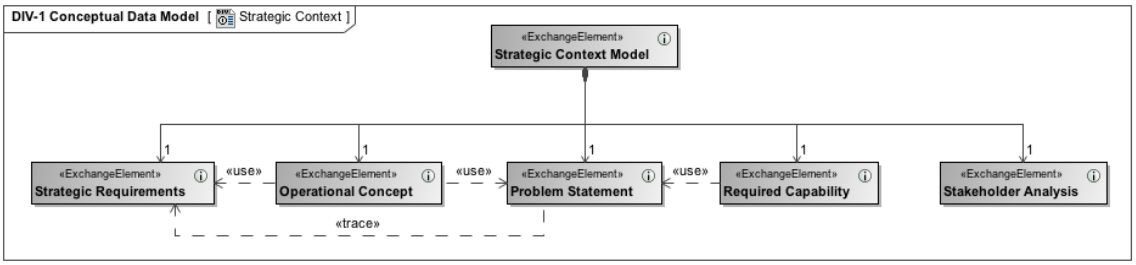


Figure 17. ECE Process Phase-2 Step-1 conceptual data model.

Table 3. *AV-2 Integrated Dictionary* of Phase-2 Step-1 Data Elements

Name	Definition
Strategic Requirements	Identification of applicable standards, rules, criteria, and conventions by authoritative source "chapter and verse." Represents a detailed "requirements" specification as derived from the context sources defined within the Architecture Study Plan.
Problem Statement	Clear identification of the specific effects to be achieved, including desired outcomes and measurable benefits.
Required Capability	High-level verbal description of the operational means by which the specified effects will be provided.
Operational Concept	Graphical depiction of the main operational context.
Stakeholder Analysis	Classifies stakeholders, lists their concerns, identifies corresponding viewpoints, and defines the views by which the system will be represented.

[40] Here *strategic* in the sense of relating to the overall technical aims and interests of the program or project, and not to a particular DoD strategic planning domain (i.e., National Security Strategy, National Defense Strategy, National Military Strategy, or Joint Operations Concepts).

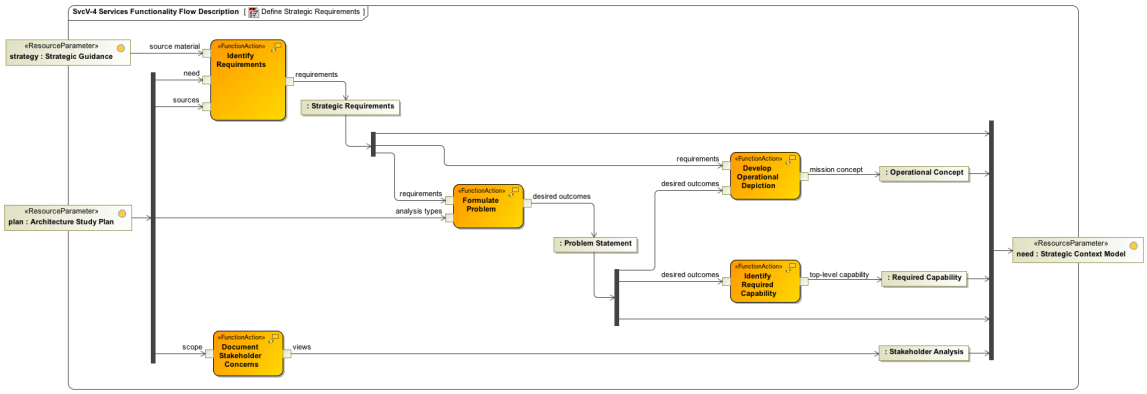


Figure 18. ECE Process Phase-2 Step-1 detail.

3.2.1.1 Identifying Requirements

Based on the list of authoritative *sources* for the standards, rules, criteria, and conventions that define the problem context, as documented in the *Architecture Study Plan* (*AV-1*), relevant strategic guidance is identified and collected for reference and use (the assumption being that such guidance exists—the production thereof being beyond the scope of this "white" paper; see Appendix A for a general discussion on requirements gathering and documentation). Of particular interest is that guidance that defines the desired effect (result, outcome, or consequence of an action or activity), including, e.g., strategic vision, goals and objectives, doctrine, mission statements and measures of effectiveness, concept of operations, scenarios, and desired system attributes. Other relevant information may include constraints such as policies, procedures, standards or rules that apply to the operational or business context of the problem. For later requirements traceability purposes, it must be verified that all such strategic guidance sources are documented. In general, as the *AV-1* is not sufficient or appropriate for this purpose (it is intended to serve as an executive summary), consideration should be given to the use of a DoDAF *StdV-1 Standards Profile* (in SysML consider a requirements diagram (*req*), or a *class* diagram in UML); for example, it is often the case that the stated purpose of the study is narrower (e.g., as defined by context "setting" and "constraints") than the strategic guidance specified (context "sources"), and so the *StdV-1* can be used to identify the applicable "chapter and verse".

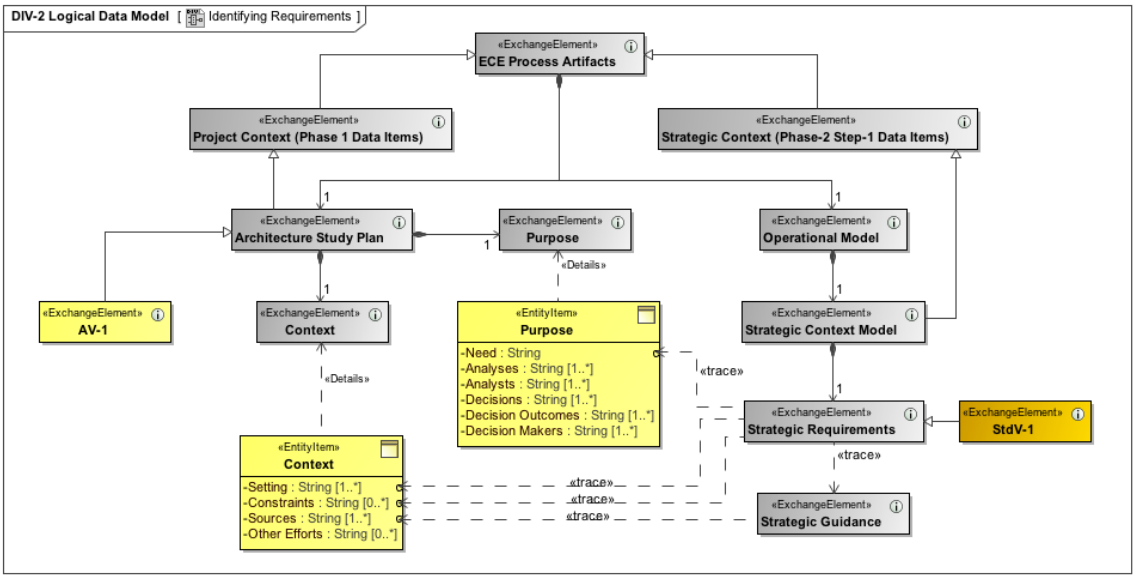


Figure 19. *Identifying requirements* DoDAF data model.

3.2.1.2 Formulating the Problem Statement

A concise *Problem Statement* is formulated from the vision and strategy (high-level goals and objectives) to provide clear identification of the specific changes that are required (cf. Appendix B, and the traceability example in Figure 90 of Appendix E). Generally this requires developing an understanding of the *effects*[41] that will produce the desired end state; of key importance is the identification of the desired outcomes and measurable benefits[42] associated with these effects,[43] which define the required validation objectives.

Figure 20. The relationship between effects and state.

Within DoDAF, a problem statement is captured in a *CV-1* diagram, such as illustrated in Figure 23 (represented by use cases (*uc*) in UML or SysML; alternatively, in SysML a *block definition diagram* (*bdd*) or *req* could be used, or a *class* diagram in UML). The availability of additional details may warrant supplemental use of an *OV-6b State*

[41] Effect—a change in a strategic environment network node (person, place, or physical object) or link (behavioral, physical, or functional relationship).
[42] Strictly, a *measure* is an operation for assigning a number to something. When a measure is applied, the number obtained is a *measurement*. A *metric* is an interpretation of the assigned number.
[43] The required performance of each effect is described by a Measure of Effectiveness (MOE).

Transition Description (to define operational "performer" states and transition rules), and possibly with a supporting *OV-5a Operational Activity Decomposition Tree* (to display operational states for later use in capturing the "desired effect" linkages to required capabilities). (In UML or SysML, this supplemental information would be captured in state machine (*stm*), *bdd*, or *class* diagrams.)

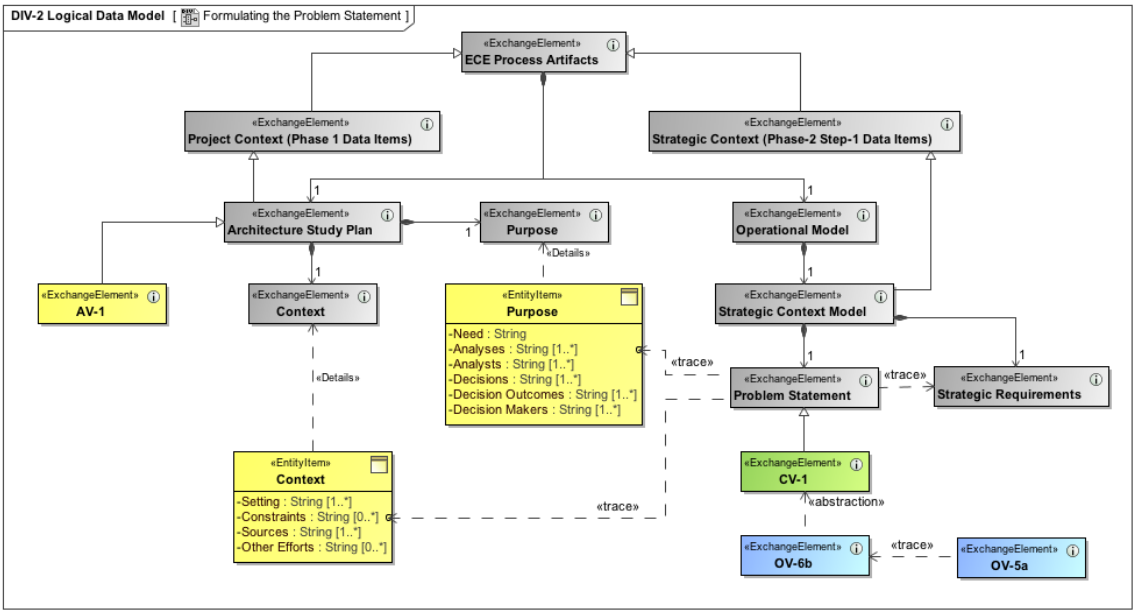


Figure 21. *Formulating the problem statement* DoDAF data model.

Because of its importance in capturing mission intent, a digression on *effects* is called for. First, note that effects have traits such as decomposability and traceability, and come in different "flavors"; for example:

- strategic,[44] operational, or tactical
- direct or indirect
- desired or intended vs. collateral
- 1st, 2nd, and 3rd order
- systemic, functional, and physical
- physical or psychological
- Diplomatic (political), Informational (or Information), Materiel (or Military), and Economic; DIME

[44] Note that the term *strategic* is overloaded. The use here is related to long-term or overall aims and interests of an enterprise (government, military, business). Elsewhere in this paper it most often relates to the overall technical aims and interests of the program or project of which this study is part.

Second, it should also be noted that—for purposes of the ECE process—the problem description should be stated in a way that is consistent with the specified analysis methods to be used and at a level that corresponds to DOTMLPF resource constraint descriptions provided (e.g., as defined in an *AV-1*). It is likely that some problem decomposition may be required (see §3.2.2.1.1), depending upon the strategic guidance available. Traditionally a statement of the strategic problem to be solved was based primarily on textual descriptions. For a models-based approach, a graphical model is required, as in the examples that follow below.

Table 4. Strategic Problem Statement Textual Examples

STRATEGIC OBJECTIVE	Indirect, 3rd Order, Systemic Effect(s)	Indirect, 2nd Order, Functional Effect(s)	Direct, 1st Order, Physical Effect
Reduce adversary's capacity to make and sustain war at the front.	Movement of military logistics are delayed in reaching front.	Road traffic halted at "choke" points; traffic diverted.	Road bridge rendered unusable to vehicular traffic.
Advance friendly air superiority; disrupt adversary air defense.	Command, control, & communications of adversary within region is disrupted and disabled.	Adversary early warning and ground controlled intercept sites are disabled.	Communications relay van is destroyed.
Sustain the populace through the winter months.	Refugees receive necessary food and shelter.	Food and supplies reach distribution points and refugee camps.	Mobility aircraft with relief supplies land at regional airports.

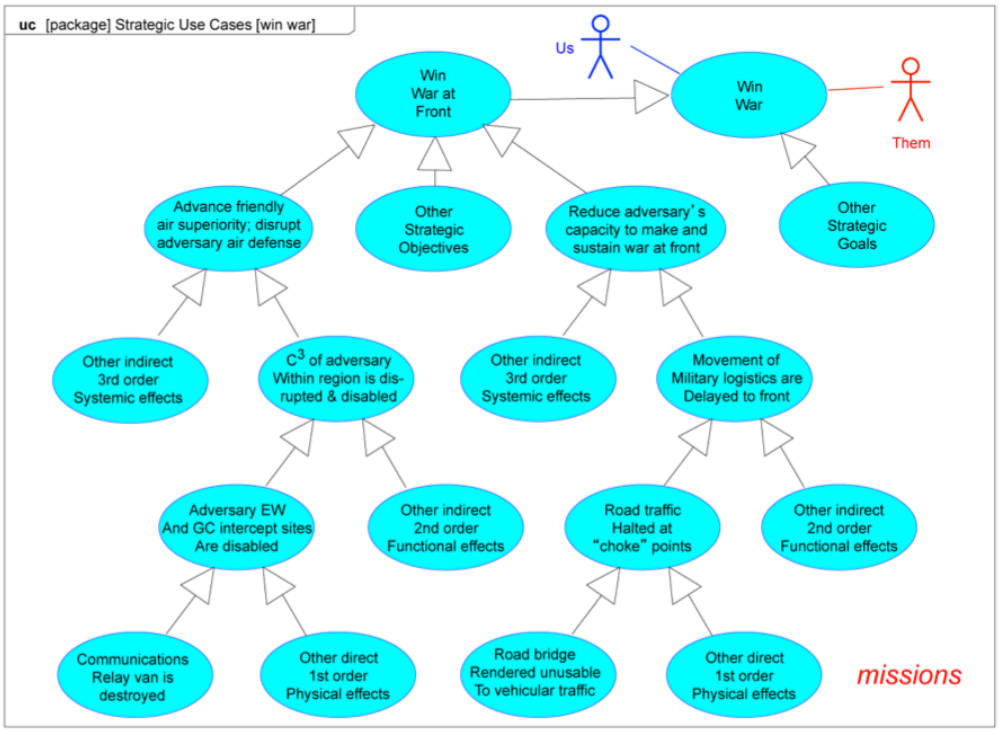


Figure 22. Strategic context graphical example (UML or SysML *uc* diagram).

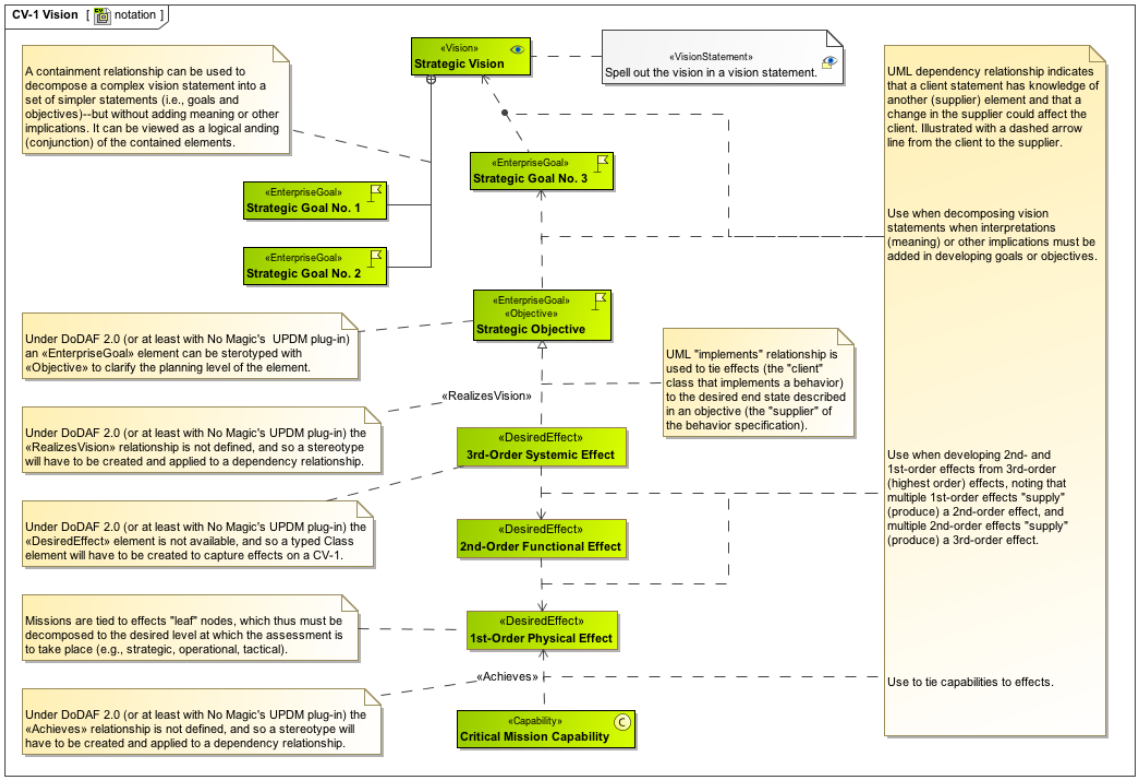


Figure 23. DoDAF *CV-1* strategic context example (model specification).

Finally, note that if the purpose of the study is driven by a particular capability gap, effects are only derived that relate directly to specific mission needs (and only these). It should be understood that at this level of the assessment, effects are expressed abstractly (e.g., hold targets at risk, provide countermeasures against surface-to-air missiles (SAMs), or communicate in a jamming environment); they must not be stated in solution-specific language nor specify optimization.

3.2.1.3 Identifying the Required Capability

After identifying the desired state changes and the effects that may produce the same, a brief description of *how* (i.e., the operational means by which) the effects will be provided or produced (still in an abstract sense) is captured with a bi-directionally traceable, *operational capability*[45] statement. Within DoDAF, identification of the

[45] As previously defined, a capability is the ability, capacity, potential, or power to successfully deploy (assemble, integrate, and manage) available resources to meet some end (e.g., perform some task or activity, produce some desired output or **effect**).

required capabilities can be by updating (or by copying and further refining) the *CV-1*—and creating or refining an *OV-6b* and *OV-5a*—developed in defining the *Problem Statement* above, such as illustrated in Figure 23.

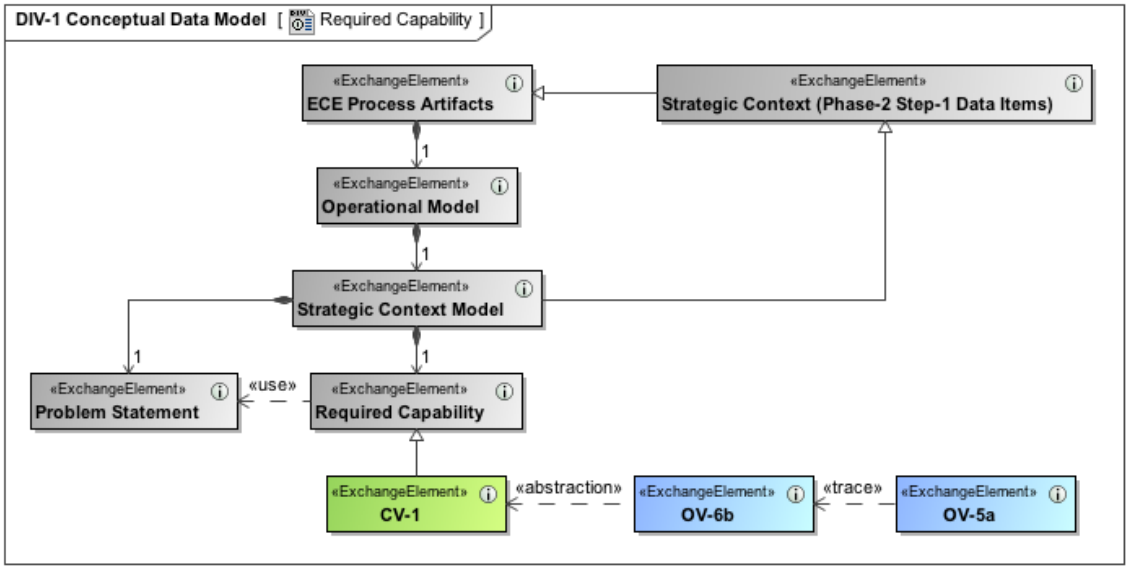


Figure 24. *Required Capability* DoDAF conceptual data model.

Although perhaps it is somewhat tedious, here follows a list of *operational capabilities* for illustration purposes only.[46]

1. Deter adversaries and reduce the need for military force to achieve national objectives.
2. Prevent the initiation and escalation of armed conflict.
3. Increase the capability of allies/coalition partners to assist in achieving security objectives.
4. Defend the United States against enemy missile attack.
5. Protect DoD personnel, dependents, facilities, and installations from terrorist or other attacks.
6. Locate and identify the capabilities of potential military adversaries.
7. Locate and identify the capabilities of potential non-military adversaries.
8. Identify the intentions of potential military adversaries.
9. Identify the intentions of potential non-military adversaries.
10. Maintain the use of the sea and littorals for U.S. military objectives.
11. Maintain the use of the air for U.S. military objectives.

---

[46] Joint Defense Capabilities Study Team, *Joint Defense Capabilities Study: Improving DoD Strategic Planning, Resourcing and Execution to Satisfy Joint Capabilities*, Final Report, January 2004.

12. Maintain the use of space for U.S. military objectives.
13. Maintain the use of information and the electromagnetic spectrum for U.S. military objectives.
14. Deny the use of the sea and littorals to adversaries.
15. Deny the use of the air to adversaries.
16. Deny the use of space to adversaries.
17. Deny the use of information and the electromagnetic spectrum to adversaries.
18. Detect, locate, and destroy adversary WMD capability.
19. Locate and destroy hard and deeply-buried targets.
20. Deny adversaries the use of their installations, facilities, and infrastructure.
21. Locate, identify, and destroy moving and time-sensitive targets.
22. Seize and control terrain.
23. Deny adversaries sanctuary in urban areas.
24. Deny sanctuary to individuals and small groups.
25. Destroy or neutralize adversary military capabilities.
26. Control the behavior of noncombatants without the use of lethal force.
27. Deny sanctuary to adversaries intermingled with noncombatants.
28. Stabilize and maintain order in Nations and non-State areas.
29. Protect deployed forces from air, sea, space, land, and information attack.

3.2.1.4 Developing Operational Depictions

Develop a graphical depiction (or depictions) of the **main** operational context (setting)—e.g., mission, class of mission, or scenario—found or referenced in the written content of the strategic guidance. This pictorial representation (model) is intended to illustrate the main operational concepts (what happens or who does what) and interesting or unique aspects of operations, and identify interactions between the subject architecture and its environment (including, e.g., external systems, organizations, and geographical distribution of assets). This model will be used in establishing the context for a suite of related operational models; this context may be in terms of phase, a time period, a mission or a location, and thus it can serve as a container for the spatial-temporally constrained performance parameters (measures). Additional textual descriptions may accompany the graphic. Following DoDAF conventions, the operational depiction is captured in an *OV-1 High Level Operational Concept Graphic* diagram. (Context can also be defined in a SysML *internal block diagram* (*ibd*), and supplemented with domain information in a *bdd*; *class* diagrams could be used within UML.)

When the concept includes operational exchanges (it would be unlikely not to), the *OV-1* should be supplemented with an *Operational Resource Flow Description* (*OV-2*) diagram, along with other views as appropriate for defining the resources (e.g., data items could be captured by a *DIV-1* without consideration of implementation or product specific issues). (In SysML use a *bdd*, or in UML a *class* diagram.)

Operational concept activity sequencing and timing information related to doctrinally-correct operational procedures, business rules, or operational constraints should also be captured, such as in a DoDAF *OV-6a Operational Rules Model* (a tabular collection of mission-oriented "what" requirements or constraints). That is, an *OV-6a* is used to describe the rules under which the architecture behaves under specified conditions, including operational control handoffs. (In SysML consider a *req*, or use a *class* diagram in UML.) Operational rules will serve to provide guidelines for the development and definition of more detailed rules and behavioral definitions that occur later in the architectural definition process. From a modeling perspective, operational rules may act upon locations, operational activities, missions, and entities in data models. Operational rules are generally expressed in a textual statement; for example: *If* (these conditions) exist, *and* (this event) occurs, *then* (perform these actions). Two different forms are found:

- **Imperative**—a statement of what shall be under specified conditions, e.g., "Battle Damage Assessment (BDA) shall only be carried out under fair weather conditions."
- **Conditional Imperative**—a statement of what shall be, in the event of another condition being met. "If battle damage assessment shows incomplete strike, then a re-strike shall be carried out."

An *AV-2 Integrated Dictionary* should also be established for the project at this time in order to provide textual definitions for all of the modeling elements generated at this level. (Note that the *AV-2* should be updated as the architecture develops.) Use of common taxonomies is highly recommended. (In SysML consider a *req*, or use a *class* diagram in UML.)

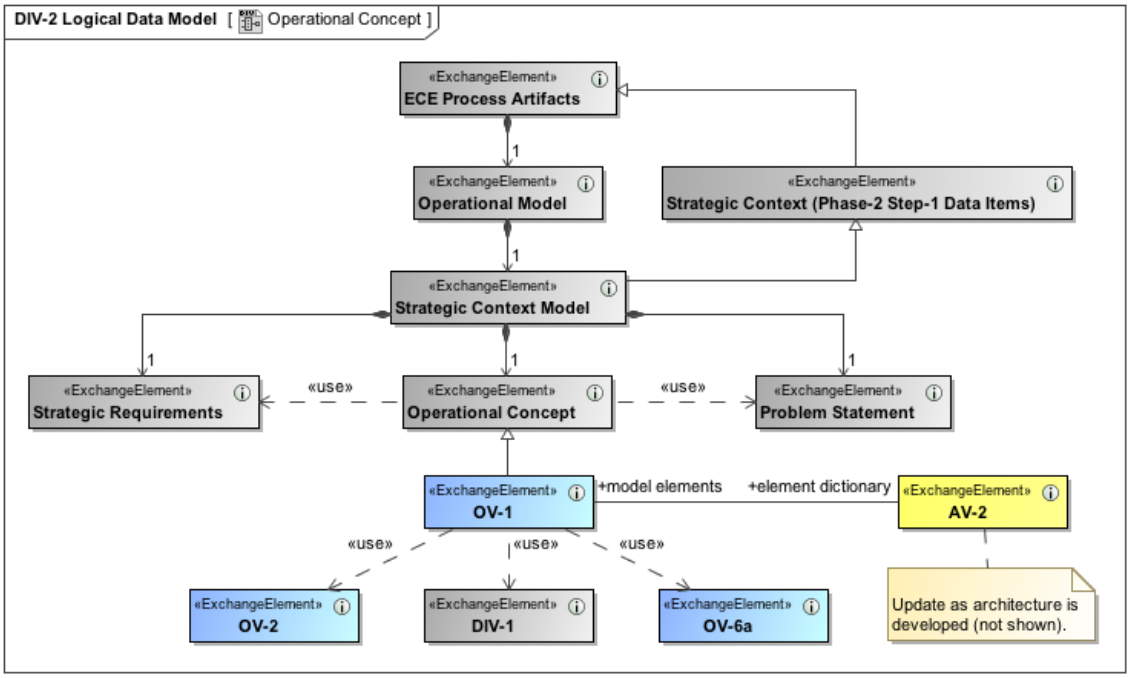


Figure 25. *Operational Concept* DoDAF data model.

3.2.1.5 Documenting Stakeholder Concerns

Given that the ECE process is being exercised for a valid use case, determination of whether or not the process technical goals and objectives are met in a satisfactory manner ultimately resides outside of the study team with one or more stakeholders. To put it differently, as the intent of an enterprise capability—or system of systems—is to satisfy "stakeholders over the life of the products that make up the system" (EIA-632 §1.1(c)), it is necessary to undertake a task to "identify stakeholders [enterprise, organization, team, or individual, or classes thereof] who will have an interest or stake in [concerns relative to] the outcome of the project" (EIA-632 Requirement 4 (a)) and take explicit measures to assure they are satisfied. In addition to this SE standards-point-of-view, it is very likely that, for any particular project, programmatic or quality management (business) process requirements apply which will also explicitly state that stakeholders must be identified (e.g., ISO 9001 §7.2.1 "customers").

The beginning point for considering stakeholders and their interests will be the organization list found in the scope statement of the *Architecture Study Plan* (*AV-1*).

However, contextual analysis of the strategic guidance is likely to identify additional stakeholders (and, if so, update the *AV-1)*.

In general, stakeholders represent a varied audience with differing, often conflicting, interests. It should go without saying, then, that it behooves the architecture study team to expend an appropriate amount of effort to identify the concerns that these varied interests have. Once concerns have been identified, an explicit effort can be planned and made to present the study results in such a way that each stakeholder can understand how their interests have been met.

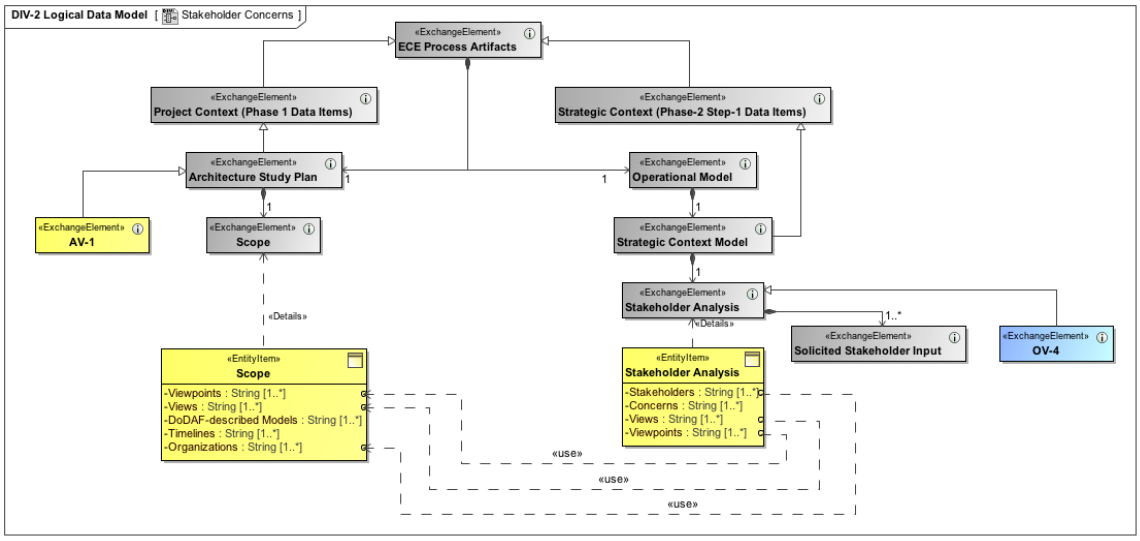


Figure 26. *Stakeholder Concerns* DoDAF data model.

A first step to consider towards identifying concerns is to categorize the stakeholders by type. (This will also help in prioritizing and reconciling differences and conflicts in requirements that inevitably will arise.) From an SE standards-point-of-view (EIA-632 Annex A), stakeholders are divided into two broad categories: *acquirer* and *other*.

> **Acquirer**: An enterprise, organization, or individual that obtains [procures] a product (good [materiel] or service) from a supplier. The acquirer can be a customer or user [buyer, customer, owner, user, or purchaser] of a desired system product, or can be a developer obtaining a lower layer product in the system hierarchy from another vendor or a developer in the role of supplier.

Other stakeholders: All stakeholders other than the acquirer.

An alternative standard (IEEE Std 1471 §5.2) indicates that, at a minimum, the stakeholders identified shall include the following:

> **Users** of the system;
>
> **Acquirers** of the system;
>
> **Developers** of the system; and,
>
> **Maintainers** of the system.

Stakeholder classification can be documented in DoDAF using an *OV-4 Organizational Relationships Chart* by using inheritance and aggregation properties. Following stakeholder identification and classification, stakeholder *concerns* relative to the system in question are identified.

> **Concerns**: those interests that pertain to the system's development, its operation, or any other aspects that are critical or otherwise important to one or more stakeholders. (IEEE Std 1471 §4.1)

Categories of concerns to consider should include the following:

- The purpose or missions of the system
- The appropriateness of the system for use in fulfilling its missions
- The feasibility of constructing the system
- The risks of system development and operation to users, acquirers, and developers of the system
- "Ilities" like reliability, maintainability, security, deployability (or product distribution), and evolvability of the system

Note that concerns are **not** system operational or technical requirements in and of themselves (although they might be loosely thought of in terms of requirements categories). Note also that the system architecture description developed must address all of the concerns identified; otherwise, by definition, it would be incomplete.

In the end, stakeholders must be polled in order to identify their concerns. A "collector" will either send a request for information to—or conduct an interview with—each identified stakeholder group in order to produce a list of that group's special (particular, unique) concerns. (It may also be advantageous to take this opportunity to "validate" the list of *Strategic Guidance Sources* previously identified in the ECE process.) The identified concerns are to be documented along with traceability to the applicable

stakeholders. In DoDAF this can be accomplished by creating a *class* for each concern, stereotyping the *class* as a «concern», and adding a *class property* (typed as *String* and named, say, "text") with the default value set to an appropriate description of the concern. The set of classes so created can then be portrayed on an *OV-4* and associated with the respective organizations.

Next, in compliance with the architectural framework to be followed, *viewpoints* are selected to provide coverage of all documented stakeholder concerns (cf. Appendix C). Rationale and traceability are established using an appropriate modeling technique (such as described above), which may require use of a "Fit-for-Purpose" view. One *view* is then defined for every *viewpoint* selected. It is suggested that a set of Fit-for-Purpose views—such as SysML *package* (*pkg*) diagrams—be developed to graphically portray this information.

Finally it should be noted here that a complete set of *view* definitions, if properly developed, defines—at least a lower bound—to the scope of the required architectural modeling effort (i.e., it identifies the architectural models that **must** be developed). Therefore it may be useful at this juncture to revisit the *architecture development schedule* and *cost* entries in the *AV-1*, and update the study project plans as appropriate (further description of which is beyond the scope of this paper).

3.2.1.6 Phase-2 Step-1 Summary

After completing the first operational modeling step: (1) strategic guidance was identified, reviewed and analyzed; (2) strategic requirements and constraints were extracted from the guidance and integrated into a coherent, consistent model—including validation objectives[47] in the form of effects to be produced along with associated measures (MOEs); (3) the required mission-level (operational) capabilities were defined; (4) a high-level operational depiction was developed; and (5), stakeholder concerns were identified and documented, *viewpoints* selected, and *views* defined.

[47] These validation objectives will be used by the stakeholder to measure "mission" success during operational testing of a new (or upgraded) enterprise or system of systems.

Intentionally blank.

3.2.2 Defining a Mission Model (Phase-2 Step-2)

The basic task of developing a mission *model* is one of analyzing and decomposing the strategic guidance in order to identify, describe, and relate the required operational activities or tasks and resource flow (e.g., information) exchanges— including timing and sequencing. This information is captured in, e.g., behavioral, data, and rule (requirements) sub-models, along with a supporting integrated system dictionary. Essential (critical) operational capabilities are also identified and linked to the enterprise-level capability under evaluation. Since a mission model is operation-centric, it will naturally be developed using methods established by, e.g., a DoDAF *operational viewpoint*; however, to be complete it will, in general, have to use *capability viewpoint* methods, as well as, e.g., define interfaces as specified in the *service viewpoint.* A second basic task is also performed in this step: identifying and validating the modeling and simulation (M&S) approach to be followed in evaluating the capability performance vis-à-vis mission. Generally speaking, the M&S approach will include identification of executable models that are to be used for demonstrating the dynamic behavior of any capabilities that are employed. This will also involve definition of the tasks that must be analyzed and associated metrics. The M&S approach itself is validated on the basis of an "as is" capability (for precedented capabilities), the mission-performance of which is used to establish a baseline. Once developed, the mission evaluation plan also serves as a validation plan for alternate concept evaluations.

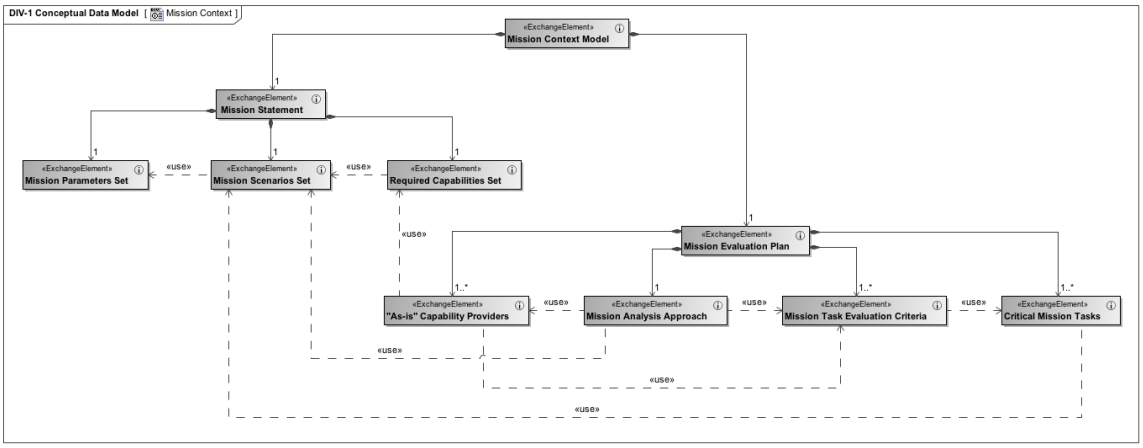


Figure 27. ECE Process Phase-2 Step-2 conceptual data model.

Table 5. *AV-2 Integrated Dictionary* of Phase-2 Step-2 Data Elements

Name	Definition
Mission Statement	A "commander's" estimate of the situation, including mission parameters, scenarios, and capabilities.
Mission Parameters Set	Mission objectives, constraints, assumptions, threats, and metrics.
Mission Scenarios Set	Possible Courses of Action (COA).
Required Capabilities Set	Functional capabilities that will produce the desired effects.
Mission Evaluation Plan	The approach to be followed in testing COA for suitability, feasibility and acceptability.
"As-is" Capability Providers	A description of how the required capabilities are currently provided, including resource descriptions and critical mission task performance.
Mission Analysis Approach	The modeling and simulation plan to be used for testing COA.
Mission Task Evaluation Criteria	Standards for each critical mission task, and other applicable attributes.
Critical Mission Tasks	Activities that materially contribute to misison success.

For organization purposes, Step-2 is broken down into two tasks: develop a mission statements (Step-2A) and develop a mission evaluation plan (Step-2B).

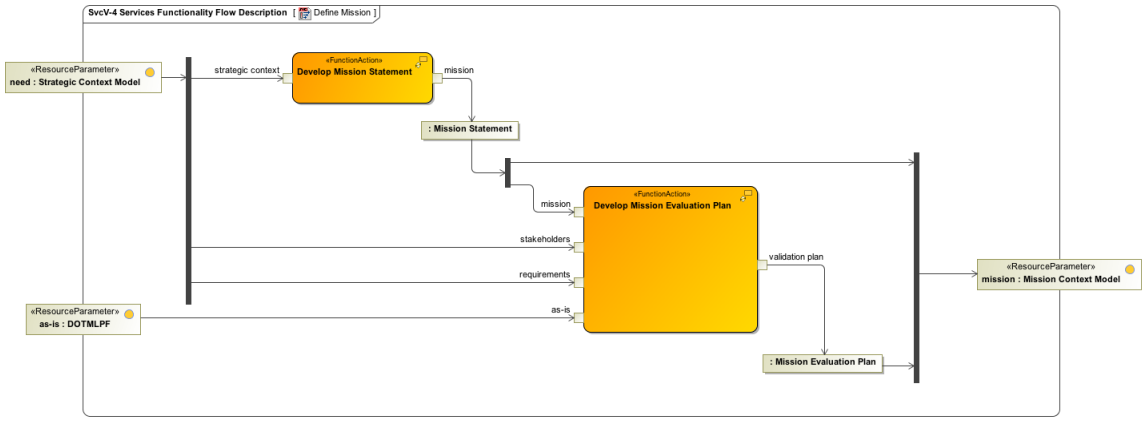


Figure 28. ECE Process Phase-2 Step-2 basic task flow.

As an aside, it should be noted that standard architectural practice develops model content by using *viewpoint model* definition templates as found in vendor tools. However, the test for architectural description completeness is not on the type or number of these

templates used to create models, but on whether or not all of the information defined in the *views* is provided by the set of models produced. (Tests for correctness and consistency also exist, as established through compliance with the applicable modeling standards.) Again, *viewpoints* and *views* are **not** *architecture* **nor** an *architectural description* in and of themselves.

3.2.2.1 Developing mission statements (Phase-2 Step-2A)

The "strategic" problem statement of §3.2.1.2 is further developed as necessary—through, e.g., decomposition—down to the desired mission analysis and DOTMLPF level. In general this will involve developing descriptions of multiple mission scenarios (i.e., different courses of action).

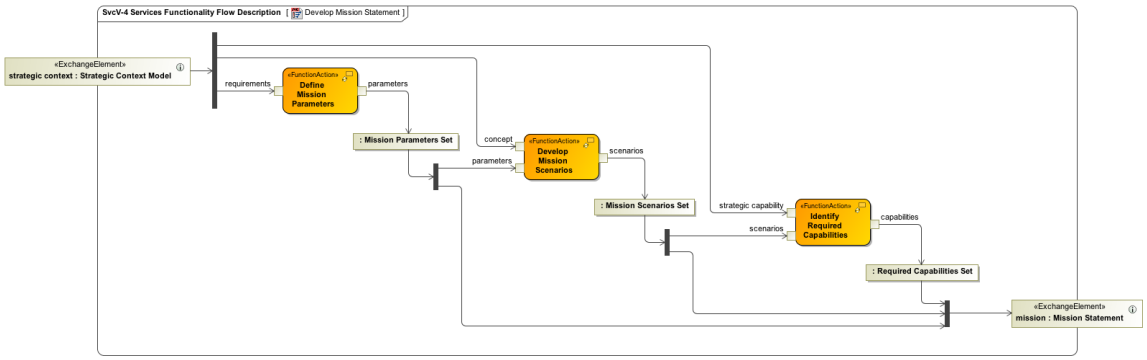


Figure 29. Process detail for generating the *Mission Statement* data item.

3.2.2.1.1 Defining mission parameters

Mission parameters are defined, including:

- Problem Statement (intent, purpose, or goal; the **why**)
  - A very short statement that describes a focused scope (central idea or mission area) and the desired future state.
- Mission Objectives
  - The problem statement is expanded to include, e.g., what, when and how.
- Mission MOEs
- Constraints and tradeoff measures
- Assumptions (including threats)
- References to applicable strategic guidance (traceability)
  - May require resolution of conflicting guidance.

Within DoDAF, this information can be captured in a *CV-1* diagram, possibly supplemented with—or, alternatively, by using—an *OV-6b* and supporting *OV-5a.* Traceability should be provided to appropriate elements in the *strategic context model*, including, of course, the *strategic requirements* of §3.2.1.1 (as captured, e.g., in DoDAF using a *StdV-1*). The "views" into the *strategic requirements* offered by the *problem statement* of §3.2.1.2 (represented in DoDAF with a *CV-1*, and possibly *OV-6b* and *OV-5a*), and any operational rules defined in support of the *operational concept* of §3.2.1.4 (e.g., DoDAF *OV-6a*), will likely be of use in this endeavor as well. (In SysML or UML, the mission parameters model may include, as appropriate, *uc*, *bdd*, *req*, *stm*, or *class* diagrams.)

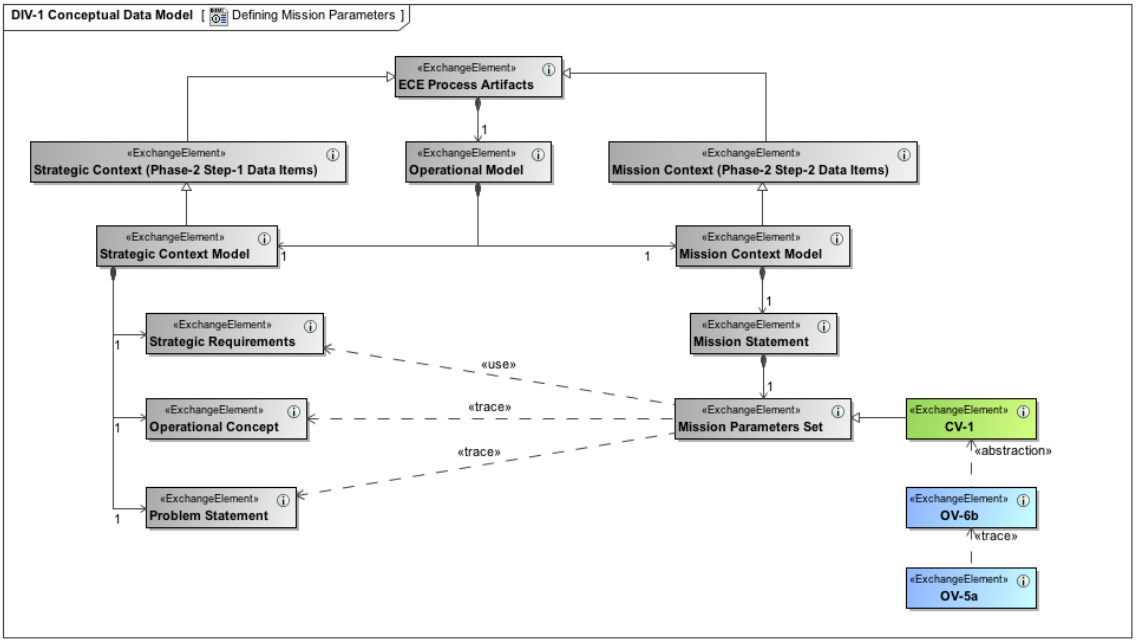


Figure 30. *Mission Parameters* DoDAF conceptual data model.

3.2.2.1.2 Developing mission scenarios

A set of well-posed scenarios are constructed for the range of conditions possible. This is a **critical** step. (Say NO! to scenario agnosticism.) Scenarios…

- start from an initiating event or process
- proceed through a logically connected & physically possible combination or sequence of features, events, and processes (i.e., they are one path through an logic tree)
- reach a defined "final state"

- provide context to:
    - connect follow-on development to mission and strategic guidance (traceability)
    - broaden the perspective by considering a wide range of relevant situations
    - define the spectrum of conditions required for the analyses
- should be developed for both "sides" of the problem (e.g., own force elements and threats)

Mission scenarios are captured in *OV-6b State Transition Description*, *OV-5b Operational Activity*, and *OV-6c Operational Event-Trace Description* models (in UML or SysML, use *stm*, activity (*act*), or sequence (*seq*) diagrams).

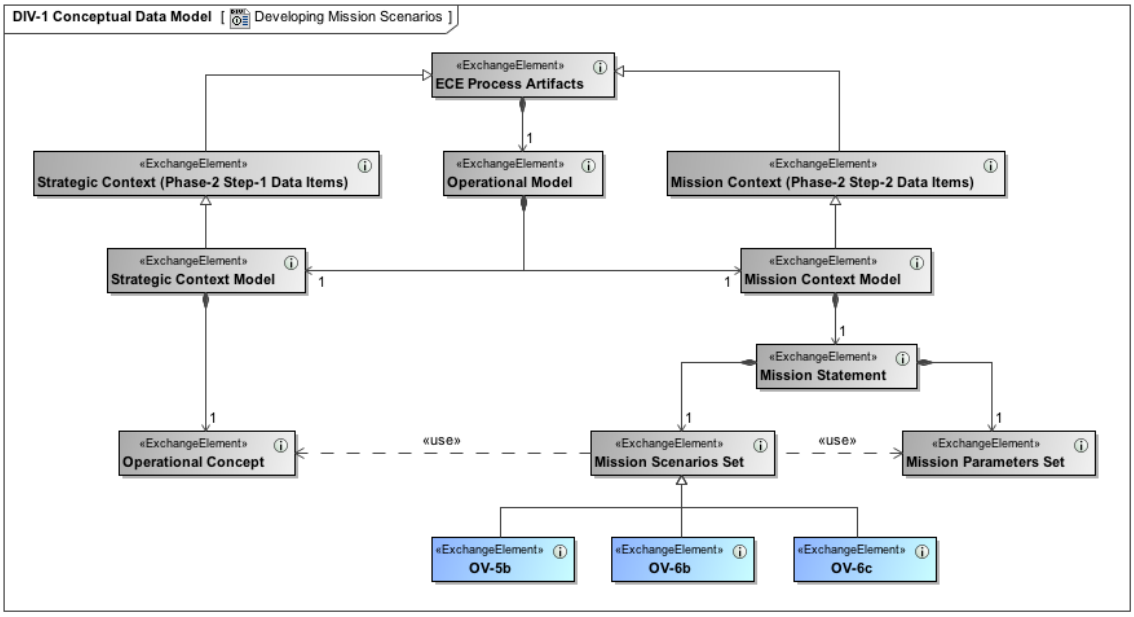


Figure 31. *Mission Scenarios* DoDAF conceptual data model.

For DoD-related efforts, once draft mission scenarios have been developed, a wealth of resources exist both to recast the scenario descriptions into common terms (as with the use of JCAs above), as well as to test for completeness and doctrinal correctness. These include the *Universal Joint Task List (UJTL)*, service-specific task lists, *Joint Common System Function List (JCSFL), Consolidated Operational Activities List (COAL),* and *Joint Mission Thread (JMT)* definitions.[48]

[48] It is a common but poor practice to use, e.g., the UJTL, JCSFL, COAL or JMT to draft mission scenarios. This effectively "traps" solutions within a box, stifling creative solutions. However, consideration of such lists *after* drafting the scenarios is generally a good thing.

3.2.2.1.3 Identifying required capabilities

First, determine which scenario (**not** mission) objectives are critical (important). These are factored from the **set** of mission scenarios (e.g., the *OV-5b, OV-6b, and OV-6c* diagrams of §3.2.2.1.2), and they correspond to common (cross-cutting) activities. It is important to note that this is a screening; non-critical objectives are handled as assumptions, or ignored all together. Within DoDAF, objectives should be captured in a *CV-1* diagram (represented as use cases in UML or SysML); traceability to higher-level objectives should also be defined.

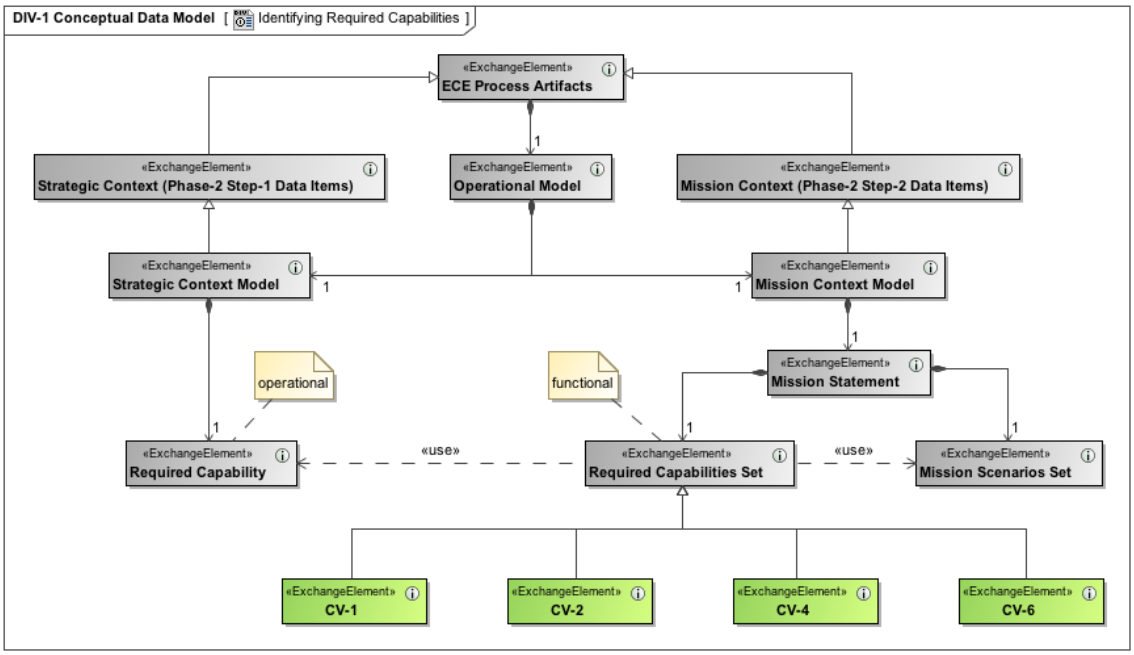


Figure 32. *Required Capabilities* DoDAF conceptual data model.

Next, identify the *functional capabilities* required to produce the effects that will generate the desired state changes (i.e., meet the screened scenario objectives). These capabilities will form the basis of the development to follow, which eliminates the problem of trying to evaluate a system in terms of capabilities *de nusquam* (from nowhere). Note that a given set of functional capabilities may not be unique; in such cases variants should be identified and developed (including evaluation of appropriate trade space).

The traceability between capabilities and objectives can be captured using a *CV-1* diagram. A direct mapping between the capabilities and the operational activities can also be provided by using a *CV-6 Capability to Operational Activities Mapping* diagram;

however, note that the *CV-6* cannot replace the *CV-1* in that it does not have any provision for documenting objectives and measures. Hierarchical relationships between the scenario- (functional) and strategic-level (operational) capabilities should be identified in a *CV-2 Capability Taxonomy* diagram. Dependencies between scenario-level capabilities should also be captured through the use of a *CV-4 Capabilities Dependencies* diagram. (In UML or SysML, use cases are partitioned into "capability" packages where the linkage between elements (actors) and the "system" translate into a provided capability.)

It is often desired (or perhaps programmatically required) to cast capabilities in terms of common names (taxonomy) and descriptions to facilitate communication with stakeholders. This can also improve higher-level capability analysis, strategy development, investment decision making, capability portfolio management, and capabilities-based force development and operational planning. For DoD, such a taxonomy exists for functional capabilities as the Joint Capability Areas[49] (JCAs). To get started, consider the following high-level capability statements examples:[50]

> **(JCA 1.) Force Support** – The ability to establish, develop, maintain and manage a mission ready Total Force.
>
> **(JCA 2.) Battlespace Awareness** – The ability to understand dispositions and intentions as well as the characteristics and conditions of the operational environment that bear on national and military decision-making by leveraging all sources of information to include Intelligence, Surveillance, Reconnaissance, Meteorological, and Oceanographic.
>
> **(JCA 3.) Force Application** – The ability to integrate the use of maneuver and engagement in all environments to create the effects necessary to achieve mission objectives.
>
> **(JCA 4.) Logistics** – The ability to project and sustain a logistically ready joint force through the deliberate sharing of national and multi-national resources to

[49] http://www.dtic.mil/futurejointwarfare/jca.htm
[50] The JCA provides a standard decomposition of these capabilities, for which see.

effectively support operations, extend operational reach and provide the joint force commander the freedom of action necessary to meet mission objectives.

(JCA 5.) Command & Control – The ability to exercise authority and direction by a properly designated commander or decision maker over assigned and attached forces and resources in the accomplishment of the mission.

(JCA 6.) Net-Centric – The ability to provide a framework for full human and technical connectivity and interoperability that allows all DOD users and mission partners to share the information they need, when they need it, in a form they can understand and act on with confidence, and protects information from those who should not have it.

(JCA 7.) Protection – The ability to prevent/mitigate adverse effects of attacks on personnel (combatant/non-combatant) and physical assets of the United States, allies and friends.

(JCA 8.) Building Partnerships – The ability to interact with partner, competitor or adversary leaders, security institutions, or relevant populations by developing and presenting information and conducting activities to affect their perceptions, will, behavior, and capabilities in order to build effective, legitimate, interoperable, and self-sustaining strategic partners.

(JCA 9.) Corporate Management and Support – The ability to provide strategic senior level, enterprise-wide leadership, direction, coordination, and oversight through a chief management officer function.

3.2.2.2 Developing a mission evaluation plan (Phase-2 Step-2B)

This step develops linkages between the required mission-scenario-level capabilities (or, to keep it simple, mission capabilities) and the critical mission tasks to be performed (i.e., an employment of forces "plan"), as well as mission performance standards and other stakeholder-specified attributes. Collectively these are needed as a means to select the required mission analysis tools (e.g., they define the input and required output that must be supported by the modeling and simulation effort; it is assumed such tools exist—the development of a means to assess a mission is beyond the scope of this paper).

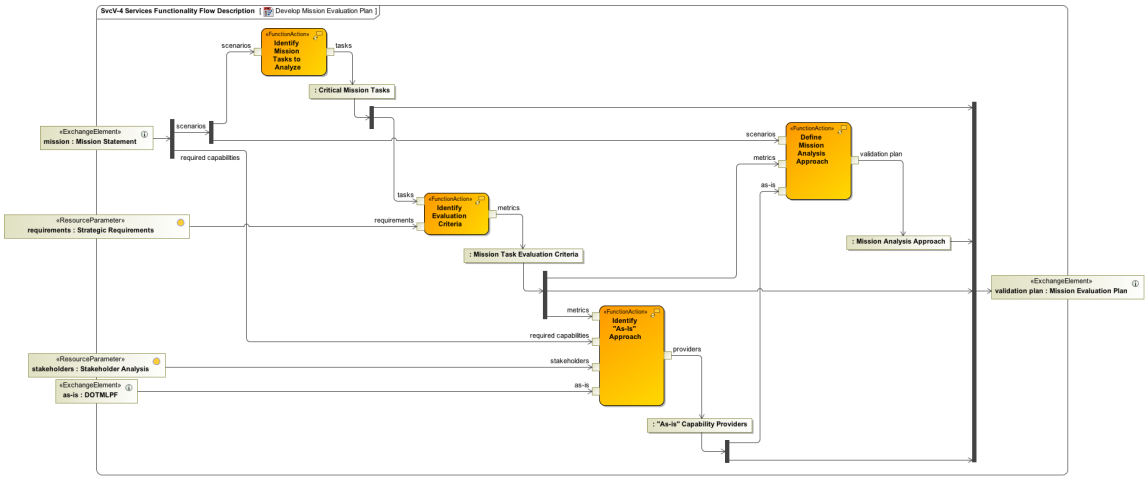


Figure 33. Process detail for generating the *Mission Evaluation Plan* data item.

3.2.2.2.1 Identifying mission tasks to analyze

An overarching mission task and function structure is developed, beginning with a review of the critical scenario objectives and corresponding mission-common activities (as captured in *CV-1* and *CV-6* diagrams above). These "critical" functions and tasks are further screened to identify those that will be analyzed. For DoD-specific efforts, guidance on which activities to consider may be available from the *Joint Mission Essential Task List (JMETL)*, or similar service-specific lists (generically speaking, *Mission Essential Task Lists*, METLs).

- When NOT to analyze:
  - The task or function does not apply to the concept or scenario.
  - The task or function is being evaluated by another study.
  - There is ample evidence that the function or task will succeed in the scenario.

The final set of mission operational activities to be analyzed (from a mission performance perspective)—subsequently referred to as *critical mission tasks*—can be captured in an *OV-5a,* and are mapped to the supplying capabilities in a *CV-6*. (In SysML or UML, tasks or functions are captured as a *block-* or *class-operation*.)

> *DON'T let the "as-is" approach or standard lists blind or prevent you from examining functions and tasks that should be assessed.*

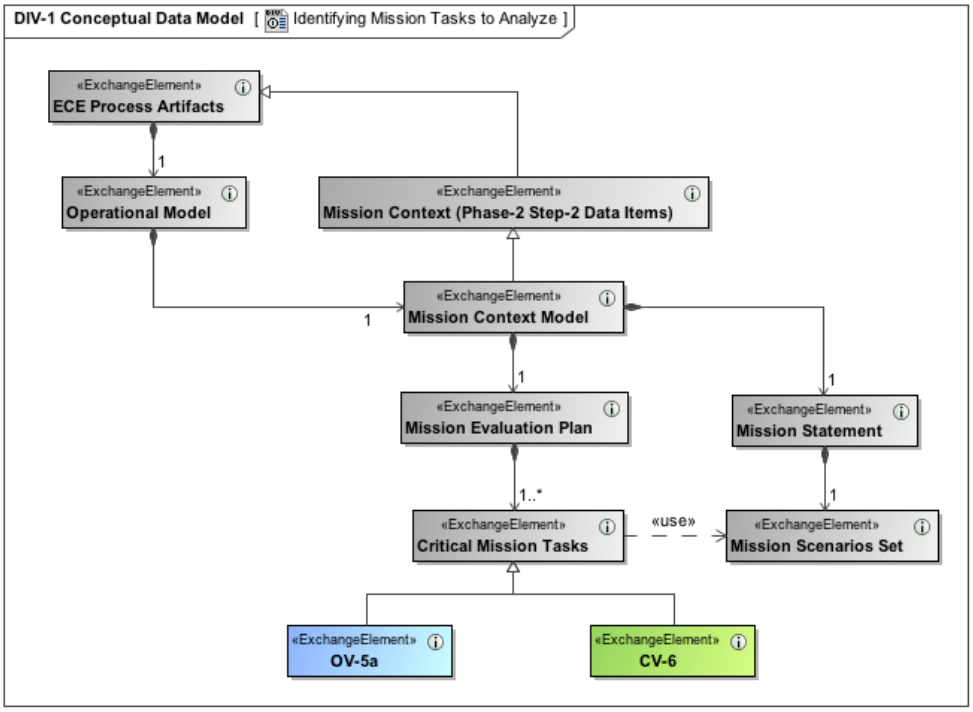


Figure 34. *Critical Mission Tasks* DoDAF conceptual data model.

3.2.2.2.2 Identifying evaluation criteria

Determine **standards** appropriate for the evaluation of how well a concept will meet mission requirements (metrics; i.e., measures of effectiveness, MOEs). Here, *standard* is defined to be "a quantitative or qualitative measure for specifying the level of performance of a task," and is referred to as a measure of performance, MOP. It is critical to understand that an MOE can be multidimensional in that it is generally expressed in terms of a set of **derived** MOPs. This means that once MOP measurements have been obtained, they can be compared to the standards to determine if performance is satisfactory. It is also possible to interpret the results (e.g., by a "roll-up" calculation) as an actual MOE for comparison purposes.[51]

For DoD-related efforts, it should be noted that the UJTL contains lists of common MOPs, some of which may apply in describing a particular mission—quantity kinds and units, but not, of course, mission specific performance values. (Cf. Appendix D.) For

[51] When evaluating a single functional capability set this is, in principle, unnecessary. However, when comparing variant capability sets it is required as any derived MOPs will be, in general, different.

some types of systems, guidance may go beyond MOEs and MOPs to include key performance parameters, or KPPs (e.g., the "Net-Ready" KPP of DoDD 4630.05).

In order to judge the utility of alternative concepts, these standards may have to be augmented with **attributes**. Here, *attribute* is defined to be "a quantitative or qualitative characteristic of an element or its actions." A common attribute, e.g., is cost, but, in general, numerous other preferences may exist (e.g., color). Like standards, attributes must be connected to the chosen scenarios and must be characterized by suitable metrics and associated value (utility) functions.

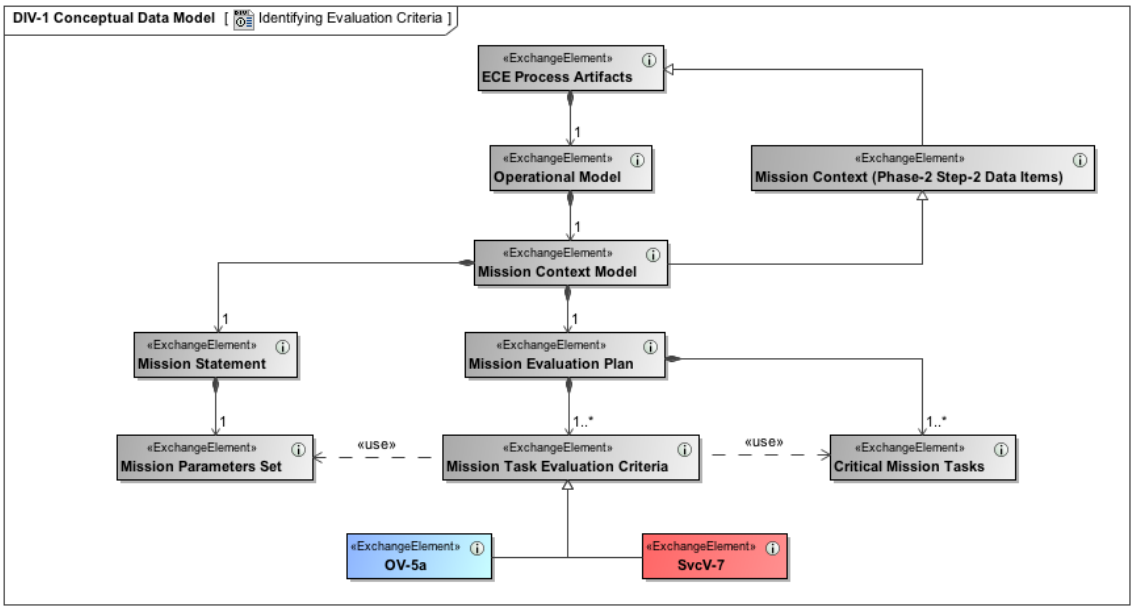


Figure 35. *Evaluation Criteria* DoDAF conceptual data model.

Within DoDAF (actually UPDM), «MeasureType» (think general measurement category;[52] e.g., *height* or *weight*) and associated «Measurement» (think "kind of quantity"[53] with a "unit of measurement"[54]—that may derived (complex)—in an ISO

[52] Aka "Value Type" or "Data Type" used to establish a more neutral term for system values that otherwise may never be given a concrete data representation. A Measure Type adds an ability to carry a unit of measure of a quantity kind associated with the value.
[53] Or "Quantity Kind"; can be measured using defined and unrestricted units of measurement. For example, length, a quantity kind, may be measured by meter, kilometer, or foot units.
[54] Or "Unit"; a particular value that can be used to specify a quantity of a dimension (i.e., quantity kind). A unit often relies on precise and reproducible measuring techniques. For example, a unit of length such as meter may be specified as a multiple of a particular wavelength of light. A unit can also use less stable or precise ways to express some values, such as costs expressed in some currencies, or a severity rating measured by a numerical scale.

80000-1 sense; e.g., *length* and *meter*, or *mass* and *kilogram*) modeling elements can be created and linked to the appropriate operational activities in an *OV-5a* diagram (in SysML, standards and attributes can be captured as block constraints, preferably by using the "quantity kinds and units" model library), or an *SvcV-7 Services Measures Matrix* might be adapted for use in this regard (e.g., using operational activities—or even mapped capabilities—as the resource). Performance requirements (specifically the relationships existing between MOPs and MOEs) should also be modeled; for example, this could be accomplished with constraints and "Fit-for-Purpose" SysML diagrams, such as illustrated below, although the actual approach to be followed will be domain specific. (Cf. APPENDIX E.)

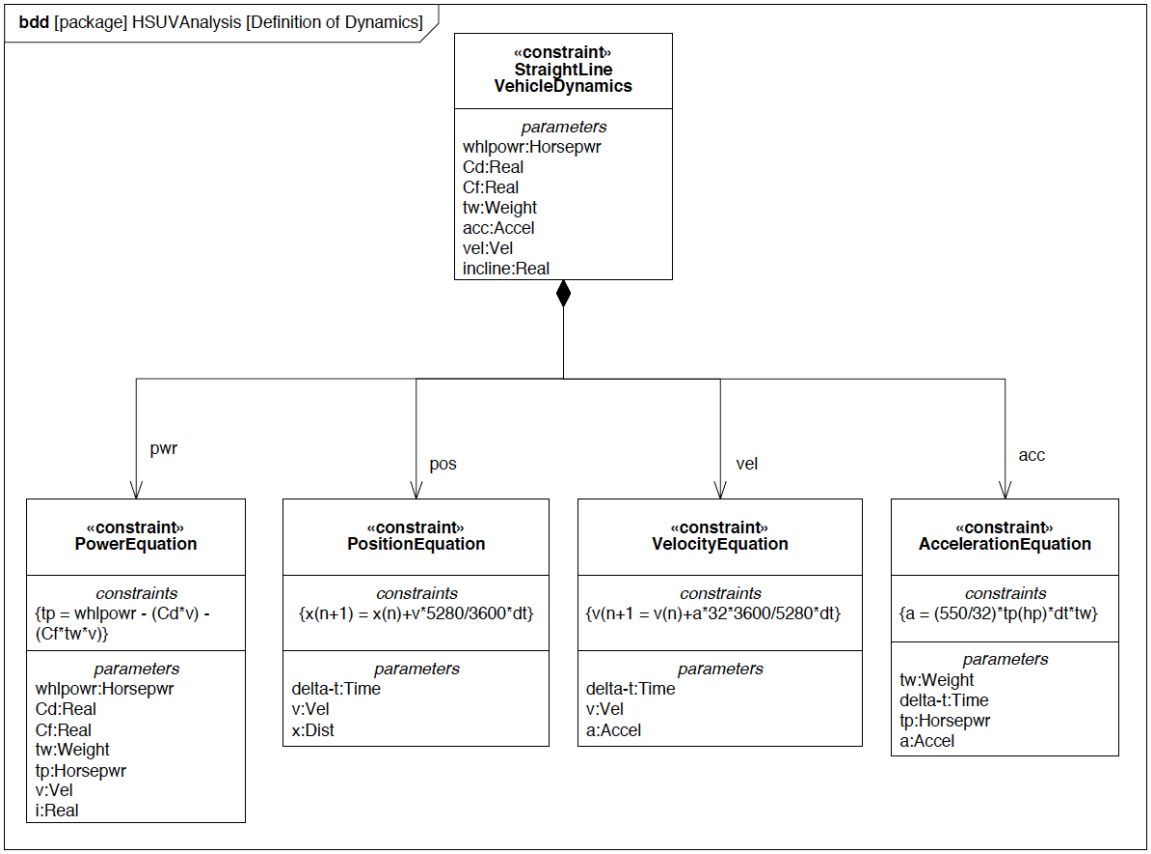


Figure 36. Vehicle dynamics mathematical constraints example.[55]

[55] OMG SysML 1.3 Figure C.31

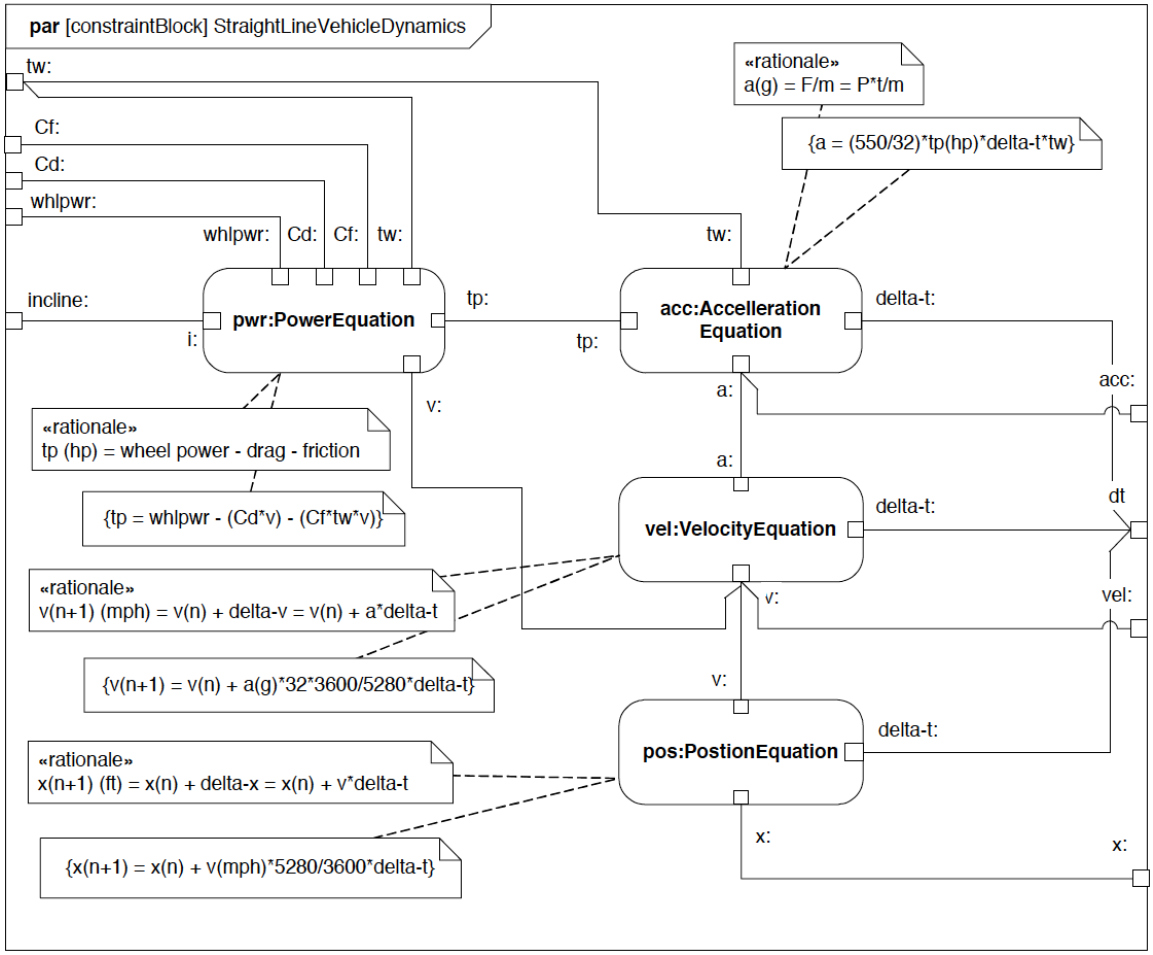


Figure 37. Vehicle dynamics mathematical model example.[56]

3.2.2.2.3 Identifying the "as-is" approach

Determine how the required capabilities are provided (**now**, by who, for precedented systems). This must be defined down to the operational unit level or element at which analysis will occur, consistent with the desired mission analysis and DOTMLPF levels previously identified, including the actions they can perform (i.e., performance against the evaluation criteria).

- e.g., soldier, squad, platoon, company, regiment, or battalion
- e.g., institutional kitchen, restaurant, home kitchen, or camp stove

[56] OMG SysML 1.3 Figure C.30

The actual organizations (or posts) that provide a capability and leadership roles are defined using an *OV-4* diagram. Personnel resource types can likewise be identified in an *OV-4*, including required skills (and so training and education, and related metrics). *OV-4s* are thus used to provide identification of four (DOTMLPF) of the seven basic resource element types (assets). Any materiel and facility resources—and doctrine, by name—planned for use in providing an operational capability (e.g., a "capability configuration") can be defined in a *SvcV-1 Services Interface Description* diagram and linked to the providing organization and associated capability, so capturing the remaining three basic resource elements (DOTMLPF). The information from these *OV-4* and *SvcV-1* diagrams can then be used to create a *CV-5 Capability to Organizational Development Mapping*; in this table, the intersection (cell) between the providing organization (row) and scenario-level capability (column) is populated with the organization's organic resources (DOTMLPF) deployed or assigned to provide said capability. (In SysML use block definition diagrams, or in UML use *class* diagrams.)

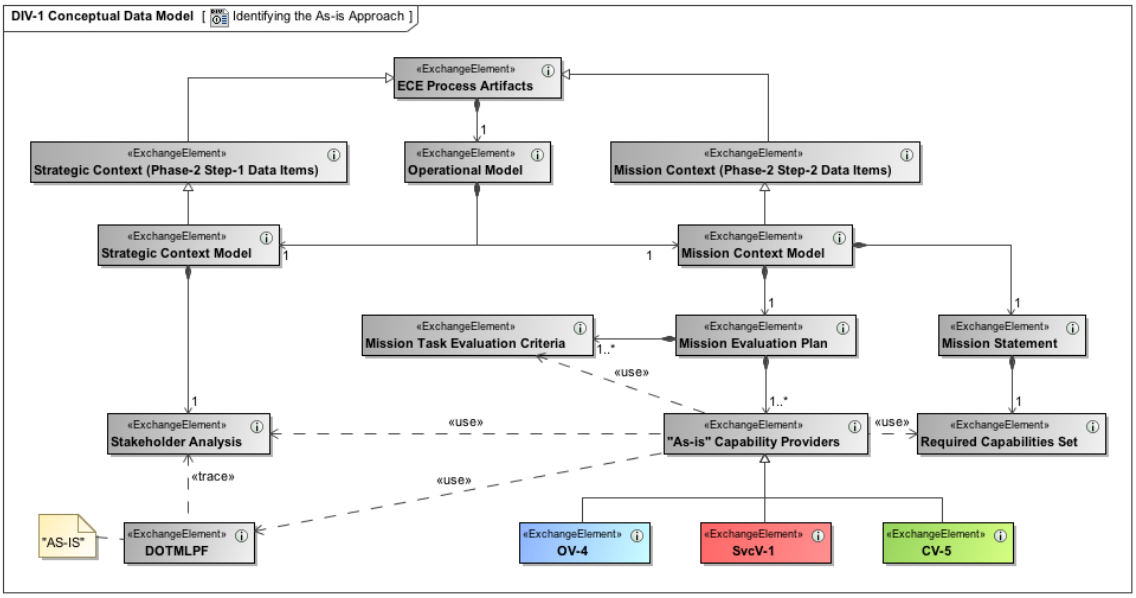


Figure 38. *As-Is Capability Providers* DoDAF conceptual data model.

3.2.2.2.4 Choosing the mission analysis approach

It is not possible to state anything in regards to the existence or performance of a capability unless it is possible to test it against specified standards (cf. §3.2.2.2.2) and conditions (e.g., various threats and operating environments). And while *test* may

eventually include "field exercises" (e.g., during operational test and evaluation), these will generally be limited for practical reasons (e.g., expense). Therefore, assessing the performance of a capability to meet mission requirements will almost always rely on some form of analysis. In the current phase of the process, analysis generally drives performance requirements development in a top-down sense.

Other than a textual entry in an *AV-1* report, DoDAF does not have a viewpoint defined to capture analysis approaches, requiring development of a "Fit-for-Purpose" *view*. Turning to SysML, the various model elements that will be used to conduct the analysis could be defined in a *bdd*. This should include a depiction of the constraint blocks and equations, and key relationships. Mission performance is defined in terms of MOEs, MOPs, and related parameters, and captured in parametric diagrams.

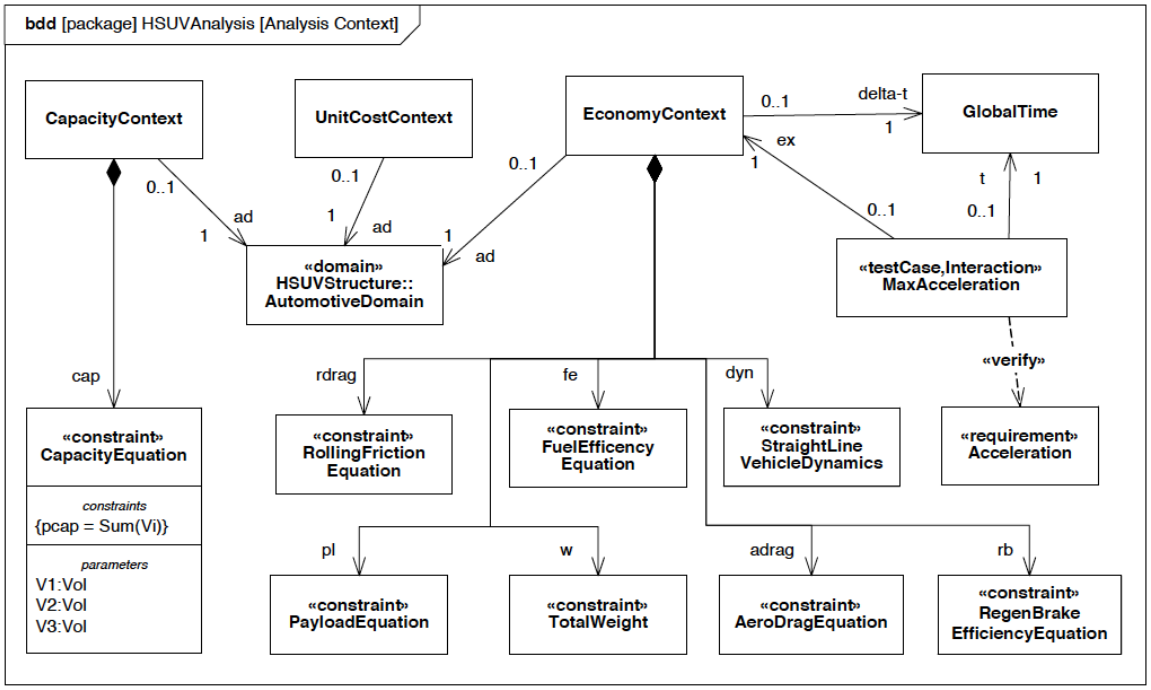


Figure 39. Vehicle engineering development analysis context example.[57]

The actual approach that will be executed by the analytical team should be updated,[58] "fleshed out," and documented in a *M&S Plan*[59] (aka modeling and analysis plan, be it a

[57] OMG SysML 1.3 Figure C.26

document or, preferably, a models-based *view* using, e.g., DoDAF artifacts as illustrated in Figure 41; cf. footnote 38 and corresponding discussion in §3.1). Going beyond the analysis context, this would include a description of the collection of tools and modeling information flows that will be used to transform the input (through mission execution using the capabilities provided by, e.g., the "as-is" DOTMLPF resources), estimate outcomes (MOPs and MOEs), and present results.

The choice of modeling technique(s) will, of course, be determined by the nature of the problem (e.g., the physics involved), the scope (operational level of the assessment), and the mission time span considered. Determination of the techniques to be used will thus begin by considering the established mission scenarios, the mission tasks that are to be analyzed, and the established evaluation criteria established. From such consideration, validation test cases are defined. Most commonly mission test cases are captured in sequence diagrams (*OV-6c, sd*), although sometimes activity diagrams can be used to advantage (*OV-5b*, *act*). Test cases must include identification of the data to be collected (*DIV-1, DIV-2*, *bdd, class*). The problem then becomes one of identifying to what level and how simulations and analyses are to be performed. Note that, in general, critical, mission-performance related elements (e.g., functions or components) are implemented as physics-based models, while non-critical elements are implemented as behavioral-based models; usually ancillary elements are ignored altogether. In the end, a balance will have to be struck between scope, technique, and level of detail due to practical (resource) limitations.

---

[58] The assessment plan (e.g., the *AV-1*) should have defined an approach to be followed for validation.
[59] DoD projects should consult DoDD 5000.59, *DoD Modeling and Simulation (M&S) Management*, and the DoD M&S initiative at http://www.acq.osd.mil/se/initiatives/init_ms.html for additional direction in this area.

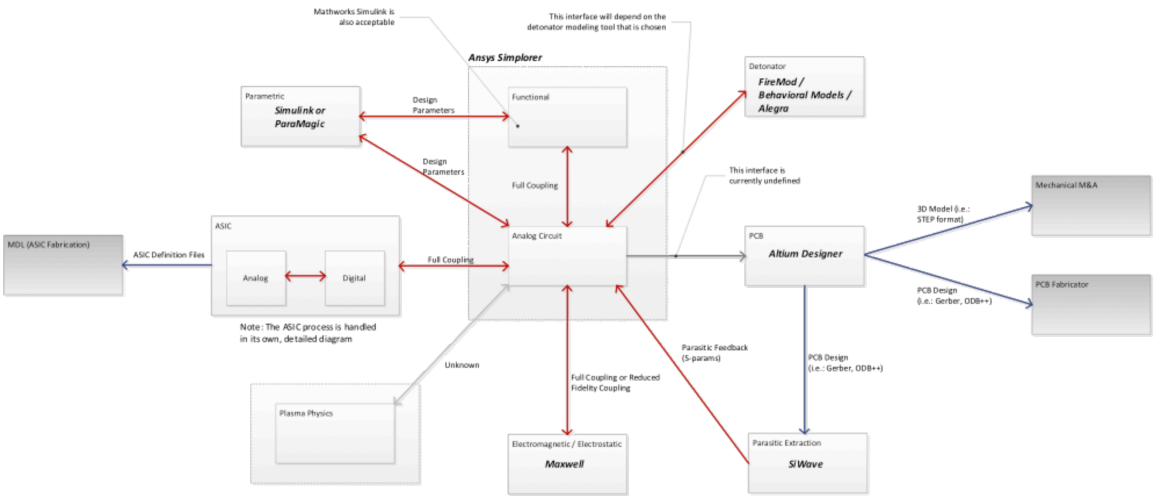


Figure 40. Example graphical summary of a modeling and analysis plan.[60]

Finally, the modeling and simulation plan is "executed" against the "as-is" capability portfolio. Analytical results are checked against actual performance data for model validation purposes. (This could be performed via the model if, e.g., SysML parametric diagrams have been constructed and the appropriate "plug-in" tool is available; that is, test results are imported and data analysis is performed to verify the results meet requirements.) The intent here is two-fold: (1) produce an *a priori* mission performance baseline against which proposes changes can be compared; and (2), establish a "validated" mission performance analysis tool that will be applied consistently against all proposed what if/to be changes (i.e., different solutions have to be evaluated using the same process, methods and tools in order to enable an "apples-to-apples" comparison; use of solution-specific tools will not produce results that can be compared in a valid trade-off analysis). The biggest challenge at present will be to interface current generation DoDAF-SysML-UML vendor tools with the necessary analysis tools (some functionality exists in this area, but is vendor dependent).

[60] Courtesy of Patrick O'Malley, SNL Organization 2627.

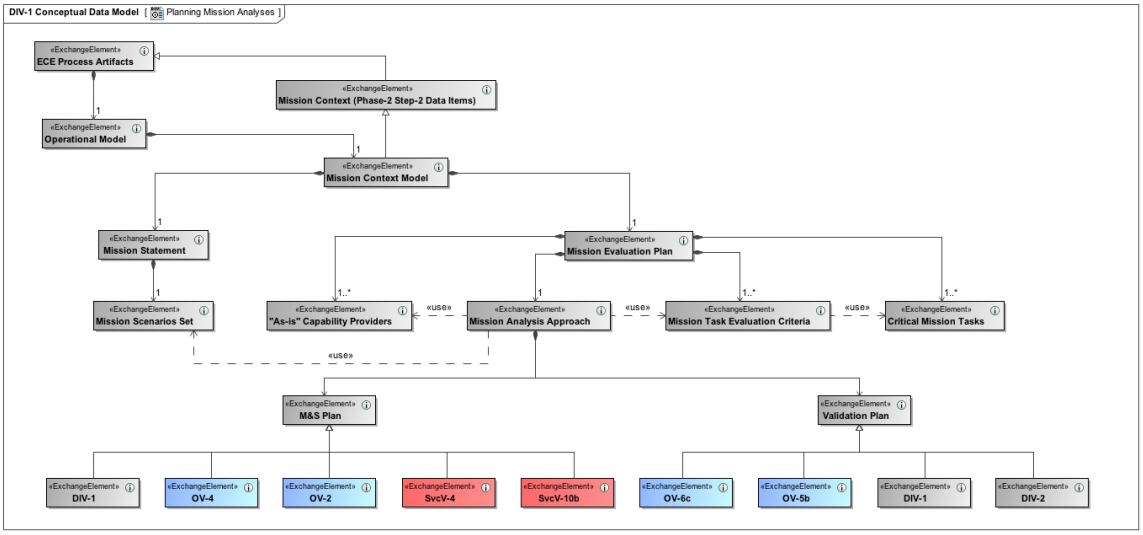


Figure 41. *M&S and Validation Plans* DoDAF conceptual data model.

3.2.2.3 Phase-2 Step-2 Summary

After completing the second operational modeling step, strategic requirements have been identified, analyzed, reconciled, and developed into one or more mission statements, each of which includes: (1) identification of critical mission tasks, associated standards (MOEs and MOPs) and important attributes; (2) allocation of mission tasks to one or more functional capabilities; (3) definition of bounding test cases (mission scenarios); and (4), a validated analytical methodology for assessing how well critical mission tasks (and so mission) will be performed given a set of appropriate capability definitions (configurations; e.g., DOTMLPF resource sets). Note that, collectively, the scenarios, tasks, standards, and assessment methods form the requisite integrated SoS verification objectives against which the integrated set of services (capabilities) should be evaluated.

3.2.3 Analyzing Requirements (Phase-2 Step-3)

The purpose of Step-3 is to translate the mission requirements modeled in Step-2 into a separate requirements specification for each critical capability identified in §3.2.2.1.3. That is, an independent definition for each capability is to be abstracted out from the mission model. Thus this step is concerned with the analysis and decomposition of the operational problem in terms of the required performance of the critical capability model elements involved in fulfilling the defined mission.

Consistent with the rest of this paper, the method described below follows a top-down modeling approach, and centers on producing self-consistent,[61] "black box," technical specifications. Once completed, each such model is often referred to as an *Elaborated Context Diagram* (ECD) since it represents a further elaboration of one or more elements of the context descriptions developed for the enterprise (mission) model of Step 1. The requirements engineering artifacts (set of modeling objects) produced in following this models-based systems engineering (MBSE) method collectively form what is also known as a *Technical Requirements Model* (TRM) of the particular capability, in this case, to which it applies.

For reference, a "black box" model describes an entity from the perspective of external observables as derived from multiple scenarios (e.g., use cases) in the form of a static, composite (i.e., scenario cross-cutting) view of input/output (I/O) flows. Generally speaking, a black box model will also include requirements in the following categories: functions; interfaces; controls; performance and quality of service; and stores. In addition, it will typically include non-behavioral (non-mission performance) requirements that have been expressed or imposed by the stakeholders.

With all of that said, however, a change in frame of reference must be introduced. While the intent is to here derive requirements—and eventually to define resource portfolios—for critical capabilities, strictly speaking a capability is a skill that arises from an ability to successfully use available resources to perform a task or produce an output (cf. §2.1.8),

[61] "Self-consistent" in that all stakeholder requirements conflicts have been resolved by this point.

and skills do not have interfaces or own resources. While it would be possible to define a system that is responsible for the desired input-to-output transformation (cf. Figure 1), preference will be given here to using the logical system construct of a *service*, in a service-oriented architecture (SOA) sense. *Services* have a number of attributes that are appropriate for this point in the problem decomposition, including technology agnosticism and abstraction. Therefore, at this tier in the architectural description, capabilities will be **allocated** to *services*, and so this step actually has the intent of producing a set of *service* ECDs (or TRMs).

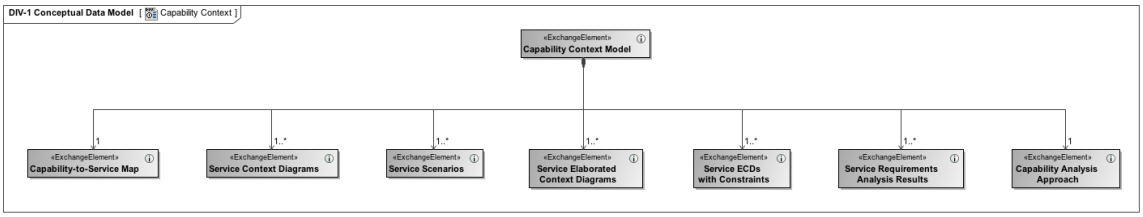


Figure 42. ECE Process Phase-2 Step-3 conceptual data model.

Table 6. *AV-2 Integrated Dictionary* of Phase-2 Step-3 Data Elements

Name	Definition
Capability-to-Service Map	Allocation of functional capabilities to services.
Service Context Diagrams	Basic context diagrams for each service, including terminations for all I/O resource flows.
Service Scenarios	Cross-cutting constructs of mission scenarios that describe required behaviors from a service perspective.
Service Elaborated Context Diagrams	Service context diagrams as elaborated with functional, performance, I/O, control, store, and physical requirements.
Service ECDs with Constraints	ECDs as elaborated with design constraints.
Service Requirements Analysis Results	ECDs annotated with service requirements variation analysis results.
Capability Analysis Approach	The modeling and simulation plan to be used for testing service-provided capabilities.

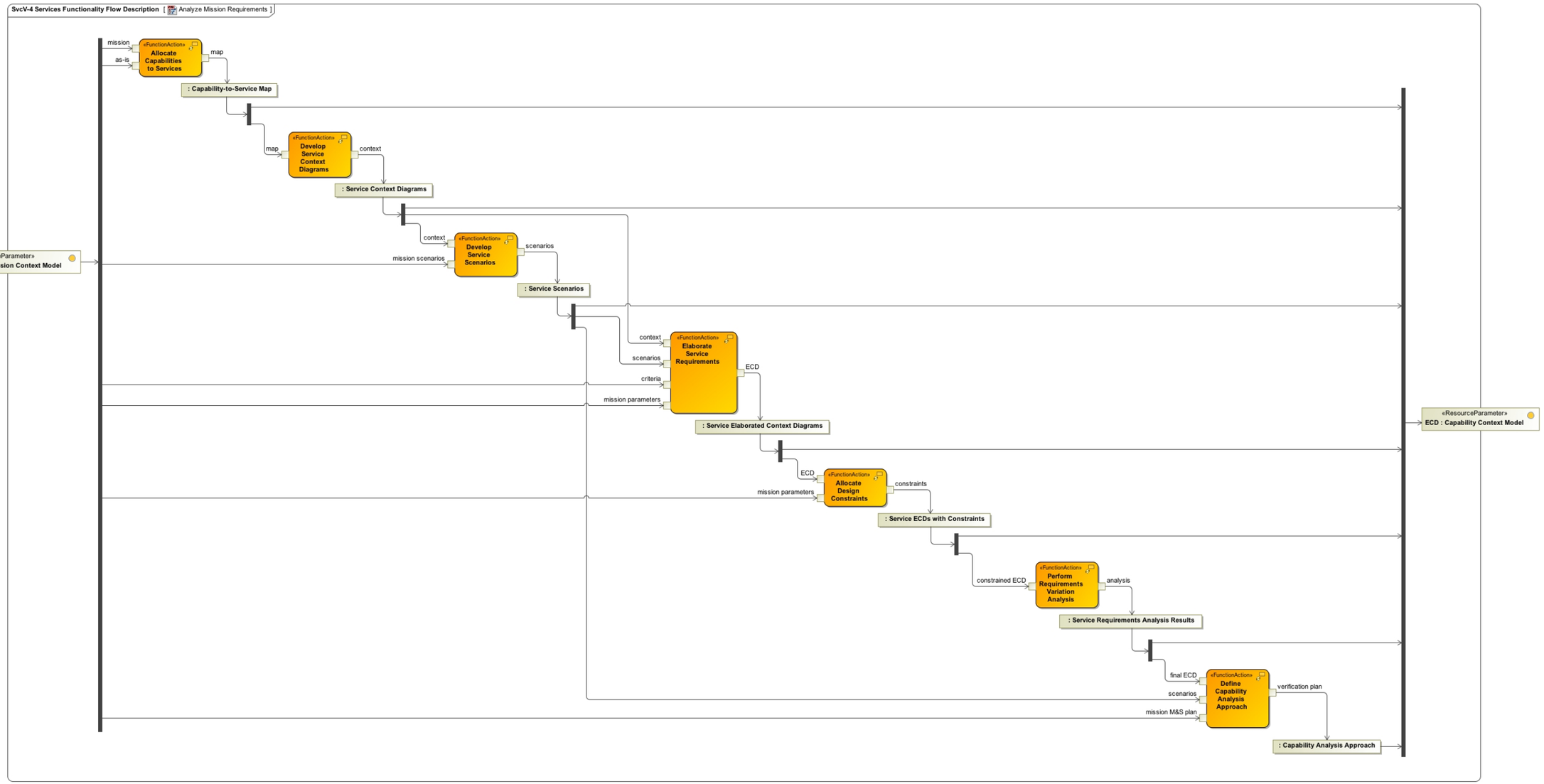


Figure 43. ECE Process Phase-2 Step-3 task flow.

Intentionally blank.

3.2.3.1 Mapping critical capabilities to services

In DoDAF, a mapping between the functional capabilities required and the named services[62,63] that will provide said capabilities is developed through the use, e.g., of a *CV-7 Capability to Services Mapping* matrix. Nominally a one-to-one mapping is used, and the resultant service hierarchy will naturally reflect the capability hierarchy as portrayed in the *CV-4* developed in §3.2.2.1.3.

However, it is possible—and preferable—to perform an initial partitioning of the problem at this step by considering the attributes that define a system-of-systems (cf. §2.1.2): operational independence, managerial independence, and geographic location. If partitioning is pursued, a copy of the *CV-4* can be updated by linking persons, person types, organizations, or organization types: (1) to allocated capabilities with "capability of performer" dependencies; and (2), to defined locations or location types in the respective element specifications. Relevant information to populate this version of a *CV-4* should be available in the *mission parameters set* produced in §3.2.2.1.1, as well as the "as-is" DOTMLPF documentation of §3.2.2.2.3. (Let the "as-is" inform but not artificially constrain the architecture at this point.) Note that service access elements can also be specified as being at a particular location or location type (traceability to operational or management oversight is, however, through the allocated capability). A service hierarchy is then composed by grouping services on the basis of these attributes (cf. §3.2.5.1). Following *CV-7* production, a graphical portrayal of the service hierarchy can be generated through the use of a *SvcV-1 Services Context Description* diagram.

Next a version of the *SvcV-1* should be developed that identifies resource flows[64] between services down to the desired "black box" level; these flows can be captured by using association relationships (e.g., as might be found in a SysML *bdd*), or by using a

---

[62] "Named" services as used here implies identification only; definition thereof is yet to be developed.
[63] Service model elements are actually typed "service access" in that they are only intended to represent the interfaces to services in order to provide abstraction from any internal logic.
[64] These flows are in reference to resources that are transformed (i.e., inputs consumed and outputs produced; cf. Figure 1) by a service—data, information, performers, materiel, and personnel—and **not** the DOTMLPF resource sets that represent a "bundle" of potential services (cf. §2.1.7).

resource-interaction-typed dependency (e.g., as might be found in a UML class diagram). Inter-service resource flows are identified from and traced to the operational exchanges found in relevant *OV-2, OV-5b* or *OV-6c* behavioral diagrams that detail the operational mission scenarios (cf. §3.2.2.1.2) otherwise defined in the operational concept (cf. §3.2.1.4); some simple data and information examples[65] include:

- Input—database query or update, sensor input, C^2 input
- Output—query result, sensor output

A third (or updated) version of the *SvcV-1* is now produced that adds a resource port to each service access (interface) for each resource flow in support of the next step (*SvcV-2* diagramming).[66]

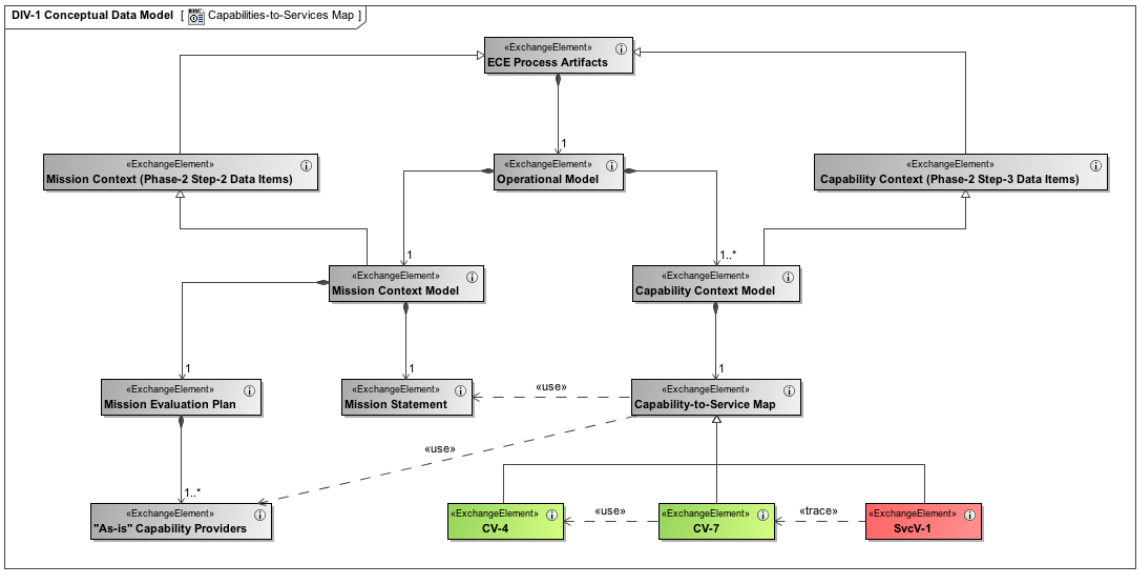


Figure 44. *Capability-to-Service* map DoDAF conceptual data model.

3.2.3.2 Developing an initial Elaborated Context Diagram (ECD)

At this point the process becomes multi-threaded in that, as noted above, detail is developed individually for all services identified as providing one or more critical capabilities, beginning with the production of a context diagram for each said service. Initially, an "unpopulated" ECD (or simply *context diagram* for short) is drafted for each

[65] As used here, "data" represent raw facts collected for processing while "information" is the processed facts, consistent with computer processing terminology; in the English language the two terms are interchangeable.

[66] UPDM, as built on SysML and UML, does not support port-to-port connections on an *SvcV-1*.

particular service on the basis of the enterprise (mission) model that identifies the relationship between a specific service and its environment—including external systems and users (i.e., other "services")—in terms of aggregated external resource flows. The construct of choice is an internal block diagram (i.e., SysML *ibd* or UML composite structure diagram); this option is identified as an *SvcV-2* in UPDM, and is consistent with the DoDAF viewpoint model, but—at least as implemented in MagicDraw—the diagram type is named *SvcV-2 Services Resource Flow* ***Internal*** *Description* [emphasis added].[67] Nominally each context diagram will contain a minimum of three services:[68] the one to be modeled as a black box, one that supplies the input, and one that receives the output; the actual elements will, of course, be selected and portrayed such that the terminations of all resource flows (i.e., I/O terminations) are identified. Appropriate I/O connections are then modeled between service ports (as identified earlier in an *SvcV-1*). Ports and connections should be consistently typed.

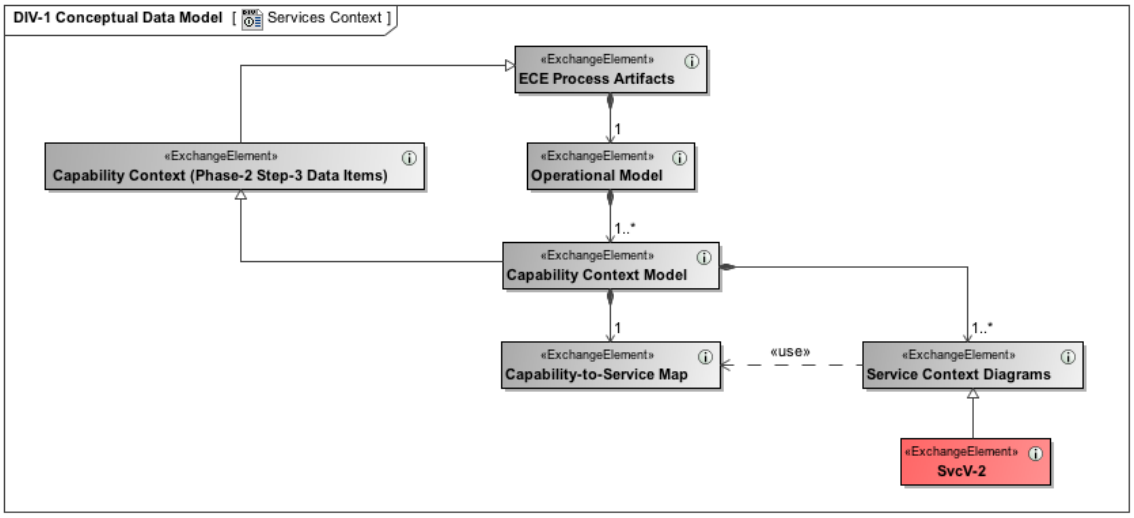


Figure 45. *Service Context Diagrams* DoDAF conceptual data model.

---

[67] DoDAF and UPDM name the *SvcV-2 model* as *Services Resource Flow Description*. The later is, however, used by MagicDraw for an *SvcV-2* constructed on the basis of a UML class diagram, which construct, while equally valid per UPDM, is not of use here. That is, MagicDraw uses different names to differentiate between the different diagram types that can serve as the foundation for the *SvcV-2* DoDAF and UPDM viewpoint model. While reasonable, it is important to note specification deviations.

[68] The services as shown on an *SvcV-2* are typed "resource role" that, **in this case only**, indicates that a "part" relationship exists with the higher-level service. Later in the architecting effort, when capability resource portfolio options ("capability configuration") are being explored in the context of (i.e., as parts of) a service, this nomenclature will actually make some sense.

(In SysML, an ECD that takes the form of a specialized type of *ibd*. Systems are represented by blocks, and arrows are used on connectors to depict the direct of item flows. Blocks are used to capture item flow definitions (stereotype of "I/O" preferred), which will later form the basis for interface control requirements and documentation. Inheritance and aggregation properties are often used to advantage in describing composite flows. Block attributes are used to capture logical and physical interface characteristics. In UML, *class* diagrams can be used.)

3.2.3.3 Developing Service Scenarios

Scenarios from the enterprise model are further developed using behavioral diagrams, as appropriate, in order to produce service scenarios. In DoDAF, the diagrams types that can apply include *SvcV-4, SvcV-10b Services State Transition Description,* and *SvcV-10c Services Event-Trace Description.* Service scenarios represent integrating, cross-cutting constructs of the set of enterprise-level (mission) scenarios (cf. §3.2.2.1.2), although it should be recognized that further development (analysis and definition) may be required. In turn, each service-level scenario will be used later in the process to elaborate (add more detail to) the requirements for the respective ECD (i.e., each scenario establishes and refines a subset of the operations and attributes of a service), as set forth in §3.2.3.4.

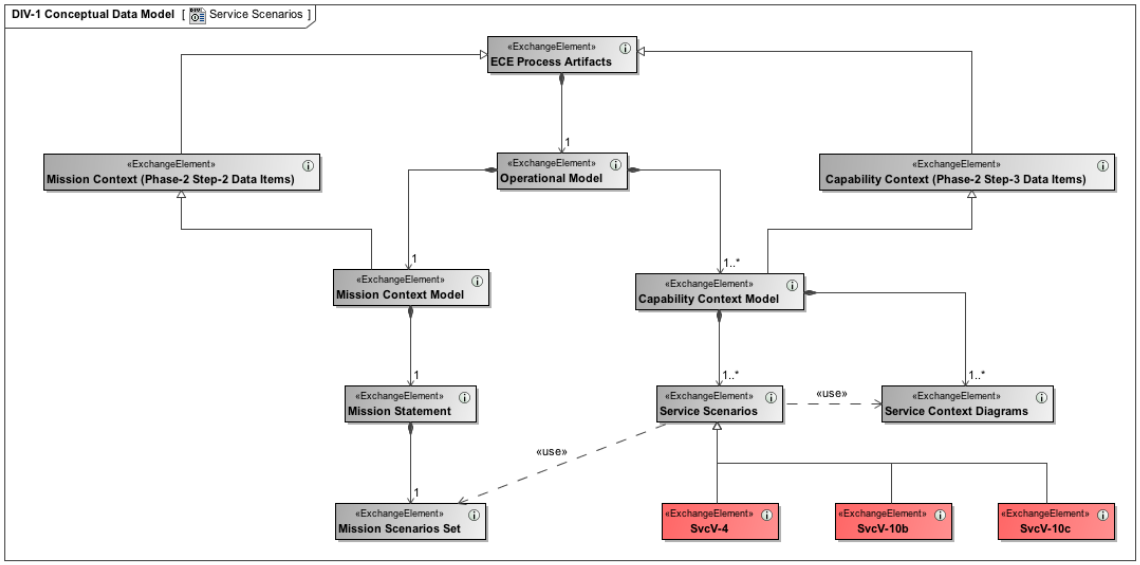


Figure 46. *Service Scenarios* DoDAF conceptual data model.

In order to avoid scenario "explosion", it is often useful to prioritize them (e.g., by probability or criticality) and identify a set of bounding cases from which the required operations and attributes will be derived. Scenario prioritization is intended to:

- ensure high-probability mission scenarios are addressed;
- ensure scenarios exercise each interfaced external service;
- ensure stressing scenarios which drive mission performance are addressed; and,
- include nominal (baseline) and exception cases (e.g., failure conditions, including those generated by external threats).

An initial set of derived service scenarios which address internal support (e.g., "hotel services") requirements may also be identified at this point. Some of these scenarios might be identifiable from the enterprise-level analysis. However, typically support requirements are handled at this point as assumptions, or ignored all together, and will evolve as the architectural design is further developed. Although not specifically addressed herein, for reference, support scenarios may include:

- performance & fault monitoring and recovery;
- security;
- configuration management;
- database administration;
- system backups, recovery, and archive;
- system installation;
- power distribution & control; and,
- environmental control.

3.2.3.4 Elaborating Service Requirements

The *service scenarios* developed in §3.2.3.3, supplemented with information from the *mission context model* of §3.2.2, are now used to add requirements detail to (i.e., elaborate) the relevant *service context diagrams* generated in §3.2.3.2. The resulting model portrayals so developed of a "black box" system (service in this case) are known as *elaborated* context diagrams (ECDs). The particular "black box" requirements of interest here can be categorized by the following list, and for which process details are provided further below:

- functional requirements;
- performance/QoS requirements (e.g., response time, accuracy, reliability, ...);

- I/O requirements (function inputs, outputs);
- control requirements (events, conditions, states and resulting function activation/deactivation);
- store requirements (e.g., data stores); and,
- physical requirements.

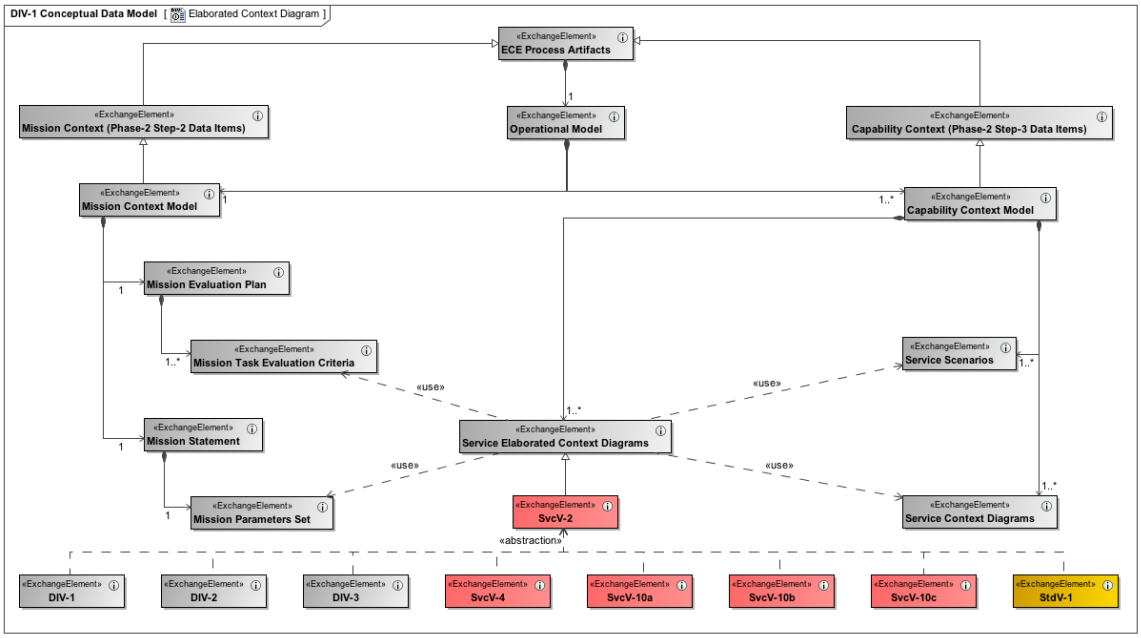


Figure 47. Service *Elaborated Context Diagram* DoDAF conceptual data model.

3.2.3.4.1 Detailing Functional Requirements

Service functions are activities identified in relevant *service scenarios* produced in §3.2.3.3 (e.g., in DoDAF, a «Function» or «FunctionAction» on a *SvcV-4*). They are captured in the respective block (class) as an operation (e.g., in DoDAF, as a «ResouceOperation» in a «ServiceAccess» block on a *SvcV-2*). Corresponding operation input, output, control and performance characteristics are documented as outlined in the sections that follow below. Algorithms and mathematical relationships are used to specify detailed functional requirements. In general, these are defined in equations, tables, flow charts, truth tables, or pseudo code that may be referenced in the function-specification documentation field or captured in an *SvcV-10a Services Rule Model;* Fit-for-Purpose views (e.g., a SysML parametric diagram) may also be appropriate. Polymorphism is applied to specify operations that have different methods dependent on the operation signature; e.g., track (air target) vs. track (ground target).

3.2.3.4.2 Detailing System Performance Requirements

Functional performance requirements (including QoS) are now defined for all service functions identified in §3.2.3.4.1. In general, these will be the critical system MOPs that impact mission MOEs as identified during development of *mission task evaluation criteria* in §3.2.2.2.2. Corresponding parameters should be defined with specified attribute values or constraints, appropriately stereotyped (e.g., «performance» or «MOP»), and associated with the service in question (e.g., in DoDAF, these are captured as a «ServiceAccess» block «Property» on a *SvcV-2*). This step may also require an initial allocation of mission performance requirements across multiple performance attributes (properties) where multiple functions (operations) are involved, with final allocation to be determined as part of an optimization and evaluation ECE process activity that takes place later; performance requirement allocation is typically found in situations that involve:

- response time;
- throughput;
- accuracy; and,
- specialty requirements (e.g., reliability, availability).

It should be noted that certain types of performance requirements, particularly those associated with mission-critical performance timelines, do not lend themselves to definition by a simple property value. Rather, they must be defined, e.g., in a timing diagram or as time constraints on a sequence diagram (*SvcV-10c*) as derived from applicable scenarios.

3.2.3.4.3 Detailing Interface Requirements

Each resource flow captured in *service context diagrams*, as developed in §3.2.3.2 (e.g., in DoDAF, on an *SvcV-2*), is reviewed—and updated if necessary—to verify it serves as an "input to" or "output from" an activity identified in a relevant *service scenarios* as produced in §3.2.3.3 (e.g., in DoDAF, a «Function» or «FunctionAction» on a *SvcV-4*). Each resource conveyed at an interface is also verified to be appropriately typed—or is typed at this point (in DoDAF by use of an «EntityItem» construct that further details the corresponding operational resource «ExchangeElement»). Aggregation and inheritance

can be applied to I/O resources to simplify and structure the content (in DoDAF, e.g., use *DIV-1* and *DIV-2* diagrams, and «LogicalDataModel» constructs). In cases where technology constraints apply (see the constraints section below), I/O physical characteristics (e.g., content encoding, signal characteristics) are appropriately modeled as well as (in DoDAF, this information can be captured in *DIV-3 Physical Data Model* diagrams, an *StdV-1* table, or as a constraint in an *SvcV-10a*).

3.2.3.4.4 Detailing Control Requirements

While "statelessness" is a preferred service attribute, this is not always possible (or perhaps not even desirable in certain cases). For example, it may be intended for a service to respond to various external events by enabling or disabling one or more functions (i.e., by changing its state). Typical states that may be found include: normal operations, off-nominal or degraded mode(s) of operation, backup and restore, maintenance, and training. External events may include explicit "C^2 signals" or implicit "control" signals (e.g., environmental conditions). Such events are specified by defining particular I/O attribute values, while noting that an event could actually be a complex Boolean or algebraic expression of multiple attributes.

This sub-step begins by reviewing, and supplementing as necessary, the *service scenarios* of §3.2.3.3 to ensure responses to all control inputs are defined (e.g., sequencing of service actions or functions) and, if necessary, further refined. In DoDAF specifying control response is accomplished by defining states, events and state transitions on an *SvcV-10b* diagram, control flows to «FunctionAction» elements on an *SvcV-4* flow diagram, or as events/messages to the service on an *SvcV-10c* with the addition of states along the time line. In addition, applicable *DIV-2* «EntityItem» attributes are verified to have been specified—or are now elaborated—with values or expressions, as appropriate, to characterize the control signal itself.

3.2.3.4.5 Detailing Store Requirements

Store requirements may be identified in relevant *service scenarios* of §3.2.3.3 as resource flow inputs to a service that must be stored to support black box operations that occur later in time relative to receipt (e.g., targeting information). Store requirements are

captured as stereotyped properties (e.g., «store») associated with the service with specified attribute values or constraints (e.g., in DoDAF, a store attribute can be captured as a «ServiceAccess» block «Property» on a *SvcV-2*). Example store attributes include:

- number of instances;
- type, size, range;
- persistence (scenario, mission persistent, archive); and,
- Create/Retrieve/Update/Delete (CRUD) frequency.

3.2.3.4.6 Physical Requirements

Any service physical requirements (such as defined in the *mission parameters set* of §3.2.2.1.1) are captured as stereotyped properties (e.g., «physical») with specified attribute values (e.g., in DoDAF, a store attribute can be captured as a «ServiceAccess» block «Property» on a *SvcV-2*). If these requirements are constraints, see below.

3.2.3.5 Identifying Design Constraints

Constraints are imposed on the architectural solutions to be developed (functional or physical). Service design constraints are passed down and allocated or derived from the enterprise-level model via the mission context model as part of the mission parameters set identified in §3.2.2.1.1 (in DoDAF this will be in the form of a *CV-1* diagram, and possibly *OV-6b* and *OV-5a*). Typical enterprise-level constraints stem from considerations that include:

- doctrinally-correct operational activity sequencing and timing;
- information assurance context (e.g., types of system or service data to be protected, such as classified or sensitive but unclassified, and expected information threat environment);
- physical threats, including environmental conditions; and,
- planning level (e.g., tactical, operational, or strategic).

"Pass-through" constraints may also be identified that will impact the physical solution to be developed downstream in the process. These may be of a type that mandates "white box" (architectural) solutions. More typically these involve COTS/GOTS, legacy systems, or standards in a requirement that imposes part of the solution (e.g., use of a specified component, interface, or algorithm).

Because design constraints can place severe restrictions on the available solution space, limit performance, and often dominate final costs, it is imperative that they be clearly identified and agreed to by the appropriate stakeholder(s). A method for identifying design constraints can be summarized as follows.

1. Identify sources of constraints (include "as-is" analysis).
2. Identify specific constraints. For physical architecture "pass-through" requirements, include consideration of the following general categories:
    a. hardware;
    b. software (application, middleware, OS);
    c. data;
    d. users/procedures;
    e. interfaces;
    f. algorithms; and,
    g. security.
3. Capture the final design constraints list and link to the respective service ECD. In DoDAF, simple constraints can be captured as a «ServiceAccess» block «Property» on a *SvcV-2* (use block constraints in SysML); however, in general, it is more appropriate to capture constraints in an *SvcV-10a*.
4. Validate design constraints with stakeholder.

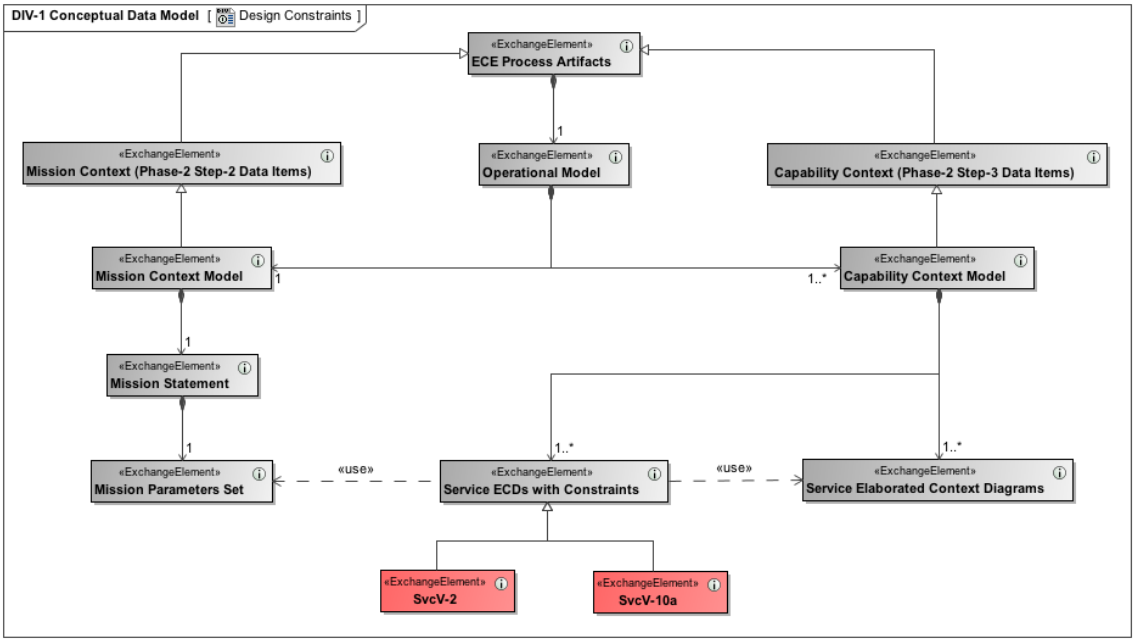


Figure 48. Service design constraints DoDAF conceptual data model.

3.2.3.6 Performing Requirements Analyses

In general, systems engineering processes specify that requirements analyses be performed to include the following types: requirements variation; trade off; effectiveness; and risk assessment. At this point in the ECE process, however, technology options and related specifications have yet to be defined, and so the usual case is to only perform a variation analysis. The concern here is to identify potential changes in requirements due to changing customers, missions, applications, or enhancements. A simple, five-step requirements variation method follows here below.

1. Identify type of change for each requirement (here service requirements). Categories to consider include:
    - changing interfaces;
    - system growth (e.g., number of users, data store);
    - increased mission performance;
    - new functionality (tasks);
    - COTS upgrades;
    - technology growth (new types of or advanced technologies); and,
    - changes in physical location or governance.
2. Define a probability of change for each requirement (e.g., high, medium or low).
3. Evaluate impact to or sensitivity of performance relative to MOEs for requirements variation to determine requirement mission criticality. (Beware of mixing qualitative, semi-quantitative, and quantitative techniques!)
4. Capture requirements variation results to support possible revision or planned evolution (multiple deliveries, aka block upgrades).
5. Provide the variation analysis results as input to a risk analysis. For example:

    Risk measure = probability of change * criticality

The results of any variation analysis performed should be captured in the definitions for the respective model elements (e.g., as a block property). In terms of graphical portrayal, no viewpoint model is defined within DoDAF to capture such information, and a "Fit-for-Purpose" view would have to be created. While such views will be highly dependent upon the nature of the requirements and analyses performed, one possibility to consider would be to simply update (or copy and update) the diagrams produced in §3.2.3.4 above, as updated with constraints in §3.2.3.5, by annotating the respective requirement model elements (e.g., with typed comment statements or figures).

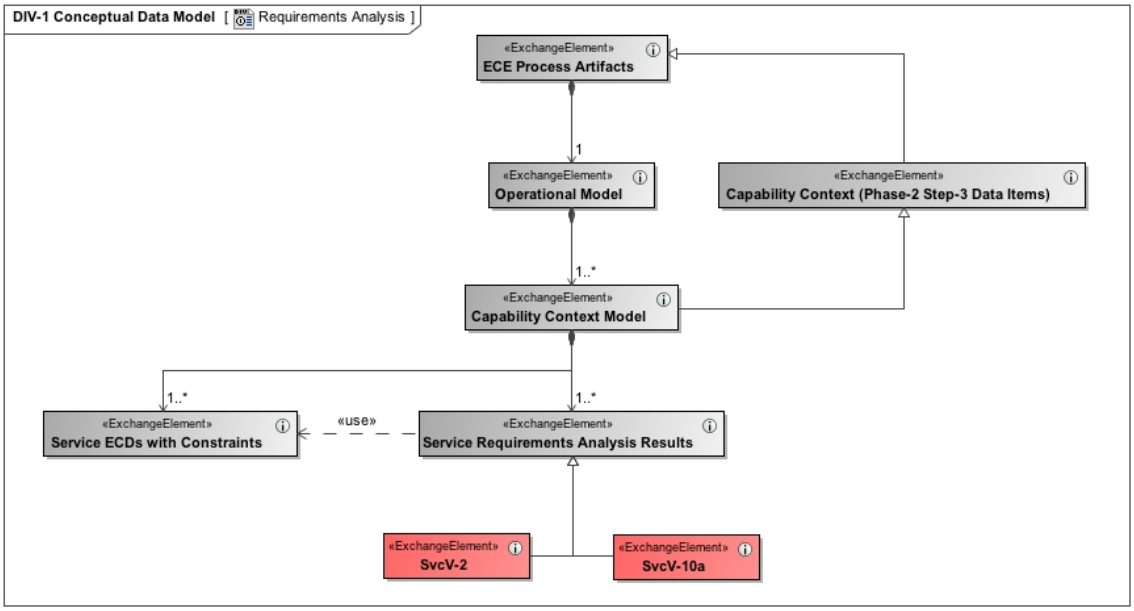


Figure 49. *Service Requirements Analysis* DoDAF conceptual data model.

3.2.3.7 Choosing the Service Capability Analysis Approach

The analytic approach to be followed (for verification in this case) should be identified, "fleshed out," and documented in a capability (service) modeling and simulation (or analysis) plan that will be executed by the analytical team (preferably a models-based *view* using, e.g., DoDAF artifacts as illustrated in Figure 50; cf. footnote 38 and corresponding discussion in §3.1 and §3.2.2.2.4). This would include a description of the collection of tools that will be used to transform resource inputs into resources outputs by exercising the service behavioral models, estimate outcomes (vis-à-vis MOP requirements), and present results.

As for the mission level, the choice of modeling technique(s) will be determined by consideration of, e.g., the nature of the problem and practical (resource) limitations. On the basis of the service scenarios, verification test cases are defined, and from which analysis approaches are selected. Most commonly service capability test cases are captured in sequence diagrams (*SvcV-10c, sd*), although sometimes activity diagrams can be used to advantage (*SvcV-4*, *act*). Test cases must include identification of the data to be collected (*DIV-1, DIV-2*, *bdd*, *class*). Ideally, service-level analysis models would be compatible with—and even serve as "plug-ins" to—mission-level analysis models.

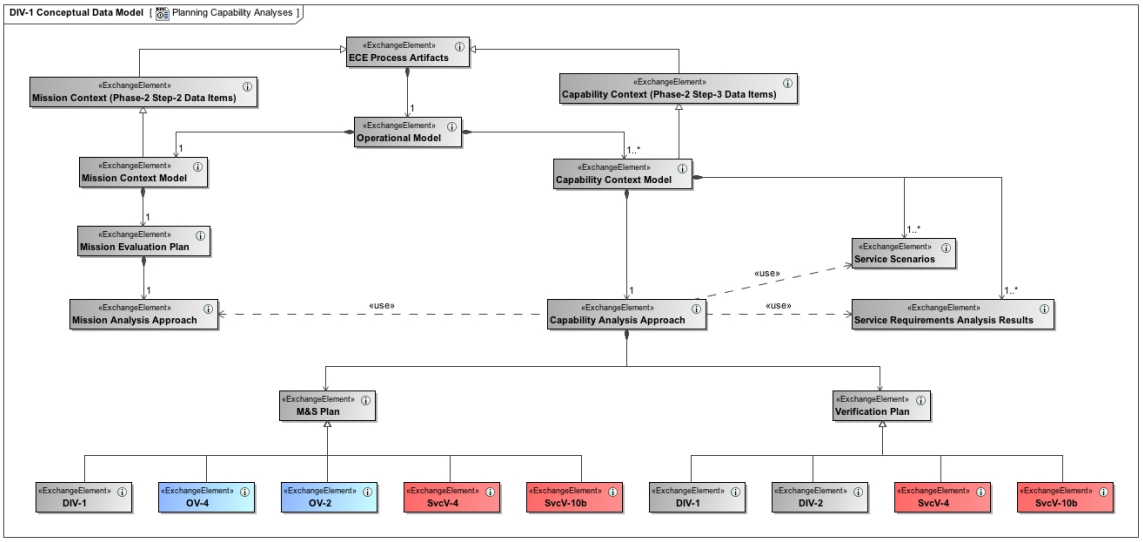


Figure 50. *M&S and Verification Plans* DoDAF conceptual data model.

The final task of this step is to "execute" the modeling and simulation plan against the "as-is" capability portfolio. Results are checked against actual performance data for model verification purposes. As before, the intent is two-fold: (1) produce an *a priori* service capability performance baseline against which proposes changes can be compared; and (2), establish a "validated" service performance analysis tool that will be applied consistently against all proposed changes. Also as previously stated, the biggest challenge at present will be to interface current generation DoDAF-SysML-UML vendor tools with the necessary analysis tools.

3.2.3.8 Phase-2 Step-3 Summary

After completing the third operational modeling step, the *Technical Requirements Model*—specification—of **each** required "black box" service (critical mission capability) has been developed. In other words, the combined set of interacting services has been specified in terms of the capabilities to be provided, along with the required levels of performance (MOPs), that will enable the mission to be satisfactorily met (MOE). The baseline includes: required functions and associated I/O; the required external interfaces; the required performance, physical, and quality characteristics that impact how well the functions must be performed or define a physical characteristic; the required control in

terms of input events and preconditions that determine when functions are performed; and the required items that the system must store including data, energy, and mass. In addition, validated analytical approaches for evaluating service-level capability performance is available, and individual “black box” verification plans have been defined that include bounding test cases (service scenarios) and performance requirements.

3.2.4 Describing Capability Behaviors (Phase-2 Step-4)

One of the key steps in the ECE process (as with any system design process) is the development of *functional* architectures. This process step can thus be described as having the objective of decomposing the operations identified in the service-based capability scenarios developed in the previous step into one or more viable functional architectures. Decomposition is carried out as deeply as needed to define the input-to-output transformations that must be performed. The choices made—by design intent or otherwise—that become reflected in a functional architecture are generally considered to have first-order impacts on issues like cost, appearance, usability, profitability, safety, and marketability on the final system (including system of systems) design. Because of this, functional architecture development also appears as a fundamental principal behind Value Engineering.

A functional architecture identifies and describes what functions (and how well) a "system" (here service capability) must perform in order to be successful, how these functions are related to each other both temporally (i.e., sequencing) and in terms of interfaces (e.g., data flows), and under what operational concepts (e.g., use cases) and environmental conditions they must be performed. The best functional architectures are technology agnostic (i.e., they do not address **how** functions will be performed), and so intentionally decouple requirements from implementation, leaving physical architecture trade spaces unbiased. Collectively the set of functional design artifacts produced serve to define a "white box" view of the system in question. Executable model(s) created on the basis of the functional architecture can be used to demonstrate the capabilities the architecture enables and to evaluate behavioral properties such as checking logical consistency (e.g., no deadlocks), functional sequencing, and resource provisioning.

The "classical" approach taken by systems engineers in developing a functional architecture—as followed here—is known as *structured* analysis. To create a robust solution, functional architectural development will rely on integrating the results of both decomposition and composition techniques. The required input is the set of documented, validated system/capability/service-level requirements, as derived from the user or

stakeholder requirements, as produced in the previous ECE process step. In review, the input for each required service capability includes: (1) an elaborated context ("black box") diagram that captures all interfaces and the required functionality; (2) system (service) scenarios that establish how and under what conditions a capability is to be invoked (typically each scenario establishes and is used to refine definition of a subset of the operations and attributes of a system); and (3), system (service capability) technical requirements and other constraints. Critical to this level of architecting is an understanding of the level of functionality achievable within program constraints and risk. Where multiple functional solutions exist that meet the full set of threshold performance (mission) requirements, trade space and risk should be analyzed and assessed against desired functional performance in order to stay within program constraints; note that while this is useful to narrow the number of functional solution sets carried forward to a manageable level, it is not necessary (nor perhaps even desirable) to down select to a single architecture.

The output of the functional architecting step serves as a necessary input into logical architecting (although, less preferably, it can be used directly as the input into physical architecting). The functional model (functional architectural description) can be decomposed into six "deliverables" as described in Figure 51 and Table 7. The requisite architecting steps and data item flows are modeled in Figure 52.

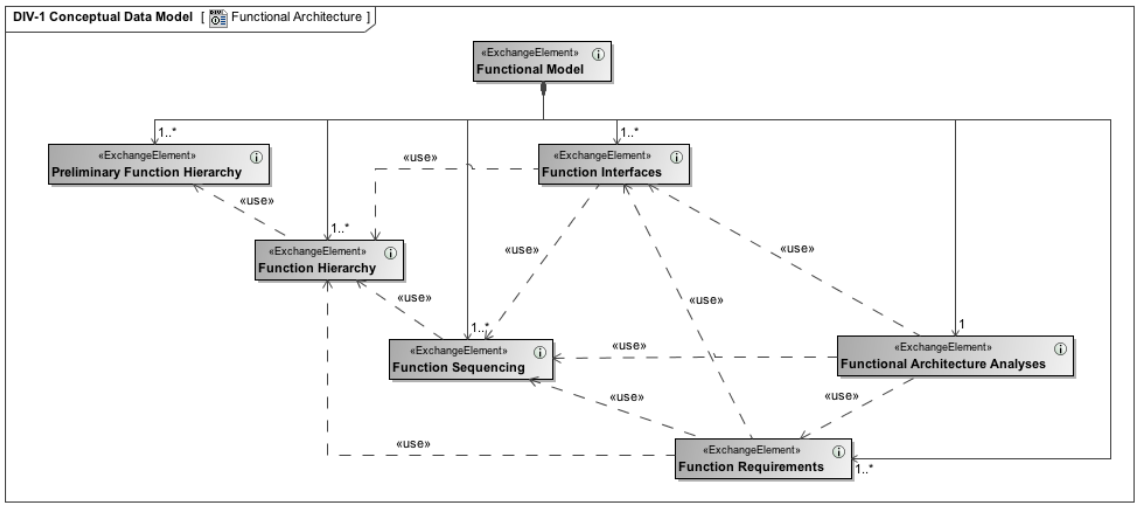


Figure 51. ECE Process Phase-2 Step-4 conceptual data model.

Table 7. *AV-2 Integrated Dictionary* of Phase-2 Step-4 Data Elements

Name	Definition
Preliminary Function Hierarchy	Functional hierarchy produced by decomposition of ECD operations.
Function Hierarchy	Functional hierarchy produced through: (1) composition by evaluation of service scenarios; and (2), reconciliation and integration with the hierarchy produced by decomposition.
Function Sequencing	Logical sequence of function execution for specified operational use of the represented capability.
Function Interfaces	Identification and definition of all function-to-function interfaces.
Function Requirements	Allocation of all pass-through, decomposed and derived requirements and applicable constraints to the functional level.
Functional Architecture Analyses	Results of functional architecture and requirements validation activities.

Intentionally blank.

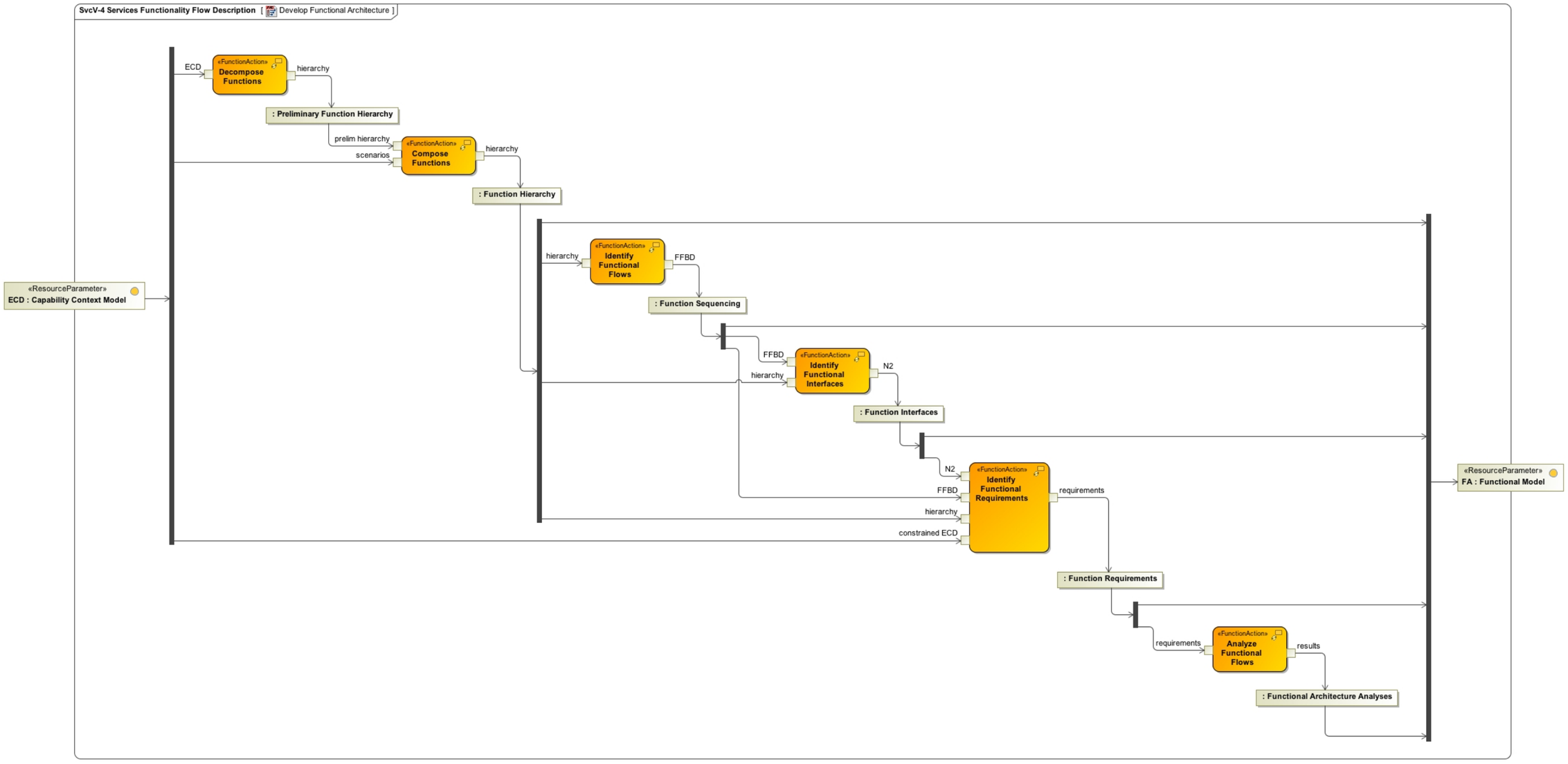


Figure 52. ECE Process Phase-2 Step-4 task flow: *Develop Functional Architecture.*

Intentionally blank

3.2.4.1 Decomposing "System" Functions

Decomposition (aka top-down structuring) begins with a top-level function (i.e., an operation identified in an ECD) that is partitioned into several, second-level functions (or sub-functions). Typically this is repeated to develop third—and even fourth—level functions, depending upon the amount of detail available in the system operational scenarios, but only insofar as needed to understand how the outputs are to be produced. The architect should avoid large numbers of functions at a given level, and eliminate details if they do not aid in understanding behavior. Alternate decompositions should be sought for in order to create trade space. Decomposition is noted to be efficient and particularly successful when the system is an update or variation of an existing system. Accepted practice captures the functional hierarchy in the form of an inverted "tree" structure (in DoDAF, use a *SvcV-4* diagram to illustrate the «Function» compositional hierarchy; in SysML use a *bdd* to capture activities and actions; in UML use a *class* diagram). Beyond the obvious design implications, a hierarchy is also important as a communication tool to both engineers and stakeholders.

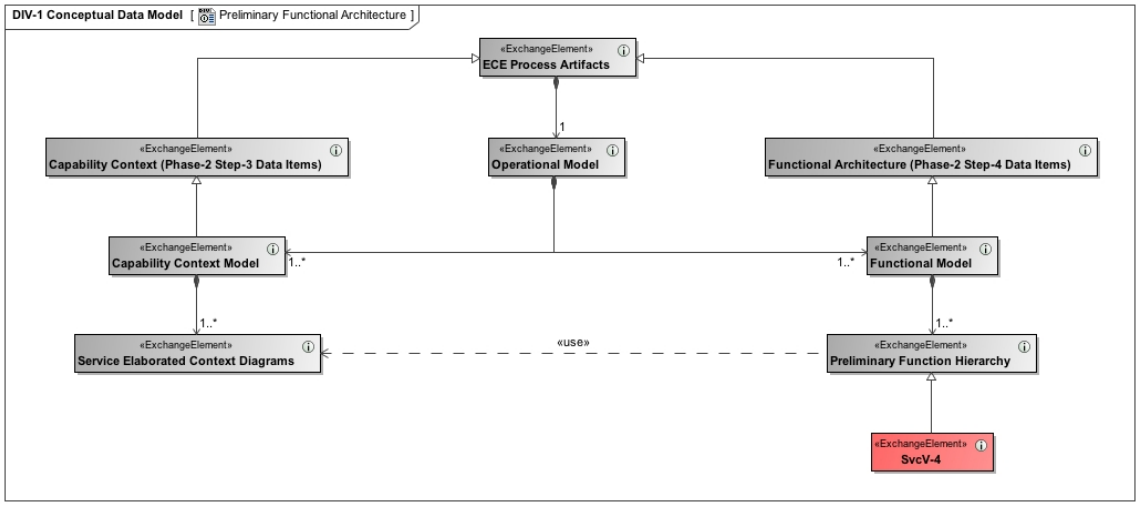


Figure 53. *Preliminary functional architecture* DoDAF conceptual data model.

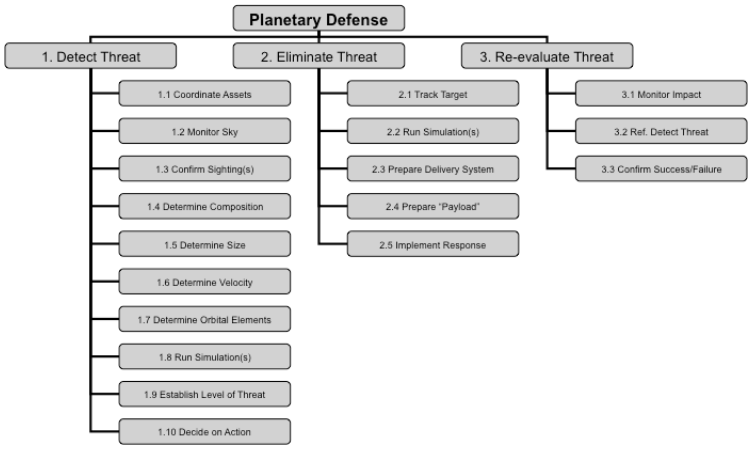


Figure 54. Functional architecture of a planetary defense system.[69]

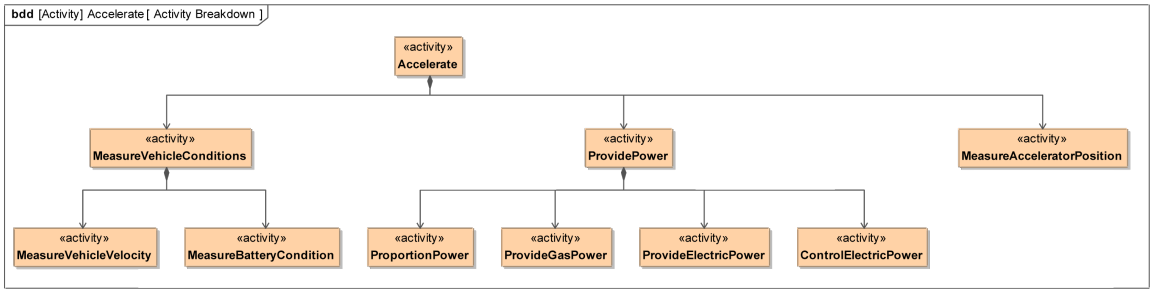


Figure 55. Functional hierarchy example in SysML.[70]

One difficulty associated with decomposition is that the procedure is somewhat unguided. However, given that the ECD supplied as input to this step was accurately and completely developed, a generic partitioning scheme can be applied as outlined below. Functions should be identified to:[71]

- interface with each external system
- receive (e.g., format or translate) each system input
- produce each system output
- control the system
- provide required system internal support services[72]

---

[69] Taylor, et al., "Planetary Defense System: An Advanced Systems Engineering Approach," *Revolutionary Aerospace Systems Conference—Academic Linkage,* May 2, 2006.
[70] Adapted from OMG SysML v1.2, p. 190.
[71] Adapted from Hatley, D. J., and I. A. Pirbhai, *Strategies for Real-Time System Specification,* Dorset House, New York, 1988.
[72] E.g., manage data or files, manage communications, manage faults or recovery operations, monitor system performance, manage system security, manage configuration, provide environmental controls, provide physical support.

Following this approach, it should be noted that decomposition must conserve I/O (i.e., use and produce all system-level inputs and outputs, adding no new ones, and with all sinks and sources identified—no I/O disappearing into or appearing out of "thin air"!). Other methods do exist, particularly in the field of Value Engineering.[73] Note that sub-functions of a function should be at the same level of abstraction.

Successful decomposition also requires that functions be identified by verb-noun word pairs. The verb is to be used to describe an action or activity that is to be performed, and **not** an objective or performance goal (e.g., *maximize* is generally not appropriate). It is highly recommended that commonly accepted taxonomies should be used as this will, in general, lead to more repeatable and meaningful results (e.g., DoD references like the *JCSFL*, or published lists of engineering functions[74]).

3.2.4.2 Composing System Functions

Composition (aka bottom-up structuring) identifies functions by modeling each "simple" or "subsystem" scenario (e.g., those involving input-function-output triplets, or a single activity from a system scenario). Classically these scenarios have been modeled using network diagrams (e.g., data flow diagrams, Petri networks, and bubble charts). Current MBSE practices would capture these as activity diagrams (e.g., DoDAF *SvcV-4* (flow)), often supported by use cases and associated, supporting diagrams.

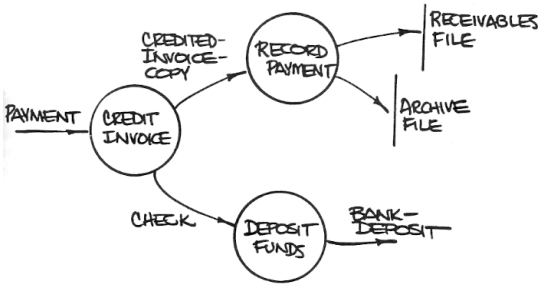


Figure 56. Data flow diagram example.[75]

---

[73] See, for example, Wixson, James R., "Function Analysis and Decomposition using Function Analysis Systems Technique," *INCOSE '99*, Brighton, England, June, 1999.
[74] See, for example, Hirtz et al., "A Functional Basis for Engineering Design: Reconciling and Evolving Previous Efforts," *Journal of Research in Engineering Design,* Vol. 13, Issue 2, March 2002.
[75] DeMarco, Tom, *Structured Analysis and System Specification*, Prentice Hall, Inc., New Jersey, 1978, figure 11, p. 47.

After all of the sub-functions have been so identified, they are organized into similar groups, then these groups into higher groups, and so on, until a hierarchy is formed from bottom to top and captured, e.g., in a *SvcV-4*. For complex systems the composition approach represents a substantial amount of work. The principal advantage is that it is so comprehensive that it is less likely to omit major functions compared to decomposition. Composition is strongly recommended when the system is unprecedented or represents a radical departure from an existing system. Most often composition is conducted to augment decomposition as a means to help assure completeness.

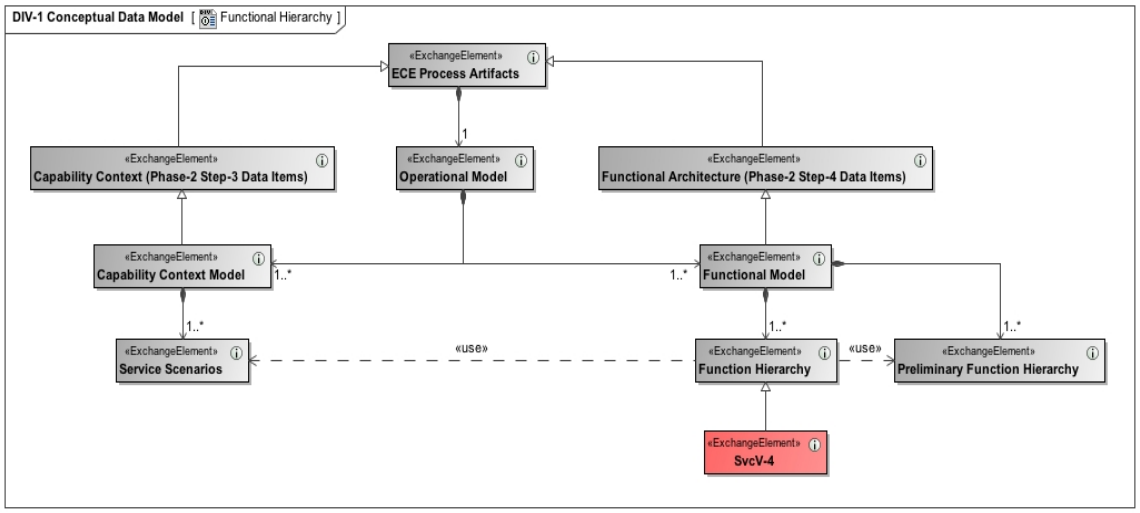


Figure 57. *Functional Hierarchy* DoDAF conceptual data model.

3.2.4.3 Mapping Functional Flows

Following development of a functional hierarchy, the *sequential* relationship (**not** interfaces) of all functions that must be accomplished by the system is defined. Each function is identified and related to the other functions in a logical sequence at the same indenture level such that any specified operational use of the system can be traced in an end-to-end path. This mapping is repeated for each lower level in turn, illustrating how sub-functions are organized to form part of larger functional areas.

Perhaps the most prevalent method for documenting flow-mapping[76] results has been through the use of functional flow block diagrams (FFBD), as illustrated below. IDEF0 models have also been used for this purpose. In current modeling paradigms, functional flow diagram equivalents can be represented in activity diagrams (DoDAF *SvcV-4*) through the use of control flows.

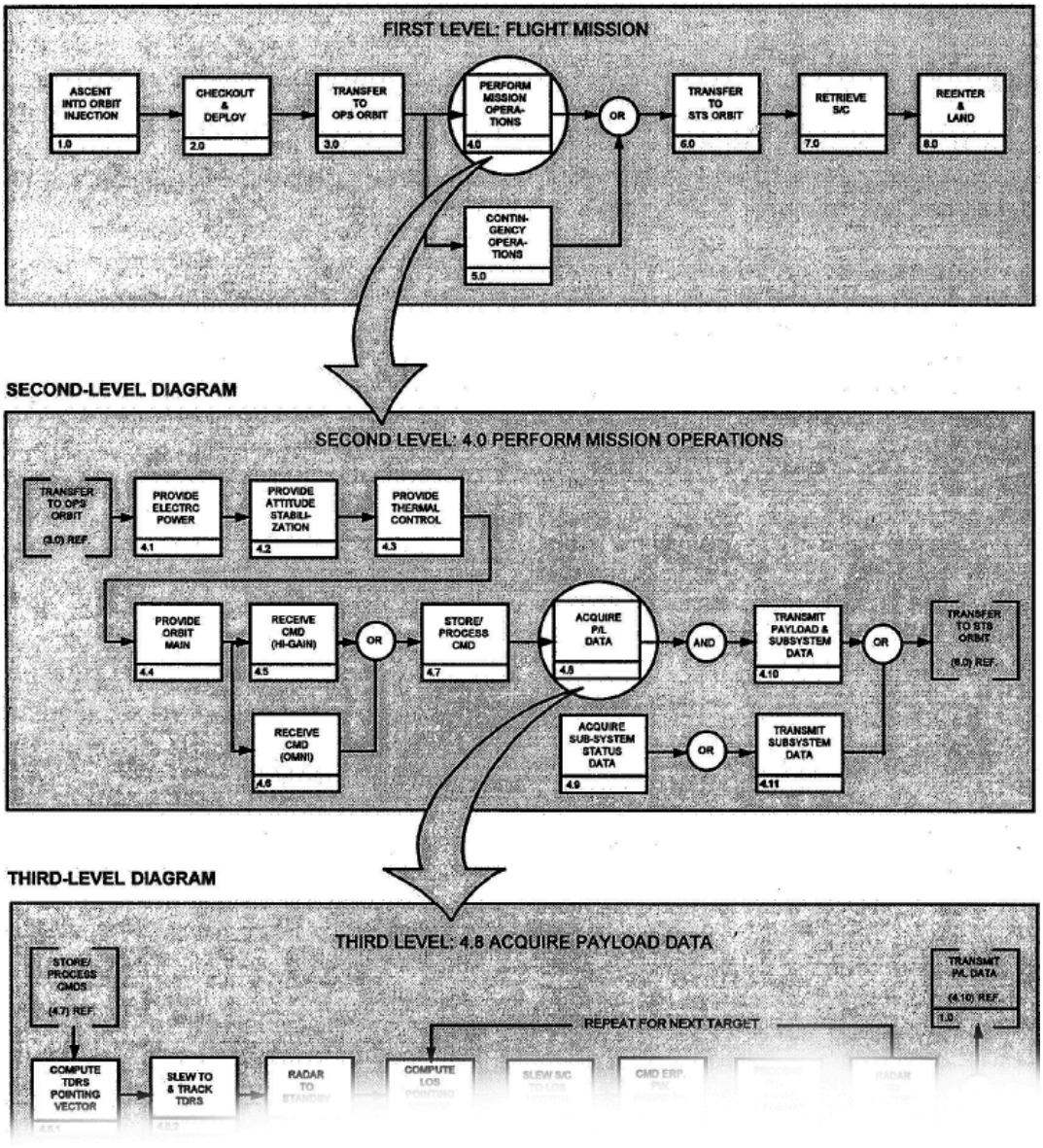


Figure 58. Functional Flow Block Diagram (FFBD) example.[77]

[76] Often this mapping exercise is called "analysis," but for this see later.
[77] *NASA Systems Engineering Handbook, NASA-SP-6105,* June 1995, p. 128.

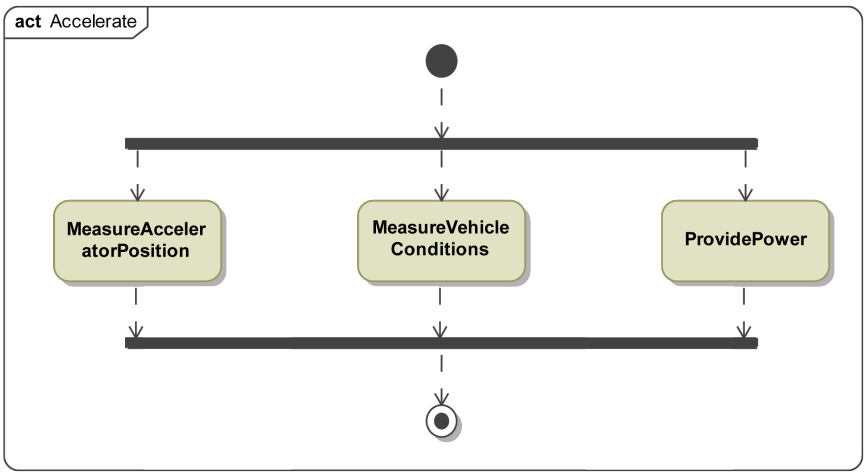


Figure 59. SysML 2nd level flow diagram vis-à-vis the hierarchy of Figure 55.[78]

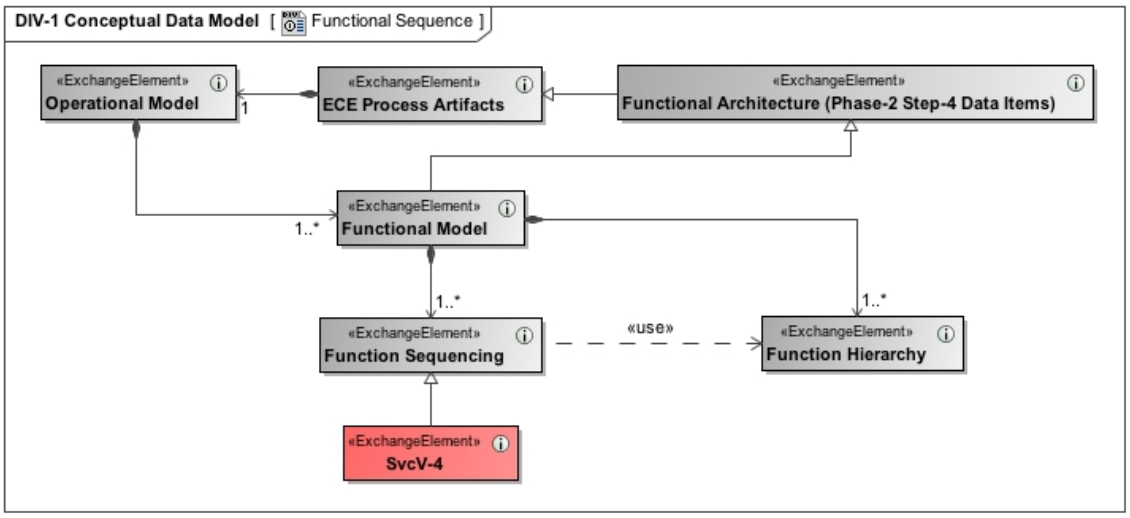


Figure 60. *Functional Sequence* DoDAF conceptual data model.

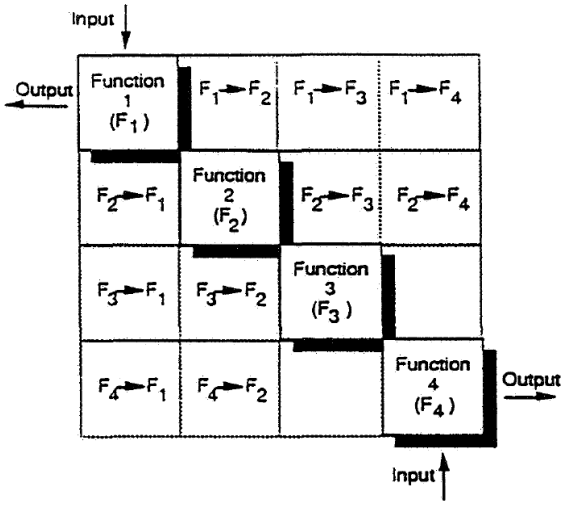
3.2.4.4 Mapping Functional Interfaces

Traditionally, the method of choice to identify and define interfaces has been the "N^2 chart" or diagram, as originally published at TRW in the 1970s.[79] (A later variant of this idea is referred to as a *design structure matrix*, or DSM, which was first published in 1981;[80] a DSM is different from an N^2 diagram in that the directional notation used for functional dependencies—feed forward and feedback—is reversed.). An N^2 diagram is

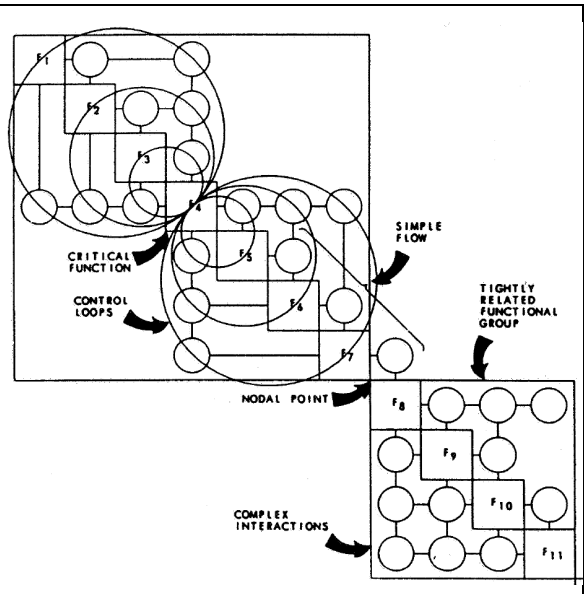
---

[78] Adapted from OMG SysML v1.2, p. 189.
[79] Lano, Robert J., *The N^2 Chart*, TRW Software Series, Redondo Beach, CA, 1977.
[80] Steward, Don, *Systems Analysis and Management: Structure, Strategy and Design*, Petrocelli Books, 1981.

constructed by placing system hierarchical-level or grouped functions on the diagonal of a square matrix; the other matrix elements represent all possible interfaces, which are methodically queried one-by-one. Where the need for an actual interface is identified, an appropriate name is assigned and reference to it is placed in the corresponding cell of the matrix (flow source row-destination column intersection); where a blank appears, no identified interface exists between the corresponding functions. Note that it may be advantageous to reorder rows and columns in order to cluster any off-diagonal interactions identified; this will facilitate later system partitioning. Once interfaces are identified, the corresponding flow definitions are developed and recorded in a convenient document and format.



Basic N^2 chart with feed-forward and feedback interfaces identified.



Key system structural features can be identified based on a completed N^2 Chart.

Figure 61. N^2 diagram features.[81]

Unfortunately for the architect, current "state-of-the-art" specifications or tools (UML, SysML, UPDM, …) do not define any table or matrix modeling construct that would enforce complete pair-wise query for interfaces between all hierarchical-level or grouped functions (although some particular tools may have custom options to allow such

[81] *NASA Systems Engineering Handbook, NASA-SP-6105,* June 1995, p. 129-130.

constructs to be defined). Rather, it is left up to the discipline of the systems engineer and architectural study team to be thorough in identifying necessary interfaces, and to develop the necessary DoDAF "Fit-for-Purpose" views. One possible, albeit forced, approach to construct a graphical display of interface information within a UML/SysML/DoDAF model would be to use methodical pair-wise queries between all activities appearing on a particular functional flow mapping activity diagram (e.g., *SvcV-4*). The results could be arranged to mimic an N^2 chart to facilitate the effort. Each such diagram, or perhaps copies thereof, would be updated with object flows (cf. any "subsystem" scenarios created as part of functional composition). Flow definitions could then be developed and activity parameter nodes or pins added to the diagrams to complete documentation of the interfaces at this level of development.

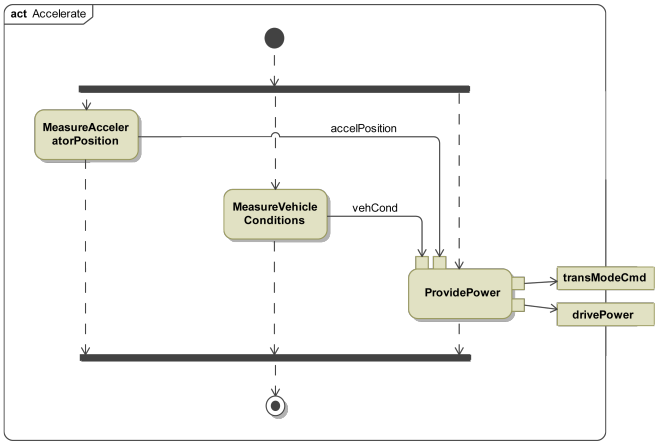


Figure 62. Example functional interface diagram (an update to Figure 59).

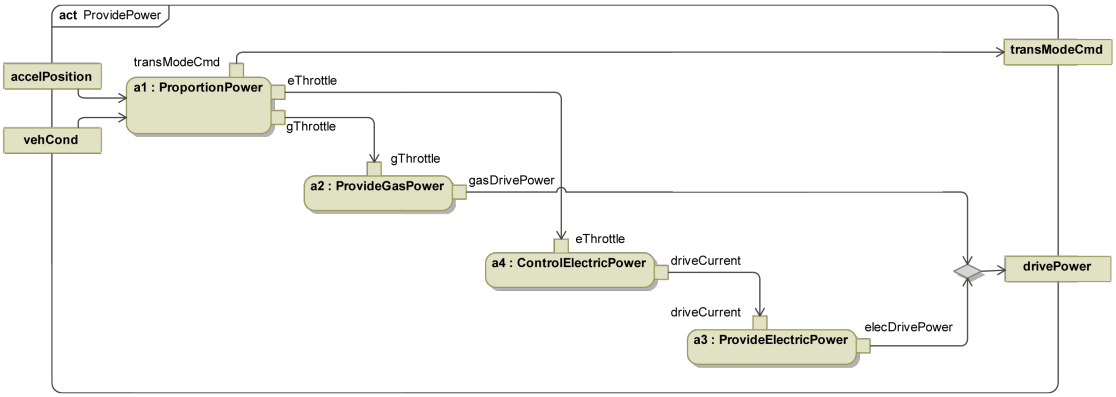


Figure 63. Example third-level interface diagram (cf. Figure 55 and Figure 62).

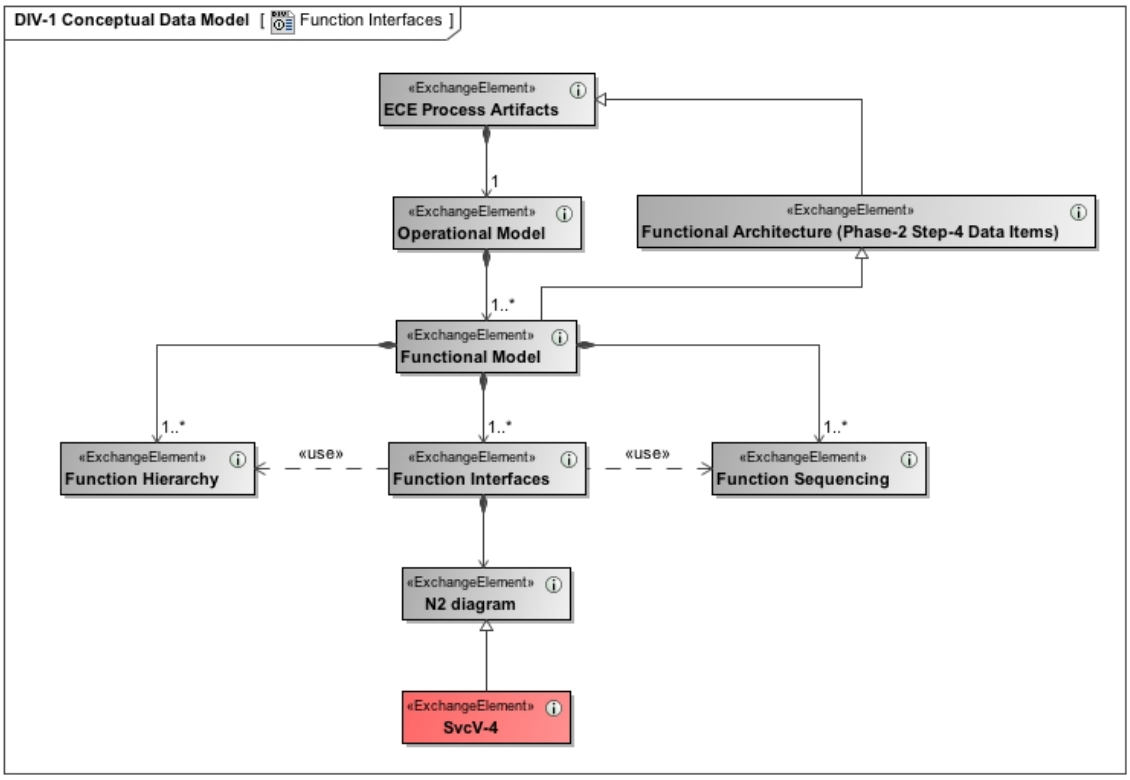


Figure 64. *Function Interfaces* DoDAF conceptual data model.

3.2.4.5 Mapping Requirements

Once the required functions and interfaces are identified, relevant system technical requirements are allocated. Generally all elements of the set of input and output (I/O) requirements should readily map to the appropriate functions that have been defined to handle the I/O. Likewise, the functional and performance requirements (e.g., MOPs) should allocate directly to, most likely, level-one of the functional hierarchy. For any lower-level functions defined, functional requirements (e.g., KPPs) will have to be decomposed or derived and allocated such that all functions are linked to required behavior (maintaining traceability). For all internal interfaces identified, associated interface requirements have to be derived and linked to the flows. Note that each functional architecture will, in general, have its own, particular set of KPPs and other derived requirements. It is also important to make note of **allocations** (rather than derivations) because these **represent potential areas for later tradeoffs**.

Within DoDAF, a "Fit-for-Purpose" view will be required; one possibility for characterizing function requirements would be the specification of "Measure Types" and "Measures" (including "actuals"), as well as "resource constraint" elements, that could then be displayed, e.g., on an *SvcV-4* diagram[82] (cf. APPENDIX E). Specification of function interfaces will likely require use of SysML extensions. Verification objectives are also established on the basis of the performance requirements developed.

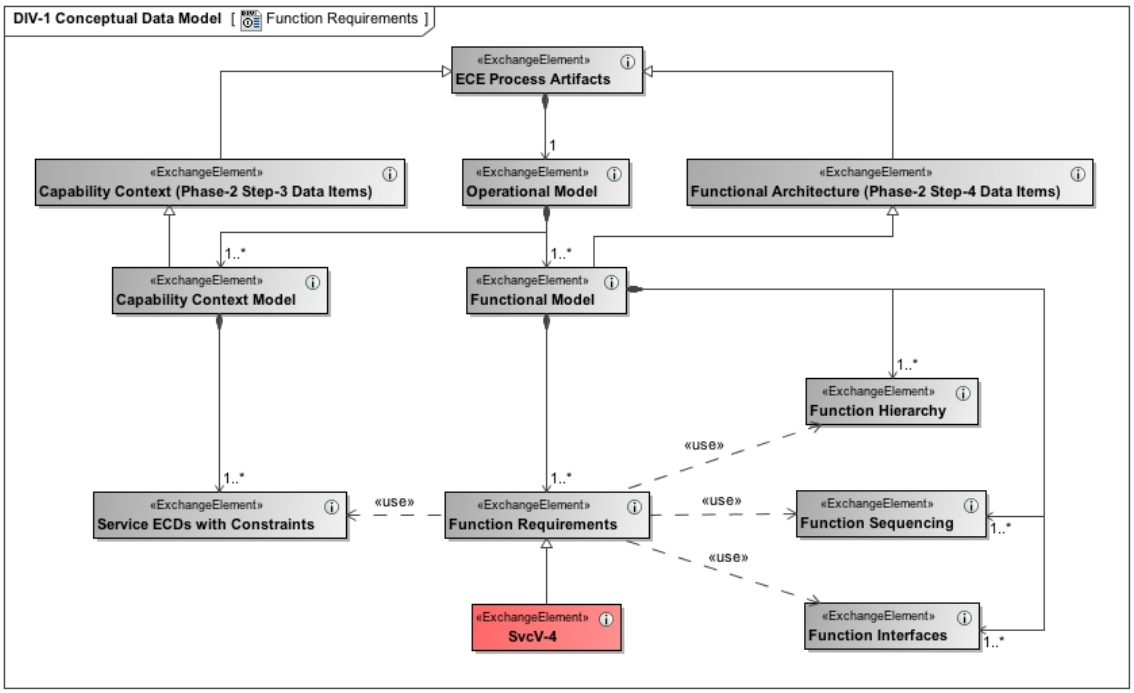


Figure 65. *Function Requirements* DoDAF conceptual data model.

3.2.4.6 Analyzing Functions

As used herein, the term *functional analysis* is restricted, for the sake of clarity, to modeling or simulation of the functional flow graphs defined through the set of functional-hierarchy-development, flow-mapping, and interface-mapping activities as described above. (Be warned, however, that very often in the published literature it is this set of activities, and not functional flow simulation, that is referred to as *functional*

[82] Note that, when using the *MagicDraw* UPDM tool (at least v18.0), the SvcV-7 model does not support specification of function measures, and the **SV-7** *Systems Measures Matrix* will, unfortunately, have to be used. The presence or stability of this "feature" in other *MagicDraw* versions or software tools is unknown.

analysis!) At a minimum it is generally considered necessary to define a functional timeline (aka Time Line Analysis, TLA) for the "system" in order to, e.g., validate any time-critical design requirements identified. When following MBSE methods, this information would be captured in sequence diagrams (e.g., DoDAF *SvcV-10c*).

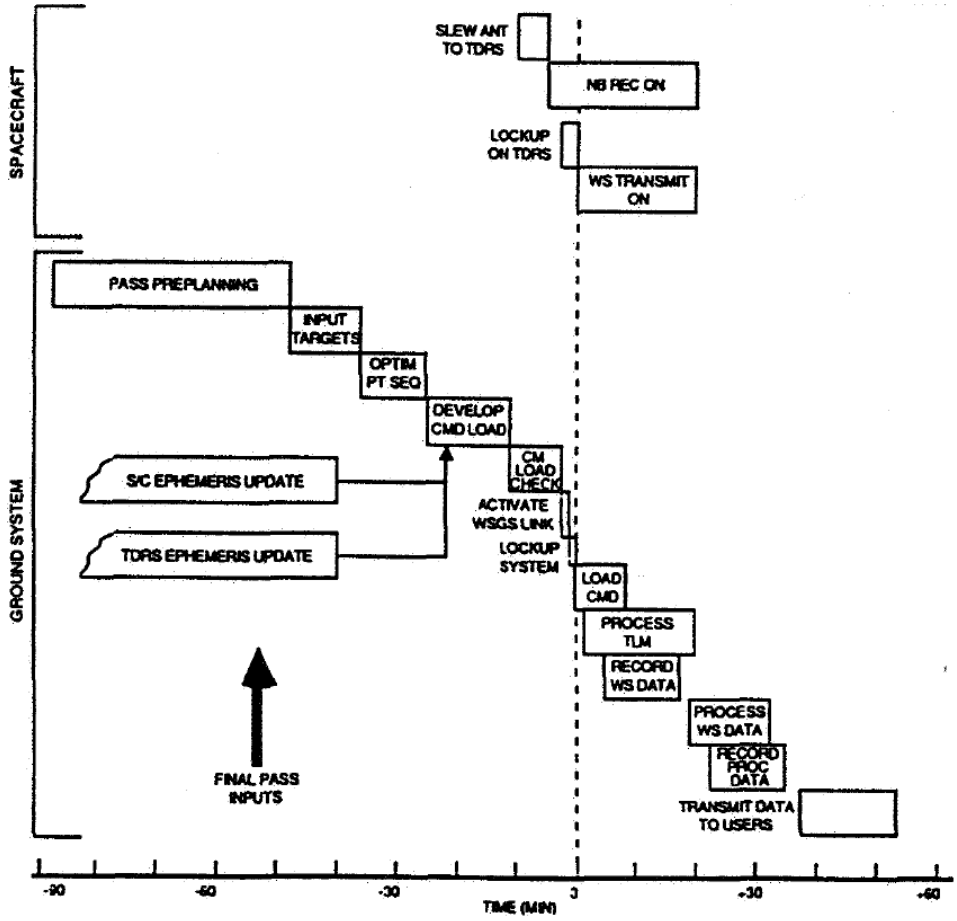


Figure 66. Example time line analysis (NASA flight mission segment).[83]

For systems with complex functional flow graphs (those exhibiting choice, iteration, or concurrent execution—i.e., those exhibiting more than simple flow behavior), additional analysis is usually desired. A "classical" evaluation method in this regard is through the use of the Petri net mathematical modeling language (or one of its derivatives) in order to determine, e.g., network node reachability, liveness, and boundedness. More recent approaches include use of business process modeling techniques (e.g., Business Process

[83] *NASA-SP-6105, p. 131.*

Modeling Notation, BPMN and Event-driven Process Chain, EPC), as well as activity diagram and state machine modeling (e.g., via the Cameo Simulation Toolkit™ marketed by No Magic) in UML or its derivatives (e.g., DoDAF *SvcV-4 and -10b*).

The results of any analyses (e.g., a functionally needed revision of timing allocations) should be reflected as needed in a requirements update.

Although the description of trade study techniques is beyond the scope of this paper, once (and if) functional analysis *per se* has finished development of multiple functional architectures that meet the full set of threshold performance (mission) requirements, trade space and risk should be analyzed and assessed. Note, however, that while it is useful to such techniques to narrow the number of functional solution sets carried forward to a manageable level, it is not necessary (nor desirable) to down select to a single functional architecture.

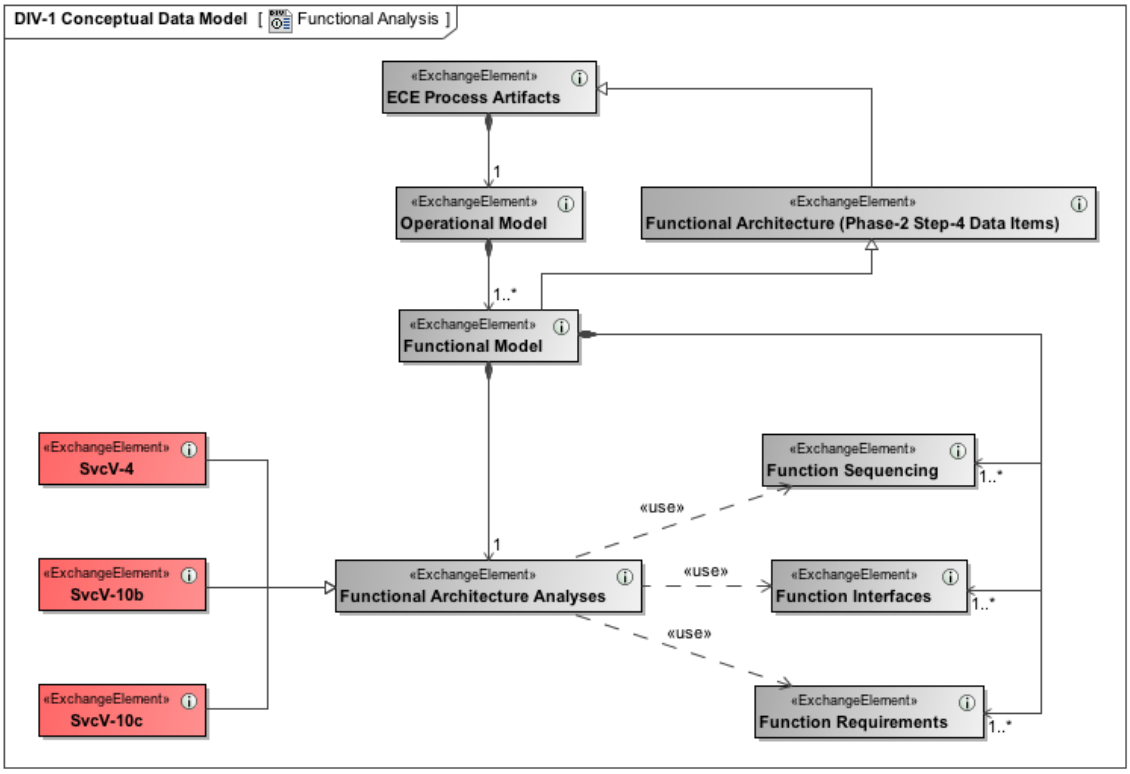


Figure 67. *Functional Analysis* DoDAF conceptual data model.

3.2.4.7 Phase-2 Step-4 Summary

After completing the fourth operational modeling step, the specifications developed for each capability have been implemented in one or more functional architectures. Requirements have been derived and allocated down to the leaf-node level. Analyses have been performed to validate the architectures. Architecture options have been narrowed or down selected through appropriate trade studies. In addition, it should be recognized that the collection of "leaf" nodes of the functional hierarchy, functional sequence and interface requirements, and functional performance requirements (e.g., KPPs) collectively define the set of test and evaluation objectives required as input into system (service) verification planning; it is likely that insights gained during *functional analysis* can also be brought to bear in test planning.

Intentionally blank.

3.2.5 Repartitioning Behaviors (Phase-2 Step-5)

The structured analysis of system functions that has preceded this point is somewhat limited in that it does not explicitly support definition of how system parameters are to be handled (e.g., storage and passing or flow of data), nor support allocation of non-functional performance related constraints. When the subsystem hierarchy is small and described by simple flow between the functions, this issue is probably manageable. However, as the functional structure size and complexity grows, the problem can become unwieldy—especially in understanding and managing the impact of future requirements- or design-change notices. Object-oriented analysis offers a flexible method of helping with this problem by combining function and parameter statements into "objects." The OMG SysML is specifically intended to support systems engineers in performing such object-oriented analysis and design (OOA/D), although other frameworks (DoDAF, as herein) and languages (UML) can be used as well.

The objective of applying OOA here is to repartition (aka refactor) functions, flows, and controls into a *logical* architecture built up of *logical* components (preferably—as noted before—**several** architectures in support of developing alternatives to be evaluated by trade studies). Note that it is still desirable to remain technology agnostic at this point, and so the idea of *logical* should be suitably abstract. (An example: for service capability modeling within DoDAF, an appropriate level of abstraction is available by ultimately defining a «CapabilityConfiguration» in terms of resource types such as «System», «Software», «Materiel», «PersonType», and «Organization Type», on, e.g., an *SvcV-1* diagram, although "sub-services" may be useful as well during decomposition.)

The desirability of conducting an OOA is also supported by the following consideration. In a "classical" SE process, all partitioning criteria were applied when developing physical architectures to implement the defined functional architecture(s). In general, most system architectures are driven by multiple partitioning criteria (e.g., modularity, isolation, reuse, COTS, constraints, commonality, data, performance, reliability, maintainability, producibility, safety, security, design responsibility, ...), yet optimization against multiple partitioning criteria is a very difficult problem. Unfortunately, in

practice, this has often meant that criteria were prioritized and screened to reduce the number to manageable levels.) Decomposition—as a method of making easier problems out of hard ones—can be applied to factor physical architecting into two steps: *logical*—which, as it is considered to be technology agnostic, is therefore aligned closely with the functional architecting step of the process previously discussed—and *physical*—that, as before, applies technology to the problem, but now with a reduced set of criteria to deal with, making evaluation of the physical solution space easier to manage.

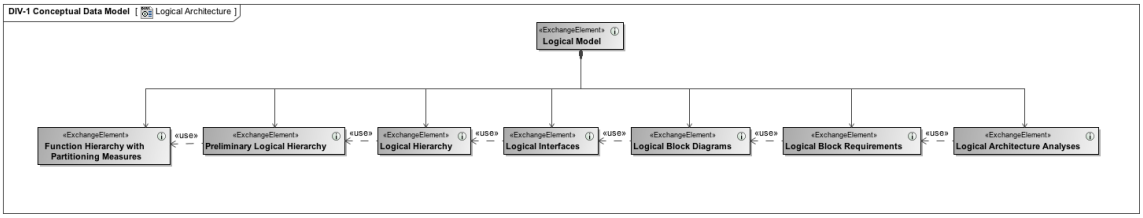


Figure 68. ECE Process Phase-2 Step-5 conceptual data model.

Table 8. *AV-2 Integrated Dictionary* of Phase-2 Step-5 Data Elements

Name	Definition
Function Hierarchy with Partitioning Measures	Measurements of functions vis-a-vis individual partitioning criteria captured as attributes.
Preliminary Logical Hierarchy	Logical hierarchy derived from the functional architecture using methods of composition or decomposition.
Logical Hierarchy	Logical hierarchy refined by considering global partitioning criteria.
Logical Interfaces	Identification and definition of all logical block-to-block interfaces.
Logical Block Diagrams	Logical subsystem internal structure diagrams.
Logical Block Requirements	Allocation of all requirements and constraints to the logical block level.
Logical Architecture Analyses	Results of logical architecture and requirements validation activities.

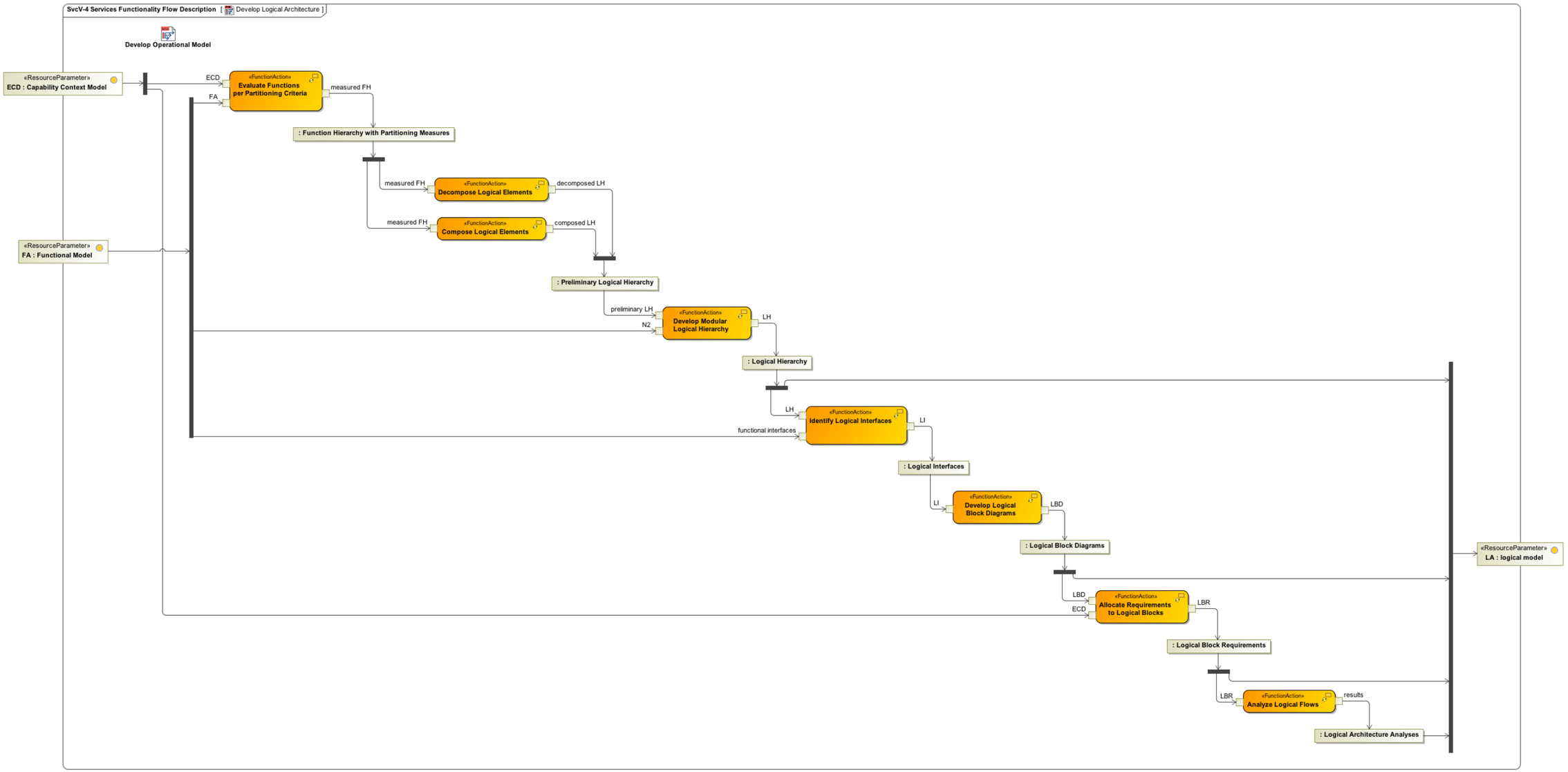


Figure 69. ECE Process Phase-2 Step-5 task flow: *Develop Logical Architecture.*

Intentionally blank.

3.2.5.1 Defining partitioning criteria

Before a proper logical architecture can be developed it is necessary to define the criteria that should be satisfied in partitioning a functional (e.g., *SvcV-4*) architecture. In general, these criteria should be found in the applicable ECD. Typical partitioning criteria used to help group functions together at some level in a *logical* hierarchy include:

- **Modularity**—group to maximize cohesion and minimize coupling.[84,85] Compare, e.g., the two "tightly related functional group[s]" connected by a single "nodal point" in the right-hand illustration of Figure 61. It should be recognized that interface issues dominate integration and verification (testing) activities, and so modularity is generally a very important consideration.[86]
- **Similarity**—group common or related functions, I/O, or data stores.
- **Changeability**—group functions whose requirements are poorly understood or likely to change.
- **Constraints**—group functions that share common constraints such as safety, security, environmental, or physical requirements.

---

[84] "Modular designs are characterized by the following: functionally partitioned into discrete scalable, reusable modules consisting of isolated, self-contained functional elements; rigorous use of disciplined definition of modular interfaces, to include object oriented descriptions of module functionality; designed for ease of change to achieve technology transparency and, to the largest extent possible, makes use of commonly used industry standards for key interfaces." Open Systems Joint Task Force, *Program Manager's Guide: A Modular Open Systems Approach (MOSA) To Acquisition*, Version 2.0, DoD. September, 2004, p. 5. Cf. Nelson, Eric M., *Open Architecture: Technical Principles and Guidelines 1.5.6*, IBM, 3-27-2008. As an example interface standard within models-based engineering (MBE) practices, consider the *Functional Mock-up Interface* (FMI) at https://www.fmi-standard.org .

[85] Coupling at functional interfaces (as defined, e.g., in an N^2 diagram) can be characterized as "binary" (all links are of equal importance), "discrete" (a measure of the "thickness of the pipe" or number of unique values or connections defined at an interface), "ranked" ("expert" opinion is used to assign ordinal or qualitative values to each interface), or "weighted" (real values are assigned to the links based on the results of sensitivity analyses or other forms of calculated metrics).

[86] Cf.: DoDD 5000.01, May 12, 2003, Certified Current as of November 20, 2007, §E1.1.27 "A modular, open-systems approach shall be employed, where feasible." Interim DoDI 5000.02, November 25, 2013, §7.d "Program management is also responsible for evaluating and implementing open systems architectures, where cost effective, ... to support continuous availability of multiple competitive alternatives throughout the product life cycle." Interim DoDI 5000.02, November 25, 2013, Enclosure 3, §14 "Program managers are responsible for applying open systems approaches in product designs where feasible and cost-effective. Open systems and open architectures provide valuable mechanisms for continuing competition and incremental upgrades. Program management will use open systems architecture design principles to support an open business model… To the maximum extent practicable, each program will leverage the guidance and procedures in the *DoD Open Systems Architecture Contract Guidebook for Program Managers* … ."

Within DoDAF, it is suggested that the functional partitioning criteria to be applied to a particular service be captured in an *SvcV-10a* table. This should include guidance on, e.g., the relative merit, priority, or even quantitative scoring metrics, of the different partitioning criteria. (As previously noted, common practice is to use a heuristic approach in applying partitioning criteria; although formal methods do exist that support partition optimization,[87] said topic is beyond the scope of this paper.) For criteria that apply individually (e.g., *similarity*, *changeability*, and *constraints*), each function is evaluated on the basis of said criteria, and attribute measures are captured in the respective modeling object (e.g., UML *class* or SysML *block*); in DoDAF, stereotyping (e.g., by functional category such as used in §3.2.4.1) and "resource constraints" are used appropriately to represent the measures, and then displayed in an *SvcV-4* diagram.

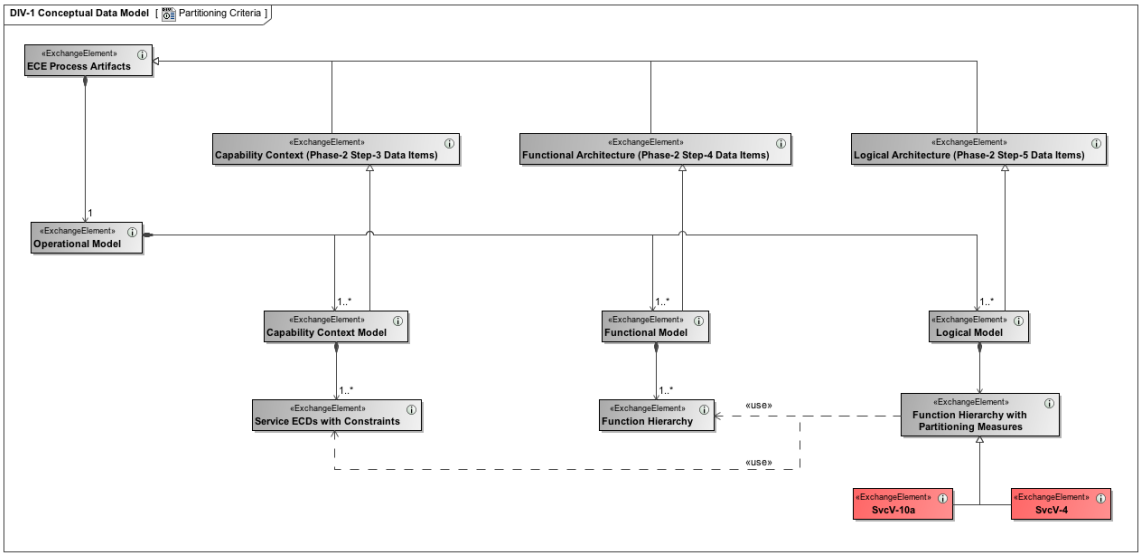


Figure 70. *Partitioning criteria* DoDAF conceptual data model.

The techniques of decomposition (§3.2.5.2 below) and composition (§3.2.5.3 below) utilize the individual partitioning criteria measures to develop a set of preliminary logical hierarchies. A *modularity* measure applies (and possibly other criteria as well), on the other hand, to a grouped set of fully interfaced functions (i.e., to a particular architecture,

[87] Cf. Otero, Richard E., and Robert D. Braun, "The State of Problem Decomposition in Engineering Design," AIAA 2009-2188, 50th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, 4-7 May 2009, Palm Springs, CA.

or at least significant portion thereof); generally this requires consideration of an N^2 chart (or DSM) such as developed in §3.2.4.4, as outlined further in §3.2.5.4 below.

3.2.5.2 Producing a logical system definition by decomposition

Create a preliminary logical hierarchy (preferably more than one) by evaluating the functional hierarchy "tree" (e.g., *SvcV-4*) diagram(s), working from top down. At any particular functional node (and corresponding logical, service element), consider different allocations—logical architecture design variants—of the supporting sub-functions to logical sub-elements (e.g., DoDAF resource types or "sub-services", or «logical» typed blocks in SysML), as guided by the defined partitioning criteria. Allocations can be captured in an *SvcV-1* diagram using «ActivityPerformedByPerformer» relationships. For each such allocation, the logical element should have a resource operation defined and specified as realizing the corresponding function. If desired, each logical hierarchy itself could also be portrayed in an *SvcV-1* diagram using, e.g., inheritance and aggregation relationships.

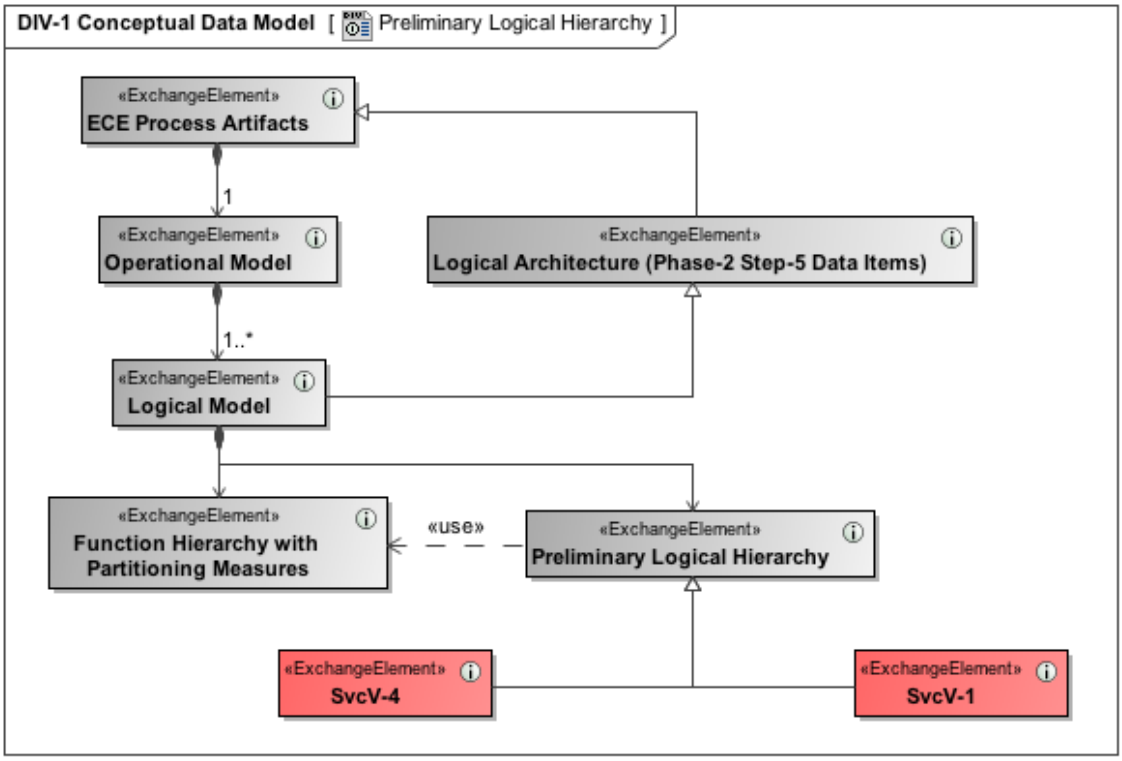


Figure 71. *Preliminary logical architecture* DoDAF conceptual data model.

3.2.5.3 Producing a logical system definition by composition

An alternative or companion approach to decomposition would be to create one or more preliminary logical hierarchies through composition—a bottom-up approach. This method begins by first creating a logical element for each leaf-level function and defining an allocation relationship between them as in the decomposition method; the results are portrayed in an *SvcV-1*. Next, logical groupings are developed by considering the partitioning criteria measures of each allocated function. For example, pair-wise comparisons[88] of the logical components could be performed, and, given a suitable "score," the pair-compared elements could be combined into a logical grouping. When completed across the bottom of the tree, the set of groupings form the next higher level of the hierarchy, and so on. Finally, just as for the hierarchies generated by decomposition, each "clean" logical hierarchy created by composition can be portrayed in an *SvcV-1* diagram, if desired.

3.2.5.4 Producing a logical system definition by considering modularity

The preliminary logical hierarchies created using the methods of decomposition and composition, as presented above, were only able to take into account partitioning criteria that could be evaluated at the level of an individual function. Functional partitioning criteria that are more global in nature—*modularity* in particular—require evaluation of metrics using an architectural description that contains functional interfaces. Generally this involves the creation and evaluation of N^2 chart (or DSM) for each preliminary architecture developed by decomposition and composition. Global functional architecture measures of partitioning criteria are then combined—heuristically or formally—with individual, function-level measures to "score" each logical hierarchy.

The method for creating an N^2 chart (or DSM) for a particular preliminary logical hierarchy begins with a **copy** of the N^2 chart (or DSM) developed per §3.2.4.4 (which may be in the form of one or more *SvcV-4* diagrams). Next, row-column pairs[89] are shifted in order to group the set of functions allocated to a particular logical element (e.g.,

---

[88] This is an example, and is not meant to restrict the logical groupings to pairs.

[89] Each row-column pair represents a single function; both must be shifted to the same row-column index in order to preserve the integrity of the information represented by the N^2 chart.

the "tightly related functional group" shown in the right-hand panel of Figure 61 could represent a logical element). Finally, a *modularity* measure is determined for the architecture on the basis of the reordered N^2 chart (or DSM).

It should be noted that it is possible (and generally desirable) to also create one or more logical architectures by beginning with a **copy** of the N^2 chart (or DSM) that was developed following the method of §3.2.4.4, and then shifting row-column pairs such that *modularity* is maximized. (In general, a single maximum may not exist; rather, several local maxima will be identified, each of which may represent a viable logical architecture that should be evaluated.) After this reordering, individual partitioning criteria can be considered to identify one or more sets of logical elements and groupings, and so logical hierarchies, which can be captured in an *SvcV-1*. As above, the resulting partitioning arrangements are then "scored."

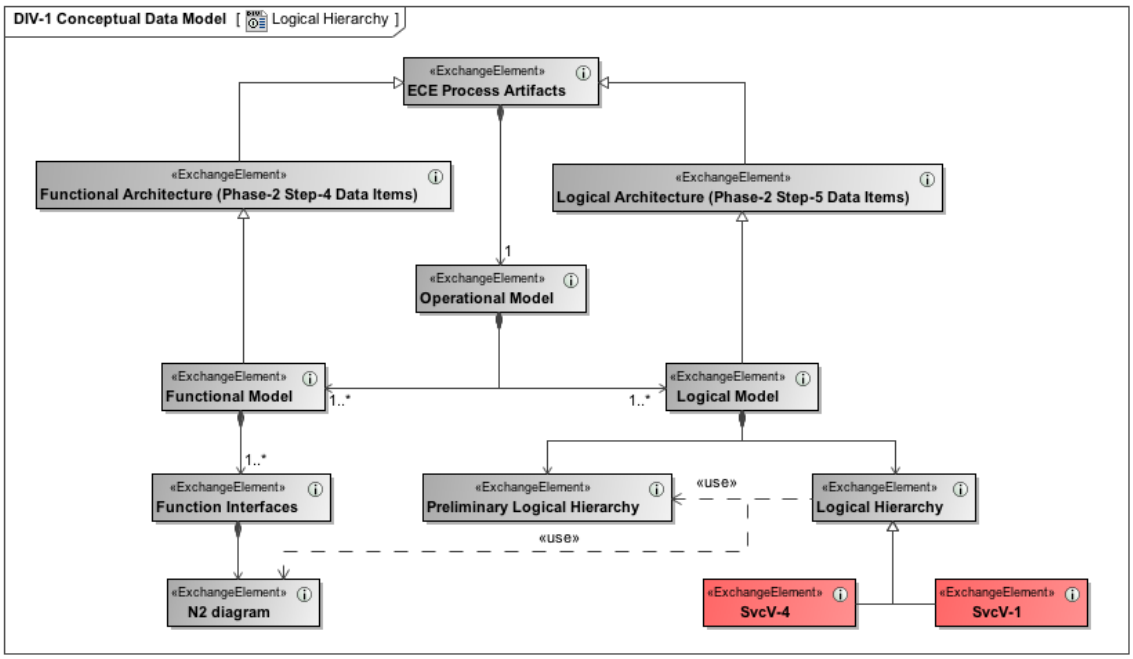


Figure 72. *Logical hierarchy* DoDAF conceptual data model.

At this point, a robust architecting process will have produced a set of logical architectures larger than can be reasonably processed. If this is the case, the partitioning measure "scores" should be used to down select to a preferred ("baseline") logical architecture. It is also strongly advised that two "runners up" architectures be carried

forward as well; these should be selected not only on the basis of the "scores," but also on the basis of representing significantly different architectural arrangements, and not simply minor variations thereof. This last point is important in supporting the identification of a robust trade space.

3.2.5.5 Diagramming logical interfaces

Each *SvcV-4* (flow) *functional* interface diagram (see, for example, Figure 63 above) is updated (or perhaps copied and updated) through the use of "swim lanes" to reflect the functional allocations established by the logical hierarchy. Flows across boundaries between the swim lanes define the exchanges at logical element interfaces.

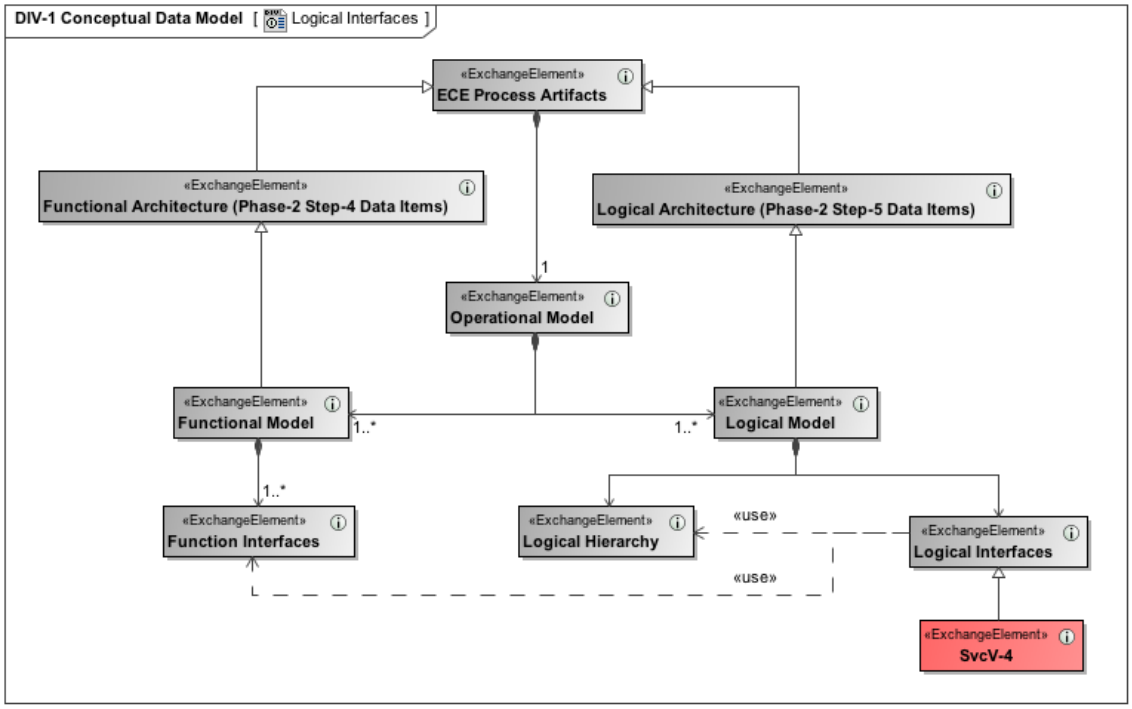


Figure 73. *Logical interfaces* DoDAF conceptual data model.

3.2.5.6 Mapping logical block structure

The information contained in each one of the logical interface diagrams (e.g., *SvcV-4* (flow) or activity diagrams, *act*, representing "subsystem" scenarios) is recast into a set of logical subsystem internal structure diagrams (SysML *ibd* or UML *composite structure* diagram); in DoDAF this is performed with a *SvcV-2 Services Resource Flow Description* along with «CapabilityConfiguration» and applicable resource type blocks.

Note that since, in general, multiple functions can be allocated to a single logical component, these diagrams will be crosscutting in nature. Finally, a composite internal block diagram (or *SvcV-2*) is completed for the system level by developing an integrated view of all logical subsystems. When the logical block diagramming activity is completed, an *SvcV-2* (*ibd*) should be available for the top (capability/service/system) and intermediate (resource/subsystem) nodes of the logical hierarchy. Each logical component should also have been reviewed by this point to ensure it is fully elaborated within the context of its application.

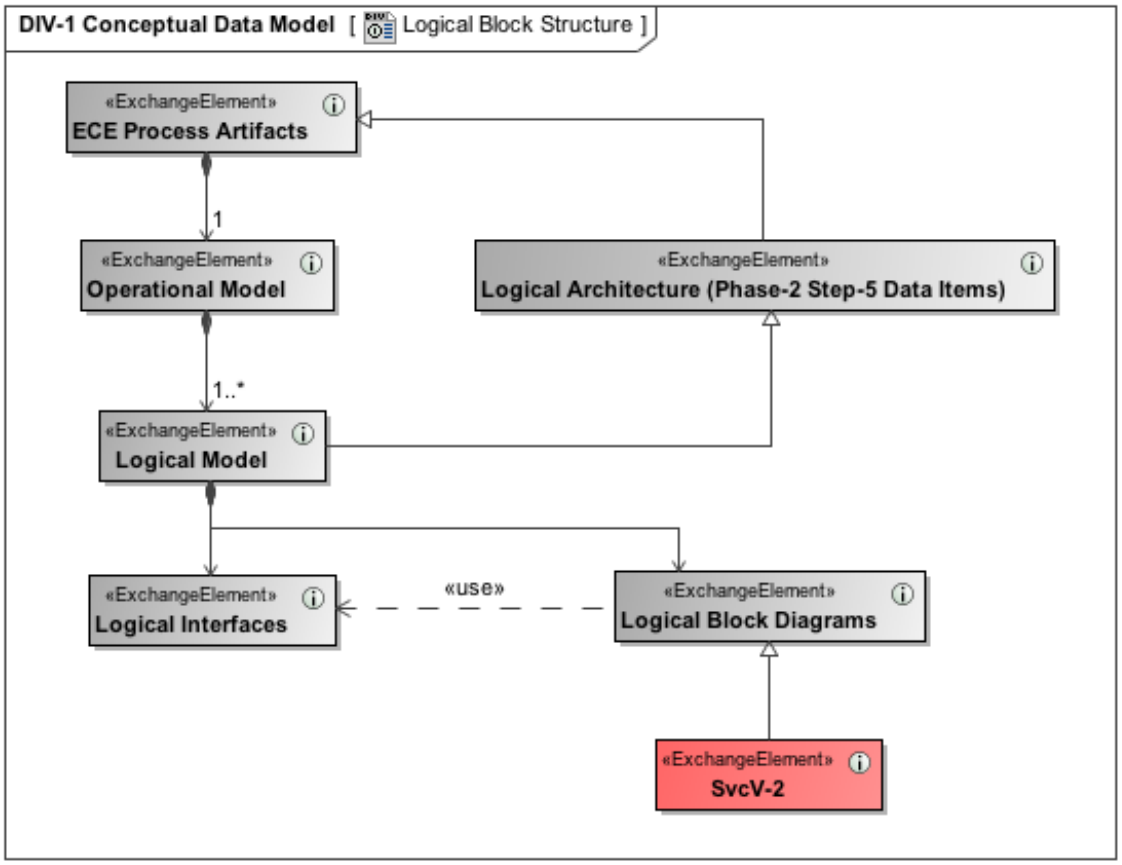


Figure 74. *Logical block structure* DoDAF conceptual data model.

3.2.5.7 Allocate requirements

All requirements allocated at the functional architecture level (cf. §3.2.4.5) and capability context—service "black box"—level (c.f. §3.2.3.5) are now allocated to the appropriate (function performing) logical components. In a models-based architectural description,

the allocation relationships should already exist implicitly for performance requirements. The primary task here is to map constraints to the appropriate logical blocks; an *SV-10a Systems Rules Model*[90] can be used for this purpose, with traceability recorded in the specifications of each rule (constraint). As needed, additional logical component requirements are derived by elaborating the scenarios, I/O flow, and control requirements. Functional verification objectives (and supporting test cases) are also recast to reflect the logical architecture and components. Each logical subsystem and component should be reviewed to ensure, e.g., the following types of requirements have been defined: functional; performance; I/O; control; store; and design constraints.

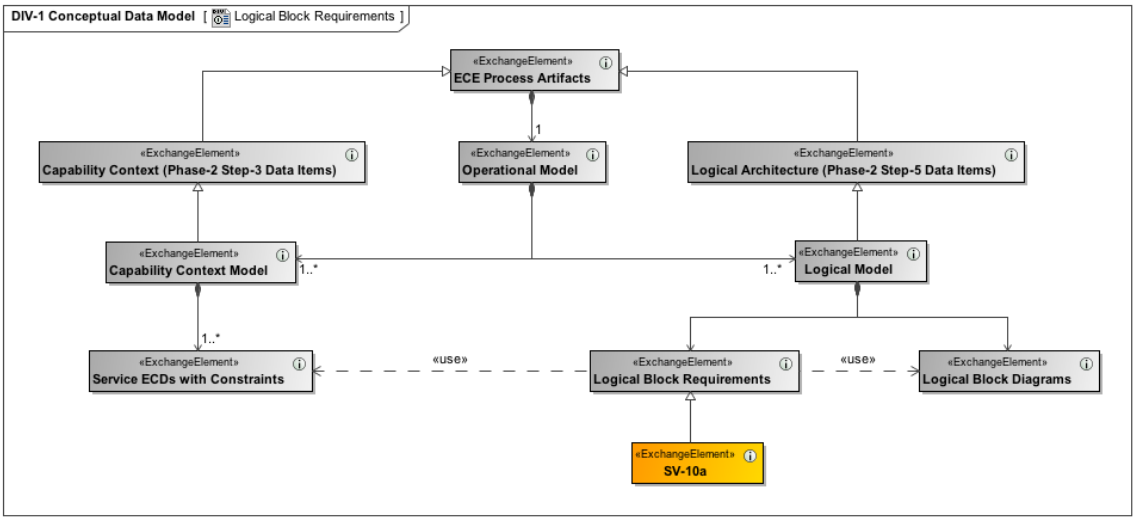


Figure 75. *Logical block requirements* DoDAF conceptual data model.

3.2.5.8 Analyzing logical flows

At a minimum it would be expected that additional TLA would be performed through the development of a set of sequence diagrams (*SvcV-10c, sd*) representing logical behavior at the system and subsystem levels (i.e., top and intermediate nodes of the logical hierarchy). With the development of a logical system description built up of logical components, it also becomes possible to extend the system analysis effort to encompass object-oriented component-based system modeling and simulation (e.g., with *Modelica*,

[90] Use of an *SV-10a* here reflects a limitation of the MagicDraw UPDM plugin as the *SvcV-10a Services Rules Model* only supports specification of rules for "service" modeling artifacts.

Simulink, or *Simplorer*), as well as with parametric modeling within SysML. While beyond the scope of this paper—and quite dependent upon the actual problem being studied—this is, nevertheless, and important aspect of the analysis to be performed as part of this step, consistent with a project-specific modeling and analysis plan. Furthermore, as the logical element types (resources) will likely include people, the specialty discipline of Human Systems Integration (HSI) may need to be brought to bear. And, finally, trade studies should be conducted to evaluate the different logical architectures developed. The results of any changes driven by analysis results should be reflected in an update to the requirements.

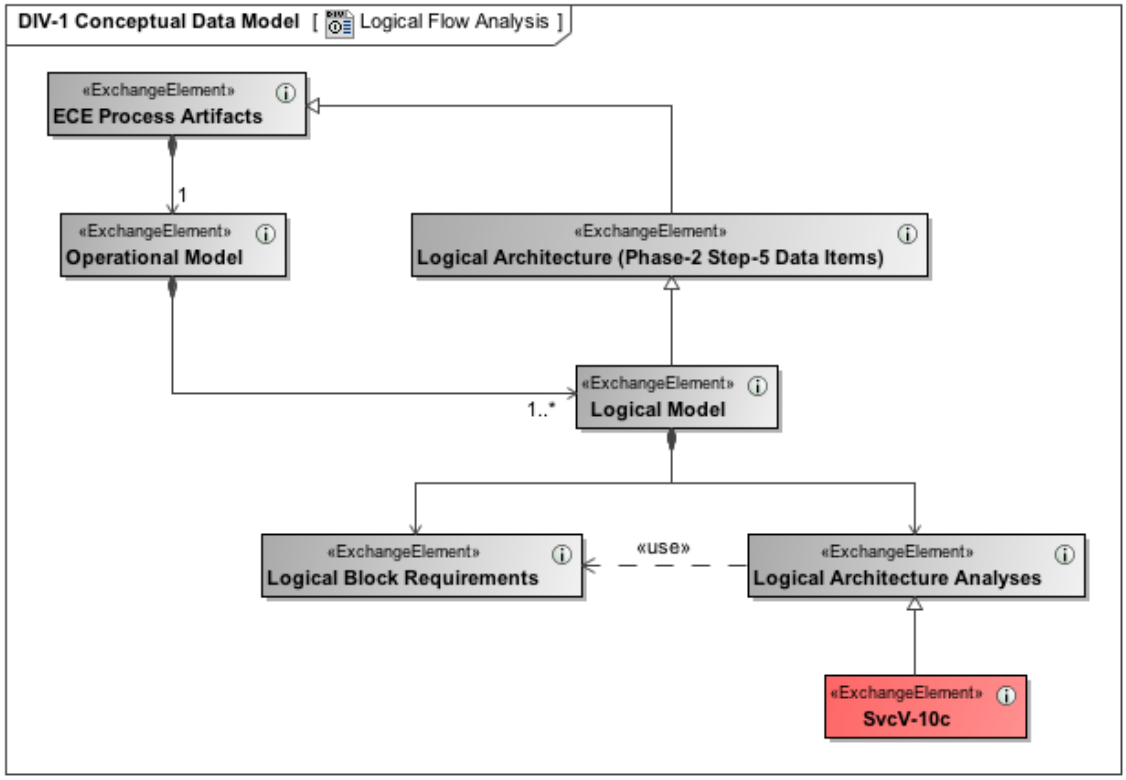


Figure 76. *Logical architecture analysis* DoDAF conceptual data model.

3.2.5.9 Phase-2 Step-5 Summary

After completing the fifth operational modeling step—repartitioning behaviors—the operational modeling phase is complete. The specifications developed for each capability have been implemented in functional and logical architectures. Analyses have been performed to validate the architectures. Architecture options have been narrowed or down selected through appropriate trade studies. Requirements have been derived and allocated down to the logical block level in anticipation of physical architecting in Phase 3.

3.3 ECE Process Phase-3. Modeling Deployed Capability

The third phase is oriented around satisfying the following two objectives:

- The shortfalls in resources needed to deliver the required enterprise capabilities are identified, along with the associated operational risks, prior to investigating potential solutions.
- The possible solution space for the enterprise capability shortfalls is identified prior to identifying a preferred option.

The approach toward meeting these objectives is to:

- Develop enterprise physical models, allocate and associate as-is/what-if/to-be resources with the fulfillment of operational capability requirements, and analyze the performance thereof vis-à-vis mission.

The first objective is met by developing a reference physical model of how operational and capability requirements are satisfied—a model of the "as-is" system except for unprecedented design cases—and, on the basis of said model, establishing a mission performance baseline. Results that appear to be unacceptable according to current strategic guidance (requirements) are identified, as well as the functional **needs** (gaps) that so arise. The second objective is met by developing one or more "what-if" or "to-be" system models that are used in investigating how mission performance can be improved relative to the baseline, including closure of gaps (e.g., the evaluation process is repeated until one or more suitable solutions are identified).

The working assumption herein is that both the structured analysis and object-oriented analysis described in the previous ECE process step were completed, resulting in definition of one or more logical architectures (the "golden rule" is a minimum of three). If this is not the case, the physical architecture partitioning criteria have to be expanded to cover the partitioning criteria that would otherwise have been applied in developing a logical architecture from the functional architecture.

The system modeling (physical architecting) step includes the following activities, which are described further below:

1. Develop generic physical architecture
    a. Identify or define the desired partitioning criteria

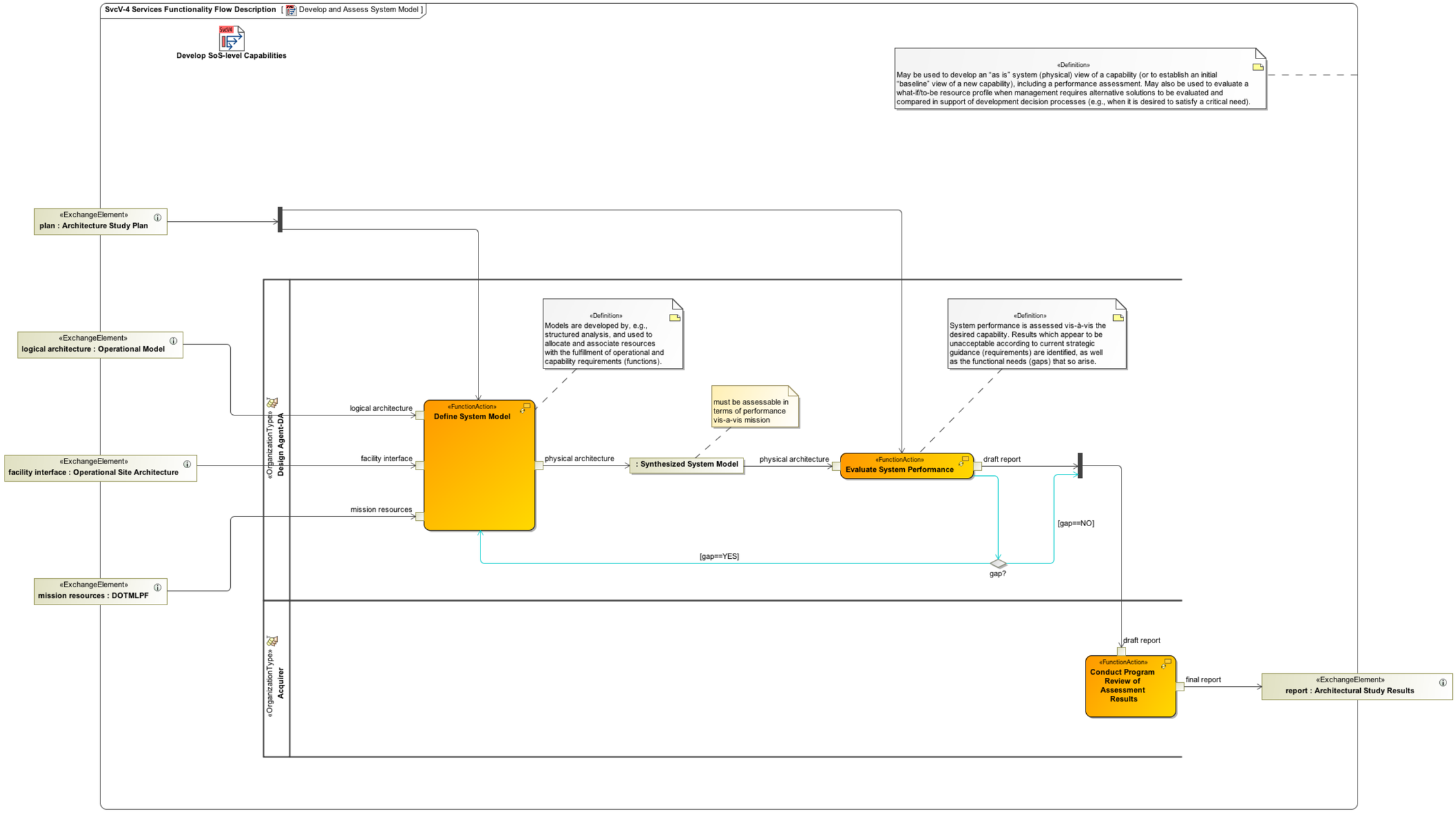
b. Develop system generic physical hierarchy
c. Allocate logical components (functions) to the generic physical components

2. Develop and assess "as-is" physical architecture
    a. Identify "as-is" component technologies
    b. Develop system-level concept simulation & analysis models
    c. Evaluate "as-is" performance
3. Develop and assess "what-if" or "to-be" physical architectures
    a. Identify alternative component technologies
    b. Identify alternative instantiated physical architectures
    c. Eliminate non-viable system concepts
    d. Perform preliminary screening of system concepts
    e. Document concept analysis candidates
    f. Develop system-level concept simulation & analysis models
    g. Evaluate performance of alternative concepts

From a DoDAF perspective, however, it should be noted that DoDAF v2.0, Volume 2, p. 201, §3.1.8, *Systems Viewpoint* reads:

> "The DoDAF-described Models within the Systems Viewpoint describes systems and interconnections providing for, or supporting, DoD functions. DoD functions include both warfighting and business functions. The Systems Models associate systems resources to the operational and capability requirements. These systems resources support the operational activities and facilitate the exchange of information. **The Systems DoDAF-described Models are available for support of legacy systems. As architectures are updated, they should transition from Systems to Services and utilize the models within the Services Viewpoint.**" [emphasis added]

How this deprecation of DoDAF System Viewpoints (*SV*s) is managed will have to be determined on a project-by-project level, and further discussion of DoDAF *SV* application herein is generally avoided. It is strongly recommended that SysML be considered for any system-level modeling artifacts that must be produced.


SvcV-4 Services Functionality Flow Description [Develop and Assess System Model]
SvcV4
Develop SoS-level Capabilities
«Definition»
May be used to develop an "as is" system (physical) view of a capability (or to establish an initial "baseline" view of a new capability), including a performance assessment. May also be used to evaluate a what-if/to-be resource profile when management requires alternative solutions to be evaluated and compared in support of development decision processes (e.g., when it is desired to satisfy a critical need).
«ExchangeElement»
plan : Architecture Study Plan
«ExchangeElement»
logical architecture : Operational Model
«ExchangeElement»
facility interface : Operational Site Architecture
«ExchangeElement»
mission resources : DOTMLPF
«OrganizationType»
Design Agent-DA
«OrganizationType»
Acquirer
«Definition»
Models are developed by, e.g., structured analysis, and used to allocate and associate resources with the fulfillment of operational and capability requirements (functions).
«Definition»
System performance is assessed vis-à-vis the desired capability. Results which appear to be unacceptable according to current strategic guidance (requirements) are identified, as well as the functional needs (gaps) that so arise.
must be assessable in terms of performance vis-a-vis mission
logical architecture
facility interface
mission resources
«FunctionAction»
Define System Model
physical architecture
: Synthesized System Model
physical architecture
«FunctionAction»
Evaluate System Performance
draft report
[gap==NO]
[gap==YES]
gap?
draft report
«FunctionAction»
Conduct Program Review of Assessment Results
final report
«ExchangeElement»
report : Architectural Study Results

Intentionally blank.

3.3.1 Developing a generic physical architecture

3.3.1.1 Defining partitioning criteria

Before a proper physical architecture can be developed it is necessary to define the criteria that should be satisfied. Typical partitioning criteria used to help group logical functions together at some level in a physical hierarchy include:

- Reuse, GFE, COTS, and other design constraints—e.g., feature-based requirements, use Part No. "XYZ."
- Physical or environmental—e.g., location, power.
- Safety and security—e.g., barriers, access controls.
- Subcontractor or development responsibility—e.g., Department 123 is responsible for power supply design.

As in logical architecting, common practice is to use a heuristic approach in applying these criteria, although formal methods do exist that support partition optimization (a topic that is beyond the scope of this paper).

3.3.1.2 Producing a generic physical hierarchy

Developing the definition of a physical architecture is done one level of a hierarchical tree at a time. At this point in the design process, a *generic* physical architecture (GPA) is defined for every logical (or functional) architecture carried forward. By *generic* it is meant that the partitioning is made without any specification of the performance characteristics of the physical resources (including software) that comprise each element. (Note that in much of the object-oriented design literature, a generic physical architecture is often referred to as a *node* diagram.) Development of the hierarchy can proceed either through decomposition or composition, following the same methods as described for functional or logical architectures (as above).

3.3.1.3 Allocating logical blocks to the generic physical architecture

All logical components (with their linked requirements) are allocated to elements (nodes) in the GPA; this may be a one-to-one or a many-to-one assignment. In addition, all design constraints are allocated to the elements as applicable.

3.3.2 Developing and assessing the "as-is" system

3.3.2.1 Mapping the "as-is" capability portfolio

The "as-is" DOTMLPF resources that represent the deployed capability baseline—as identified in §3.2.2.2.3—are linked to the appropriate GPA nodes. Make note of any gaps in coverage.

3.3.2.2 Developing physics-based framework and "as-is" component models

Although the details are beyond the scope of this paper, at this point it is necessary to develop appropriate system-level physics-based models that will form the "backbone" or basis by which the effectiveness (performance against mission needs) that the "as-is" and alternative system concepts will be evaluated. Details should be documented in a Modeling & Analysis plan, as noted for other steps. Note that it is generally considered to be a requirement for the analytical framework to be fixed across all as-is/what-if/to-be evaluations in order to avoid questions concerning the validity result comparisons. Only "component" level details will differ to accommodate application of different technologies within the framework.

3.3.2.3 Assessing the "as-is" capability

Logical verification objectives and supporting test cases are recast to reflect the physical architecture and components. Test cases are evaluated using the physics-based models, compared to requirements, and documented as the baseline. Deficiencies or gaps are duly noted. Sensitivity analysis should be performed to understand margin and risk vis-à-vis mission success.

3.3.3 Developing alternatives

Where the ECE process has been invoked, e.g., to develop changes to a baseline capability as a result of perceived needs or actual operational failures, alternatives should be methodically investigated in order to support rational decision making processes.

3.3.3.1 Identifying alternatives

For each component (element or leaf node) in the GPA, identify different solutions or technologies (specific instantiations) that can likely be used to satisfy the allocated

requirements and constraints (noting that a "component" may be a person or people performing some role). The more "what-if" options identified means there is a greater chance that the best alternatives will be considered in the final analysis (the exception being the case where a single, specified "to-be" capability configuration is being evaluated against a baseline). That is, even if some of the options are never selected in the final set of alternate architectures, it is generally considered that a great advantage accrues to the effort by generating a creative set of choices. The alternatives identified for each component should be documented in *Technical Basis Reports* (TBRs).

3.3.3.2 Instantiating alternative physical architectures

The list of component alternatives is combined with the GPA in order to identify alternative *instantiated* physical architectures, where *instantiated* refers to the use of specific technologies to implement the system. The two most frequent techniques used to represent this combination are the *morphological box* and the *trade tree*, discussions and examples of which follow below.

A morphological analysis (MA) divides a problem into segments and identifies *at least two* solutions for each segment. These solutions represent an important opportunity for creativity, and, conversely, any segment with only one fixed choice represents a constraint on the problem-level solution. Typically, a problem being evaluated using MA is portrayed as a table with the columns (or the rows) representing the problem segments (i.e., the generic components of one GPA). The row entries (or columns) are filled with the alternate specific instantiations identified for each component. *One* alternative instantiated physical architecture is represented by a selection of one component instantiation for each generic component.

The second common approach is to use a *trade tree*, which is a graphical method of capturing alternatives. In this method, each nodal layer (decomposition or branch level) represents a problem segment, and, from each node, a branch is added for each proposed segment solution. When a node only supports one branch, a constraint exists on the problem-level solution. Each path though the tree—from the root node to a leaf node—

represents one alternative instantiated physical architecture. Thus the total number of alternatives is given by the number of leaf nodes.

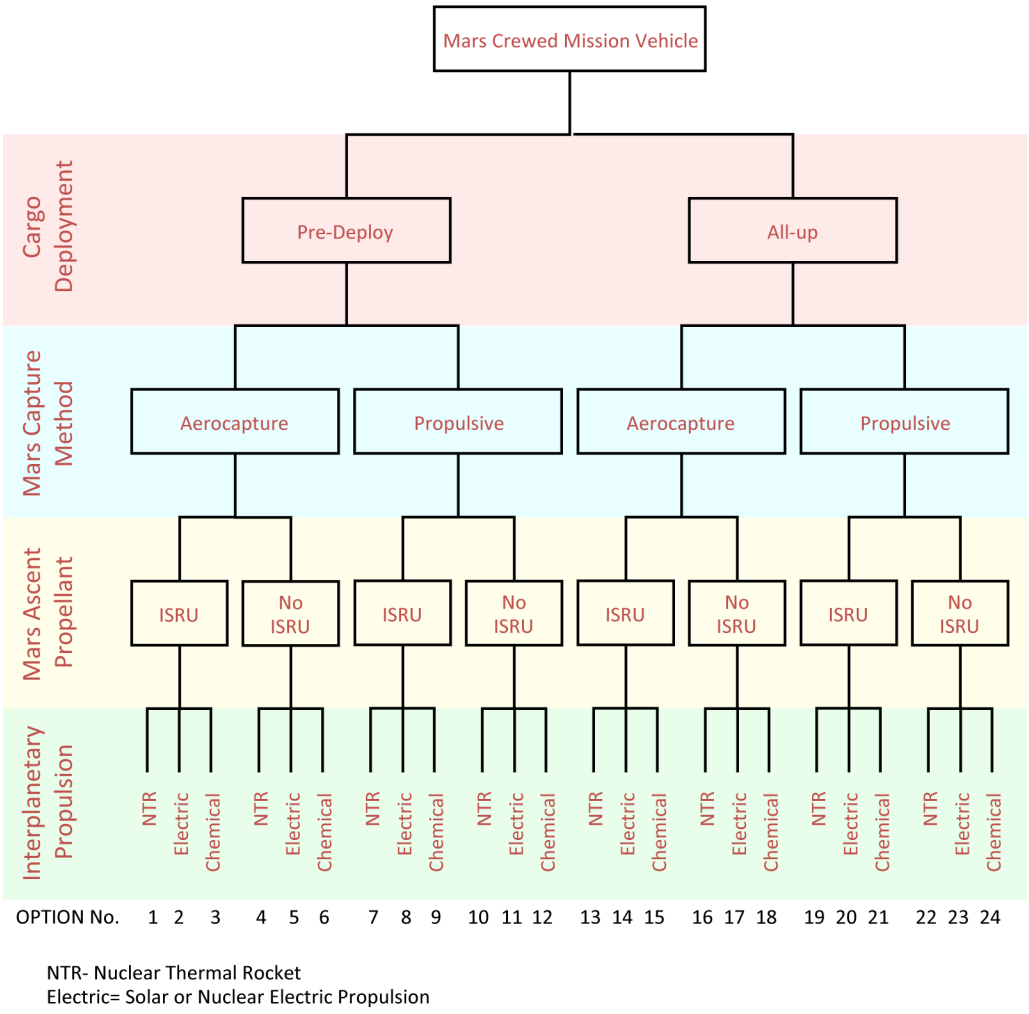


Figure 77. Top-level trade tree example (NASA Manned Mars Mission).[91]

3.3.3.3 Eliminating infeasible alternatives

It is not to be expected that the technologies associated with all component-level (resource) alternatives are compatible at a system (service) level, and thus any instantiated physical architecture defined by a mix of *incompatible technologies* is

[91] Adapted from NASA Mars Assessment Working Group study as presented in Guerra, Lisa, "Trade Studies Module," *Space Systems Engineering*, version 1.0, NASA Exploration Systems Directorate, 2008.

eliminated at this point in the process from further consideration. In general, identifying infeasible combinations proceeds by performing pairwise comparisons between all component alternatives across problem segments (but not within). Managing the comparisons can be, e.g., through the use of an upper triangular matrix, or perhaps through the "roof" (correlation matrix) of a "House of Quality" planning matrix used in the Quality Function Deployment (QFD) method. If a *trade tree* or equivalent is being used to define the different instantiated physical architectures, non-viable alternatives should be "pruned."

3.3.3.4 Producing a manageable option set

When a design problem only contains a few components—each with several technology options—the number of alternatives to evaluate can be quite large. For real systems there are usually millions of possible combinations, evaluation of which is likely intractable, making a preliminary screening necessary. The exact screening criteria will depend on the available analysis resources and the number of alternatives to be carried forward. Possible technical screening criteria might include:

- technical maturity of a component option
- similarity of alternatives
- perceived uncertainty in cost, risk, and effectiveness estimates
- sensitivity of system performance to key assumptions
- insensitivity to uncertain requirements
- vulnerability to threats
- flexibility
- reliability
- maintainability

In addition, there may be "political" (managerial) disqualifying factors to evaluate as well, such as an advocated alternative, or contributions to longer-term organizational goals.

Since this preliminary screening is usually done with limited data derived for alternatives whose definitions are still in transition, alternatives should be given the benefit of the

doubt. A corollary is that while this preliminary screening will have to rely on the experience and expectations of the system designers involved in the process, care must be taken that individual biases do not eliminate promising options; rational thinking is beneficial!

The bottom line is that the potentially large number of alternative instantiated physical architectures must be reduced to a smaller number of serious contenders that can be evaluated within programmatic constraints. On the other side, it is common but unfortunate that many design teams only consider one or two architectures in any detail, making it very likely that they are missing several creative, high-quality designs.

In the end there is no "formula" for doing this, except that screening criteria must be identified and agreed to before hand, and then must be methodically applied to the identified options. It is imperative to document the basis for eliminating each alternative from consideration at the time it becomes clear that it is "pruned." This documentation should be included in the final conceptual design report, and it will provide an audit trail that may be very important in the event the results are questioned or if new technology developments occur that invalidate assumptions made.

3.3.3.5 Documenting screened concepts

Documenting the final set of instantiated physical architectures (system concept alternatives) that are to be analyzed in further detail is required. In a formal, document-centric sense, this usually takes the form of a Technical Description Document (TDD) that is developed for each alternative, and for which standard templates are available. Model-based concept descriptions using a MBSE OOA/D methodology are another, preferred possibility. These descriptions should be used as "living documents" that are populated with information as it is developed throughout the definition and analysis process (i.e., in the verification and validation (V&V) activities, and trade studies, that will follow). To be complete, TDDs are required to include complete traceability (e.g., from component-level requirements up).

3.3.3.6 Physics-based framework updates with "what-if/to-be" component models

As noted earlier for the "as-is" model, while the details are beyond the scope of this paper it is necessary here to develop appropriate component-level physics-based models that will integrate with the analytical framework. Following integration, each concept analysis candidate is evaluated in terms of mission effectiveness against established test scenarios. Work should be guided by an approved Modeling & Analysis plan, as noted for other steps. Results should likewise be documented in a suitable venue.

3.3.3.7 Performing trade studies

Following an approved formal decision making methodology (details of which are beyond this paper[92]), the results from concept performance evaluations and mandated constraints are compared and a recommended solution is identified.

3.3.4 ECE Process Phase-3 revisited.

A "fourth" phase is initiated—typically following a phase-gate review—when management requires alternative solutions to be evaluated and compared in support of development decision processes (e.g., when it is desired to satisfy a critical need). What this really amounts to is that Phase 3 is repeated for selected alternatives. (Although note that radical innovation may require earlier phases to be repeated as well.) It is generally desirable to identify and evaluate alternatives that cover the joint spectrum of possibilities of both new material (technology) and non-material ("soft" wares) approaches, including combinations thereof.

---

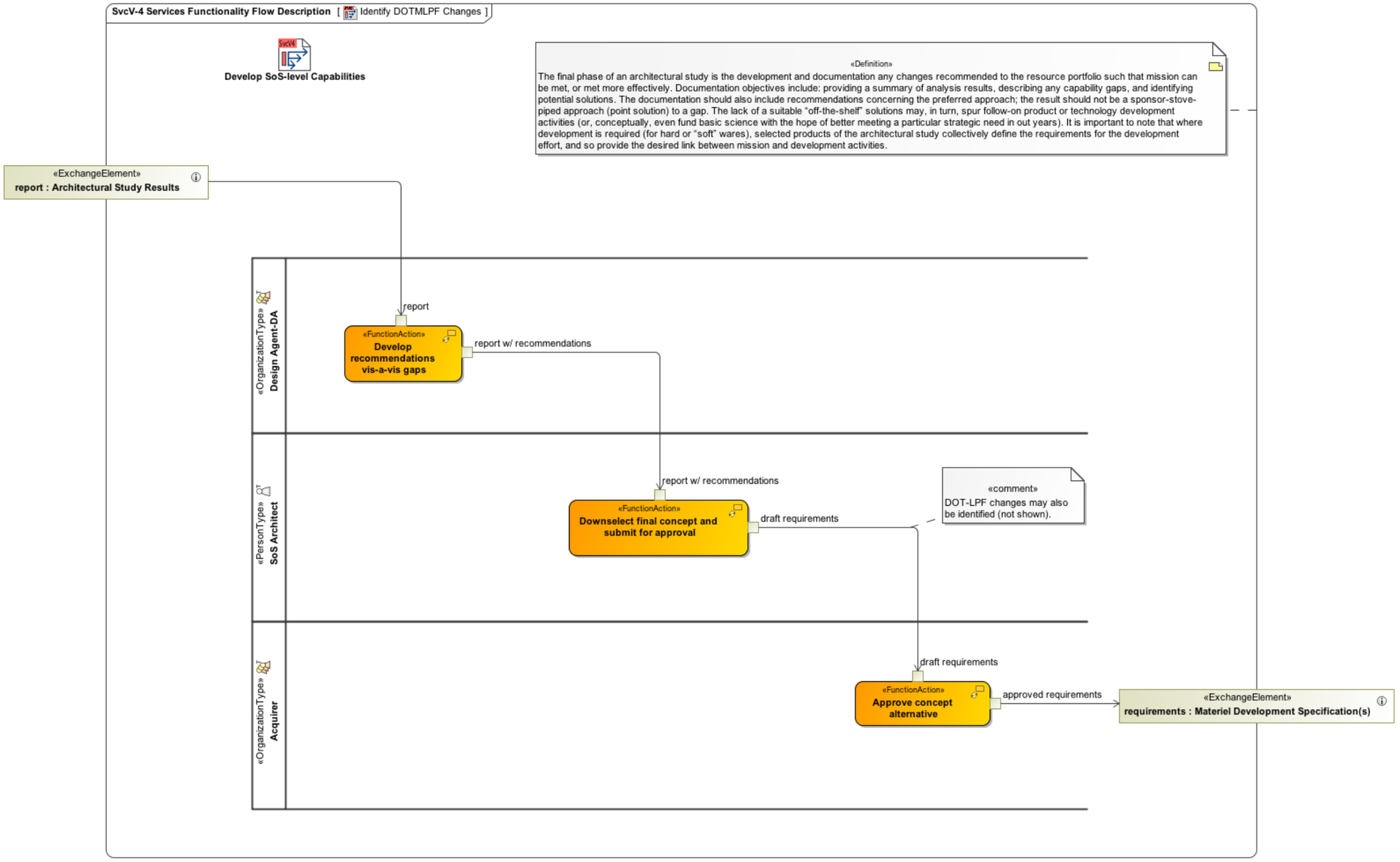
[92] Cf. Defense Acquisition University, "Trade Studies," Chapter 12 in *Systems Engineering Fundamentals*, Defense Acquisition University Press, Fort Belvoir, VA, January 2001; *Analysis of Alternatives (AOA) Handbook: A Practical Guide to Analyses of Alternatives*, Office of Aerospace Studies, AFMC, Kirtland AFB, NM, July 2010.

Intentionally blank.

3.4 ECE Process Closeout.

The final step of the ECE process is the production of a document that captures the outcomes of Phase-1 through -3, with objectives that include: providing a summary of analysis results, describing any capability gaps, and identifying potential solutions. The final report should also include recommendations concerning the preferred approach, which should not be a sponsor-stove-piped approach (point solution) to a gap. The lack of suitable "off-the-shelf" solutions may, in turn, spur follow-on product or technology development activities (or, conceptually, even fund basic science with the hope of better meeting a particular strategic need in out years). It is important to note that where development is required (for hard or "soft" wares), selected products of the process collectively define the requirements for the development effort, and so provide the desired link between mission and design and development activities.

Intentionally blank.


SvcV-4 Services Functionality Flow Description [Identify DOTMLPF Changes]
Develop SoS-level Capabilities
«Definition»
The final phase of an architectural study is the development and documentation any changes recommended to the resource portfolio such that mission can be met, or met more effectively. Documentation objectives include: providing a summary of analysis results, describing any capability gaps, and identifying potential solutions. The documentation should also include recommendations concerning the preferred approach; the result should not be a sponsor-stove-piped approach (point solution) to a gap. The lack of a suitable "off-the-shelf" solutions may, in turn, spur follow-on product or technology development activities (or, conceptually, even fund basic science with the hope of better meeting a particular strategic need in out years). It is important to note that where development is required (for hard or "soft" wares), selected products of the architectural study collectively define the requirements for the development effort, and so provide the desired link between mission and development activities.
«ExchangeElement»
report : Architectural Study Results
«OrganizationType»
Design Agent-DA
report
«FunctionAction»
Develop recommendations vis-a-vis gaps
report w/ recommendations
«PersonType»
SoS Architect
report w/ recommendations
«FunctionAction»
Downselect final concept and submit for approval
draft requirements
«comment»
DOT-LPF changes may also be identified (not shown).
«OrganizationType»
Acquirer
draft requirements
«FunctionAction»
Approve concept alternative
approved requirements
«ExchangeElement»
requirements : Materiel Development Specification(s)

Intentionally blank.

4 SUMMARY

By following a top-down approach in determining requirements: strategic planners link the purpose of an enterprise with strategic goals and objectives (mission); enterprise architects link mission to capability performance—and future development—needs; and system analysts evaluate alternative resource mixes in providing the desired capabilities, linking the results to development (material or non-material) requirements when new solutions are called for. The causal chain so described—and documented—provides the necessary assurance that development, design, and deployment activities are mission oriented, and, therefore, that the solution produced by the effort will be capable of supporting the desired level of capability improvement (generate the desired effects) once it is available and integrated at the enterprise level.

Intentionally blank.

APPENDIX A. Eliciting Stakeholder Requirements

At times requirements engineering needs dictate that stakeholders must be polled in order to "collect" system "requirements." The assigned "collector" will either send a request for information to—or conduct an interview with—each identified stakeholder group in order to produce a list of that group's special (particular, unique) requirements. Unfortunately the summation of collected responses are often defined to be the system requirements; this creates a number of problems such as:

- Practically guarantees that the device or function of a particular stakeholder group will be represented whether it is needed or not.
- The requirements so established are often on the basis of state-of-the-art technology rather than on the basis of meeting higher-level needs.
- There is a compelling tendency to mix technical solutions with requirements.
- Individual quantified requirements lose (or never have) traceability to mission requirements.

Rather, the information collected should inform the requirements development process as outlined below, not be the process!

Because the requirements collected typically use narrative language—possibly of a subjective nature or with poor grammar—it is generally necessary to expend some effort rewriting them before they can be used (think risk reduction). Basic grammar rules and desirable attributes associated with "good" requirements should be followed. In summary, however, a requirement has four parts: (1) conditional statement; (2) subject; (3) *shall* verb phrase; and (4), clarifying phrase. A typical list of requirement attributes might include:

- correct
- consistent
- unambiguous
- verifiable
- succinct
- feasible
- necessary

- simple, singular, independent
- design-free (*what* not *how*)
- sufficient and complete
- traceable
- externally observable
- reviewed, approved, and baselined by the stakeholders, yet modifiable (i.e., under configuration management, CM, control)

Requirements that do not meet these desired characteristics are "risky." The risk of concern here is that generated by poor problem definition (i.e., the wrong problem being solved), not that created by the selected technology or design solution per se.

APPENDIX B. Vision, Goals And Objectives

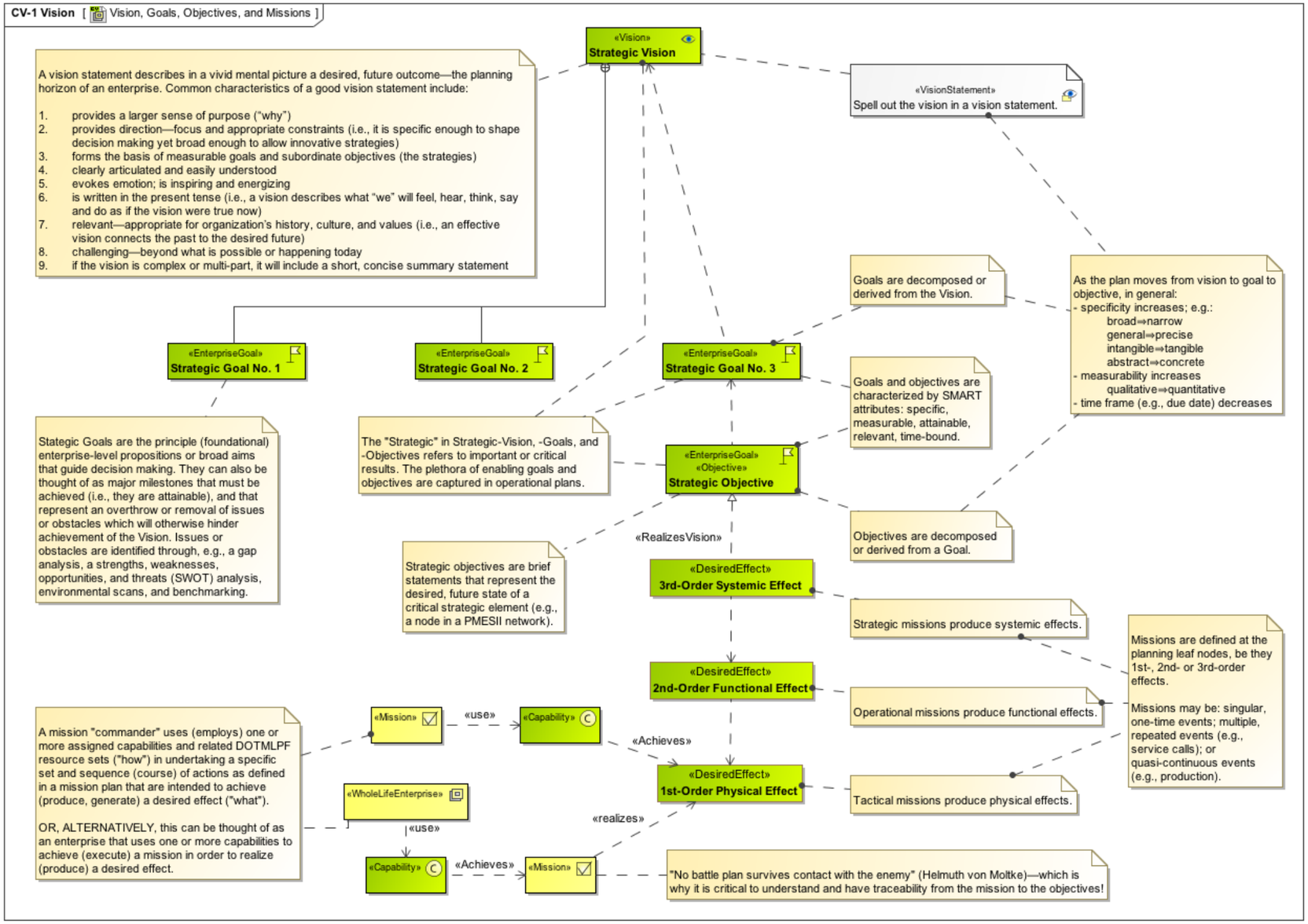


Figure 78. Basic descriptions and considerations of strategic vision, goals and objectives.

Intentionally blank.

APPENDIX C. A Digression on *Views* and *Viewpoints*

The approach toward assuring that the motivating concepts of a system are effectively communicated to the stakeholders—to demonstrate their concerns have been addressed—requires the introduction of several concepts embodied in the following definitions:

> **View**: "A representation of a whole system from the perspective of a related set of concerns." (IEEE Std 1471 §3.9)
>
> **Viewpoint**: "A specification of the conventions and rules for constructing and using a view. A pattern or template from which to develop individual views by establishing the purposes and audience for a view and the techniques for its creation and analysis." (IEEE Std 1471 §3.10)

That is, a *view* is a particular, limited, crosscutting look at an architectural description intended to portray the system information needed to address a particular concern. (And the set of views provides, by definition, a complete architectural description, given that all stakeholders and their concerns have been addressed.) A *viewpoint*, on the other hand, provides definition of a standard way to communicate the information content found in a *view.* In different words: a *view* defines **what** information is to be presented, while a *viewpoint* defines **how** that information is to be presented. From these definitions it can be asserted that viewpoints should not be system specific (unlike the stakeholders, views or architectural descriptions), but, rather, are intended to be patterns (a language, if you will) that address system-independent issues (i.e., the concerns) like security, performance, structure, data, etc. To clarify the relationships that exist between system, architecture, architectural description, stakeholders, concerns, views, and viewpoints, consider Figure B-1 (reproduced from IEEE Std 1471 Figure 1).

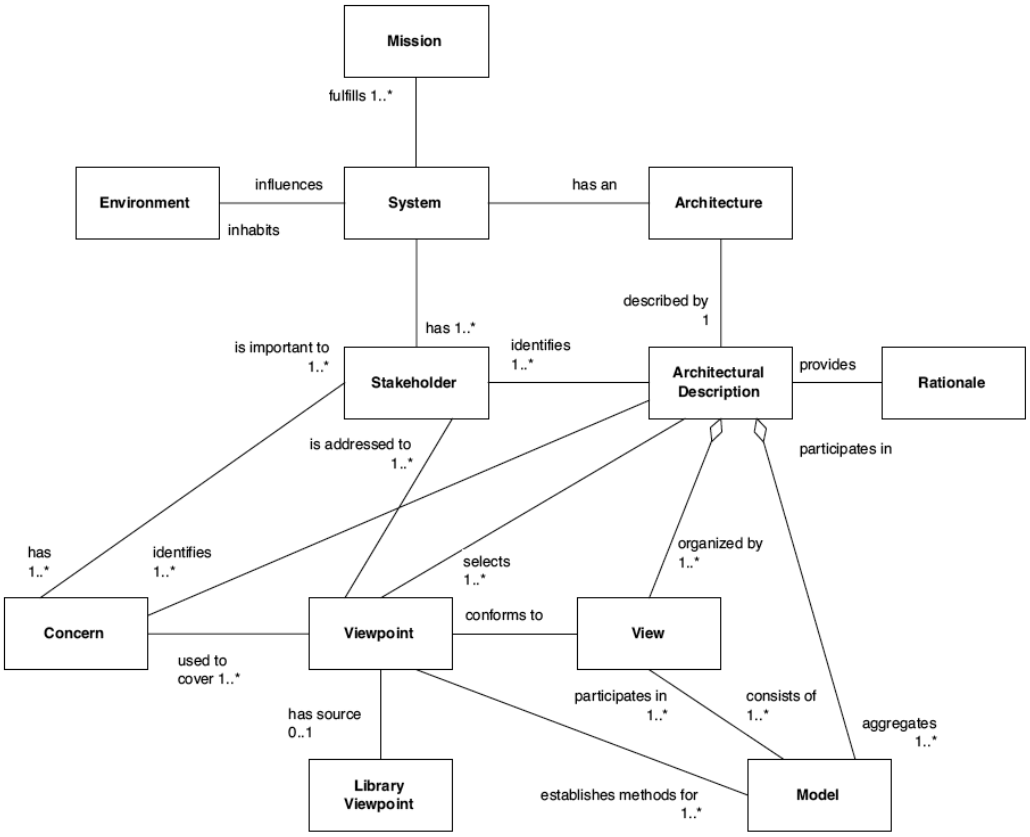


Figure 79. Conceptual model of architectural description.

Viewpoint Expounded

If it still remains somewhat fuzzy, however, of just what a *viewpoint* is, consider the following description of the expected content (here IEEE Std 1471 §5.3):

> "Each viewpoint shall be specified by
>
> a) A viewpoint name,
>
> b) The stakeholders to be addressed by the viewpoint,
>
> c) The concerns to be addressed by the viewpoint,
>
> d) The language, modeling techniques, or analytical methods to be used in constructing a view based upon the viewpoint,
>
> e) The source, for a library viewpoint (the source could include author, date, or reference to other documents, as determined by the using organization).
>
> A viewpoint specification may include additional information on architectural practices associated with using the viewpoint, as follows:

— Formal or informal consistency and completeness tests to be applied to the models making up an associated view

— Evaluation or analysis techniques to be applied to the models

— Heuristics, patterns, or other guidelines to assist in synthesis of an associated view

Viewpoint specifications may be incorporated by reference (such as to a suitable recommended practice or previously defined practice).

An AD shall include a rationale for the selection of each viewpoint.

The rationale shall address the extent to which the stakeholders and concerns required in 5.2 are covered by the viewpoints selected under this clause."

The discussion of *viewpoints* in IEEE Std 1471 also includes helpful examples in Annex C and Annex D. It is important to note that *viewpoints* are **required** for standards-based approaches:

> "An AD [architectural description] shall identify the viewpoints selected for use therein ... [and] shall include a rationale for the selection of each viewpoint. ..."[93]

As performed for *concerns* (above), *viewpoints* can be documented by using UML classes. Then, by portraying the viewpoints on a diagram (e.g., *OV-4*) along with the stakeholders and concerns, appropriate *association* relationships can be established. As an alternative, note that SysML provides a "viewpoint" stereotype[94] that aligns fairly well with IEEE Std 1471 which includes the following attributes: stakeholders, purpose, concerns, languages, and methods.

View Expounded

Following *viewpoint* selection, one *view* is defined for every *viewpoint* (a one-to-one relationship).[95] As defined above, a *view* is a "representation of a whole system from the perspective of a related set of concerns,"[96] but it must be understood that this "representation" is one of organization (a selective index, if you will) and **not** the actual

---

[93] IEEE Std 1471 §5.3
[94] OMG SysML 1.3 §7.3.2.5
[95] Be warned that the use of the terms "view" and "viewpoint" within UPDM is **not** consistent with IEEE Std 1471, SysML or DoDAF as followed herein.
[96] IEEE Std 1471 §3.9

architectural model content. This should be apparent when considering the fact that, at this point in the ECE process, details of the architecture have yet to be captured. Rather, a *view* can be thought of as being analogous to a graphical user interface (GUI) "behind" which the software (S/W) coding has yet to take place—at this point in the process it is non-functional. Consider as an example the following package diagram:[97]

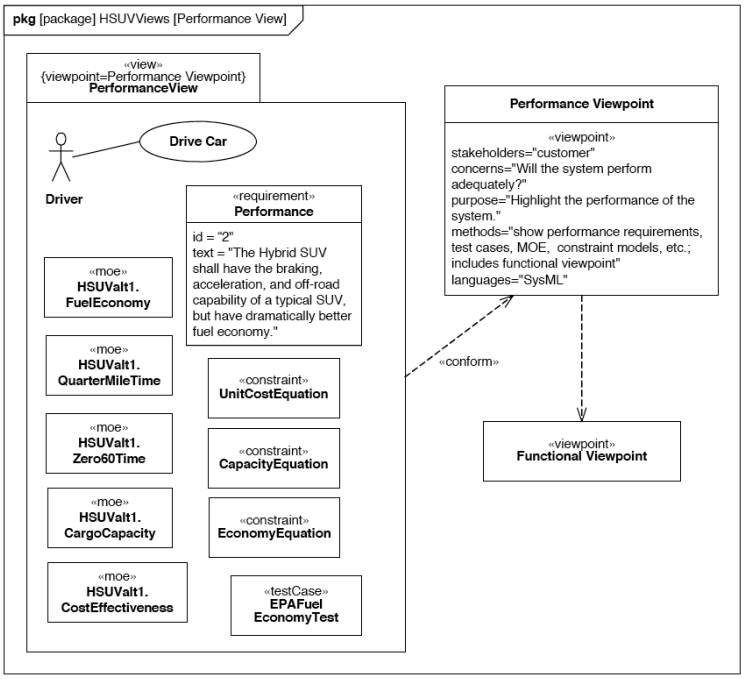


Figure 80. Example view: establishing a performance view of the user model.

In the words of the SysML specification:[98] "A view can only own element import, package import, comment, and constraint elements." That is, a view diagram does not "own" the architectural description, but is an extended (stereotyped) package that is used to group (point to) the set of model diagrams (behavior, structure, or requirement) or model elements (e.g., typed blocks) necessary to demonstrate that the system will satisfy the concerns of the respective stakeholder. In some sense the set of *views* as developed at this point can also be thought of as providing a validation plan for ensuring stakeholder *concerns* are met.

---

[97] OMG SysML 1.3 Figure C.27
[98] OMG SysML 1.3 §7.3.2.4

Architectural Framework

While IEEE Std 1471 provides specification of what a *view* and a *viewpoint* is, and OMG SysML provides guidance on how to document a *view* and *viewpoint* in a graphics-based model, neither standard provides actual *viewpoint* specifications that can be used to define *views* and the supporting *models.* This need can be filled most readily by the adoption of an architectural framework—as opposed to the effort that would be required to develop such a framework "from scratch"—which is where DoDAF comes in. (Note also that the *models* framework of choice should have been selected in ECE process Phase-1 and documented in the *scope* section of an *AV-1*.)

DoDAF provides a means of representing architecture content—it defines *viewpoints*—that is prescribed for use by all DoD organizations (although "Fit for Purpose" views are allowed[99] to cover the presentation needs of customers that are not addressed by the generic nature of the framework). From a standards-compliant perspective, *some* of the viewpoint elements specified by IEEE Std 1471 can be documented within the descriptions found in the DoDAF *AV-1* report (cf. Figure 14), although the intended **strong** linkage (traceability) between stakeholders, concerns and viewpoints would be missing. In part this is supplied by a close reading of the different viewpoints as defined within DoDAF, which in some, perhaps abstract, sense, contain implicit information regarding generic stakeholders and their concerns; only for the "Capability Viewpoint" is explicit mention made that the intended use is "to address the concerns of Capability Portfolio Managers" by describing "capability taxonomy and capability evolution." The DoDAF viewpoints also include model descriptions (the "modeling techniques" of IEEE Std 1471) to be used in constructing the view used in organizing the architectural description in conformance to each particular viewpoint (e.g., the various *AV-, CV-, DIV-, OV-, StdV-,* and *SvcV-* nomenclature used in reference to diagrams presented earlier in this paper are references to particular viewpoint model descriptions to which they were intended to comply—each dash number representing a specific model type within the

[99] Cf. DoDAF Vol. 2, §3.1.9.

particular viewpoint). Therefore, if the *AV-1* scope description is updated to make reference to a set of viewpoints, each viewpoint, in turn, at least implies, in a generic sense, the type of stakeholder and the concerns that will be addressed by using one or more of the models defined therein to generate the content of a *view* that describes a specific system in a standardized way. The stakeholder (actual organization) information captured in an *OV-4* (as described above) could then be typed, or inheritance and aggregation used, as appropriate, by the various categories of stakeholders found in the DoDAF viewpoint descriptions, and supplemented with DoDAF concern statements. If necessary, one or more Fit-for-Purpose views (viewpoints) can be defined as well.

APPENDIX D. An Introduction to DoD MOEs, MOPs and KPPs

Mission Need Statement

A mission need statement (MNS) identifies and describes the mission need (deficiency) in terms of a stated mission, the mission objectives, and the capabilities required to execute the mission (generally speaking, **not** in terms of equipment or system-specific performance characteristics). An MNS also includes a validated, projected threat the mission is expected to have to counter, as well as constraints (e.g., infrastructure and environments).

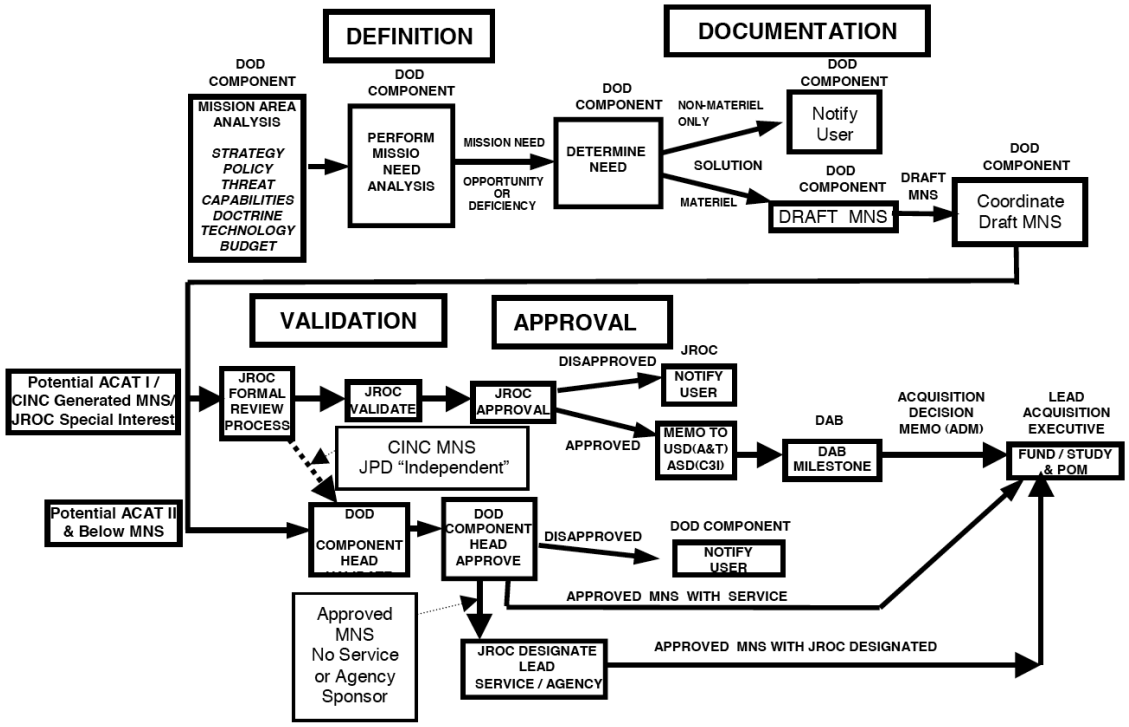


Figure 81. Basic MNS development process (from CJCSI 3170.01).

Figure 82. Simple mission need statement for illustration purposes only!

Mission Tasks

Mission tasks (MTs) are derived directly from the deficiencies (mission needs) identified in the MNS (and only these). They are usually expressed in terms of general tasks to be performed to correct the deficiencies (e.g., hold targets at risk, provide countermeasures against SAMs, or communicate in a jamming environment). MTs must not be stated in solution-specific language nor specify optimization.

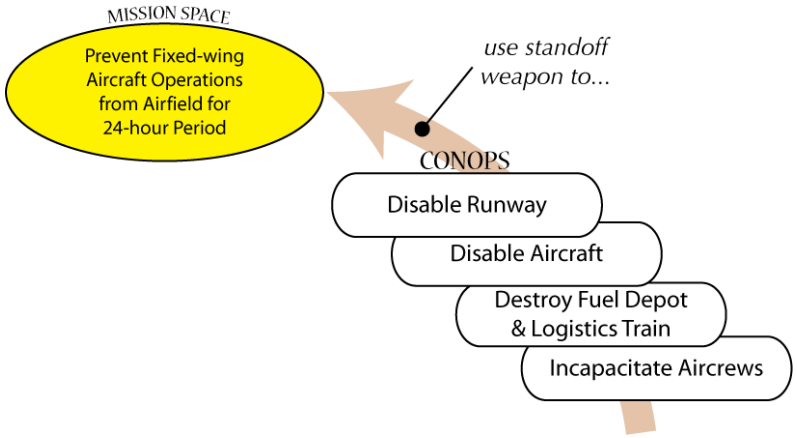


Figure 83. Simple mission tasks for illustration purposes only!

Measures of Effectiveness

Measures of effectiveness (MOEs) are developed to describe measures of how well a task is to be performed by a capability used in executing a mission. That is, MOEs contain the details of measuring proficiency in performing a task described by an MT, and they should normally represent raw (kind of) quantities (e.g., numbers of something or frequencies of occurrence) and not be of a complex or derived type; the preference is for quantitative MOEs, but at times it may be necessary to use semi-quantitative or even qualitative measures. One or more MOEs support each MT (multiple MOEs should be independent of each other, although they may be hierarchical), and they are MOEs only in relation to an MT (no quantity is inherently an MOE). MOEs are used to compare alternatives—they are used to express "worth" in effectiveness analyses and cost-effectiveness comparisons—and so are to be independent of the nature of the alternatives; they may also be used to investigate performance sensitivities to variations of key mission assumptions.

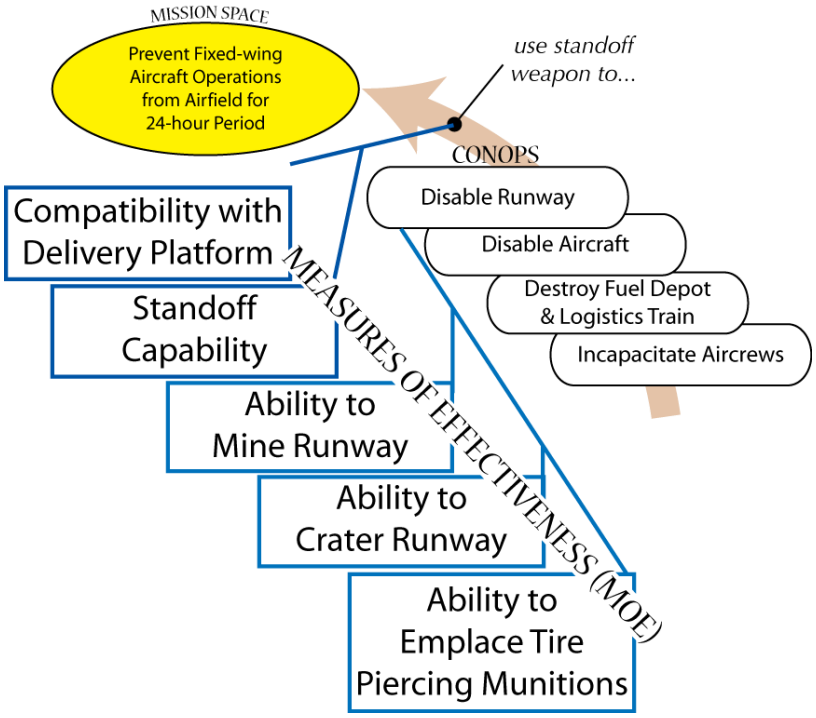


Figure 84. Example capabilities measured by MOEs for illustration purposes only!

Examples of high-worth (highly-significant) MOEs that might be used in DoD mission or campaign analysis-of-alternatives (AoA) models include:

- Time to accomplish high-level objectives
- Targets placed at risk (e.g., a target is at risk when an aircraft arrives undamaged at the weapon release point)
- Targets negated ("killed")
- Level of collateral damage
- Friendly survivors
- Numbers and types of resources used (e.g., sorties flown, attrition rate, bombs dropped)

Measures of Performance

Measures of performance (MOPs) are typically quantitative measures of system characteristics (e.g., range, velocity, mass, scan rate, weapon load-out) chosen to enable calculation of one or more MOEs. MOPs are generally universal to all alternatives, in which case they are generally only indirectly reflected in system performance parameters published in a system requirements document (SRD). In some cases MOPs have to be

system specific, and so are directly reflected in an SRD. Publication of MOPs should be accompanied by a MOE sensitivity analysis that assesses the impact of uncertainties or variations in MOP values.

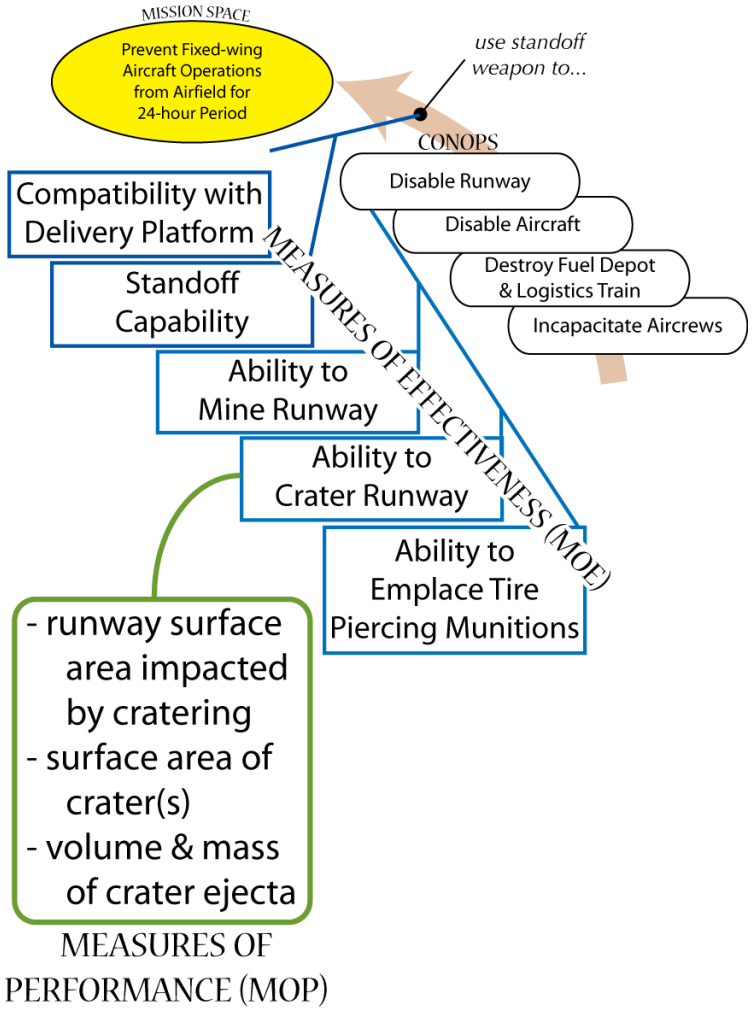


Figure 85. Example MOPs for illustration purposes only!

Materiel (Technology) Alternatives

As noted above, MOEs and supporting MOPs are operational-centric, and are intended to be agnostic to the alternatives considered to meet the mission need. Once an operational model is developed, including, e.g., definition of related MOEs and MOPs, various alternative capability configurations of the system model are synthesized.

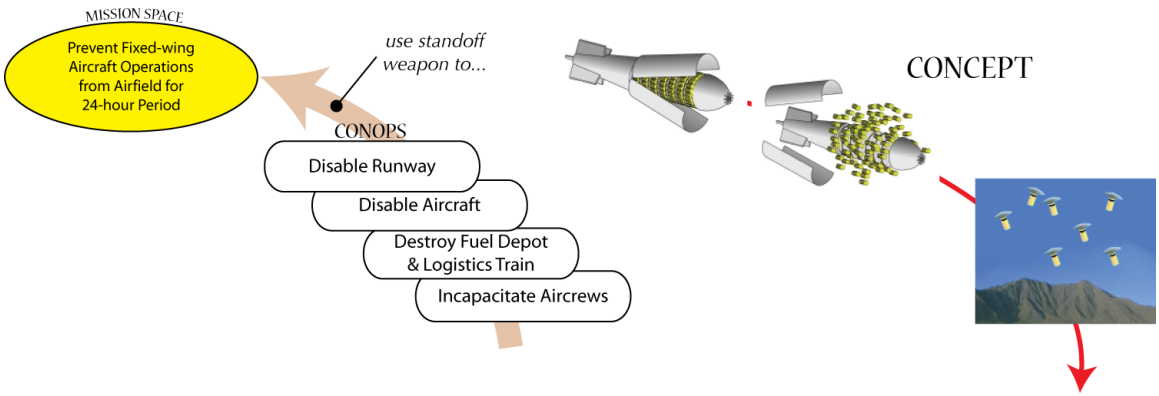


Figure 86. Example concept (1 of *n* alternatives) for illustration purposes only!

Effectiveness Analysis

Once a set of viable alternatives has been defined, mission effectiveness for each concept must be evaluated in terms of military worth (e.g., conduct an AoA). This mission performance analysis includes appropriate consideration of the MOEs, MOPs, threats, scenarios, and concept of operations (CONOPS).

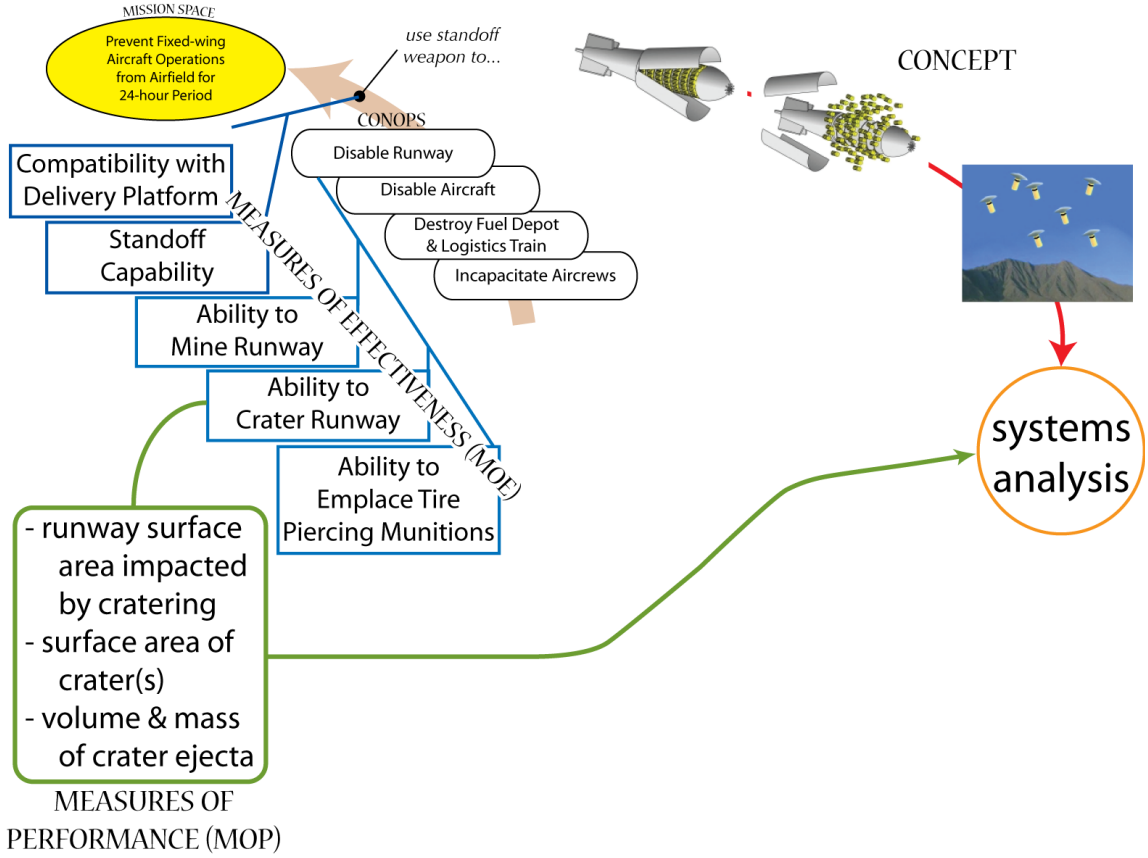


Figure 87. Example information flow in a concept effectiveness analysis.

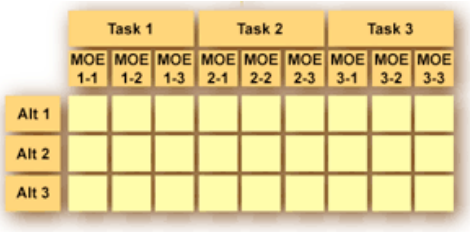


Figure 88. Illustration of a table for mission performance comparisons.

Refine Selected Concept

Once a preferred concept is identified (down selected) on the basis of mission effectiveness and other critical attributes (e.g., cost), an appropriate set of system requirements must be developed. Of particular interest here, said SRD shall include key performance parameters (KPPs) that collectively establish the determination of the MOPs that apply. That is, just as MOPs are alternative-agnostic measures of system characteristics that enable calculation of one or more MOEs, KPPs are measures of a particular alternative that enable calculation of the MOPs, and so MOE(s).

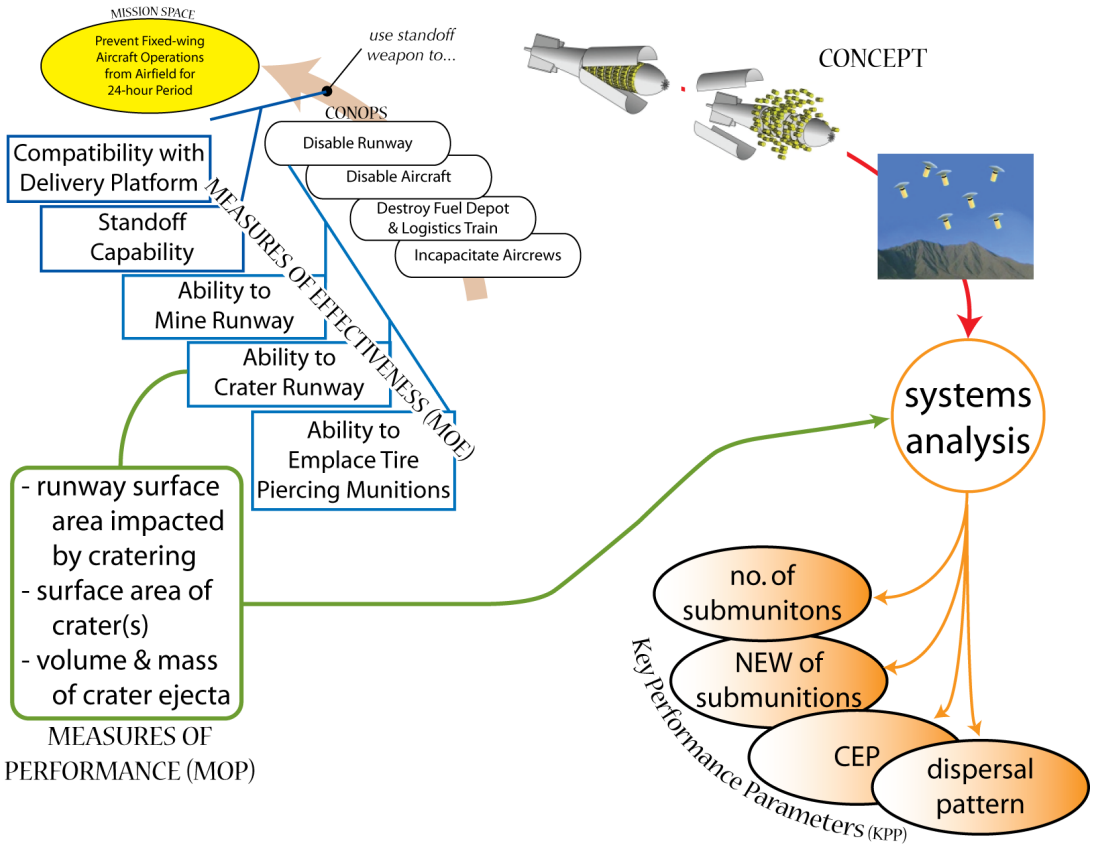


Figure 89. Example relationship between KPPs, MOPs, and MOEs.

Note that the “key” in KPP is generally interpreted by DoD directives as a limit on the maximum number of KPPs that apply for a given system—eight! Also note that there are related, but different concepts referred to as Technical Performance Parameters (TPPs) and Technical Performance Measures (TPMs) which are not addressed herein.

Intentionally blank.

APPENDIX E. Capturing Requirements in DoDAF

Technical requirements are of two basic types: performance requirements and constraints. In this sense, *performance requirements* define both what must be done (aka functional requirement) and how well it must be done (aka functional performance requirement), and are related directly to mission performance parameters such as MOE, MOP and KPP (cf. APPENDIX D). *Constraints* are boundary conditions or limitations on the solution space that are not directly related to required mission performance.

Traditionally, requirements management would be text based (e.g., using worksheets of some appropriate style). However, this approach tends to "break" current state-of-the-art models-based techniques. Behavioral and structural models within DoDAF offer a means to capture performance requirements from an MBE perspective; in addition, DoDAF offers a variety of "transition" techniques to capture text-based requirements and link them into an architectural model, as outlined below. (While beyond the scope of this appendix, also note that programmatic requirements can be captured in DoDAF as well—e.g., in *CV-3* and *PV* diagrams.)

All Viewpoint

The *AV* models are used to capture overarching aspects of an Architectural Description. In particular, the *AV-1* provides executive-level summary information, including context (cf. §3.1). From a requirements perspective, context includes both setting (e.g., mission, doctrine, relevant goals and vision statements, concepts of operation, and scenarios) as well as a list of authoritative sources for the standards, rules, criteria, and conventions that are used in the architecture. All requirements found in the Architectural Description should have traceability to *AV-1* context properties, although this may be a challenge in practice within the DoDAF model due to vendor tool limitations; use of Fit-for-Purpose views should be anticipated. Test measurements associated with verifying that *AV-1* requirements have been complied with can be defined in a Fit-for-Purpose view, such as illustrated in Figure 10 and Figure 90.

Capability Viewpoint

The strategic context—e.g., vision and high-level "goals, together with the desired outcomes and measurable benefits"—for the capabilities described in an Architectural Description are captured using diagrams based on a *CV-1* model. Construction of a *CV-1* is discussed in §3.2.1.2 and APPENDIX B. For an example of how measures can be incorporated in a *CV-1*, see Figure 8, where Fit-for-Purpose notation was adapted from UML and SysML. To establish traceability from the strategic context to the *AV-1*, a Fit-for-Purpose view can be used, such as the SysML *bdd* example in Figure 90.

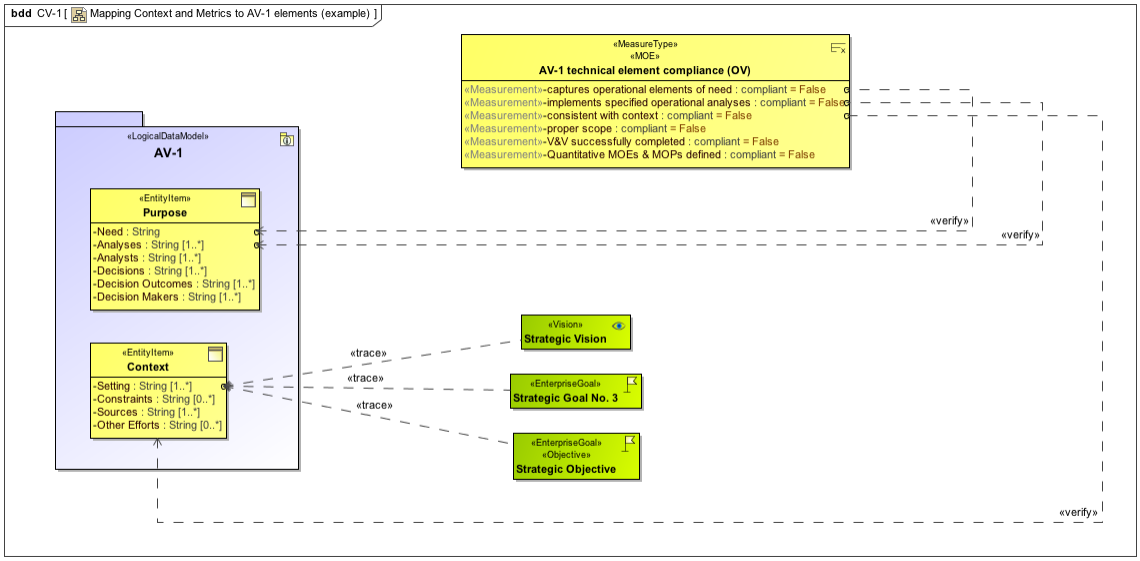


Figure 90. Example view illustrating strategic context and measures traceability.

Operational Viewpoint

DoDAF defines the *OV-6a* viewpoint model to capture relevant business (enterprise) or operational (mission-related) rules, as found in the strategic guidance defined in the applicable *AV-1*. DoDAF also specifies that the top-level rules are to serve as "guidelines for the development and definition of more detailed rules and behavioral definitions that should occur later in the Architectural definition process." This intended use—both the capturing and serving—implies (and rightly so) that such rules should have traceability. Strictly speaking, the rules captured in an *OV-6a* can be applied to any architectural element defined in any diagram built following an operational viewpoint (*OV*) model; such elements include, for example, a node, operational activity, entity item, mission,

exchange element, or operational exchange. Operational rules are likewise, per DoDAF, intended to be able to be presented in any *OV*. DoDAF states that the rules themselves will be specified in English in one of two natural language forms: imperative or conditional imperative statements.[100] Oddly, DoDAF does not make provision for defining performance requirements with an operational viewpoint, stating,[101] rather, that "Measures of Effectiveness (MOEs) and Measures of Performers (MOPs) are measures that can be captured and presented in the Services Measures Matrix model" (i.e., an *SvcV-7*). A similar model—the Systems Measures Matrix, *SV-7*—is defined in DoDAF, although interestingly the associated text in DoDAF v2.0, Vol. 2, §3.1.8.2.8, also makes reference to use of the *SvcV-7* model for capturing MOEs and MOPs! However, it should be noted that existing vendor tools (at least *MagicDraw*) have the following limitations:

- An *OV-6a* can only be represented as a table.
- Rules can only be constraints (although they may be named as different types of constraints, such as assertion, derivation, agreement, guidance, or policy).
- Rules cannot be shown on *OV* diagrams except as an internal "class" (e.g., performer or operational activity) constraint; rules that apply to other types of elements cannot be portrayed.
- No inherent mechanism is provided for upwards or downwards traceability.
- Measures can generally only be linked to services in an SvcV-7 (and to resources in an SV-7).

To provide a means for traceability, graphical portrayal, detailed (e.g., mathematical) specification, and analysis of operational constraints, it is recommended that each *OV-6a* constraint be "applied" to a SysML constraint block (cf. §3.2.2.2.2). Fit-for-Purpose views can then be developed by, e.g., use of *bdd* and parametric (*par*) diagrams.

To illustrate the interplay of capability and operational viewpoints, and SysML Fit-for-Purpose views, in identifying operational rules (at least as currently constrained by vendor tools), consider the following abbreviated example that follows after the MOE/MOP/KPP discussion of APPENDIX D. First, an *OV-6a* is created to capture the constraint (rule) governing the mission at hand, as in Table 9. Next, a *CV-1* is created (or

---

[100] Cf. 3.2.1.4 and APPENDIX A.
[101] DoDAF v2.0, Vol. 2, §3.1.6.2.8.

updated) to reflect mission-capability-effect-measure relations, and relevant (derived) constraints and parameters are defined (e.g., through the model containment tree interface); the *CV-1* is also annotated with mission objective and threat capabilities (cf. Figure 5), as illustrated in Figure 91. To graphically portray the constraints and parameters so defined, SysML constraint blocks and a *bdd* are used, as in Figure 92. Finally, the constraints are used in a parametric model, as shown in Figure 93, to enable use of effects measures as a metric vis-à-vis the mission MOE.

Table 9. OV-6a Operational Rules Model Example

Applies To	Rule Specification	Rule Kind
Disrupt OPFOR airfield operations.	Fixed-wing aircraft operations suppressed for 24 hours.	Constraint

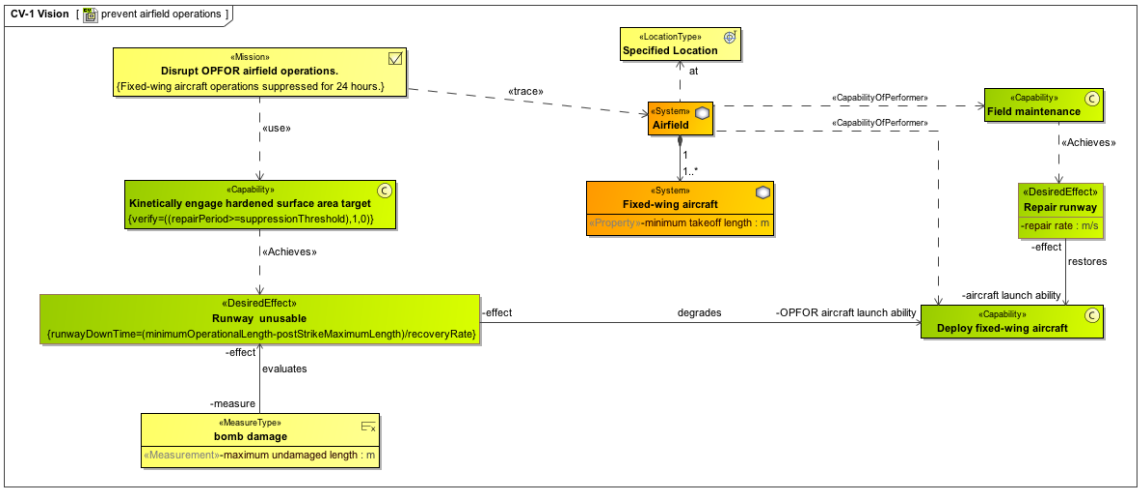


Figure 91. Example capability-effect-impact cycle for illustration purposes only!

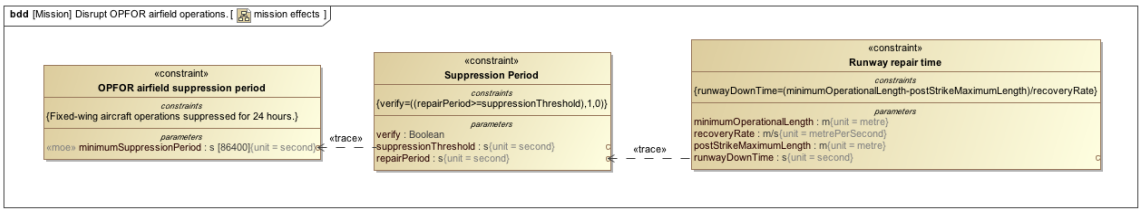


Figure 92. Example mission-related constraint set for illustration purposes only!

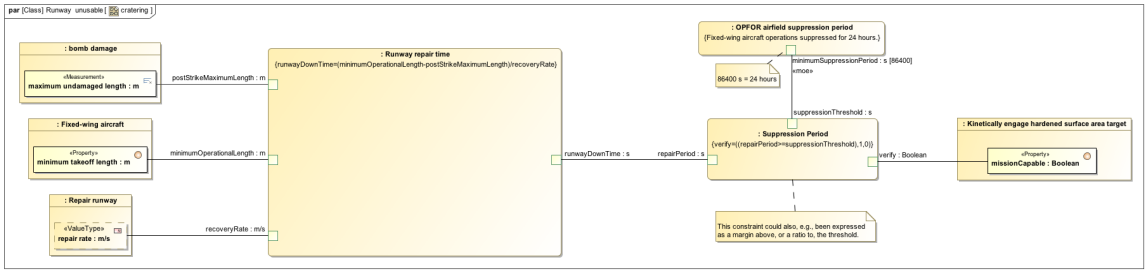


Figure 93. Example MOE metric parametric model for illustration purposes only!

Services Viewpoint

DoDAF defines two viewpoint models related to requirements capture at the services layer: the *SvcV-7* and the *SvcV-10a*. The *SvcV-10a* can be thought of as a model analogous to an *OV-6a*—with all the DoDAF model and vendor tool limitations thereof (see above)—that is intended to capture constraints on the service-level architecture, including those that apply, e.g., to resource flow, service function, data, and port elements. In contrast, the *SvcV-7* is intended to depict qualitative and quantitative "measures (metrics) of resources." (*Resource* is defined by DoDAF[102] to be "Data, Information, Performers, Materiel, or Personnel Types that are produced or consumed." However, consider also the definition found in §2.1.7.) Of particular note is the statement that "It specifies **all** of the measures [emphasis added]. ... One of the primary purposes of SvcV-7 is to communicate which measures are considered most crucial for the successful achievement of the mission goals assigned."[101] This includes, as noted earlier, MOEs and MOPs. Collectively these statements can be interpreted to mean that an SvcV-7 is appropriate for use to capture performance requirements within all viewpoint domains, and not just for services (although this is not supported in current tools)!

In UPDM, two *SvcV-7* matrices are defined. One of these is called "typical," and which includes provision for identification of "Measure Type" (think general measurement category) and "Measurement" (think ISO-80000-1 "kind of quantity" **and** "unit of measurement"). The other UPDM *SvcV-7* matrix type is called "actual," and which includes provision for "Measure," "Metric," and "Intention" fields for each entry that actually correspond, respectively, to the formal definitions of measurement and unit of

[102] DoDAF v2.0, Vol. 2, §2.2.1.

measurement, along with a notation concerning whether the "Measure" is a specified requirement or actual performance value. Existing vendor tools capture a "Measure Type" as a model object and each corresponding "Measurement" as an attribute thereof; this allows for some improvement over *OV-6a* and *SvcV-10a* constraints in supporting graphical displays of requirements information, although the allowable linkages (association types) are limited (e.g., Fit-for-Purpose stereotypes are likely required to capture traceability and satisfy relationships). For detailed specification and analysis purposes, it is recommended that each "Measure Type" be linked to a SysML constraint block, coupled with use of appropriate SysML diagram types, as previously illustrated.

Standards Viewpoint

The DoDAF *StdV-1* provides a means to capture authoritative references for the strategic guidance defined in an Architectural Description *AV-1*, as well as any other business or doctrinal standards deemed applicable to the particular architecture as it is developed. In addition, it should be noted that said references serve as the provenance for all business (enterprise) or operational (mission-related) rules invoked (i.e., all requirements should trace to one or more of these references). The *Standards Profile* model defined by DoDAF also serves to collate and associate references with elements of the architecture. This association can be thought of as a **weak** SysML requirements "allocate" (or, in the opposite sense, "satisfy") relationship; it is weak in that DoDAF §3.1.7.2.1 states: "Note that an association between a Standard and an architectural element should not be interpreted as indicating that the element is fully compliant with that Standard."

UPDM and vendor tools like MagicDraw implement the *StdV-1* as a table similar to the *OV-6a* and *SvcV-10a* except that model elements (class objects of predefined types[103] *Functional Standard*, *Technical Standard*, and *Protocol*) are used to capture reference definitions. This fact enables reference elements to be used in graphical views, and so

---

[103] *Functional Standards* set forth rules, conditions, guidelines, and characteristics (UPDM §6.7.2). *Technical Standards* document specific technical methodologies and practices to design and implement (UPDM §6.7.12). *Protocol* is a standard for communication (a MODAF, not DoDAF, compliant element; UPDM §6.7.3). Note also that MagicDraw allows such elements to be refactored into a generic *Standard* element.

facilitate elucidation of rule traceability. For example, if the mission in Figure 91 is associated with a "Design Reference Mission Document" via a *StdV-1*, traceability, in some sense, for mission requirements can now be displayed on the *CV-1* as illustrated in Figure 94. Or, for example, a graphical portrayal can be used (although traceability has to be added), as illustrated in Figure 95.

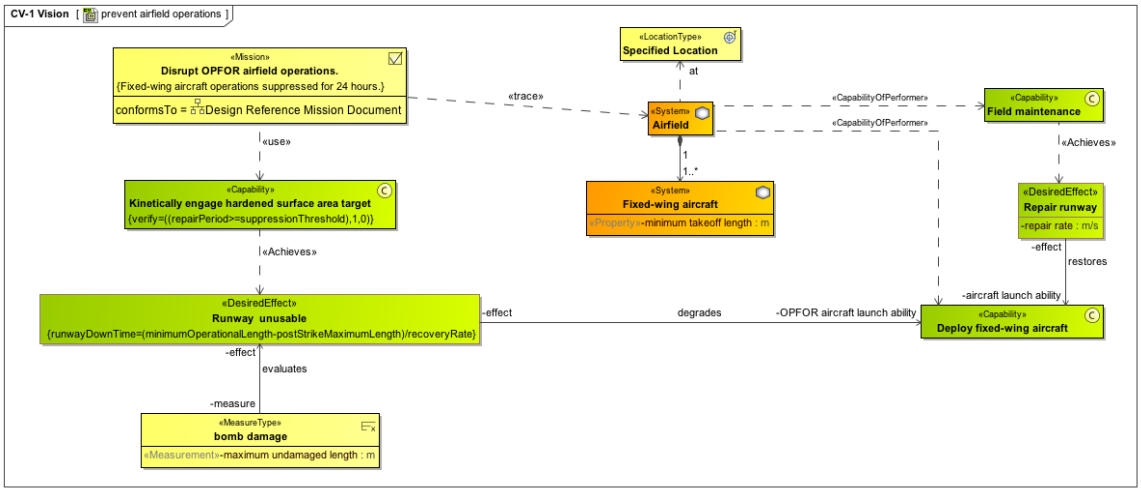


Figure 94. Example capability-effect-impact cycle with rule traceability.

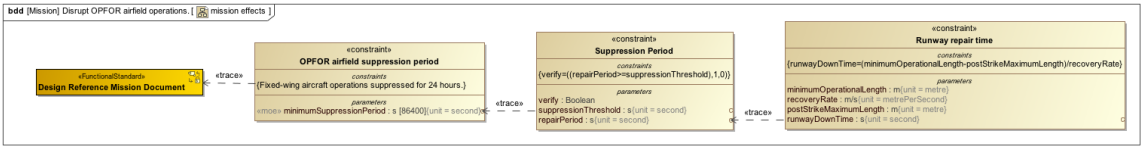


Figure 95. Example mission-related constraint set with rule traceability.

Intentionally blank.

DISTRIBUTION:

1	MS0899	Technical Library, 9536 (electronic copy)


Sandia National Laboratories