# The role of architecture frameworks: Lessons learned from ISO/IEC/IEEE 42010*

## (EXTENDED ABSTRACT)

Rich Hilliard, r.hilliard@computer.org

This paper examines the central role of an architecture framework within the larger endeavour of the architecting of enteprises, systems-of-systems and complex systems. The paper presents lessons learned from the development and use of IEEE 1471:2000, *Recommended Practice for Architectural Description of Software-intensive Systems* and its successor, ISO/IEC/IEEE 42010:2011, *Systems and software engineering — Architecture description*. The lessons learned address areas including: modeling and notations, method and process definition, and tooling, with recommendations for the continued evolution of the NATO Architecture Framework.

## 1 Background

IEEE 1471:2000, *Recommended Practice for Architectural Description of Software-intensive Systems*, was the first formal standard addressing the architecture of *systems*, where "the term *system* encompasses individual applications, systems in the traditional sense, subsystems, systems of systems, product lines, product families, whole enterprises, and other aggregations of interest". [3, §4.1]

ISO adopted the IEEE standard in 2007 as ISO/IEC 42010. Subsequently ISO and IEEE produced a joint revision, published as ISO/IEC/IEEE 42010, *System and software engineering — Architecture description*.

The first edition provided a conceptual model of architecture description and best practices for same. The first edition has been widely used in software, systems and enterprise architecting. The current edition refines the first edition and adds requirements on *Architecture Frameworks* and *Architecture Description Languages*.

This paper will discuss some of the lessons learned from the development of the Standard (from 1995 to the present) and insights gained from widespread use of the Standard during that time, with suggestions for NAF's evolution.[1]

## 2 Role of an architecture framework

The notion of an *architecture framework* has been around since the 1970s.[2] The ISO Standard defines the term in this way:

> **architecture framework**: conventions, principles and practices for the description of architectures established within a specific domain of application and/or community of stakeholders [4, §3.4]

The Standard builds upon this notion to define minimum requirements on any framework, as one means of establishing conformance to the Standard. These requirements are expressed in terms of the conceptual model of Architecture Description introduced in 2000, in the first edition of the Standard, IEEE 1471.

The requirements on architecture frameworks are briefly outlined in table 1. These, and the requirements deriving from them, will be discussed in greater detail in the remainder of the (full) paper.

The fundamental goal of an architecture framework is to codify a common set of architecture practices within a community: for the sake of understandability, commonality and synergy—reducing the need for individual architects to re-invent the wheel; and to promote interoperability. Not merely interoperability of systems but also of architects and other stakeholders within the domain of interest. To achieve this goal, the Standard establishes its minimal requirements on architecture frameworks in terms of their content and presentation.

*Submission to **IST-115 Symposium on Architecture Definition & Evaluation**

[1] For full definitions and requirements, see the Standard itself, or visit the 42010 website http://www.iso-architecture.org/42010/.

[2] For a survey of existing frameworks, see http://www.iso-architecture.org/42010/afs/.

An architecture framework shall include:

**a)** information identifying the architecture framework;

**b)** the identification of one or more concerns (per §5.3);

**c)** the identification of one or more stakeholders having those concerns (per §5.3);

**d)** one or more architecture viewpoints that frame those concerns (per §7);

**e)** any correspondence rules (per §5.7).

An architecture framework should include conditions of applicability.
An architecture framework shall establish its consistency with the provisions of the conceptual model in 4.2.

Table 1: Requirements on Architecture Frameworks [4, §6.1]

# 3 Lessons learned from IEEE 1471 / ISO 42010

The lessons learned may be summarized as falling into these general areas:

**Ontology-based:** Use clear terminology and sound conceptual foundations;

**Interest-driven:** Stakeholders and their concerns drive successful architectures;

**Open and extensible:** Design architecture frameworks as you would design systems—to be open and extensible;

**Framework as foundation:** Not only for the definition of products (including architecture descriptions), but also architecting methods, tools and processes;

**Governance:** Use well-defined conformance mechanisms to promote the items above.

Each area is briefly sketched below; to be elaborated upon in the (full) paper.

## 3.1 Ontology-based: terms and concepts

To be *useful*, an architecture framework should be *useable*. This means, it must be understandable and presented in a form that can be acted upon.

The first ingredient of understandability is a crisp, clear conceptual foundation and terms reflecting those foundations. IEEE 1471 was built upon an explicit conceptual model, or *ontology*, and this is continued in 42010. See figure 1 for a part of the core ontology of the Standard, centered on Architecture Descriptions (ADs) as products.[3]

To maximise understandabililty, and therefore interoperability, it is useful to align one's terms and concepts with existing practices, such as within architecting communities. NAF's use of the terms *architecture view* and *architecture viewpoint* is not consistent with community practice. NAF's term "subview" implies that a model is wholy owned by a view. This is sometimes true, but more generally models are shared between views to reduce redundancy [4, §5.6]. Model is the preferable term. A view consists of one or more models.

Capturing an ontology in an explict (meta) model is helpful, however, although the notion of meta model is widely used in enterprise architecture, few (if any!) of the framework meta models are complete—even by their own definitions![4] In the present case, the NAF meta model (NMM) addresses the *subject matter* of NATO systems (e.g., Actors, Capabilities, Locations, Information Requirements) but has nothing to say about:

> ... the main principle of an architecture description framework, which is to provide *a common way of describing architectures*... [NAF §1.7.6][5]

The NMM does not include architecture constructs such as those found in figure 1 above from the Standard. Architects need both domain constructs and architecture constructs.

[3] There are analogous models for Architecture Frameworks and Architecture Description Languages, because each of these is "composed of" viewpoints; but this paper will use the AD case for exposition.

[4] For discussion, see: http://www.iso-architecture.org/42010/docs/Hilliard-On-Metamodels-in-42010.pdf.

[5] NAF references are to NAF version 3, http://www.nhqc3s.nato.int/ARCHITECTURE/_docs/NAF_v3/ANNEX1.pdf.

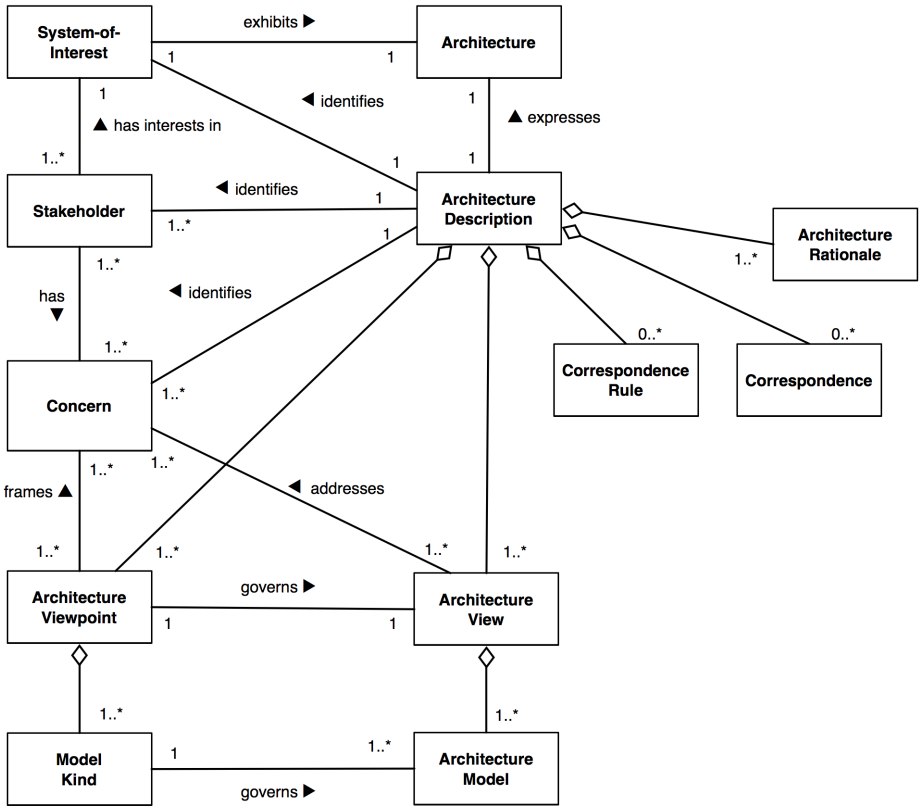


Figure 1: Contents of an architecture description

## 3.2 Interest-Driven: Stakeholders and their Concerns

A key lesson of architecting, dating back to Vitruvius, is that architecture is multi-disciplinary because complex entities—*whether buildings or systems*—have a multitude of interested parties (termed *Stakeholders*), each with specific interests (termed *Concerns*).

One consequence is that different forms of presentation and analyses are relevant for the varying interests (Concerns) of the various audiences (Stakeholders). The identified Concerns then serve as a key index into an architecture description. They guide readers where to look for information on certain topics. They make possible a simple check on coverage of the areas of interest of an architecture's numerous stakeholders: *Are my concerns covered? Is the Architect making progress?*

It is therefore valuable to *(i)* treat Concerns as first-class entities within an architecture framework and *(ii)* key to maintain a linkage of problem and solution spaces: traceability of stakeholders and their concerns with the representations used (viewpoints and models).

NAF (chapter 2) provides a useful analysis of stakeholders, communities of interest, and their requirements, but does not follow through on maintaining actual traceability between Concerns and the descriptive apparatus (viewpoints and model kinds) chosen to frame and consequently address those concerns (in NAF chapters 3 and 4).

## 3.3 Open and extensible

The most important lesson learned from the past 20 years of architecture framework development is this: *you will never finish defining the ontology of a given domain of interest.* For any of the numerous published frameworks, it is trival to identify gaps in their polished meta models. A fixed ontology is a luxury not borne out by previous frameworks. Architecture frameworks have existed since the 1970s—even before John Zachman coined the term "framework for information systems architecture". The ontology of frameworks has evolved as our understanding of enterprises, information systems and software has evolved. The earliest frameworks knew nothing of object-oriented programming and design; later frameworks invariably included objects. Early frameworks did not include notions like service—yet, no self respecting framework today would ignore service oriented architecture. There

is no reason to believe this evolution will not continue. An architecture framework is—at best—a "starter set" of Concerns, Stakeholders, Viewpoints and Model Kinds for Architects within the domain of interest [1].

> All problems in computer science can be solved by another level of indirection. [David John Wheeler]

Instead of trying to define it once-and-for-all, give architects the starter set and a means to extend the framework in a *principled* manner. In NAF v3, extension is permitted but discouraged. The proper "unit of extension" is the Model Kind: being a specific form of representation, with its own mini-meta model to address one or more identified concerns.

### 3.4 Framework as foundation

A robust architecture framework can serve as foundation for all aspects of architecting, beyond its central role in architecture description. This means a framework has implications for methods and processes, and tooling.

There can be no one process associated with an architecture framework for the same reason that architecture is multi-disciplinary: multiple Stakeholders and Concerns. There are many viable processes that will be undertaken within an enterprise. A robust framework should support multiple methods and processes, such as for creation, evaluation and governance [2, 5, 6].

Beyond considerations of any over-arching processes, which will be specific to particular uses and goals, much of "what to do next" and "how to document it" will be determined by viewpoint and model-kind specific techniques. Over-arching processes should not get in the way of these, which should be available off-the-shelf. As such, they are as important to preserve in repositories for architects as previous architecture descriptions (because they are immediately reusable).

Inclusion into a NAF repository of viewpoints, model kinds and ontologies for architects and other stakeholders. One goal of an enterprise should be make available a repository of useful, proven techniques, so that individual architects need not reinvent the wheel. These are as important, if not more so to architects, than a repository of (existing) architectures. These resources become the 'architect's handbook' of practices essential to the maturing of the practice.

### 3.5 Governance

Conformance is a key means to achieving the usability and thus interoperability of architecture descriptions and other products. Currently, NAF defines conformance in terms of meta model consistency. This is not sufficient in itself, for the reasons discussed above. Robust conformance should support: extension, reusable model kinds and methods, as well as end-product architecture descriptions.

All of the above issues will be elaborated upon in the (full) paper.

## References


[1] David Emery and Rich Hilliard. Every architecture description needs a framework: Expressing architecture frameworks using ISO/IEC 42010. In Rick Kazman, Flavio Oquendo, Eltjo Poort, and Judith Stafford, editors, *Proceedings of the 2009 Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture (WICSA/ECSA 2009)*, pages 31–40. IEEE Computer Society Press, 2009.

[2] Christine Hofmeister, Philippe Kruchten, Robert L. Nord, Henk Obbink, Alexander Ran, and Pierre America. A general model of software architecture design derived from five industrial approaches. *The Journal of Systems and Software*, 80(1):106–126, 2007.

[3] *IEEE Std 1471, IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*, October 2000.

[4] *ISO/IEC/IEEE 42010, Systems and software engineering — Architecture description*, December 2011.

[5] Mark W. Maier and Eberhard Rechtin. *The art of systems architecting*. CRC Press, 2nd edition, 2000.

[6] Henk Obbink, Philippe Kruchten, Wojtek Kozaczynski, Rich Hilliard, Alexander Ran, Herman Postema, Dominick Lutz, Rick Kazman, Will Tracz, and Ed Kahane. Report on Software Architecture Review and Assessment (SARA). Technical Report version 1.0, The SARA Working Group, February 2002.