Every Architecture Description Needs a Framework: Expressing Architecture Frameworks Using ISO/IEC 42010

David Emery
DSCI, Inc.
demery@dsci-usa.com

Rich Hilliard
r.hilliard@computer.org

Abstract

The current draft revision of ISO/IEC 42010 (IEEE Std 1471) proposes a formalization of architecture framework *within the ontology of the standard. This paper discusses the origin of the concept, motivates its standardization in ISO/IEC 42010, and argues that a well-defined architecture framework should be a key component of any architecture description. The paper describes the application of the proposed construct to several well-known architecture frameworks.*

Keywords: *architecture description, architecture framework, standardization, architecture viewpoint, model correspondence, model correspondence rule.*

1. Introduction

1.1. History of frameworks and IEEE 1471

Within system and software engineering, the term *architecture framework* dates back to the 1970s. One can find numerous definitions on the World Wide Web; this one is representative:

> An enterprise architecture framework, or architecture framework for short, is a prefabricated structure that you can use to organize your enterprise architecture into complementary views. (http://www.architectureframework.com/faq/)

John Zachman's framework for information systems architecture is often cited as the inspiration for modern architecture frameworks [23]. A Web search will yield numerous examples of architecture frameworks in software, system and enterprise architecture (some of these are discussed below).

Although the influence of these concepts on IEEE Std 1471^{TM}–2000, *Recommended Practice for Architectural Description of Software-intensive Systems* was pervasive, the direct impact was minimal – there is one paragraph [8, Annex B]:

> An organization desiring to produce an architecture framework for a particular domain can do so by specifying a set of viewpoints and making the selection of those viewpoints normative for any AD claiming conformance to the domain-specific architecture framework. It is hoped that existing architecture frameworks – such as the ISO Reference Model for Open Distributed Processing (RM-ODP) [9], the Enterprise Architecture Framework of Zachman [23], and the approach of Bass, Clements, and Kazman [1] can be aligned with the standard in this manner.

A key idea of IEEE 1471 was to introduce *architecture viewpoints* as a mechanism to codify best practices in architecture description. Each viewpoint specifies the architecture concerns to be addressed, the interested stakeholders, and the notations, models and methods to be used to create, interpret and analyze a view resulting from applying that viewpoint.

1.2. Library viewpoints in IEEE 1471

Reflecting then-current practice of the late 1990s, IEEE 1471 introduced *library viewpoints* to allow users to record "reusable" viewpoints, The term was inspired by programming languages such as Ada which maintain a *program library* as a container for modules that were available to be shared and used in new programs. Library viewpoints allow users to specify a set of viewpoint definitions which could be reused from one architecture description (AD) to another to capture approaches such as Kruchten's 4+1 view model [12] and RM-ODP.

The expectation was that sets of well-defined viewpoints could be codified within organizations as corporate knowledge, or as a part of architecture methods or as formal standards. IEEE 1471 included annexes showing how RM-

ODP and IEEE/EIA Std 12207.0–1995, *Information Technology — Software life cycle processes* could be expressed with library viewpoints. Similarly, working papers produced by the IEEE Architecture Working Group cover other approaches that were not formal standards, such as 4+1, Zachman, and the US Department of Defense Architecture Framework (DODAF) [22]. Our own work has also borne this out; we put several library viewpoints "on the shelf" to be considered each time we approached a new project [5]. The software architecture literature provides several examples of viewpoint sets: Kruchten's 4+1 view model, Siemens' "four views" [7], Garland and Anthony [6], Rozanski and Woods' viewpoints and perspectives [18] and SEI's *Views and Beyond* [3]).

Since the publication of IEEE 1471, system and software architecture practice has continued to evolve. Enterprise architecture, building on the Zachman framework for information systems architecture, has also emerged, as exemplified by recent enterprise architecture frameworks such as the DODAF [22], MODAF [21] and TOGAF [19], and by standards such as ISO RM-ODP and ISO 15707, based on the *Generalized Enterprise Reference Architecture* (GERA) [10].

Today many, if not most, architects must work within one or more architecture frameworks, as directed by their organization, their clients, the method they employ, or due to other forces. Since an original design objective of IEEE 1471 was to establish a common frame of reference, or ontology, for architecture description [4], it seems worthwhile to extend that ontology to address architecture frameworks. The next section discusses extensions to IEEE 1471 to allow the expression of architecture frameworks like those listed above.

2. Frameworks in ISO/IEC 42010

In March 2006, IEEE 1471 was adopted by ISO as an international standard following a fast-track ballot, with an agreement to conduct a joint ISO and IEEE revision [11] . The joint revision, being conducted by ISO/IEC JTC1/SC7 Working Group 42,[1] will be designated ISO/IEC 42010 (and IEEE Std 42010) with the new title, *Systems and Software Engineering — Architecture Description*.

The current working draft, ISO/IEC WD4 42010, provides this definition:

> **architecture framework** conventions and common practices for architecture description established within a specific domain or stakeholder community

[1]*42* because Architecture is the answer to Life, the Universe and Everything.

Building upon the requirements for specifying architecture descriptions in IEEE 1471, the draft standard specifies requirements for architecture frameworks. These may be summarized as follows: a framework must identify a set of interested stakeholders, a set of their architecture concerns, a set of viewpoints framing those concerns and any correspondences to be enforced between views resulting from applying those viewpoints.

Figure 1 shows how the proposed concept of framework would fit into the IEEE 1471 ontology.

2.1. Model correspondences

A consequence of the use of multiple views in architecture descriptions is the need to express dependencies – in particular, consistency – between those views.

In the 2000 edition of IEEE 1471, the only consistency requirement on an AD was to record any known inconsistencies between its views. At the time of standardization, there was no well-established practice to be codified for expressing consistency or other dependencies between views.

ISO/IEC WD4 42010 introduces a mechanism called *model correspondences* for expressing relations between two or more architecture models. Since architecture views are composed of architecture models in the IEEE 1471 ontology (see Figure 1), model correspondences can be used to relate views to express consistency, traceability, refinement or other dependencies.

The draft standard does not specify a format for model correspondences. They could be captured as relations, tables, or graphically. Traditionally, tables are used to capture traceability. Kruchten graphically depicts relations between a Logical view and a Process view by associating elements with arrows [12, figure 9]. Kruchten's example could be rendered in a tabular form, or as a mathematical relation, as shown in (1).

$$\begin{aligned} LogicalToProcess = \\ (\mathsf{flight}, FlightAgent_1), ...(\mathsf{flight}, FlightAgent_n), \\ (\mathsf{profile}, FlightAgent_1), ...(\mathsf{profile}, FlightAgent_n), \\ (\mathsf{clearance}, FlightAgent_1), ...(\mathsf{clearance}, FlightAgent_n), \\ (\mathsf{location}, AeronauticalServer), \\ (\mathsf{airspace}, AeronauticalServer) \end{aligned} \tag{1}$$

where flight, $\mathsf{profile}$, $\mathsf{clearance}$, $\mathsf{location}$ and $\mathsf{airspace}$ are *active objects* in the Logical view; and $FlightAgent_1$, ... $FlightAgent_n$, and $AeronauticalServer$ are *Agents* in the Process view.

When a view contains architecture models of differing levels of abstraction, one usage for correspondences is to relate such models by refinement. For example, the architect might want to capture that:

$$\textsf{model } ST_2 \textbf{ refines } \textsf{model } B_3 \tag{2}$$

where ST_2 is a state transition model and B_3 is a behavior model. Notice that this model correspondence does not even refer to elements of either model – it simply associates two models.

Often, an architect will seek to impose a constraint between types of models and then demonstrate that constraint is satisfied by the architecture. ISO/IEC WD4 42010 introduces *model correspondence rules* to express such constraints on two or more architecture models. The following are examples of model correspondence rules; defined with reference to various viewpoints (not defined herein):

1. Every behavior model B_k must be refined by a state transition model ST_q.
2. Each active class in the Logical view must be mapped to one or more agent tasks in the Process view (taken from 4+1).
3. The version identifiers of each architecture view must match.
4. Each element in the Logical view should be classified as a *principal*, *resource* or *action* in the Trust view.
5. Within a Trust view, each threat in its threat model should be linked to at least one measure or mechanism in the security model.

Example 1 expresses a refinement relationship between two types of models. Example 2 formalizes the rule for the model correspondence in (1). In the draft standard, a model correspondence rule *holds* when a model correspondence satisfies it. Example 3 demonstrates an n-ary relationship, which refers to one attribute of the views an architecture description, but makes no reference to architecture elements in those views. Example 4 uses the ontology of one view (a Trust view) to classify the elements of another view, expressing a form of traceability. Example 5 states a traceability-like constraint between threats in one model and countermeasures in a second model.

A model correspondence rule can be expressed within an AD or as a part of specifying an architecture framework. Depending on where it is captured, the rule applies either to the individual AD or as a general constraint on any AD applying that architecture framework.

Correspondences have been a part of several approaches, including 4+1 (as noted above) and RM-ODP. The choice of the term *correspondence* in ISO/IEC WD4 42010 was motivated by these precedents. Model correspondences in ISO/IEC WD4 42010 are compatible with viewpoint correspondences in RM-ODP, in the following sense: all RM-ODP viewpoint correspondences can be expressed as ISO/IEC WD4 42010 model correspondences (but not vice versa). The key differences are:

- viewpoint corresondences are binary, whereas model correspondences are n-ary;
- viewpoint correspondences relate elements within RM-ODP viewpoint specifications, whereas model correspondences can also relate models themselves or their attributes, without reference to any elements.
- the choice of the term "model correspondence" rather than "view correspondence" reflects that although RM-ODP views are *homogeneous*: a single viewpoint language is used per viewpoint specification, ISO/IEC 42010 allows *heterogeneous* views: each view is composed from one or more architecture models, where each model may utilize a different modelling language. It is critical to be able to express correspondences between models in different modelling languages, not just between views.

Linington provides a very useful overview of viewpoint correspondences in RM-ODP [13]. Boucké *et al.* survey recent mechanisms for expressing relations on architecture views and models and put forth a taxonomy of these mechanisms [2].

3. Specifying frameworks

This section describes the proposal for architecture frameworks in ISO/IEC WD4 42010.

3.1. Anatomy of a framework

An architecture framework is determined by:

- a set of architecture-related concerns;
- a set of stakeholders holding those concerns;
- a set of architecture viewpoints which *frame* (i.e., cover) those concerns; and
- a set of model correspondence rules.

Figure 1 shows the ingredients of architecture frameworks and the relations between a framework and an architecture description to which that framework is applied.

Under IEEE 1471, an architecture description addresses *known* concerns of *known* stakeholders for the system of interest. Architecture frameworks introduce a level of indirection to this: the stakeholders and concerns for a system's architecture may not be known when the framework is defined. This situation also arises with library viewpoints and was recognized in the 2000 edition of IEEE 1471. The definitions of *stakeholder* and *concern* in the standard take this into account: stakeholders may be individuals, teams, organizations or classes (of individuals, teams or organizations); and concerns may be fine-grained or very broad in scope. Architecture methods and architecture frameworks can provide additional guidance in this area, for example, to define typical stakeholders or taxonomies of concerns.

Common practice indicates that framework developers often have in mind known or established stakeholders within the domain of the framework. For example, in enterprise architecture the chief information officer is an established stakeholder for virtually any enterprise. Framework developers may postulate likely stakeholders for specific systems or classes of systems within their domain. Accounting systems, for example, are likely to have chief financial officers and auditors as stakeholders of any architecture. When a system is intended for a specific business unit, that unit's management (operations, financial, executive) will be stakeholders for architectures of interest within that unit.

The stakeholders motivate the set of concerns which the architecture framework will focus upon. Identifying the architecture-related concerns determines the choice of viewpoints to be included. The principal content of an architecture framework is the set of viewpoints. Each viewpoint must meet the minimal requirements of ISO 42010:2007. The spirit of these requirements is that for each view, or *map* of the architecture, the viewpoint is the *legend* for that map. A map's legend explains the symbolic conventions used in the map and establishes the basis for interpreting that map. Each viewpoint establishes the notations, models, techniques and methods to be used in architecture descriptions resulting from applying the framework.

An architecture framework may include model correspondence rules to interrelate required views and models, as discussed in 2.1.

Beyond the minimal requirements, framework developers are free to add additional ingredients to a framework. These could provide additional guidelines and requirements to specify not only models and tools, but also architectural principles and rules, patterns and styles, delivery formats, expectations for rationale and decision capture, possibly process definitions for constructing views and using the results. For example, the DODAF AV-2 requirement for an integrated data dictionary applies across the architecture description. This could be expressed as a required viewpoint for all architecture descriptions that conform to DODAF, but it often captures additional terms used within the architecture description that do not necessarily relate to elements of the system of interest as presented in the AD.

An architecture framework conforming to the standard is expected to build upon the core ontology in the standard, optionally documenting its extended ontology with an explicit metamodel.

3.2. The realization: every AD has a framework

Consider an architecture description conforming to ISO/IEC 42010. It identifies a set of viewpoints which have been selected either to be reused from existing library viewpoints or developed specifically for this AD and a set of correspondence rules to be asserted between the models of those viewpoints. The AD also contains a set of views (one per viewpoint) and a set of correspondences that must satisfy the correspondence rules.

Although ISO/IEC WD4 42010 does not specify any process, it implies that the identification of viewpoints should precede the creation of the associated views, and that correspondence rules be defined before the correspondences.[2] Thus an AD should use viewpoints and correspondence rules that have been chosen to produce an effective AD. The idea the viewpoints and correspondence rules are chosen and integrated to fill specific needs for this AD is really the same process as selecting viewpoints and correspondence rules to form an architecture framework – i.e. a "good" set of viewpoints and correspondence rules for a given AD *is* an architecture framework.

Therefore, architecture descriptions already include some framework-like content. *Why not require in the standard that every architecture description include exactly one framework?* This proposal is under consideration by Working Group 42.

There are some real advantages to doing this. The framework becomes a "container" for stakeholders, concerns, viewpoints and correspondence rules. Thus an AD would consist of:

- its architecture framework (i.e., stakeholders, concerns, viewpoints and correspondence rules);
- its set of views and correspondences (satisfying the architecture framework);
- its rationale and required administrative information.

There are also some disadvantages. The proposal would impose a specific methodological choice, namely to develop an architecture framework along with each architecture description. This could easily result in a proliferation of "too

[2]In practice, this is usually an iterative process: an initial set of viewpoints and correspondence rules are defined, then some conforming views and correspondences are produced, as commonly happens in design [17]. With increased understanding of the architecture and its rendering in the AD, the viewpoints and correspondence rules may be adjusted to better define what the views and correspondences need to express.

many" frameworks. There are also potential issues for users of multiple frameworks. For example, an AD which was required to conform to the DoDAF but which added additional viewpoints to frame concerns not expressible within DODAF, such as information security, would have to name its augmented framework and argue that this was a legitimate extension of the DODAF. This could cause problems with the reviewer community for such an AD.

3.3. Applying the realization

With the introduction of frameworks, the draft standard defines three kinds of conformance:

1. conformance of an architecture description (as in IEEE 1471);
2. conformance of an architecture framework (as described in 3.1);
3. conformance of an architecture description to an architecture framework.

Making architecture frameworks a point of conformance opens new possibilities for interoperability and knowledge sharing in the architecture community. The first step in this direction was made with viewpoints: a viewpoint is a reusable form of knowledge for modeling architectures with respect to a specific set of architecture concerns. There are a number of viewpoints and viewpoint sets in the literature today (see 1.2), giving the architect a starting point when confronting a new system when specific concerns are raised. Having a standardized definition of architecture framework and requirements on specifying conforming frameworks allows sharing of knowledge between communities that previously were speaking different languages. The common basis provided by the ISO/IEC 42010 ontology makes these discussions possible.

Using the architecture framework construct, we can look at how architecture frameworks are used in practice.

4. Using the architecture framework construct

In this section, we demonstrate how ISO/IEC WD4 42010's proposed architecture framework construct can be used to understand, clarify and diagnose some current uses of architecture frameworks.

Architecture reviews. A key tenet of the ontology of IEEE 1471 is that the architecture description of an architecture is distinct from the architecture itself; the AD being an artifact and the architecture being a conceptualization. Experience evaluating architectures and architecture descriptions indicates that a "good" AD is a prerequisite to a review of the architecture it describes. Reviewing an AD is a process in its own right, separate from reviewing the architecture it describes. An approach for AD evaluation is provided by [15]. Evaluation practices can also make use of this separation, factoring evaluation into:

1. evaluate the architecture framework;
2. evaluate the views and other content against the framework;
3. evaluate the architecture with the knowledge gained from review of the AD.

Adopting the approach that each AD should result from an explicitly captured architecture framework enables investment in tools and techniques for evaluating an AD against its architecture framework and assessing the AD's conformance to same. Thus an effective architecting method could consist of the organization's baseline architecture framework, modeling tools as required by the viewpoint descriptions and correspondence rules specified in the framework, and review and evaluation techniques. Architects, engineers and even non-technical stakeholders (e.g. acquisition personnel, user or customer representatives) can be trained in the tools and modeling techniques used by the organization's baseline framework.

Zachman. Zachman's framework for information systems architecture is usually depicted with a matrix of 6 rows and 5 columns [23]. The columns, labelled Data/What, Function/How, Network/Where, People/Who, Time/When and Motivation/Why, can be understood as architecture concerns with respect to the enterprise of interest in IEEE 1471 terms. The rows, labelled Planner, Owner, Designer, Builder, Programmer, and User, can be understood as stakeholders. Each row is called a *stakeholder view* or *role perspective* by Zachman and is intended to be *a complete view of the system* (as is also required in IEEE 1471). Each cell of the matrix depicts a perspective, or way of viewing the subject. Each cell can be understood as a model type, in terms of ISO/IEC WD4 42010. Each row, in effect, defines a viewpoint comprised of 6 models. The implied life cycle-like ordering of the rows from Planner to User can be captured via model correspondence rules (see example 2). The idiom of layers, or levels, of refinement is common to a number of architecture frameworks, as noted above.

DODAF. DODAF defines three "views" – Operational, Systems and Technical – or *viewpoints* according to ISO/IEC 42010. DODAF does not distinguish views and viewpoints, which significantly complicates their exposition. As viewpoints, the DODAF's definitions are incomplete: stakeholders and concerns are not identified. This makes it difficult for DODAF users to understand *why* they are modeling, and *when* they are done. DODAF defines 29 architecture products in detail; each related to a view(point). These architecture products correspond to architecture models in ISO 42010 terms. The DODAF (version 1.5 and earlier) also has some products that can be considered as correspondences between the views. For example, the System View SV-6 work product, *Systems/Services Data Exchange Matrix* establishes a correspondence between the systems and services defined in other SV products, and the information exchanges defined in the OV-3 work product.

GERAM. The *Generalized Enterprise Reference Architecture and Methodology* found in ISO 15704:2000 [10] is currently being revised using the terms and definitions of ISO/IEC 42010:2007. The latest draft version of ISO CD 15704, *Reference-base for enterprise architecture and models*, is an architecture framework (in the sense of this paper) for enterprise reference architectures. It identifies areas of concern to stakeholders in the domain of industrial automation. It specifies modeling properties for use in that domain and several specific viewpoints to be modeled that produce architectural and operational views for a manufacturing enterprise. It makes use of correspondence relationships, principally in the context of the enterprise life cycle, model genericity, and modeling viewpoint, to form a cohesive framework.

Kruchten's 4+1. Using the ontology from ISO/IEC WD4 42010, we can say the 4+1 architecture defines 5 viewpoints: Logical, Development, Process, Physical and Scenarios. As discussed above, Kruchten addresses "correspondences between the views" such as "logical to process", "logical to development" and "process to physical". In practice, a successful 4+1 description can be measured in part by how complete these associations are (e.g. *Is every software element allocated to at least one process running on at least one computer?*) With ISO/IEC WD4 42010, these criteria of the 4+1 approach can be captured with model correspondence rules.

Essential Project. The Essential Project (http://www.enterprise-architecture.org/) is developing free (GNU GPL) tools for enterprise architects. The toolset arrives with a built-in enterprise architecture framework, consisting of four "layers": Business, Information, Application and Technology. Within each layer are three views labelled Conceptual, Logical and Physical. Although the term *layers* seems to imply ordering, none is stated in the Essential documentation. However the three views within each layer again suggest levels of refinement which can be captured with model correspondence rules. Each layer also has a metamodel with additional relations expressed between elements within the views which are probably *models* in the ISO/IEC 42010 sense. If one gathered up the Conceptual views (i.e., models) of each of the Essential Project's layers (i.e. views), the content would be very close to the RM-ODP Enterprise viewpoint – if these two frameworks had both been specified in a uniform manner, using the proposed framework construct with stakeholders and concerns identified, this commonality might have been easier to discern and utilize.

Mixing Frameworks. Organizations often mandate the use of a particular architecture framework (e.g. DoDAF), while its architects may feel more comfortable working within another framework (e.g. 4+1). Sometimes an AD is required – due to contractual obligations or organizational mandates – to conform to more than one existing framework (e.g. *The Contractor shall deliver DODAF products.* and *The Contractor shall deliver an AD using the 4+1 view model*). In such cases, an early decision for the architect is how to accommodate and use multiple preexisting frameworks within an approach that works for the project.

A benefit of the proposal described here is that it facilitates merging of frameworks in a principled manner by aligning each framework to the standard model and using stakeholders and concerns to structure the solution. To continue the above example: an AD could select 8 viewpoints: 3 from DODAF (Operational, Systems and Technical) and 5 from 4+1 (Logical, Development, Process, Physical and Scenario). In addition, the relationship between elements in the DODAF viewpoints and the 4+1 viewpoints could be specified as model correspondence rules. E.g.:

> Each *node* in the DODAF SV-3 product (*Systems-Systems / Services-Systems / Services-Services Matrix*) must be 1-1 with a *computational node* in the 4+1 Physical view.

Applying multiple architecture frameworks to form the basis for an AD requires substantial analysis and specification of how the concepts in each framework relate to the others. It may be helpful if each framework and its viewpoints have an explicit metamodel, as discussed (but not required) in the draft standard. It may be helpful to capture the results of such an analysis to make it reusable as an architecture framework itself, i.e. *Here is how we produce an AD that meets both DODAF and 4+1.* Alignment with the ISO/IEC WD4 42010 ontology and application of the framework construct facilitates the merging of pre-existing architecture frameworks.

5. Conclusion

The concept of *library viewpoint* in IEEE 1471 reflected the state of the art in the early days of software and enterprise architecture. By not constraining an architecture description with a fixed set of viewpoints, but recognizing that reuse of viewpoints was desirable, IEEE 1471 provided a strong basis for the ISO/IEC 42010 revision to standardize a notion of *architecture framework*. This was accomplished as a simple extension to the existing IEEE 1471 ontology through the addition of *model correspondence rules* to capture explicit relationships between models defined by viewpoints within an architecture description. Model correspondence can also be used to express relations among multiple architecture descriptions, to express relations within a product family or system of systems, or even across architecture frameworks.

Applying the proposed ISO/IEC WD4 42010 ontology to frameworks in current use (e.g. 4+1, GERAM, RM-ODP, Zachman, DODAF, MODAF) has shown its applicability to capture and relate these architecture frameworks to each other, and more importantly, to provide a clear definition of conformance of an architecture description to its framework.

Insofar as any framework exists to collect and relate viewpoints to enable the architect to construct useful, consistent architecture descriptions, this paper demonstrates that every architecture description should have a clearly defined, explicitly stated architecture framework. A framework should capture an organization's best practices and consistently implement methodological or technical requirements, including conformance with other architecture frameworks. Once an organization has defined a baseline framework for use within its domain, stakeholder and concern sets, and relevant methods, the organization can more easily capitalize investments in evaluation [15], training and automated tools.

5.1. Future Work

Currently ISO/IEC 42010 provides little normative guidance for architecture rationale or decision capture. Developers of architecture frameworks can consider means to capture rationale or design decisions, based on the specialization provided by frameworks for a given domain or method. Similarly, frameworks may consider specifying patterns or related design principles that are either recommended or even mandated for use in architecture descriptions.

Frameworks can and probably should be at the core of architecture methods and processes. There is discussion within ISO/IEC JTC1/SC7 on the relationship between the ISO/IEC 42010 and ISO's system and software engineering process model standards (ISO/IEC 15288 and ISO/IEC 12207, respectively). Architecture framework requirements can be used to evaluate processes (such as "architecture maturity models").

Capturing and maintaining consistency across models in an architecture description is, in general, a research problem (with respect to what to relate and how to maintain those relations) as well as a toolsmithing problem (for implementing these relations). Frameworks can be used to constrain architecture practices to support automated consistency checking, but this should not prevent an architecture description from clearly addressing stakeholder concerns.

Another future area of application of the architecture framework construct may be somewhat counterintuitive: standardization of architecture description languages (ADLs). There is a long history of ADLs in software architecture [14]; more recently the community has seen the development of languages for system and enterprise architecting (for example, SysML [16] and ArchiMate [20], respectively). Such languages tend to address multiple stakeholders, multiple concerns, and often include several types of models and viewpoints.

The scope of ArchiMate is defined in this way:

> An architecture is typically developed because key people have concerns that need to be addressed by the business and IT systems within the organization. Such people are commonly referred to as the "stakeholders" in the system. The role of the architect is to address these concerns, by identifying and refining the requirements that the stakeholders have, developing views of the architecture that show how the concerns and the requirements are going to be addressed, and by showing the trade-offs that are going to be made in reconciling the potentially conflicting concerns of different stakeholders. Without the architecture, it is unlikely that all the concerns and requirements will be considered and met. [20]

This certainly suggests ArchiMate is close to an architecture framework in all but name. SysML is of a different character. It is described as a "general-purpose modeling language for systems engineering". The obvious question an architect might ask is, *Is SysML appropriate for architecting?* The SysML specification offers this insight:

> SysML has extended the concept of view and viewpoint from UML to be consistent with the IEEE 1471 standard. In particular, a viewpoint is a specification of rules for constructing a view to address a set of stakeholder concerns, and the view is intended to represent the system from this viewpoint. This enables stakeholders to specify

aspects of the system model that are important to them from their viewpoint, and then represent those aspects of the system in a specific view. Typical examples may include an operational, manufacturing, or security view/viewpoint. [16]

So in both cases, it is legitimate for an architect to ask: *What stakeholders and concerns does the language intend to address, and what model types and viewpoints does the language provide for addressing those concerns?* This is, of course, the essence of an architecture framework, as argued above. Capturing this information within the specification of a framework provides the necessary links from the concepts behind the languages and tools to their application within a well-defined approach.

5.2. Contributing to the Revision

ISO/IEC 42010 is entering its first ballot this year (2009), via both ISO and IEEE balloting practices. Informal review of previous drafts has shown acceptance of the approach to specifying architecture frameworks described here, but of course new reviewers may provide new perspectives that impact the draft. The current schedule calls for completing the final draft in 2010 with likely adoption in 2011.

Existing frameworks should consider the value of meeting the conformance requirements in ISO/IEC 42010 and provide feedback on the suitability of the current proposal. In some cases, this will require changes to existing terminology (e.g. the items called "viewpoints" in RM-ODP would be "views" under the ISO/IEC 42010 framework). This introduces potential confusion in the user and tool community for each framework. However, as the complexity of architecture practices as reflected by "mandated" frameworks grows, the need to relate one framework to another through a standard ontology and vocabulary is also likely to grow.

Interested parties can participate in the ISO revision through their national member bodies. Individuals may also participate through the IEEE Architecture Working Group (http://www.iso-architecture.org/ieee-1471/).

Acknowledgement. Thank you to Richard Martin for valuable comments on an earlier draft of this paper, and to the anonymous reviewers for their feedback.

References

[1] L. Bass, P. Clements, and R. Kazman. *Software Architecture in Practice*. Addison Wesley, 1998.

[2] N. Boucké, D. Weyns, R. Hilliard, T. Holvoet, and A. Helleboogh. Characterizing relations between views. In R. Morrison, D. Balasubramaniam, and K. Falkner, editors, *Software Architecture: Second International Conference, ECSA 2008 Paphos, Cyprus, September 29-October 1, 2008 Proceedings*, number 5292 in Lecture Notes in Computer Science, pages 66–81, 2008.

[3] P. C. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, R. Nord, and J. Stafford. *Documenting Software Architectures: views and beyond*. Addison Wesley, 2003.

[4] W. J. Ellis, R. Hilliard, P. T. Poon, D. Rayford, T. F. Saunders, B. Sherlund, and R. L. Wade. Toward a recommended practice for architectural description. In *Proceedings of 2nd IEEE International Conference on Engineering of Complex Computer Systems, Montreal, Quebec, Canada, October 21–25, 1996*, 1996.

[5] D. E. Emery, R. Hilliard, and T. B. Rice. Experiences applying a practical architectural method. In A. Strohmeier, editor, *Reliable Software Technologies – Ada-Europe '96*, number 1088 in Lecture Notes in Computer Science. Springer, 1996.

[6] J. Garland and R. Anthony. *Large Scale Software Architecture: A Practical Guide Using UML*. John Wiley and Sons, 2002.

[7] C. Hofmeister, R. L. Nord, and D. Soni. *Applied Software Architecture*. Addison-Wesley, 2000.

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

[9] ISO. *ISO/IEC 10746-1 Information technology – Open Distributed Processing – Reference Model: Overview*, December 1998.

[10] ISO. *ISO 15704 Industrial automation systems — Requirements for enterprise-reference architectures and methodologies*, 2000.

[11] ISO. *ISO/IEC 42010 Systems and software engineering — Recommended practice for architectural description of software-intensive systems*, July 2007.

[12] P. B. Kruchten. The "4+1" view model of architecture. *IEEE Software*, 28(11):42–50, November 1995.

[13] P. Linington. Black cats and coloured birds – what do viewpoint correspondences do? In *Proceedings of the 4th International Workshop on ODP and Enterprise Computing (WODPEC 2007)*. IEEE Digital Library, October 2007.

[14] N. Medvidovic and R. Taylor. A classification and comparison framework for software architecture description languages. *IEEE Transactions on Software Engineering*, 26(1):70–93, 2000.

[15] R. Nord, P. Clements, D. Emery, and R. Hilliard. A structured approach for reviewing architecture documentation. Technical Report CMU/SEI-2009-TN-XXX, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 2009.

[16] *OMG Systems Modeling Language (OMG SysML^{TM}) version 1.1*, formal/2008-11-01 edition, November 2008.

[17] D. Parnas and P. Clements. A rational design process: How and why to fake it. *IEEE Trans. Softw. Eng.*, 12(2):251–257, 1986.

[18] N. Rozanski and E. Woods. *Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perpectives*. Addison Wesley, 2005.

[19] The Open Group Architectural Framework (TOGAF), 2007.

[20] *ArchiMate 1.0 Specification (draft)*, November 2008.
[21] MOD Architecture Framework, Version 1.2, 2008.
[22] US Department of Defense Architecture Framework, Version 1.5, 2007.
[23] J. A. Zachman. A framework for information systems architecture. *IBM Systems Journal*, 26(3):276–292, 1987.

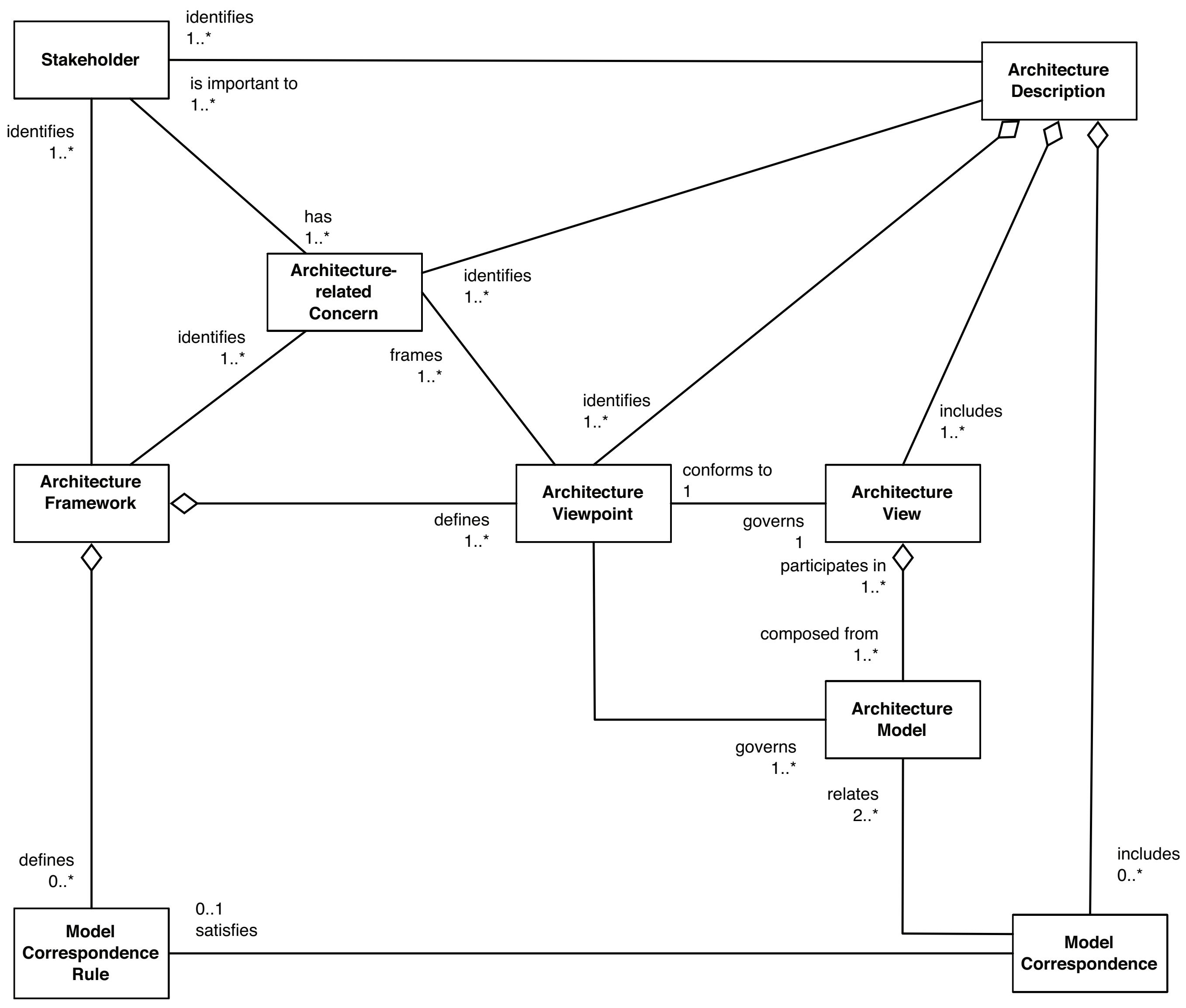

Figure 1. Architecture frameworks in ISO/IEC WD4 42010