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 42010:2007 introduces a formalization of architecture framework within the ontology underlying 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 ISO/IEC 42010 construct to several well-known architecture frameworks. Keywords: architecture description, architecture frameworks, standardization, viewpoint, model correspondence, model correspondence rule.

1. Introduction

1.1. History of frameworks and IEEE 1471

Within systems and software engineering, the term architecture framework dates back to at least the 1970s. One can find numerous definitions of the term 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. [1]

John Zachman’s framework for information systems architecture is often cited as inspiration for modern architecture frameworks [2]. A Web search will yield numerous examples of architecture frameworks in software, systems and enterprise architecture (some of which will be discussed below).

Although the influence was pervasive, the direct impact of architecture frameworks on IEEE Std 1471™–2000, Recommended Practice for Architectural Description of Software-Intensive Systems, (IEEE 1471) was minimal – there is one paragraph of discussion [9] 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) [10], the Enterprise Architecture Framework of Zachman [2], and the approach of Bass, Clements, and Kazman [2] can be aligned with the standard in this manner.

A key idea of IEEE 1471 was the introduction of architecture viewpoints to codify best practices for the creation and use of architecture views within an architecture description (AD). A viewpoint specifies the architecture concerns to be dealt with, the stakeholders addressed, and the languages, models, methods and techniques used to create, interpret and analyze any view (resulting from applying that viewpoint).

*Submission to WICSA/ECSA 2009.
1.2. Library viewpoints in IEEE 1471

Reflecting current practice in the mid-1990s to define sets of “reusable” viewpoints, IEEE 1471 included the notion of a library viewpoint which would allow users to capture approaches such as Kruchten’s 4+1 view model[13] and RM-ODP (which actually calls itself an architecture framework), by providing a set of viewpoint definitions that could be reused from one AD to another. The notion and terminology was influenced by programming languages such as Ada, that maintain a program library as a container for modules that could be shared and reused by a new program.

The expectation at the time was that sets of well-defined viewpoints would be codified and reused, either as organizational defaults or even via formal standardization. Our own work in architecture has borne this out; we collected a set of library viewpoints that were examined as candidates each time we approached a new architecture project[6]. The software architecture literature also evidences many viewpoints and viewpoint sets (e.g., Kruchten’s 4+1 view model[13], Siemens’ “four views”[8], Garland and Anthony[7], Rozanski and Woods’ viewpoints and perspectives[19] and SET’s Views and Beyond[4]).

Over the last 15 years, two primary disciplines of architecture practice have evolved. One is software architecture, exemplified by the work cited above. The other is enterprise architecture, building on the Zachman framework for information systems architecture, and exemplified by recent frameworks such as the DODAF[23], MODAF[22] and TOGAF[20]. The term “framework” is often explicitly used in these enterprise architecture practices. ISO RM-ODP and ISO 15707, based on the Generalized Enterprise Reference Architecture and Methodology (GERAM)[11], also constitute frameworks in the sense discussed here.

IEEE 1471 contained informative annexes that explained RM-ODP and IEEE/Std 12207.0–1995, Information Technology — Software life cycle processes, in terms of library viewpoints. Similarly, working papers produced by the IEEE Architecture Working Group cover other approaches that were not de jure standards, such as 4+1, Zachman, and the US Department of Defense Architecture Framework (DODAF)[23].

Today many, if not most, software, systems and enterprise architects operate within one or more architecture frameworks, as directed by their organization, their clients, the method they employ, or due to other forces. Since one of the original design objectives of IEEE 1471 was to establish a common frame of reference, or ontology, for architecture descriptions[5], it seems a worthwhile goal 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[12]. The joint revision, being conducted by ISO/IEC JTC1/SC7 Working Group 42[1] will be ISO/IEC 42010 (and IEEE Std 42010) with the new title, Systems and Software Engineering—Architecture Description. One area deemed to be sufficiently mature to standardize is the notion of architecture framework. The current draft, ISO/IEC WD 42 10 (or ISO 42010:201X), defines it as follows:

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

Building upon the requirements for specifying architecture descriptions in IEEE 1471, the draft standard also specifies requirements for architecture frameworks which may be summarized as follows: a framework must identify: a set of stakeholders, a set of their architecture concerns, and a set of viewpoints framing those concerns. A framework, according to the draft standard, must also identify any correspondences to be enforced between views resulting from applying those viewpoints. (Model correspondences are discussed in[2.1]) Figure[1] shows how this new concept would fit into the IEEE 1471 ontology.

2.1. Model correspondences

A consequence of using 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 requirement on consistency was that an AD must record any known inconsistencies between its views. At that time, there was no well-established practice to be codified for expressing consistency or other dependencies.

142 because Architecture is the answer to Life, the Universe and Everything.
ISO 42010:201X introduces a general mechanism, *model correspondences*, for expressing relations between two or more architecture models. Since views are composed of models in the IEEE 1471 standard, correspondences can be used to relate views to express consistency, traceability, refinement or other dependencies.

Model correspondences can 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 [13] figure 9. That example could be rendered in a tabular form, or as a mathematical relation, as shown in (1).

\[
\text{LogicalToProcess} = \\
\{(\text{flight}, \text{FlightAgent}_1), \ldots (\text{flight}, \text{FlightAgent}_n), \}
\{(\text{profile}, \text{FlightAgent}_1), \ldots (\text{profile}, \text{FlightAgent}_n), \}
\{(\text{clearance}, \text{FlightAgent}_1), \ldots (\text{clearance}, \text{FlightAgent}_n), \}
\{(\text{location}, \text{AeronauticalServer}), \}
\{(\text{airspace}, \text{AeronauticalServer}) \}
\]

where \text{flight}, \text{profile}, \text{clearance}, \text{location} and \text{airspace} are active objects in the Logical view; and \text{FlightAgent}_1, ... \text{FlightAgent}_n, and \text{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:

\[
\text{model } D_2 \text{ refines model } R_3
\]

where \(D_2\) is a design model and \(R_3\) is a requirements model. Notice that this model correspondence does not even refer to elements of either model – it simply associates two models.

In most cases, the architect will want to impose a constraint between models and then demonstrate that constraint is satisfied. For this, ISO 42010:201X introduces *model correspondence rules* as a constraint on two or more architecture models which is enforced on a model correspondence. Rules allow an architect to express expectations within a framework or within an individual architecture description, about relationships between models (and therefore views).

The following are examples of model correspondence rules; defined with reference to various viewpoints (not defined herein):

1. Every requirements model \(R_k\) must be refined by a design model \(D_q\).
2. Each active class in the Logical view must be mapped to one or more agent tasks in the Process view (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. Each threat in threat model should be linked to at least one measure or mechanism in security model.

Example 1 expresses a refinement relationship between two types of models. Example 2 formalizes the rule for the \(\text{LogicalToProcess}\) correspondence given above. Example 3 demonstrates an n-ary relationship, which refers to one attribute (version identifier) of an architecture description’s views, but no architecture elements. Example 4 uses the ontology of one view (a Trust view) to classify the elements of another view. Example 5 states a traceability-like constraint between “problems” (threats) in one model and “solutions” (countermeasures) in a second model.

A model correspondence rule could be expressed either as part of an individual AD or as a part of specifying an architecture framework. A rule applies either to the individual AD or as a general rule on any AD employing that framework.

Correspondences have been part of several approaches, including 4+1 (as noted above) and RM-ODP. As Kruchten writes in his 1995 paper:

The various views are not fully orthogonal or independent. Elements of one view are connected to elements in other views, following certain design rules and heuristics [13].
The choice of the term *correspondence* in ISO 42010:201X was motivated by Kruchten’s and RM-ODP’s usage of the term. Model correspondences in ISO 42010:201X are designed to be compatible with viewpoint correspondences in RM-ODP, in the following sense: all RM-ODP viewpoint correspondences can be expressed as model correspondences (but not vice versa). The key differences are:

- RM-ODP’s viewpoint correspondences are binary; whereas model correspondences in ISO 42010:201X are n-ary;
- viewpoint correspondences relate elements within RM-ODP viewpoint specifications (i.e., *views* in ISO 42010 terminology), whereas model correspondences can also relate models themselves or their attributes (without reference to elements).

Linington provides a very useful overview of viewpoint correspondences in ODP [14]. Boucké *et al.* survey applications and mechanisms for expressing relations on architecture views (and their models) in the literature and put forth a taxonomy of these mechanisms [6]. The ISO 42010:201X construct has been designed with these perspectives in mind.

3. Specifying frameworks

This section describes the elements of an architecture framework as proposed in ISO 40201:201X.

3.1. Anatomy of a framework

Figure 1 shows the ingredients of an architecture framework and the relationships among an architecture framework and architecture descriptions. 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

Under IEEE 1471 (ISO/IEC 42010:2007), architecture descriptions address known concerns for known stakeholders for the system of interest. Architecture frameworks introduce a *level of indirection* in that the stakeholders for a system’s architecture are not known when the framework is defined. However, 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 can postulate likely stakeholders for specific systems or classes of systems within their domain. Accounting systems, for example, are likely to have both Chief Financial Officers and Auditors as stakeholders for any architecture. When the accounting system is for a specific business unit, the business unit management (operations, financial, executive, etc.) will be stakeholders for architectures of interest within that business unit.

This situation with architecture frameworks also arises with library viewpoints, and was recognized in the 2000 edition of IEEE 1471. This is one motivation for the definition of stakeholder in the standard, which allows the architect to perform some abstraction in identifying stakeholders. Sometimes it is important to identify individuals; in other cases, a class will suffice:

**system stakeholder** – An individual, team, or organization (or classes thereof) with interests in, or concerns relative to, a system

Thus the developer of an architecture framework needs to consider the known and likely stakeholders for systems and architecture descriptions of systems. These stakeholders motivate the set of architecture-related concerns that the architecture framework will focus upon. A conforming architecture framework must identify these architecture-related concerns.

Identifying the concerns to be addressed leads directly to the choice of viewpoints to be included. The primary content of an architecture framework is this set of viewpoints. Each viewpoint must meet minimal requirements from ISO 42010:2007. The spirit of these minimal requirements on viewpoints is that if each view is a map of the architecture, the viewpoint provides the legend for that map. A map legend explains the symbolic conventions used, and provides the basis for interpreting
the map. The viewpoint establishes the notations, models, techniques and methods to be used in architecture descriptions resulting from applying the framework. Beyond the minimal requirements, framework developers are given substantial latitude by the standard in the contents of viewpoints, they can be highly descriptive specifying not just models and tools, but also required content in models, patterns and styles, delivery formats, rationale and design capture, even process steps for constructing the viewpoint.

An architecture framework may include model correspondence rules that associate models or model elements with each other, as discussed in [5.1].

Finally, an architecture framework conforming to the standard is expected to extend, but not contradict, the core ontology in ISO 42010:201X. The developer of an architecture framework may add additional ingredients to a framework, such as the DODAF AV-2 requirement for an integrated data dictionary that 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.) The architecture framework can optionally document its extended ontology with an explicit metamodel.

3.2. The realization: every AD implicitly has a framework

Consider an architecture description conforming to ISO 42010:201X. It identifies a set of viewpoints which were selected to frame stakeholder concerns that are either reused from library viewpoints or developed specifically for this AD, and a set of correspondence rules to be asserted between the models of the AD. The AD also has the set of views (one per viewpoint) and a set of correspondences that must satisfy the correspondence rules.

Although ISO 42010:201X 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. 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.

In practice, many architecture descriptions include some framework-like content. Why not require in the standard that every architecture description contain exactly one framework? 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.

This proposal was rejected by the working group, not because it was technically unsound but due to the pragmatic consideration that it would result in “too many” 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. The working group was concerned this would 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 [5.1]);
3. conformance of an architecture description to an architecture framework.

[64x116]In practice, this is usually an iterative process, where an initial set of viewpoints and correspondence rules are defined, then some conforming views and correspondences are produced, akin to design [18]. 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.
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 [22], giving the architect a starting point when confronting a new system, when specific concerns are raised (e.g., reliability, security). Having a standardized definition of 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 42010 ontology, makes these discussions possible.

With the addition of 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 42010:201X’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 [16]. 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 42010:201X. Each row is therefore a view comprised of 6 models. Notice the implied life cycle-like ordering of the rows from Planner to User – this ordering can be captured via model correspondence rules (see example [2]). Layers of refinement are seen in a number of frameworks, as noted above.

DODAF. DODAF defines 3 “views” – Operational, Systems and Technical – which would be termed viewpoints under ISO 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) has some products that can be considered as correspondences between the views. In particular, the System View SV-6 work product, Systems/Services Data Exchange Matrix can be considered as a correspondence between the systems/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 is currently being revised using the terms and definitions of ISO/IEC 42010:2007 [11]. The latest version of ISO/CD 15704, Reference-base for enterprise architecture and models, 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 enterprise life cycle, model genericity, and modeling viewpoint, to form a cohesive framework.

Kruchten’s 4+1. Using the ontology from ISO 42010:201X, we can say the 4+1 architecture defines 5 viewpoints: Logical, Development, Process, Physical and Scenarios. As discussed above, Kruchten, in his original paper, addresses “Correspondences Between the Views”, detailing associations of “logical to process”, “logical to development” and “process to physical”. In practice, a successful 4+1 description was 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 42010:201X, 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 3 views labelled Conceptual, Logical and Physical. Although the term layers seems to imply ordering, none is stated in the Essential documentation. However the 3 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 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 frameworks had both been specified in a uniform manner, with stakeholders and concerns identified, as proposed here, this might be easier to discern.

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 DOODAF products. and The Contractor shall deliver an AD using the 4+1 model).

In such cases, an early decision for the architect is how 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 would 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 this analysis to make it reusable itself, i.e. Here is how we produce an AD that meets both DODAF and 4+1. Alignment with the ISO 42010:201X 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 42010 revision to standardize a notion of architecture framework. This was accomplished as a simple extension to the existing IEEE 1471 (ISO/IEC 42010:2007) ontology
through the addition of model correspondence rules to capture explicit relationships between models employed by viewpoints within an architecture description. Although not defined by the draft standard, model correspondences could also be used to express relations across multiple architecture descriptions (e.g. to express relations in a product family or system of systems) or even across architecture frameworks.

Applying the proposed ISO 42010:201X 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 [16], training and automated tools.

5.1. Future Work

Currently ISO 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 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 [15]; more recently the community has seen the development of languages for system and enterprise architecting (for example, SysML [17] and ArchiMate [21], 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. [21]

This certainly makes it 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. [17]
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? Capturing this information as part of the specification of a framework provides the necessary links between the concepts behind the languages and tools and the application of these languages and tools with a well-defined approach.

5.2. Contributing to the Revision

ISO 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 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 42010 framework). This introduces potential confusion in the user and tool community for each framework. However, as the complexity of architecture practices, as represented by “mandated” frameworks grows, the need to relate one framework to another through a standard ontology and vocabulary is likely to also 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

Figure 1. Adding architecture framework to the IEEE 1471 ontology (note: Architecture Framework, Model Correspondence, and Model Correspondence Rule are new with ISO 42010:201X)