ANSI/IEEE 1471 and systems engineering

Article *in* Systems Engineering · June 2004

DOI: 10.1002/sys.20008

CITATIONS
61

READS
1,710

3 authors, including:

Rich Hilliard
Institute of Electrical and Electronics Engineers
65 PUBLICATIONS **874** CITATIONS

SEE PROFILE

ANSI/IEEE 1471 and Systems Engineering[•]

Mark W. Maier
The Aerospace Corporation
15049 Conference Center Dr
Chantilly, VA 20151
Mark.W.Maier@aero.org

David Emery
The MITRE Corporation
7515 Colshire Drive, MS N110
McLean, VA 22102-3481
emery@davebert.mitre.org

Rich Hilliard
P.O. Box 396
Littleton, MA 01460
r.hilliard@computer.org

Abstract: ANSI/IEEE Standard 1471-2000 is the *Recommended Practice for Architectural Description of Software-Intensive Systems*, developed by the IEEE's Architecture Working Group (AWG) under the sponsorship of the Software Engineering Standards Committee of IEEE. ANSI/IEEE 1471 is the first formal standard[1] to address the content and organization of architectural descriptions. The standard defines the structure and content of an *architectural description* (AD) and incorporates a broad consensus on best practices for such descriptions. Although ANSI/IEEE 1471 was conceived as a software-focused standard, this paper argues that it is equally applicable to any system; hence appropriate for use as a part of systems engineering to describe system architectures. This article reviews the concepts of ANSI/IEEE 1471, the rationale for their selection, and demonstrates its application in systems engineering.

ARCHITECTURE FRAMEWORKS

The notion of architecture has entered into both the domains of software and systems engineering in recent decades. There are several threads to this entry. Many invoke a direct analogy with the notion in civil engineering, while others are built on particular practices largely drawn from the information technology industry. Some background on what problems are being addressed by what goes by the term "architecture" is helpful in understand the motivation for the ANSI/IEEE 1471 work and related frameworks.

There are two system development scenarios most commonly associated with a need for an architectural approach. These are not the only scenarios, but are the most common. In the first, the potential sponsor of a system development engages in a period of concept exploration before making a decision to go forward with development. At the end of the concept exploration phase the sponsor may decide to go forward and begin searching for suppliers,

[•] Penultimate draft of paper to appear in *Systems Engineering: the Journal of the International Council on Systems Engineering*.

[1] A wide variety of architecture frameworks exist as de-facto standards in particular communities, but ANSI/IEEE 1471 is the first formal standard in the sense of being a product formally approved by a recognized standards body.

may decide to abandon the project, or may decide to greatly refocus the effort and explore entirely different concepts. To support this decision, the sponsor needs a package of integrated information that spans objectives or requirements, major physical or non-physical components and their interfaces, alternative development strategies, cost estimates, and revenue or utility estimates. The level of detail or fidelity of these various elements must be sufficient so that the sponsor can make a decision at acceptable risk, but will normally contain far less detail that will eventually be developed by the supplier.

In the second scenario, a system-of-systems integrator has oversight responsibility for a number of systems at various stages of development, from operational to still hypothetical. The integrator's primary concern is that the collection of systems will jointly operate to produce desirable attributes (e.g. interoperable data exchange, more rapid response, greater reliability than any individual component system, reduced total cost). The integrator does not have full control over the collection of systems, but is able to impose some set of constraints on their development, structure, or operation to achieve his overall objectives. As in the first scenario, the responsible person requires an integrated package of information that spans multiple aspects, including objectives (usually in terms of the attributes of the collection as a whole), design features of the members of the collection, shared interfaces, development timelines, and other aspects. The level of detail required in this integrated description is once again much less than is relevant to the detailed development of each individual system in the collection, but must be sufficient to allow the responsible party to make oversight decisions in constraining the component systems.

Some authors have argued that these scenarios involve features of practice that are directly analogous to how a traditional civil architect works, and that lessons from that civil practice can be carried over to complex, high-technology system developments (see [Rechtin, 1991] and [Maier-Rechtin, 2000] for detailed arguments to this effect. Others, for example [Bass, et al., 1997], make similar arguments with less direct appeal to the civil architecture analogy. An essential feature of the two scenarios is that the decision maker requires a systems engineering description that is integrated across multiple aspects or views and whose consistency and completeness criteria are focused on decisions well above the level of detailed development.

Consider, under scenario one, a sponsor contemplating the development of a new satellite communications system. The decision to go-ahead will be determined by whether the expected benefit-to-cost ratio is acceptable and the uncertainty in returns and cost is within some risk tolerance of the sponsor. Being able to make that estimation is does not require the depth of understanding of requirements, or other design elements, that would be appropriate for

contracting with a supplier, or authorizing full scale development, or making a launch decision. But, that go-ahead decision does require integrated information across the fields of requirements (in business or operational terms), physical design, cost, and other programmatics. We refer to the integrated package of information as an "architecture description."

Because of the importance and frequency of such decisions, there have been a number of attempts to codify the notions of architecture and architecture description. We will refer to such attempts as "architecture frameworks" for purposes of discussion and comparison. An *architecture framework* is characterized by one or more of the following ingredients:

I-1: a definition of the term *architecture*, possibly with conditions on its use (e.g., scope of applicability, intended purpose);

I-2: a conceptual framework explicating key concepts and terms pertaining to architecture, sometimes explicitly separating the notions of *architecture* from *architecture description* and sometimes leaving the distinction implicit;

I-3: an approach to describing architectures, typically by placing minimum conditions on those descriptions;

I-4: architectural methods, perhaps including an operational concept of how architectural descriptions are created, analyzed, interpreted, and realized; and

I-5: a theory of how architecture and architectural descriptions fit into the larger context.

Not every architecture framework will manifest all of these ingredients, nor be required to for our purposes. Rather, one would start with (I-1) and look for work that defines *architecture*, then follow that work to see what other ingredients might be present. We have characterized architecture framework in this somewhat open-ended way for a very particular purpose: to allow some basis for comparison and understanding of architectural concepts across a wide range of work. In particular, across the fields of software and systems engineering, in a way that will hopefully permit increased understanding. In the remainder of this section we will briefly explain these ingredients. In the next section, we will introduce ANSI/IEEE 1471 with special reference to it as an architectural framework. In the following section, we will compare a number of architectural frameworks.

There are a number of definitions of the term "architecture" in the literature (I-1). The minimum expectation we have for an architectural framework is that it includes some definition of this term. We will discuss several below. The choice of definition will obviously have implications for all other aspects of the framework.

A conceptual framework (I-2) captures the key concepts and terms of interest, their definitions and

interrelationships. The extent of the conceptual framework determines the scope of the architectural framework. A conceptual framework may be as simple as a glossary of definitions, or a model of concept their properties and relationships, or a formal ontology.

Architectural frameworks of interest have typically been created for practical use. A key element of such frameworks will be techniques for describing architectures (I-3). Not all frameworks explicitly separate the two concepts of architecture and architecture description, although they all do at least implicitly. Such frameworks will have some notion of architectural description (AD) as an artifact, and will address such questions as, *What constitutes an AD? What should an AD be about?* etc. The approach to description bridges the potential gap between the theoretical (I-1, I-2) and practical (I-4, I-5) ingredients of an architectural framework.

Beyond an approach to architectural description, some frameworks establish methods to assist in the creation, use, analysis, interpretation and realization of architectural descriptions (I-4).

Lastly, some frameworks take a broader view of architecture, based on their scope of the term, to capture how architecture fits into a wider context of use (I-5). This context may include an overall approach to systems development and change processes, the relations of architectural methods to life cycles, other processes and methods, and the relation of architectural descriptions to other artifacts (such as requirements or designs).

ANSI/IEEE 1471

Background

ANSI/IEEE 1471, the IEEE *Recommended Practice for Architectural Description of Software-Intensive Systems* [ANSI/IEEE, 2000] was developed in response to the recent and widespread interest in software architecture and the emergence of common practices in that field which could be standardized. The standard was produced between 1995 and 1998 by a group of approximately thirty participants, and over 150 international reviewers. It went to ballot in 1999 and was approved by the IEEE Standards Board in September of 2000. The ANSI Board of Standards approved its use as an American National Standard in 2001.

When IEEE initiated development of IEEE 1471, there were four stated goals for the standard [Ellis, et. al 1996]:

1. Focus on architectural practices for software-intensive systems without precluding more general systems whenever possible. *Software-intensive systems* includes computer-based systems ranging over software

applications, information systems, embedded systems, systems-of-systems, product lines and product families, i.e., any system where software plays a substantial role in the development, operation, or evolution of a system.

2. Establish a framework and vocabulary for architectural concepts.
3. Identify and promulgate sound architectural practices.
4. Choose elements to standardize that will facilitate the continued evolution of architectural practices and technologies.

ANSI/IEEE 1471 was written at a time when there was significant active investigation into software architecture. A good reference to related work of the period is [Bass, *et al.*, 1997]. More recent work in software architecture incorporates many of the core ideas of ANSI/IEEE 1471, in particular the use of multiple views and the abstraction of views into *viewpoints*: templates reusable across multiple description instances [Clements, *et al.* 2002]].

As a framework, ANSI/IEEE 1471 defines *architecture* (I-1), presents a conceptual framework (I-2) and embodies a theory and practice of architectural descriptions (I-3) based on that conceptual framework. Within the standard itself, the conceptual framework is introduced via a class diagram (see figure 1). Various elements of the conceptual framework will be detailed in the remainder of this section.

ANSI/IEEE 1471 does not incorporate or require specific architectural methods (I-4). It was designed to be *method-neutral*, and therefore intended to be useful with both existing architectural methods [e.g., Kruchten, 1995], and serve as a foundation for new methods. Similarly, ANSI/IEEE 1471 does not assume or prescribe the life cycle context within which concepts of architecture are to be applied (I-5). Therefore it is *process-neutral*, and may be applied within various system life cycles.

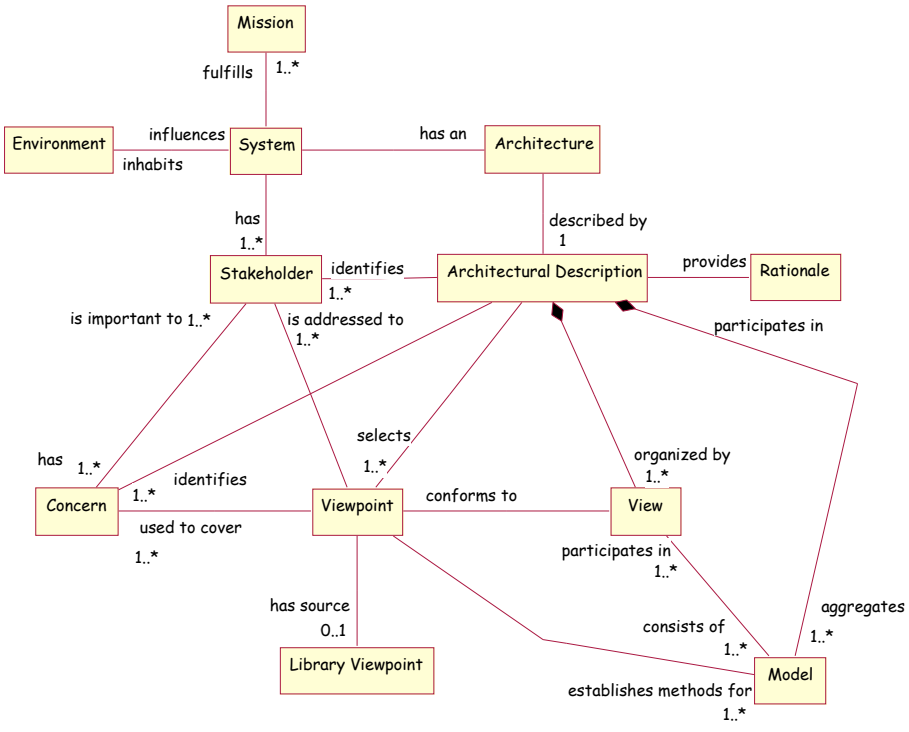

Figure 1: The ANSI/IEEE 1471 conceptual framework expressed as a UML class diagram

Defining "Architecture"

In ANSI/IEEE 1471, an architecture is an attribute (admittedly, a very complex attribute) of a system. The focus of ANSI/IEEE 1471 is architectural description rather than architecture. That is, it is about architectural blueprints rather than architectural style, but one has to have some understanding of what architecture is to be able to standardize conventions for its description. Broadly speaking, the architecture of a system is that which is essential or unifying about a system. It is that set of system attributes that largely determines the system's value, cost, and risk. The definition in ANSI/IEEE 1471 is:

> **architecture**. the fundamental organization of a system embodied in its components, their relationships to each other and to the environment and the principles guiding its design and evolution.

There are several key ideas in this definition. First, architecture is a *conceptual* attribute of a system—its fundamental organization. By this definition, a system *always* has an architecture, whether or not somebody has taken the trouble to write it down, whereas an architectural description, as a concrete artifact representing an architecture, may or may not exist. Therefore, the architectural description is not the architecture; it is a description of something that is fundamentally conceptual, following the common understanding in the civil building world. If

we asked five architects to write down an architectural description of the same pre-existing building without collaborating, we would not expect to get back five identical descriptions. Each might choose different techniques and emphasize different points, but it would be clear they were describing the same building. Thus architectural descriptions can vary even when the fundamental property, the architecture, remains the same. Second, architecture embodies "fundamental" things about a system. Here we mean "fundamental" in the sense of an abstraction of things that are important about the system as a whole. We cannot know what is fundamental about a system without knowing *fundamental to whom?* Therefore "fundamental" must be interpreted in the context established by the stakeholders of the system. Third, an architecture is to be understood in context, not in isolation. To understand a system's most fundamental characteristics (i.e., its architecture) we must understand how the system relates to and is embedded in its environment. This definition of architecture is motivated by the case of civil architecture: an architecture is articulated from the viewpoints of its stakeholders, their interests determine its fitness for purpose, and this is all to be understood in its environmental context.

Scope of ANSI/IEEE 1471

ANSI/IEEE 1471 sets requirements for *architectural descriptions* (AD), which are defined as documents produced to describe a system's architecture. In this sense ANSI/IEEE 1471 is similar to a standard for blueprints. It defines the equivalent of symbology and drawing conventions, but does not define the full range of drawings needed for an adequate description of any particular system. The most important elements of ANSI/IEEE 1471 are as follows:

1. A normative set of definitions for terms including: architecture (I-1), architectural description, architectural stakeholder, architectural concern, architectural view, and architectural viewpoint (I-2);
2. A separation of the concepts of "architecture" and "architectural description" to facilitate separating standards for how architectures are described (analogous to blueprint standards) from standards on how systems should be constructed (analogous to building codes or zoning laws) (I-3);
3. A conceptual framework for how these concepts are implemented in the context of the many uses of architectural descriptions (I-2); and
4. A set of normative requirements on the elements of an architectural description of a system and the relationships among those elements, including a notion of conformance for well-formed architectural descriptions (I-2, I-3).

ANSI/IEEE 1471's conceptual framework is described in the next section. Its normative content (requirements on ADs) is presented in the following section.

Conceptual Framework

This section defines some of the key elements of the ANSI/IEEE conceptual framework. The elements we discuss are architectural stakeholders and concerns, architectural views and viewpoints.

Stakeholders and Concerns

ANSI/IEEE 1471 includes a system's stakeholders and their architectural concerns as fundamental elements in an architectural description.

Stakeholders are those individuals, groups and organizations with some architectural interest in the system [Gacek *et al.*]. ANSI/IEEE 1471 requires that an architectural description identify the stakeholders for the system under description and document those stakeholders as a part of the architectural description of that system. In addition, ANSI/IEEE 1471 requires a particular set of stakeholders to be considered during the identification of stakeholders. The architect must consider as stakeholders: users, acquirers, developers, and maintainers of the system. This set was kept small in the spirit of making required only those elements on which there was broad agreement. For systems engineering use, this set needs to be taken metaphorically (see *Architecture in Systems Engineering*, below).

An *architectural concern* is a topic of interest to one or more stakeholders pertaining to the architecture. In ANSI/IEEE 1471, concerns are simply things the stakeholders care about, the things that allow the system to have value to its stakeholders, or those attributes of the system that affect their willingness to engage in its development. Concerns may pertain to the system's operations, its development or other aspects of the system in its context. Concerns may be requirements-oriented or design-oriented. Concerns may take the form of requirements, user needs, wants or desires or design constraints. Concerns may be fine-grained (e.g., *Does this architecture allow the system to meet requirement R1.4.5?*) or coarse-grained (e.g., *Does this architecture have an approach to fault-tolerance?*). There is a growing body of work on "concern-oriented" approaches to software architecture (see for overview, [Kandé, 2003]); some of these approaches will be equally applicable to systems engineering.

Views, Viewpoints, and Architectures

Another fundamental tenet of ANSI/IEEE 1471 is that an architectural description is organized into *multiple architectural views*. This follows both conventional civil architecture practice and emerging approaches in the

software and system engineering communities. The rationale for this practice, codified in ANSI/IEEE 1471, is that an *architectural view* represents the architecture of the (whole) system with respect to a specified set of related architectural concerns. For example, a collection of structural models representing the whole system forms a structural view of that system. As noted above, architectural concerns originate with the stakeholders. If concerns are questions about the architecture, each architectural view is a model which answers some of those questions.

Each architectural view in an AD is defined relative to an *architectural viewpoint*. A viewpoint captures the rules for constructing and analyzing a particular kind of view. It is a template for a view and can be reused across many architectural descriptions.

A key decision within ANSI/IEEE 1471 is to separate the notion of view and viewpoint; both concepts are important to understanding architectural practice. The following physical examples may help to motivate the distinction. If one looks at the front of a chair and the front of a car one sees different things. A chair and a car are different objects, but both can be "viewed" from the front. ANSI/IEEE 1471 would call the concept of looking from the front the "viewpoint" and what one actually sees when looking at a particular object from the front the "front view." Thus, the viewpoint is *where one looks from*, the view is *what one sees*. This relationship is illustrated in Figure 2. It has been explored at greater length elsewhere [Maier, 1998].

Of course, for complex systems a "front view" is of little importance, instead views are used to capture the collection of models representing function, cost, or other key concerns. The concept is the same, though the modeling languages and inter-view relationships are much more sophisticated than in the physical, three-dimensional example just given. Separating view and viewpoint is important when writing standards, defining architecture methods, and establishing enterprise architectures, although not essential if one's scope is simply the description of a single system.

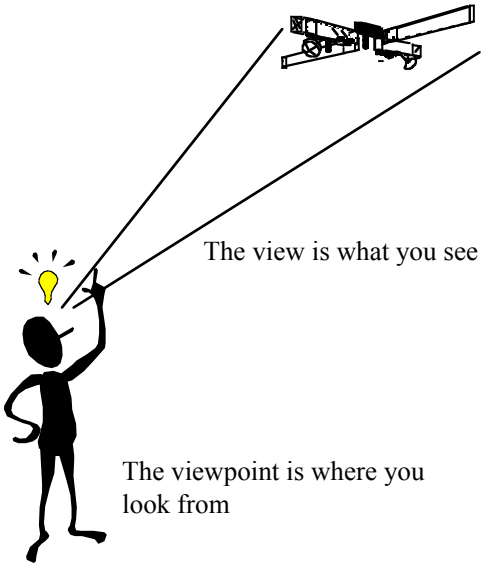

Figure 2: Viewpoint and view are just abstractions of familiar geometric ideas

An early debate among the authors of ANSI/IEEE 1471 was what views should be required in an architectural description? At the time, there were several architectural methods (such as Kruchten's 4+1 view model [Kruchten, 1995]), that each espoused a specific set of views. A fundamental decision in ANSI/IEEE 1471 was not to require any specific views in an AD, leaving this decision to users of the standard, based on the particulars of their system, its domain, the architectural method to be used, conventional practices within their organization and other considerations. Instead, ANSI/IEEE 1471 allows—actually requires—architects to select viewpoints most appropriate to their application and then document those viewpoints as a part of the AD. While not making particular viewpoints mandatory, ANSI/IEEE 1471 imposes mandatory requirements on how they are chosen. This allows the flexibility that developers need while also adding a desired element of rigor to the structure of architecture descriptions.

This decision about "viewpoint freedom" has several consequences. First, it avoided otherwise irresolvable problems in establishing a consensus on which modeling techniques, notations or description languages should be required by the standard. Instead, the architect must identify which "viewpoint languages" are to be used in each view. Second, not freezing the standard on particular viewpoints appears to be consistent with the current state of architectural practice which exhibits a wide diversity of viewpoints. Perhaps in the future, it will be possible to narrow in on some of these. Third, it provides a means for architects to identify and select best practices from around the community. It is therefore an extensibility mechanism allowing architects a principled basis to use the

best notations, methods and "analysis techniques indigenous to the various quality … communities" [Barbacci, *et al*.]. Fourth, it promotes good practices by linking viewpoint selection to outstanding architectural concerns. This is built into the requirements of ANSI/IEEE 1471: by requiring that stakeholders be explicitly identified for each architectural description, and that viewpoints that address those stakeholders' concerns be chosen for each description. [Hilliard, *et al.*, 1996]

The extensibility mechanism just described is not found in other architectural frameworks. The user of the standard is not constrained to specific viewpoint languages. Users are free to adopt other standards that specify viewpoints to be used by the user's organization, facilitating reuse of viewpoints across architectural descriptions and across organizations. Existing standards such as ISO RM-ODP, the C4ISR Architecture Framework, and the US DOD Architecture Framework can be accommodated this way, as described below. Extensibility is also a mechanism for advancement of the discipline. It provides a mechanism for codifying approaches, comparing them across methods, and advancing the state of the practice.

Finally, as we argue below, viewpoint freedom makes possible the standard's application to systems engineering. The broader nature of concerns common in systems engineering (e.g., physical properties, integration with diverse external systems, manufacturability) can be readily accommodated.

Using ANSI/IEEE 1471

An architectural description *conforms* to ANSI/IEEE 1471 if it meets the requirements in clause 5 of the standard. These requirements include identifying the key stakeholders for the architecture, factoring the description into views, selecting well-defined techniques for documenting each of these views (the architectural viewpoints to be used), and insuring consistency between views. The required relationships for a conforming description are shown in Figure 1. Specifically, to conform with ANSI/IEEE 1471 an architectural description must contain all of the elements shown in Table 1.

1. Various documentary elements, such as date of issue, issuing organization, change history, summary, etc.
2. An identification of all stakeholders considered in formulating the architecture of the system.
3. An identification of all of the stakeholder concerns considered in formulating the architecture of the system.
4. Models of the system's architecture, organized into views, with a one-to-one relationship between the views provided and the viewpoints specified.
5. Each view is underwritten by a viewpoint. Just as a map has a legend or set of conventions; a view must have a related set of conventions.
6. Identification and description of all viewpoints used in the architectural description. Each viewpoint must be specified by a name; one or more stakeholders to be addressed; one or more concerns to be addressed; the language, modeling techniques, or analytical methods used in constructing a conforming view; and the source for the viewpoint. The use of explicitly or even formally defined viewpoint languages is encouraged, but the current state-of-the-art does not allow it to be required.
7. The rationale used to select the described architecture and evidence that significant alternatives were considered.
8. Each identified concern (3) is addressed by at least one view.
9. A list of all known inconsistencies among views, identified using explicit consistency and completeness criteria to check both within and between views.

Table 1. Normative elements of ANSI/IEEE 1471

Within a conforming architectural description, the concerns of all of the stakeholders must be identified and all concerns must be mapped to at least one of the provided architectural views (through the viewpoints). Since users, acquirers, developers, and maintainers are all required stakeholders, their concerns are required to be addressed within the chosen viewpoints. For example, it is reasonable to expect that the functions provided by the system will be a concern of the users. Therefore, to satisfy users as stakeholders, a conforming description *must* have a functional viewpoint, or a viewpoint that otherwise covers the concern of functionality provided by the system. In this way there is, in effect, a required set of viewpoints (and thus views), as long as the actual stakeholders of the

system possess reasonably common kinds of concerns.

While ANSI/IEEE 1471 embodies ingredients (I-1), (I-2) and (I-3) discussed above, it is neutral on the other ingredients of an architectural framework. In the remainder of this section, we highlight a few of these considerations.

Although certain elements of an architectural process are "induced" by the requirements on content of an AD, there is no architecting process either assumed, described or required by ANSI/IEEE 1471. An implementing organization is free to use any process as long as it results in a conforming description. Each implementing organization must extend the framework of ANSI/IEEE 1471 to make its own AD process and standards. Implementing organizations can use this flexibility to adapt ANSI/IEEE 1471 to various purposes, including system-level descriptions. IEEE 1471 does not assume any particular system or software life cycle into which "architecting" has been embedded. Its use is therefore compatible with a range of life cycle styles from a traditional waterfall development, to more recent incremental/iterative approaches.

A consequence of this is that there is no relationship prescribed between requirements and architecture in the standard. Some approaches to architecture make a life cycle assumption (I-5) that requirements definition in the classic sense (i.e., an unambiguous and complete definition of what the system must do) precedes development of the architecture. A good example of this approach is how ANSI/IEEE 1471 integrates with IEEE/EIA 12207 a software development and life cycle standard. In other approaches (such as product line approaches), the architecture may be formulated prior to specific requirements analyses. Being life cycle- and process-neutral, ANSI/IEEE 1471 is compatible with either life-cycle approach. In some situations requirements owners can be considered the only stakeholders, and the requirements themselves the only concerns. ANSI/IEEE 1471 applies equally in situations where the architect and client must search for stakeholders, elicit and elaborate their concerns, and form a stable constituency for the construction of the system, *prior* to the specification of a system's requirements. In this case the classic acquisition requirements come from the architecting process and could be captured in a "requirements view". Having architects "search for" stakeholders may go beyond what some would consider architecting, but it is explicitly accommodated within ANSI/IEEE 1471.

Choosing and Writing Viewpoints

A key contribution of ANSI/IEEE 1471 is the idea that a viewpoint associates a set of concerns about an architecture with how the architectural description will address these concerns, in terms of languages and notations,

models and analytical techniques, methods, etc. The intent is that an organization can develop a library of viewpoints as it gains experience with architecting. At the same time, each system is different, and often an existing viewpoint must be modified or perhaps a new viewpoint must be defined for the specific system at hand.

A viewpoint must define stakeholders [*who*] and concerns addressed [*what*], and define modeling language(s) and other analytical techniques [*how*] used to construct a conforming view. There are a number of example definitions of viewpoints in the annexes of ANSI/IEEE 1471. None of the examples is short enough for inclusion in this paper, but they can be given quickly in outline form. A use-case viewpoint would typically have system users as its stakeholders, the provided behavior as their concerns, and standard graphical textual use-case representations as the language [Larman, 1998 (esp. Chap. 6)]. A functional decomposition viewpoint might have implementing engineers as the stakeholders, precise functional specification as the concern, and graphical and textual data flow decomposition as the language [Hatley, 2000, (Chap. 4)].

ANSI/IEEE 1471 does not impose any particular degree of formality on architectural descriptions. The degree of formality is determined by the choice of description languages in the viewpoints. If the user of the standard wishes to use formal languages, they can be specified in the selected viewpoints. If the user wants little formality then informal languages can be specified. This flexibility is both a strength and a weakness of the standard. Because the standard takes such an open position relative to the languages that can be selected, conformance to the standard is no guarantee that the descriptive modeling has been carefully carried out. On the other hand, openness of the standard toward language selection ensures that users can adapt to their local needs. In particular, it makes it much easier to adapt ANSI/IEEE 1471 to system use because it does not force the adoption of particular software architecture description languages.

The selection of viewpoints for a given description also supports the notions of consistency and completeness checking across views in the description. ANSI/IEEE 1471 does not formally define consistency and completeness among views, but the ideas can be clearly articulated within the conceptual framework of the standard [Maier, 2000b]. Consistency means that the provided views should not mutually define something that is impossible. Put another way, there should exist at least one feasible implementation of the system, which, if reverse engineered into views, would match the views provided in its ANSI/IEEE 1471 architecture description. If two views describe the same thing in incompatible ways, it would be impossible to build a single system that matched both descriptions. In ANSI/IEEE 1471 we distinguish between intra-view consistency (which is a matter of good choice of description

languages) and inter-view consistency (which is a difficult analytical challenge).

Completeness is a more difficult concept. Within ANSI/IEEE 1471 a complete architecture description is one that "covers" all identified stakeholder concerns. Completeness should always be understood as complete relative to something, where the something is the set of identified stakeholder concerns. That is, the description must be fit for the purposes of its sponsor, which may be quite partial with respect to fitness for other purposes. This is an important distinction that often arises in practice. Consider the case of developing a satellite. A set of architecture plans produced for the satellite program sponsor may be quite adequate for the sponsor to make a decision on whether or not to go ahead with the project and yet still be far from complete with respect to what the actual satellite contractors will need during construction. We expect an extended process of design refinement to occur after the basic acquisition has been made. Similarly, there may be architectural descriptions that are quite adequate for limited purposes (such as evaluating system interoperability) but inadequate for the purpose of contracting for system construction. By understanding completeness to mean completeness with respect to identified stakeholder concerns, rather than a more abstract notion, it is possible to use ANSI/IEEE 1471 to rationalize architecture frameworks targeted at relatively limited audiences.

An essential idea of the ANSI/IEEE 1471 approach to architecture is that architecture descriptions must explicitly flow from the selection of stakeholders and concerns. Instead of mandating a particular description language or description products, ANSI/IEEE 1471 mandates that the stakeholders and concerns for the architecture be explicitly identified, and that all description elements flow from those stakeholders and concerns identified. While this is hardly original in systems engineering practice, it is in contrast of many of the existing standards. For example, the U.S. DOD Architecture Framework is built on mandating particular description products, and stakeholders are mentioned only in passing. Placing the stakeholders and their concerns in a central position helps in subsequent analysis activities. By having the concerns explicit there is a natural way to identify the quality factors of interest, and thus the appropriate subjects of analysis.

ARCHITECTURE IN SYSTEMS ENGINEERING

In this section, we discuss the applicability of ANSI/IEEE 1471 in systems architecting. In this discussion, we raise three issues: (1) What can the application of ANSI/IEEE 1471 offer to systems architecting? (2) Are there impediments to its application? (3) Is it compatible with other, existing systems architecting practices?

The elements of ANSI/IEEE 1471 as an architectural framework are equally applicable to systems architecture.

It was clear to the ANSI/IEEE 1471 developers that the practices being recommended were not unique to software, but applied equally to more general systems. Because the standards effort was under the sponsorship of the Computer Society, there was no effort to explicitly extend the work beyond software-intensive systems. The definition of architecture seems to codify existing practice, if one construes "components" to include hardware and people as well as software components. The conceptual framework of stakeholders, concerns, views and viewpoints also applies directly—since there is nothing in the framework limiting its scope to software.

ANSI/IEEE 1471 requires a particular set of stakeholders to be considered for a conforming architectural description. An AD must identify users, acquirers, developers, and maintainers of the system as stakeholders. For systems engineering use, this set needs to be taken metaphorically. A wider set of system stakeholders might include: operators, producers, testers, deployers, trainers, disposers, the public, management, competitors, and regulators.

The position of ANSI/IEEE 1471 with respect to stakeholders and concerns makes it quite appropriate for systems engineering. ANSI/IEEE 1471 does not assume, as many software engineering standards do, that requirements are already worked out and that basic tradeoffs have been made. ANSI/IEEE 1471 places the beginning of the analysis directly with the primary stakeholders, which is where it should be to maximize the extent to which systems engineering can productively affect the system. ANSI/IEEE 1471 also facilitates interaction between systems engineering and software engineering by using a compatible set of concepts to describe both system and software architectures. This approach is likely to improve work practices in those systems where software dominates development (the original target of ANSI/IEEE 1471).

ANSI/IEEE 1471 and Other Standards

As ANSI/IEEE 1471 developed, it became clear that it would work best if it meshed with existing architecture standards (e.g., ISO Reference Model-Open Distributed Processing (RM-ODP) [ISO, 1996], U.S. C4ISR Architecture Framework [C4ISR, 1997], U.S. DOD Architecture Framework [DODAF, 2003]). In general, ANSI/IEEE 1471 is compatible with the existing standards, although architectural descriptions developed in accordance with existing standards are not automatically conformant/conforming to ANSI/IEEE 1471. In some cases the concepts and models in the existing standards need to be refined to be ANSI/IEEE 1471-conformant. This refinement can be an important vehicle for reconciling the diverse framework standards.

IEEE 1471 is not a replacement for these other standards; it is an organizing framework intended to supplement

such standards, by providing specific content requirements on architectural descriptions. Annex D of the standard addresses its relationship to the IEEE/EIA 12207 standard on software life cycle processes and the ISO/IEC Open Distributed Processing—Reference Model (RM-ODP). The approaches taken in this annex are described below. The Open Group Architecture Framework (TOGAF) is also moving toward harmonization with IEEE 1471. The requirements of US DoD Architecture Framework (DODAF) can be cast as requiring ADs to have at least three particular viewpoints. The three required viewpoints correspond to the three DODAF views (operational, system, and technical views), each with a specified set of representations (in the DODAF product descriptions). Seen in this way, the DODAF is an additional requirement placed on ADs for certain classes of system that facilitates the evaluation of those ADs from the perspective of the requiring group. On the other hand, some additional information must be added to a DODAF architecture description to enable it to conform to ANSI/IEEE 1471. In particular, the specific stakeholders and concerns to be addressed by the DODAF architecture description must be explicitly stated. This helps to rationalize the DODAF views and products by establishing those stakeholder concerns that each DODAF product is intended to address.

Use of ANSI/IEEE 1471 can facilitate the evolution of domain-specific architecture and architecture description standards by installing common language and concepts. ANSI/IEEE 1471 may be very useful for extending the architecture frameworks that exist and reconciling them with each other. Viewed through the lens of ANSI/IEEE 1471, existing architecture frameworks may be better understood by other communities. ANSI/IEEE 1471 forces the question of explicitly considering the stakeholder audience for each architecture description. A domain-specific standard that provides information required for one group of stakeholders may be a good standard, but ANSI/IEEE 1471 can identify the deficiency of such a framework when additional stakeholders not included in the domain-specific description standard have to be introduced.

Annex D of the standard specifically addresses its relationship to IEEE/EIA 12207 [IEEE, 1998] and ISO RM-ODP [ISO, 1996]. Both can be relatively reconciled with ANSI/IEEE 1471, though by different means. IEEE/EIA 12207 takes a conventional, top-down decomposition approach to software development. In this case IEEE/EIA 12207 becomes the process standard and ANSI/IEEE 1471 a standard for one of the process-required descriptions. This can apply more generally to other process-oriented standards. ANSI/IEEE 1471 describes a specific viewpoint that can reconcile the standards by making the viewpoint normative within an organization's architecture descriptions. RM-ODP already falls into a five-viewpoint structure. It can be reconciled with ANSI/IEEE 1471

simply reorganizing the contents of each viewpoint.

A more complex example shows how the ANSI/IEEE 1471 concepts integrate into existing systems engineering methods. The Hatley-Pirbhai method [Hatley, 2000] can be recast as three required viewpoints. The three resulting viewpoints, in ANSI/IEEE 1471 terms, are shown in Table 2.

Viewpoint Name: Requirements **Stakeholders**: Users, developers **Concerns**: Input-output behavior delivered by the system, functional decomposition of the system **Modeling language**: Data Flow Diagrams with Hatley-Pirbhai control flow extensions. Used to produce a single decomposition tree of system functions, with external interface resolving enhancements. **Consistency and Completeness Analysis Methods**: Documented series of consistency rules
Viewpoint Name: Structure (called "architecture model" in [Hatley-Pirbhai, 2000]) **Stakeholders**: Developers **Concerns**: Identification of system components and their interfaces **Modeling language**: Tree of module diagrams, arranged in two layers (connection diagrams and flow diagrams) **Consistency and Completeness Analysis Methods**: Documented series of consistency rules
Viewpoint Name: Allocation **Stakeholders**: Developers, acquirers **Concerns**: Mutual consistency of requirements with structure **Modeling language**: Allocation table relating primitive functions in requirements model tree to modules in the structure model **Consistency and Completeness Analysis Methods**: All primitive functions allocated to exactly one module, every module allocated at least one primitive function. Structure data flow consistency with allocated functions.

Table 2. The three viewpoints of the Hatley-Pirbhai method

Other standards can be similarly recast as additional normative extensions on ANSI/IEEE 1471 ADs. This is illustrated in Table 3. The table shows that the primary mechanism is the casting of the other standards' view-like concepts into ANSI/IEEE 1471 viewpoints. This highlights the value of ANSI/IEEE 1471 as an organizing framework for ADs in different domains, and potentially a means for improved understanding between domains. It also highlights the fact that many existing architecture frameworks do not specify a set of views large enough to satisfy all concerns, just those of particular stakeholder groups. Equally, some of the frameworks may over-specify by requiring views not necessary for certain purposes. There is nothing wrong with this, as long as system developers do not think that the resultant incomplete expression of a system's architecture is actually complete.

For an existing architecture framework to produce architectural descriptions that are conformant to ANSI/IEEE 1471, the framework must:

- Require the inclusion of administrative material in accordance with the initial parts of section 5 of ANSI/IEEE 1471;
- Require explicit identification of stakeholders and concerns, either individually by architecture description or generally as part of the framework definition;
- Explicitly define the required viewpoints, including the stakeholders and concerns addressed and the languages used; and
- Explicitly lay out the intra-view and inter-view consistency criteria, even though they may not be formal.

A framework may also lay out a more explicit and normative process to follow in constructing architecture descriptions, though it need not do so to produce ANSI/IEEE 1471-conformant descriptions. Based on these criteria, the ISO RM-ODP [ISO, 1996] and Kruchten's 4+1 [Kruchten, 1995] frameworks are closest to producing ANSI/IEEE 1471-conformant descriptions. Neither explicitly requires identification of stakeholders and concerns, but both deal with the issue implicitly. Both have explicitly defined viewpoints, although their terminology is somewhat different. Both use moderately formal notations within their viewpoints.

Applying the US DoD's Architecture Framework (or its predecessor the C4ISR Architecture Framework) does not produce ANSI/IEEE 1471-conformant architectural descriptions, although it could be modified to do so. The DODAF does not require that stakeholders and concerns be explicitly identified. The framework's development was oriented toward acquisition supervisors whose primary concerns were interoperability and its evaluation. In practice the DODAF has often been used to produce architecture descriptions for acquirers in early stages of development, a task it is not necessarily well suited for. The DODAF requires three views with defined notations, but they differ somewhat from what is required in ANSI/IEEE 1471. The three viewpoints in the DODAF are broader than the viewpoints in ANSI/IEEE 1471. In normal use of ANSI/IEEE 1471 one would divide the three DODAF viewpoints

4+1 [Kruchten, 1995]	ISO RM-ODP [ISO, 1996]	C4ISR [C4ISR, 1997]	Zachman [Zachman, 1987]
Logical	**Enterprise**	**Operational**	**Data**
Process	**Information**	**System**	**Function**
Physical	**Computation**	**Technical**	**Network**
Development	**Engineering**		**People**
Scenarios	**Technical**		**Time**
			Motivation

Table 3. Examples of the viewpoints/views required by well-known architecture frameworks. The views and viewpoints in the standards are not necessarily precisely equivalent to IEEE 1471's, but show the possibilities for reconciliation.

into smaller parts targeted at smaller groups of concerns. The notations in the DODAF are also very informal. ANSI/IEEE 1471 allows this, but most users of ANSI/IEEE 1471 seek a greater degree of formality, at least to the level of an explicit schema for each architecture notation.

One potential point of conflict with other frameworks is ANSI/IEEE 1471's insistence that architecture includes stakeholders and concerns rather than being solely concerned with the physical structure of the system (as in some architecture description languages). This is really a conflict of terminology rather than disagreement over good practices. The conflict occurs because some groups use the term "architecture" to refer only to the initial parts of a physical design and assume that requirements are essentially a fixed input to architecture development. Systems engineers would agree that requirements should flow from stakeholders and concerns, but might argue that these issues should be resolved before architecture work begins. ANSI/IEEE 1471 takes the broader approach that architecting considers requirements and design in partial parallelism [Maier, 2000a]. ANSI/IEEE 1471 can be used in environments where requirements and design are more strictly separated by having a process that requires that views be resolved in a particular order.

We have already discussed how ANSI/IEEE 1471 is intended to be process-neutral. It can therefore be applied within the context of systems engineering process standards such as ANSI/EIA 632, *Processes for Engineering a System* and ISO/IEC 15288, *Systems Engineering—System Life Cycle Processes*. ISO/IEC 15288 was developed as the systems engineering companion to ISO/EIA 12207 (discussed above). It provides a common framework of processes to develop and manage systems engineering efforts. ANSI/EIA 632 is another process standard, providing an "integrated set of fundamental processes to aid a developer in the engineering or reengineering of a system". There are several points in the fundamental processes wherein an architectural description might be produced or applied:

- Technical Management: a strawman architecture may be prepared for costing and planning purposes
- Acquisition and Supply: an AD may form part of the basis for a acquirer-supplier agreement
- System Design, Solution Definition: an AD may be prepared to document system architecture for stakeholders and developers
- Product realization: an AD may be prepared as blueprints for developers
- Technical Evaluation: an AD may be prepared as the basis for verification, validated against requirements

CONCLUSIONS

In this paper, we have introduced ANSI/IEEE Std 1471 and described its characteristics as an architecture framework. We have contrasted it with other architectural frameworks and then moved this discussion from software engineering to systems engineering. A key observation is that the conceptual framework on which

ANSI/IEEE 1471 is based is not particularized for software systems. We have demonstrated its applicability to systems engineering through a discussion of its use with system engineering standards and techniques.

While written as a standard for software-intensive systems, ANSI/IEEE 1471 is equally applicable to more general systems in providing a template for architectural descriptions. A conforming architectural description will incorporate many of the consensus best practices in architectural description, although a great deal of good practice (e.g. selection of viewpoints as a function of system type) lies outside the scope of the current version of ANSI/IEEE 1471. A particularly important application of ANSI/IEEE 1471 in systems engineering may be to reconcile and rationalize the wide array of architecture frameworks now becoming popular.

REFERENCES

ANSI/EIA Std 632, *Processes for Engineering a System*.

ANSI/IEEE Std 1471-2000, *Recommended Practice for Architectural Description of Software-Intensive Systems*.

Barbacci, M. R., Klein, M. H., Weinstock, C. B., Principles for Evaluating the Quality Attributes of a Software Architecture. Software Engineering Institute Technical Report, CMU/SEI-96-TR-036, 1996.

Bass, L, Clements, P., Kazman, R., *Software Architecture in Practice*, Addison Wesley, 1997.

C4ISR Architecture Framework, version 2.0, U. S. Department of Defense Architecture Working Group, December, 1997.

Clements, P., et al., Documenting Software Architectures: Views and Beyond, Addison Wesley, 2002.

Department of Defense Architecture Framework, version 1.0, U.S. Department of Defense Architecture Working Group, 2003.

Ellis, W. J., Hilliard, R., Poon, P. T., Rayford, D., Saunders, T. F., Sherlund, B., and Wade, R. L., Toward a Recommended Practice for Architectural Description, *Proceedings of 2nd IEEE International Conference on Engineering of Complex Computer Systems*, Montreal, Quebec, Canada, October 21-25, 1996.

Gacek C., Abd-Allah, A., Clark, B. and Boehm, B. W., "On the definition of software system architecture", *Proceedings of the First International Workshop on Architectures for Software Systems*, Seattle, WA, 1995.

Hatley, D., Hruschka, P., Pirbhai, I., Process for System Architecting and Requirements Engineering, Dorset House, 2000.

Hilliard, R., Rice, T. B, Schwarm, S. C., "The Architectural Metaphor as a Foundation for Systems Engineering", in Proceedings of the 1996 Symposium of the International Council on Systems Engineering, 1996.

Hillliard, R., IEEE Std 1471 and Beyond. in *SEI Workshop on Software Architecture Representation, 16-17 January 2001*. Software Engineering Institute Special Report CMU/SEI-2001-SR-010.

IEEE/EIA 12207.0-1996, *Standard for Information Technology—Life Cycle Processes*, March 1998.

ISO/IEC 10746-3:1996, *Information technology—Open Distributed Processing-Reference Model: Architecture*, 1996.

ISO/IEC 15288, *Systems Engineering—System Life Cycle Processes*, 2003.

Kandé, M. M., A concern-oriented approach to software architecture. Doctoral dissertation, École Polytechnique Fédérale de Lausanne, 2003.

Kruchten, P. B., A 4+1 View Model of Software Architecture, *IEEE Software Magazine*, pp. 42-50, Nov. 1995

Larman, C., Applying UML and Patterns, Prentice Hall, 1998.

Maier, M. W., and Rechtin, E., The Art of Systems Architecting, second edition, CRC Press, 2000.

Maier, M. W., Model Organization through Viewpoints and Views, in Proceedings of International Council on Systems Engineering Mid-Atlantic Regional Conference, pp. 6.2-1 to 6.2-9, 2000.

Maier, M. W., Reconciling System and Software Architectures, in *Proceedings of the Eighth Annual International Symposium of the International Council on Systems Engineering,* pp. 99-106, 1998.

Zachman, J.A., A framework for information systems architecture. *IBM Systems Journal*, 26(3), 1987.

AUTHOR INFORMATION

Mark W. Maier was educated at Caltech and the University of Southern California. He received B.S. and M.S. degrees from Caltech, and the EEE and Ph.D. in Electrical Engineering from USC. While at USC he was a section head at the Hughes Aircraft Company, Radar Systems Group where he led signal processing efforts in advanced radars and electronic warfare systems, and held a Howard Hughes Doctoral Fellowship. He was also a corporate instructor in the Hatley-Pirbhai structured analysis and design method. He is currently a Distinguished Engineer at the Aerospace Corporation. He has supported a variety of national security space customers. Previously, he was an Associate Professor of Electrical Engineering at the University of Alabama in Huntsville (UAH). While at UAH he was the faculty advisor to the SEDSAT-1 student satellite project, which successfully launched a microsatellite in 1998.

Maier is the co-author, with Eberhardt Rechtin, of “The Art of Systems Architecting.” He was a member of the IEEE 1471 writing team and was the chair of the ballot resolution committee.

David Emery is a Principal Engineer in MITRE's Embedded Systems department, providing systems and software engineering on a variety of military command and control and weapon systems. He previously worked for Hughes Aircraft of Canada, Siemens Research and Computer Sciences Corporation, and served on active duty with the U.S. Army. He is active in both the IEEE and the ACM, and has participated in several international standards activities, including IEEE 1471, IEEE POSIX and ISO Ada. Mr. Emery has been honored with the IEEE Third Millenium Medal, Outstanding Contribution and Meritorious Service awards, and selection to the IEEE Computer Society's "Golden Core". SIGAda recently awarded him its Outstanding Contribution Award. He is published on Ada programming language bindings, software portability and architectural approaches for software-intensive systems. His paper (co-authored with R. F. Hilliard and T. B. Rice) *Experiences Applying a Practical Architectural Method,* won Best Paper award at Ada-Europe '96.

Rich Hilliard is a consultant. He was Chief Technology Officer and co-founder of ConsentCache, Inc. Previously, he was Director of Architecture at ephibian.com (formerly, ISIS 2000) where he led their architecture consulting efforts. From 1994-1998, he was a lead architect at the MITRE Corporation, where he was a founding member of the Chief Architects' Office of the Air Force's Electronic Systems Center (ESC/DIA). In that capacity, he was a major contributor to the architecture for the Air Force's Integrated Command and Control System (C2STA). From 1991 to 1994 he was a member of the Ada 9X Mapping/Revision Team, which developed the ISO standard for Ada 95. In the early 1980s, he was one of the developers of the IDEF methods for the Air Force's Integrated Computer-Aided Manufacturing (ICAM) program.

He served as technical secretary of the IEEE Architecture Working Group, which developed IEEE Std 1471-2000.