Toward a Recommended Practice for Architectural Description*

Walter J. Ellis
Software Process and Metrics
ellis@cse.ogi.edu

Richard F. Hilliard II
The MITRE Corporation
rh@mitre.org

Peter T. Poon
Jet Propulsion Laboratory
peter.t.poon@jpl.nasa.gov

David Rayford
Comerica Incorporated

Thomas F. Saunders
The MITRE Corporation
tfs@mitre.org

Basil Sherlund
Comerica Incorporated
she@acm.org

Ronald L. Wade
E-Systems
rwade@ix.netcom.com

Abstract

The Architecture Planning Group (APG) was chartered by IEEE's Software Engineering Standards Committee to set a direction for incorporating architectural thinking into IEEE standards. In this paper, we present a framework for architectural thinking and use it to review existing architectural practices; summarize the work of the APG and its recommendations; and provide the rationale for those recommendations.

1. Introduction

It is recognized that architecture should have a strong influence over the life cycle of a system. In the past, hardware architectural aspects were dominant, whereas software architectural integrity, when it existed, was first to be sacrificed in the course of system development.

The cost of software development and the increasing complexity of software systems has changed the relative balance. Today, software technology has matured. The practice of systems development can benefit greatly from adherence to architectural precepts, at both the systems and software systems levels. However, the concepts of architecture are not yet consistently defined and applied over the life cycle.

To address this, IEEE's Software Engineering Standards Committee (SESC) chartered an Architecture Planning Group (APG) to set the direction for incorporating architectural thinking into future IEEE standards.

The APG began its work August 19, 1995, in Montreal. At that meeting, we adopted the charter included as appendix A. Over the course of the next eight months, we met regularly to develop an Action Plan for SESC. In April 1996, that Action Plan was delivered to SESC and accepted. We have subsequently initiated an Architecture Working Group (AWG) to implement the APG recommendations.

In its deliberations, the APG set these goals for itself and any subsequent Architecture Working Group:

1. To define useful terms, principles and guidelines for the consistent application of architectural precepts to systems throughout their (full) life cycle.
2. To elaborate architectural precepts and their anticipated benefits for software products, systems and aggregated systems ("systems-of-systems" [13, 17]).
3. To provide a framework for the collection and consideration of architectural attributes and related information for use in IEEE Standards.
4. To provide a useful road map for the incorporation of architectural precepts in the generation, revision and application of IEEE standards.

It was the consensus of the APG that there is a mode of architectural thinking applicable to systems which is much wider than only software. *Every system or subsystem has an architecture.* As such, the APG work attempts to address all relevant software standards and practices and to consider interdisciplinary perspectives from a systems' life-cycle perspective. Although the IEEE SESC is software oriented, the intent is to produce standards and guides which can broadly

*In *Proceedings 2nd IEEE International Conference on Engineering of Complex Computer Systems, Montreal, Quebec, Canada, October 21–25, 1996.*

support all disciplines involved in developing systems. This "wide scope" was encouraged and affirmed by SESC and will be reflected in the framework and recommendations described below.

In order to assess the impact of the wide scope interpretation of architecture, and to identify the standards potentially impacted, the APG undertook an analysis of recent and on-going activities in the community related to "architecture." Every effort was made to contact professional associations to determine the extent of efforts in establishing architectural standards and guidelines within the industry. Keyword searches were performed on IEEE standards as well as those of other standards setting organizations. Keyword searches were also performed using several search engines on Internet. The effort identified several hundred potential efforts. These were narrowed through a process of determining whether architecture was a primary or secondary focus. Elimination criteria consisted of documents that had architecture as a by-product or where the word was used in a descriptive sense rather than a development perspective. A list of approximately twenty standards and in-process works represents a starting point for the harmonization and coordination efforts of the Architecture Working Group with other efforts.

2. Architectural Framework

To meet our first goal of defining "useful terms, principles and guidelines," we began our work by initiating the development of a conceptual framework (or, frame of reference) for talking about architecture. In formulating the framework, we found the building metaphor – the analogy between software systems architecture and traditional building architecture quite useful [12, 21].

For perspective, we began with the common, "pre-systems" definition of the word:

> **Architecture** 1. The art or science of building or constructing edifices of any kind for human use. 2. The action or process of building. 3. Architectural work: structure, building. 4. The special method or style in accordance with which the details of the structure and ornamentation of a building are arranged. 5. Construction or structure generally.[1]

Each of these senses of the word are applicable in the systems context.

There is at present much interest and activity in system architectural thinking (for example, [9, 15, 22]). These activities would benefit from a common framework of terms and concepts. The key term is "architecture" itself. We considered the current IEEE 610.12–1990 definition of "architecture" [11]:

> **Architecture** The organizational structure of a system or component.

However, this definition was not adequate for our purposes. It has several limitations. (1) It does not distinguish an architectural level of structure for a system from the design or physical structure of that system. (2) It does not separate architectural concerns, such as the interaction of the system with other systems, from other concerns, such as details of construction. (3) It offers no basis for treating architectures as engineering objects.

Looking at a number of other definitions in the literature (for example, [4, 7, 25]), we found most suffered from these or other limitations. We settled on the following definition:

> An *architecture* is the highest-level concept of a system in its environment.

This definition addresses (1) by stating that architecture pertains to the *highest-level* of a system, distinguishing it from design and implementation concerns. As Mary Shaw admonished the First International Workshop on Architectures of Software Systems [8], "Let's not dilute the term 'architecture' by applying it to everything in sight." Our definition acknowledges (2) by reminding us that systems are situated *in their environments*, and that the architecture is in part a recognition and response to that environment. But it still does not directly address (3) — a "concept" is not particularly useful as an engineering construct. Following recent IEEE work on software design [10], we distinguish the architecture itself from *descriptions* of that architecture:

> An *architectural description* is a model — document, product or other artifact — to communicate and record a system's architecture.

This distinction allows us to retain the idea that every system has an architecture, while advocating that the *explicit* description of a system's architecture allows that system to be brought under "engineering control." The form and content of architectural descriptions may then be subject to standardization — not the architectures themselves.

There is an established engineering approach to bringing such complex entities under control, which is to "separate concerns" by identifying one or more viewpoints [23]. Therefore, our framework advocates that an architectural description be organized into well-defined pieces:

> An architectural description conveys a set of *views* each of which depicts the system by describing domain concerns.

Meszaros defines an architectural view as: "a way of looking at an architecture. Each view may have a different concept of components and relationships" [19]. In our

[1] *Oxford English Dictionary*, 1933 edition.

framework, each "way of looking" will be determined by the interests of the users of that view. A system exists in many domains — it is therefore useful to regard architecture as a multi-disciplinary practice [18]. Views offer a way to get a handle on this. Some common architectural views are listed below, reflecting typical systems concerns reflected at an architectural level of description.

- behavioral, dynamic, operational views [16, 14, 4]
- data, data flow, information views [5, 7]
- development, maintenance views [3, 6]
- distributed, network views [25, 6]
- functional, activity views [24, 25]
- logical views [14, 20]
- static views [14, 7]
- physical views [14, 4]

The selection of relevant views will usually be determined by the architect — in consultation with other key stakeholders of the system. The conventions by which a view is depicted will vary with the particular architectural technique or method employed; however, the engineering principles governing views may be generic — for example, completeness, the principle that an architectural view should represent the whole system from a single, well-defined perspective, is adopted in various methods [20, 25].

The notion of "components" recurs throughout systems and software thinking about architecture, as a construct for capturing the major elements of the "structure or construction" of a system. Of course, these elements do not exist in isolation but, as Meszaros observes, are related. The APG framework postulates two kinds of relations: connections (among components), and constraints (on components, or connections).

The Architecture Planning Group chose to define these terms as follows:

- *Components* are the major structural elements in a view; such as functions in a functional view, data models in a data view, or hardware in a physical view.
- *Connections* are the major relations between components of a view. They may be "run-time" relationships like control or data flow, or other dependencies.
- *Constraints* represent laws the system must observe; constraints apply to components and connections. There are three kinds of constraints:
  - Constraints reflecting performance, functional, or non-functional requirements (such as security, fault tolerance, quality)
  - Style and protocol rules, and
  - Laws of nature which constrain resources

The use of components and connections (or connect*ors*) seems to be a *de facto* standard in current architectural thinking, although the community is split on the latter term. Abd-Allah, Boehm, Gacek, Kazman, Abowd, Bass, for example, use "connection." Garlan, Shaw, and Allen use "connector." Luckham uses both terms to distinguish the relation between two components (a connection) from the entity which realizes that relation (the connector). In the present document, we use *connection* to encompass both notions, recognizing that we may want to refine this at a future time.

2.1 Uses of Architectural Description

To identify architectural concepts and principles relevant to the potential users of a future standard, we realized we needed to understand the roles which architectural descriptions may play within system life cycles. While development of a full "concept of operations" for architectural descriptions is left to the Working Group, the APG identified two fundamental ways of applying architectural precepts: *architecture as design* and *architecture as style*.

The first way is to use an architectural description as the vehicle for expressing high-level system characteristics that define and organize its major elements and their interrelationships. The architectural description is used to communicate between client and developer to aid clarification of requirements and assess their impact on system design. The architectural description is often developed through an evolutionary process from the initial expression of a system concept as a high-level abstraction to one of a more detailed and tangible expression that is widely accepted as being an expression of design.

The second way is to use a subset of the information used in a full architecture description to capture a style to facilitate certain common attributes among systems, ranging from system compatibility, interoperability, (component) interchangability, to (system) replacability. An *architectural style* is a set of patterns or rules for creating one or more architectures in a consistent fashion. There are many ways to capture and communicate a style [1, 2]. A *reference model* can be used to embody a style. Style is a partial characterization of a system; it does not represent the complete architecture for a system, but is a template for specifying the architecture of a specific system.

Architecture as design is useful for individual product development, analogous to the design of individual buildings, whereas *architecture as style* is also useful for harmony among products, analogous to the design and planning of cities. Consider the following examples:

1. Individual software products have usually had an architectural concept established prior to implementation. However, all too often, conflicts between immediate user requirements and the architecture are resolved in favor of the requirements. The architecture is compromised over time, making the product less tolerant of modification or enhancement. More prominent attention to architecture can and would make software products more manageable over their life-cycle.

2. The architectural design of a new system can often benefit from an understanding of previous architectural designs for similar or related systems. However, the relative merits of one architecture vs. another for addressing a specific constraint varies according to the mix of other constraints upon the new system. A systematic approach to architectural description would aid this understanding by facilitating "reuse" of architectural knowledge.

3. Modern software development practices have evolved an even stronger impetus to adopt attention to architecture. In recent times, the state of the software practice has begun to include reuse of software products. Such reuse is only possible when expected behavior is consistent with actual behavior. As such, encouraging the software community to articulate and observe architectural style rules is likely to facilitate the further reuse and the continued maturation of software engineering as a discipline.

4. While it is important for an architect to understand the functional aspects of a system, it may be critical for the success of that system to embody an architectural view of those who must interface with it. If we look at a mass transit map of any major metropolitan area we see how various subsystems interconnect, where various stops are located, and perhaps travel times between locations. We would not, however, attempt to use the map for determining distance, or our precise location from a geometric perspective. Such maps are neither rendered to scale nor do they show every turn, rise, or fall. Yet, the view is critical for the successful use of the system.

5. In recent years Planned Unit Developments (PUD) have gained in popularity. From an architectural perspective the components that make up the unit are the various entities within the proposed development area. This allows the abstraction of many facets: roads (interfaces), electric, water (support services), population density (complexity).

6. Most major appliances have located on them a block diagram. For instance, a refrigerator might have a diagram on the back depicting the compressor, defroster, lights, temperature control, water location for ice maker, power cable, etc. Although not to be confused with the refrigerator itself, the diagram is one view of the device.

These examples meet the principles stated above, in that that they convey specific perspectives, dealing with domain concerns. In each case what has been described as an end product descriptor, probably had its beginning as a development document to convey conceptually what the device, product, or service was going to look like.

3. APG Recommendations

The primary result of our deliberations was an Action Plan, delivered to SESC. That Action Plan made several recommendations.

First, the Architecture Planning Group recommended that SESC approve two new Project Authorization Requests (PARs) for:

1. a *Recommended Practice for Architectural Description*; and

2. a *Guide* to the new Architectural Description standard.

Second, recognizing that architectural issues were of a wider scope than any single IEEE standard, the APG recommended that the APG Action Plan be disseminated to the SESC Working Group Chairs in order for the Chairs to review and apply the architecture precepts set forth therein for the generation and revision of IEEE Software Engineering Standards.

Third, the APG recommended that the APG Action Plan be disseminated to other affected disciplines so that the framework of terminology and architectural precepts set forth therein can be useful in establishing a dialog with others in those disciplines in order to coordinate their responses during the standards development process.

Fourth, the APG recommended that the Architecture Working Group established by the above PARs address the following topics in the preparation of the *Recommended Practice* and *Guide*:

- How does architecture fit into life cycle?
- How do architecture documents relate to other life cycle documents?
- How are architectures documented?
- Who are the stakeholders for an architecture?

- What architectural methods or processes are defined?
- What kinds of analyses may be applied to architecture models?

Finally, the Architecture Planning Group recommended that the following "concepts of operations" for an architect be explored and supported by the resulting Standard and Guide:

1. Software architects will require projections of available technologies to plan not a point solution, but a means to evolve a system including the user.
2. Architectural planning will include teams of experts in the relevant engineering disciplines of hardware, software, and human factors. For example, just as hardware should be selected which supports good software engineering, the software architecture should allow hardware evolution.
3. Time-to-complete a system will be balanced against time-to-obsolescence. System capabilities will be made operational and evolved iteratively through simulation if necessary. Too often, designs of the seventies are implemented in the nineties chasing a technology that will never be in use — obsolete before it is implemented.
4. Maturing the concept of prototype, architects will plan systems to include the user through early implementation of capabilities in the evolution to system solutions. Planning must continue throughout system evolution so that future capabilities are commensurate with the hardware engines available to support them. Like the weather, prognostication of the availability of storage, speed, and input/output devices will be necessary to judge whether the system will be reasonably current when implemented. A reasonable time for implementation can then be determined, and a reasonable subset of capabilities can be designated initial. Systems are not just architected as designs, they are planned to have initial capabilities which will evolve to solutions including user inputs.
5. Architects will plan with logistics, support, evolution, and continuing quality.
6. The expression of architecture will provide for the conveyance of lessons learned, suitability, etc., for the use of architectural solutions.

4. Conclusion

SESC has determined that there is sufficient technical basis and community interest in architecture to implement the Architecture Planning Group's recommendations for an Architecture *Working* Group which will undertake the definition of a *Recommended Practice for Architectural Description*, and companion *Guide*. The first two quarterly meetings of the Working Group took place in May and July 1996. The IEEE Architecture Working Group may be contacted at: `ieee-awg@spectre.mitre.org`.

Acknowledgments Philippe Kruchten (Rational) provided helpful comments on a draft of this paper as a member of the Architecture Working Group.

References

[1] A. Abd-Allah and B. W. Boehm. Reasoning about the composition of heterogeneous architectures. Technical Report USC–CSE–95–503, University of Southern California, 1995.

[2] G. Abowd, R. Allen, and D. Garlan. Using style to understand descriptions of software architectures. *ACM Software Engineering Notes*, 18(5):9–20, December 1993.

[3] B. W. Boehm. Software process architectures. In *Proceedings of the First International Workshop on Architectures for Software Systems*, Seattle, WA, 1995.

[4] Defense Information Systems Agency. *Technical Architecture Framework for Information Management (TAFIM)*, version 2.0 edition, 1995.

[5] L. Druffel, N. E. Loy, R. A. Rosenberg, R. J. Sylvester, and R. A. Volz. Information architectures that enchance operational capability in peacetime and wartime. Technical report, US Air Force Science Advisory Board, February 1994.

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

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

[8] D. Garlan, editor. *Proceedings of the First International Workshop on Architecture for Software Systems*, Seattle, WA, April 24–25 1995. Published as CMU–CS–TR–95–151.

[9] D. Garlan. Research directions in software architecture. *ACM Computing Surveys*, 27(2):257–261, 1995.

[10] *IEEE Std 1016, Recommended Practice for Software Design Descriptions*, 1987.

[11] *IEEE Std 610.12, Glossary of Software Engineering Terminology*, 1990.

[12] R. F. Hilliard II, T. B. Rice, and S. C. Schwarm. The architectural metaphor as a foundation for systems engineering. In *Proceedings of Sixth Annual International Symposium of the International Council on Systems Engineering*, 1996.

[13] I. Jacobson, K. Palmkvist, and S. Dyrhage. Systems of interconnected systems. *ROAD*, May–June 1996.

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

[15] H. W. Lawson, W. Rossak, and H. R. Simpson. Working group report – CBS architecture. In *Proceedings of the 1994 tutorial and workshop on systems engineering of computer-based systems*, Los Alamitos, CA, 1994. IEEE Computer Society Press.

[16] D. C. Luckham, J. Vera, and S. Meldal. Three concepts of system architecture. Technical Report CSL-TR-95-674, Stanford University, July 1995.

[17] M. W. Maier. Architecting principles for systems-of-systems. In *Proceedings of Sixth Annual International Symposium of the International Council on Systems Engineering*, 1996.

[18] M. W. Maier. System architecture: An emergent discipline? In *1996 IEEE Aerospace Applications Conference Proceedings*, volume 3, pages 231–246, 1996.

[19] G. Meszaros. Software architecture in BNR. In *Proceedings of the First International Workshop on Architectures for Software Systems*, 1995.

[20] M. Moriconi and X. Qian. Correctness and composition of software architectures. In *Proceedings of ACM SIGSOFT '94: Symposium on Foundations of Software Engineering*, New Orleans, LA, December 1994.

[21] D. Perry and A. Wolf. Foundations for the study of software architecture. *ACM SIGSOFT Sofware Engineering Notes*, 17(4), October 1992.

[22] E. Rechtin. *Systems architecting: creating and building complex systems*. Prentice Hall, 1991.

[23] D. T. Ross, J. B. Goodenough, and C. A. Irvine. Software engineering: process, principles, and goals. *IEEE Computer*, 8(5):17–27, May 1975.

[24] SofTech. Integrated Computer-Aided Manufacturing (ICAM) Task I – Final Report. Technical Report AFML–TR–78–148, Air Force Materials Laboratory, 1978.

[25] J. F. Sowa and J. A. Zachman. Extending and formalising the framework for information systems architecture. *IBM Systems Journal*, 31(3):590–616, 1992.

A. Architecture Planning Group Charter

The Architecture Planning Group will define for the Software Engineering Standards Committee the statement of direction for incorporating architecture into the set of IEEE standards for software engineering. Every system or subsystem has an architecture, as defined by IEEE 610.12–1990. Every system with software has a software view of that architecture. This planning group will define terms, principles and guidelines for software architecture, not in isolation, but integrated with the views of other disciplines.

Planned Tasks

1. Define a framework for relating the concept and principles of software architecture to software and systems engineering.

2. Examine selected IEEE software engineering standards for:

   a. their conformance to the framework of software architecture that has been defined in task 1.

   b. their applicability and use in Software architecture.

3. Produce an Action Plan with recommendations for incorporating software architecture into IEEE standards for software engineering. (E.g., obsolete old standards, modify existing standards, propose new standards). Provide recommendations for the Software Engineering Standards Committee to work effectively within the systems communities.