IEEE SESC Architecture Planning Group: Action Plan

Foreward

The definition and application of architectural concepts is an important part of the development of software systems engineering products. The Software Systems Engineering Planning Group was created by IEEE SESC in April 1995, under the chairmanship of Basil Sherlund. At its August 1995 meeting, SESC recommended that the name of the group be changed to the “Architecture Planning Group” to operate with the following 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 (the *Standard Glossary of Software Engineering Terminology*). 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. (For example, obsolete old standards, modify existing standards, propose new standards). Provide recommendations for the Software Engineering Standards Committee to work effectively within the systems communities.

Planning Group Members

The members of the Architecture Planning Group are:

W. Ellis (SPM)
R. Hilliard (MITRE)
P. Poon (JPL)
T. Saunders (MITRE)
B. Sherlund, chair (COMERICA)
R. Wade (NSTC)

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 the first to be sacrificed in the course of system development.

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

This planning group seeks to provide a useful road map for the incorporation of architectural precepts in the generation, revision, and application of IEEE standards. Although the IEEE SESC is software oriented, the intent is to produce standards and guides which can broadly support all disciplines involved in developing systems.

1.1 Purpose

The purpose of this plan is fourfold:

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).
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 road map (approach) for applying architectural concepts to IEEE Standards.

1.2 Scope

As defined by IEEE 610.12-1990, every system or subsystem has an architecture:

architecture. the organizational structure of a system or component

However, the 610.12 definition is not sufficient for developing comparative attributes, for preserving an architectural structure over the life-cycle of a system, or for developing a consistent technical foundation for improving how architectures are developed, acquired and maintained. There is a need to expand beyond the definition provided IEEE 610.12-1990 to develop a useful standard for architecture. This plan is intended to address all relevant software standards and practices and to consider interdisciplinary perspectives in a systems' life-cycle context.

1.3 Audience of Action Plan

Our primary audience for this action plan is SESC and the working groups that will be developing IEEE Software Standards. In addition, this plan is intended to encourage dialog among users and advocates of related standards. There is a secondary audience: the broader community of users, developers and stake holders in the community of systems.

2. Need Statement

There are at least two fundamental ways of applying architectural precepts: architecture as "design"; and architecture as "style." (See Figure 1.)

The first 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 their impact on system design. The architectural description is developed in 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 is to use a subset of the information used in a full architecture description to capture style and protocol standards that can be used to facilitate certain common attributes that promote system-to-system consistency.

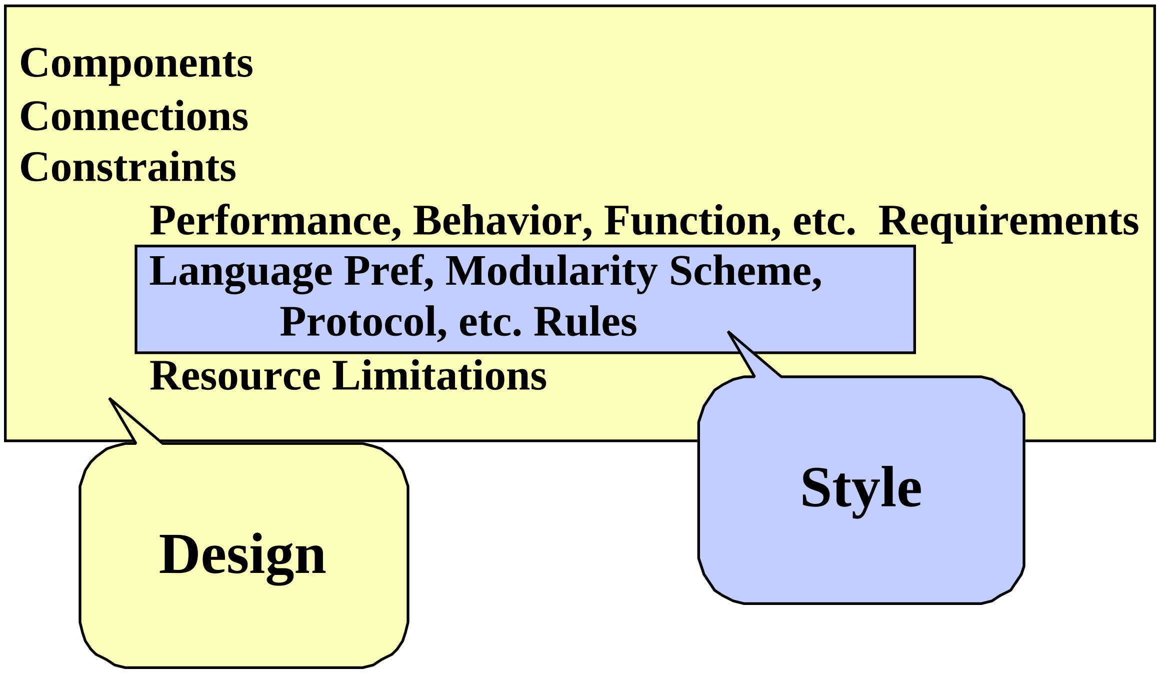
Components
Connections
Constraints
Performance, Behavior, Function, etc. Requirements
Language Pref, Modularity Scheme, Protocol, etc. Rules
Resource Limitations
Style
Design

Figure 1. Relationship Between Architecture as “Design” and “Style”

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:

Individual software products usually have had a software architecture concept established prior to implementation. However, all too often, conflicts between immediate user requirements and the software 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.

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.

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 observe architectural style rules will facilitate the further development of this practice which, in turn, is important for the continued maturation of software engineering as a discipline.

2.1 Terms of Reference

NOTE: Italics in this section indicate terms being defined. Bolding indicates terms that are defined subsequent to their first use.

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

An *architectural description* is a model (document, product or other artifact) which:

- conveys a set of **views** each of which depicts the system by describing domain concerns. (See Figure 2.)
- consists of **components, connections** and **constraints**. (See Figure 1.)

An architectural *view* represents the whole system from a single perspective. E.g., functional view, operational view. (See Figure 2.)

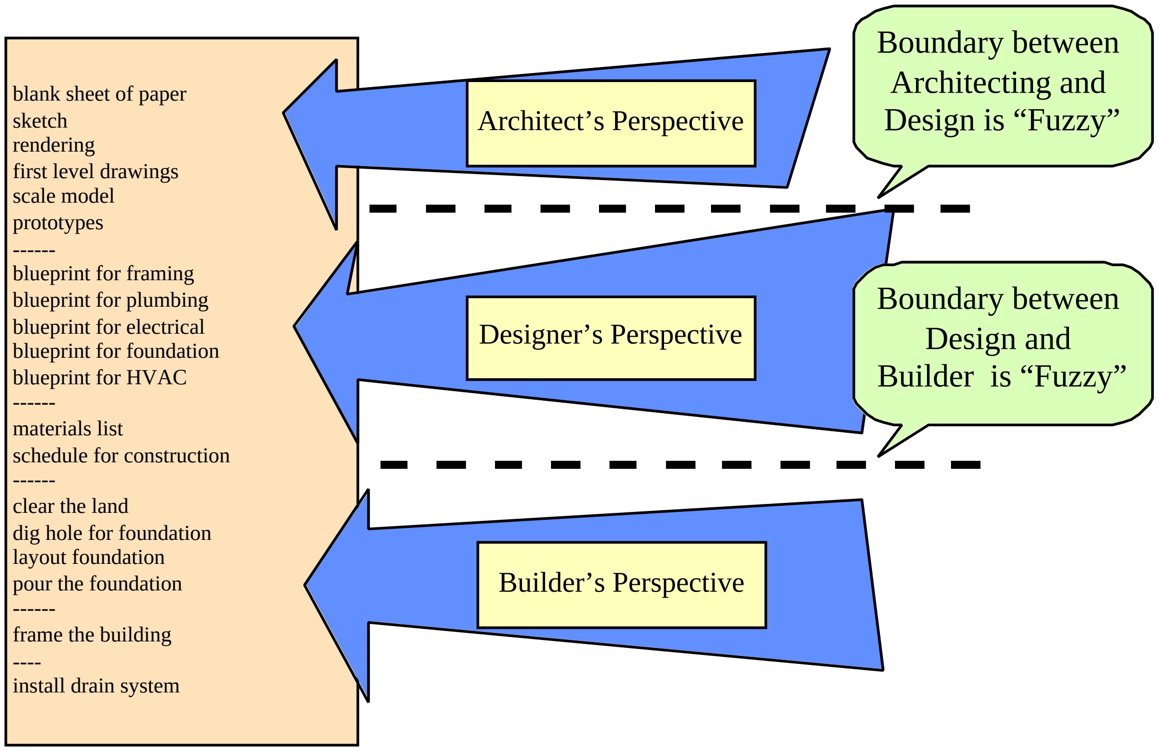

Figure 2. Evolutionary Perspective of Architecture, Design, and Development

Components are the major structural elements in a view. E.g., functions in a functional view, 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 which reflect performance, functional, and non-functional (security, fault tolerance, quality, business, marketing, etc.) requirements,
- **style** and protocol rules, and
- laws of nature which constrain resources.

An architectural *style* is a pattern or set of rules for creating one or more architectures in a consistent fashion. Style is included within the characterization of a system.

A *reference model* can be used to embody a style. It does not represent the complete architecture for a system, but a template for building a specific architecture.

A *domain specific model* can be used to embody a design of one or more systems reflecting needs of an application area. It does not represent the complete design for a specific system but is a design template.

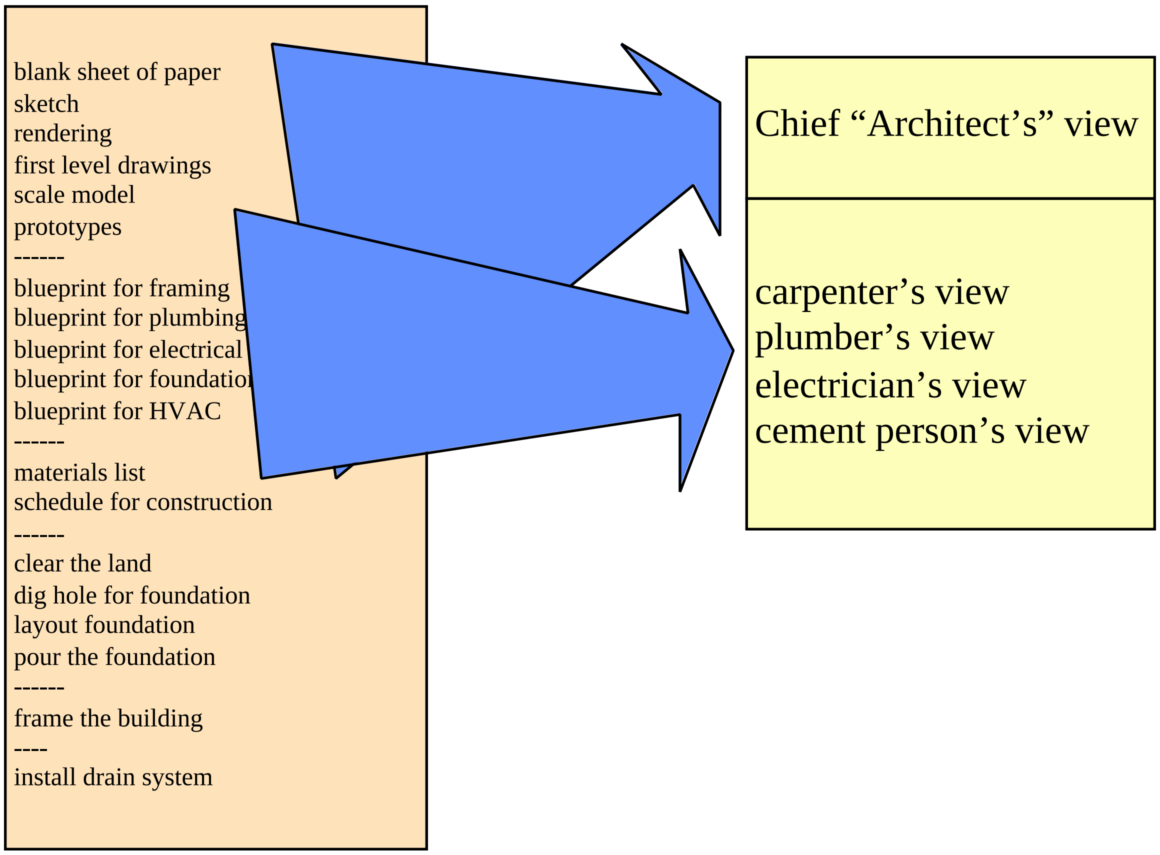

Figure 3. Architecture "Views" as a Function of Domain (of expertise or interest)

Architecting is the process of developing an architecture from concept to architectural description. The process is an evolutionary activity in which the selection of views, and details of components, connections, and constraints are determined as increasingly detailed representations. The evolutionary stages of the process migrate from "architecture" work to "design" work (and eventually to "implementation" work). The boundaries between these evolutionary phases are necessarily "fuzzy." (See Figure 3.)

2.2 Applicable User Expectations

This document will lead to a set of IEEE standards and guides that will enable architecture to be expressed, communicated, and evaluated in a consistent manner among system architects. The standards and guides will include consideration for the interdisciplinary teams from system, hardware, software and human factors engineering in future systems design. The IEEE standards and guides will encourage incorporation of lessons learned in prototyping, software engineering, and hardware evolution through independence, and system evolution.

3. Existing Related and In-work Standards

Although virtually all standards developed and maintained by the SESC will require coordination and consideration of architecture aspects, the ones listed below may be directly affected.

3.1 List of Standards and Preliminary Applicability

IEEE Std 610.12-1990 Standard Glossary of Software Engineering Terminology. This standard should be updated to remain consistent with emerging terms related to architecture. This will require a high degree of interaction and coordination because of the overlap in the approval process.

IEEE Std 982.1-1988 Standard Dictionary of Measures to Produce Reliable Software. This standard should be updated to include metrics associated with architecture. Coordination on how to include metrics should be built into any proposed architectural standard.

IEEE Std 982.2-1988 Guide for the Use of IEEE Standard Dictionary of Measures to Produce Reliable Software. This standard should be updated to match any changes to 982.1.

IEEE Std 1002-1987 Standard Taxonomy for Software Engineering Standards. This standard should be modified to include how architecture fits into the overall picture of software engineering. This would be beneficial from both the perspective of utilizing it as a method of planning as well as the basis for classifying the emerging standard.

IEEE Std 1058.1-1987 Standard for Software Project Management Plans. This standard is a candidate for revision because of the close relationship between the development of an architecture and the development of the project plans associated with the effort.

IEEE Std 1062-1993 Recommended Practice for Software Acquisition. Architecture should be a major consideration in the acquisition of quality systems. Several of the Clauses should be considered for modification during the development of the architectural standards. Specifically the Clauses 1, 2, 4, and 5, these are, respectively, Scope, References, Life Cycle, and Process Steps.

IEEE Std 1063-1987 Standard for Software User Documentation. Users often have a need to understand the architecture of a system in abstraction to understand how best to use it. This is often called an "operational view." This will require careful consideration on how to incorporate architectural standards into this existing standard.

IEEE Std 1074-1991 Standard for Developing Software Life Cycle Processes. This standard includes reference to architectural aspects of the design activity. This and related information need to be coordinated as the architectural standards are developed.

ISO / IEC 12207 Information technology – Software Life Cycle Processes. This standard has just been approved. Architectural standards could have impact in two specific areas: first, Life Cycle Processes under the sub-headings of both Acquisition and Development, second, in the Supporting Life Cycle Processes under the sub-heading of Documentation.

ISO/IEC JTC1 / SC7 / WG10. The purpose of this working group is to define the organization and management for the development of a suite of international standards for software process assessment. It has been recommended that the IEEE Architectural Working Group include this group in the dissemination of material as it is developed. Research indicates that the proposed architectural standard from the IEEE working group would likely be one of the documents produced as a part of the process in developing software in the international arena.

IEEE ECSB TC. This technical committee is in the process of defining what architecture is in the realm of Computer Based Systems (CBS). They have produced several iterations of a document that defines a framework that allows for discussion. The material which is continually under development is taking a parallel path on many elements with the IEEE Architectural Planning Group, and will prove to be a valuable aid in the review of the work that will be performed. The groups are sufficiently different in nature that there will not likely be any conflict in the generation of documents to serve the IEEE community.

3.2 Sources of In-work Standards and Preliminary Applicability

Standards Based Architecture (SBA). A part of the U.S. DoD's Technical Architecture Framework for Information Management (TAFIM). There is a major revision effort underway in the area of Information Technology. The TAFIM has also recently been adopted by X/Open.

Architecture Reporting and Monitoring System (ARMS). A system developed for the monitoring of architecture aspects of systems. Points of contact: USAF Space and Missile Systems Center; Capt L. Scott Thomason, and The Aerospace Corporation; Mr. Robert Weber

ISO JTC 1 / SC 18 Document processing and related communication. WG 3 Open document architecture (ODA) is a potential candidate for consideration.

ISO TC 184 / SC 5 Architecture and communications.

- WG 1 Framework for Computer Information Management (CIM) systems integration is a potential candidate for consideration.
- WG 2 Communications and interconnections is a potential candidate for consideration.
- WG 3 Industrial automation vocabulary is a potential candidate for consideration.

4. Approaches to Determine User Needs

The following general approach was adopted for the identification and recording of user needs:

- Identify the categories of users
- Send invitations to various users
- Request users to submit written inputs: on their needs; on emerging technology; and on emerging regulations
- Prepare a draft based on inputs
- Request users to provide written comments on the prepared draft
- Iterate the draft based on integrating the comments

5. Recommendations

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

1. an Architecture Description Standard; and
2. a Guide to the new Architecture Description Standard.

Second, the planning group recommends that this Action Plan be disseminated to the SESC Working Group Chairs for the Chairs to review and apply the architecture precepts set forth here in the generation and revision of IEEE Software Engineering Standards.

Third, the planning group recommends that this Action Plan be disseminated to other affected disciplines so that the terminology and architectural precepts set forth can be useful in establishing a dialog with the other disciplines to coordinate their responses during the standards development process.

Fourth, the planning group recommends that the working groups established by the above PARs prepare the architecture standard and guide to address the following topics:

How does architecture fit into the system (hardware and software) life cycle(s)?

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?

How to evaluate architecture?

How to accommodate or trade-off requirements which impact architecture

How do systems architecture and software architecture relate?

Lastly, the planning group recommends 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 in mind.
6. The expression of architecture will provide for
   - the conveyance of lessons learned, suitability, etc., enabling a continuous accumulation and application of architectural knowledge; and
   - the communication among key stakeholders.
7. Architecture provides the precepts for design, which guide subsequent development, and style, which promotes system to system consistency.