

Systems Architecture: A View Based on Multiple Impacts

Copyright © 2006 by Tom Gilb.

Iver Holtersvei 2, NO-1410 Kolbotn, Norway, Tom@Gilb.com, www.Gilb.com, +47 66801697

Abstract. In order to properly support the systems engineering process, the systems engineering profession needs to consciously adopt a more productive view of systems architecture. Many existing definitions of systems architecture are too narrow: they focus too much on ‘structure.’ The focus needs to be shifted to the impact on requirements.

Introduction

What is ‘Architecture?’ OK, we know what conventional ‘Architecture’ is in the context of the structure of buildings. So, the question is, ‘What is ‘*Systems Architecture*’ in the systems engineering context?’ There are many different opinions, and many standard definitions. But I want to argue that most of these are missing some essential truths about systems architecture. They seem to universally miss the point that architecture is addressing system requirements: especially the aim to achieve the required performance and resource levels (a set of defined targets and constraints). Instead, they get hung up on the nature of the solutions (and narrow ideas of those solutions, such as ‘structures’).

For example:

Architecture:

Defined As: The organizational structure of a system or component.

Source: Dictionary of Computing Terms, IEEE 630-90, 1990.

Architecture:

Defined As: “the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution.”

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

Figure 1. Two examples of IEEE definitions of !Architecture".

Figure 2 provides another example of a definition of architecture. The problem I have with this definition is that it includes three notions of requirements. Architecture should consist of designs to impact requirements, not the requirements.

So what do I think the definition of systems architecture should be? To answer, I am going to rely on some work, which I did to explain my ideas to Mark Maier. It was in response to the Architecture chapter in ‘The Art of Systems Architecting’ (Maier and Rechtin 2002).

Readers should note that whenever I use the term ‘architecture’ in this paper, I mean ‘systems architecture,’ unless the context clearly shows I am discussing some other type of architecture.

Architecture:

Defined As: A high level design that provides decisions about:

- purpose (What problem(s) that the product(s) will solve)
- function description(s) (Why has it been decomposed into these components?)
- relationships between components (How do components relate in space and time?)
- dynamic interplay description (How is control passed between and among components?)
- flows (How does data or in-process product flow in space and time?)
- resources (What resources are consumed where, in the process or system?)

Source: FAA-iCMM Appraisal Method Version 1.0 A-19, INCOSE Conference CD, June 1999, Brighton UK.

**Figure 2. Another definition of !Architecture."
This contains three requirement notions.**

A Definition for !Architecture"

Here is my core definition for ‘Architecture’:

Architecture

Concept *192 October 26, 2003

The ‘architecture’ is the *set of design artifacts*, which are selected or exist to impact a set of stakeholder requirements, by constraining, or influencing, related systems engineering decisions.

Figure 3. From the Planguage Glossary (Gilb 2004).

In other words, system architecture is a level of design best placed to efficiently control all other engineering decisions; so that the totality of stakeholder requirements are best served.

My definition of ‘Architecture’ has the following intents:

- to bring in the concept that architecture is related to multiple requirements, and must be judged in terms of its satisfaction, and optimization degree, for multiple performance and resource levels (targets and constraints)
- to avoid the notion that architecture is performed only in one instance; it can exist and have evolved, even in a ‘new’ system. (In fact, there can be an overall architecture that constrains any development of any systems within its scope. A systems architecture does not necessarily apply only to one ‘system’. Also if systems are being integrated, you might have to resolve conflicts amongst different systems architectures: that is, you could have more than one existing systems architecture.)
- to avoid the notion that architecture is necessarily formally specified: ‘Architecture specification’ may, or may not, exist.
- to differentiate architecture from other engineering design, by invoking the notion that architecture is distinguished by the agreement that it has the power to constrain the

decisions of all other systems engineering levels.

Interpretations of Terms used in the Definition

Because readers will have so many different interpretations, I will define some of the terms, which I use in my definition of 'Architecture.' I use these terms as follows:

- **Design Artifacts:** Design artifacts are the design concepts. They can be extremely broad in scope covering everything imaginable and discernable, which is intended to satisfy requirements, and which is intended to constrain other design, operational environment, or life cycle activity. In particular, design goes way beyond the traditional notion of structure, and organization. For example, it includes notions of agreements and contracts, social mores, and motivation: all of which never seem to get mentioned in the conventional definitions. It is also intended to cover all discernible mechanisms, which are operating at the architectural level, no matter who selected them, when they were selected, or if the formal 'architects' are aware of them.

The notion intended is that the architecture is seen as a 'set' of arbitrarily different designs (literally, *any set* of designs) for impacting or controlling the attributes of a system.

- **Requirements** consist of the following requirement types:

- function (*what* the system must do)
- performance (*how well* it must perform its functions)
- resource (*how much* it costs to perform its functions – development and operational resources)
- design constraints (*any required design*)
- condition constraints (*any required conditions*)

Requirements dictate the systems architecture. Requirements have characteristics that influence the design decisions. They are an interpretation of the stakeholders' needs. Of course, requirements can never be absolutely mandatory; there must always be some means of determining current requirement priority, resolving conflicts, and making hard choices.

Requirements are key inputs into the architecture engineering process. The architecture engineering process determines, articulates, and handovers to system builders, the appropriate architecture; that is, the architecture with the certified system characteristics to meet the system requirements.

- **System:** The system is any arbitrarily delineated system or sub-system that anyone chooses to study or deal with.

- **Stakeholder:** A stakeholder is any individual, organizational grouping or other entity, internal or external to a specific system, which observably has requirements regarding the system.

- **Systems Engineering Decisions:** Systems engineering decisions are decisions output by any systems engineering process about any notion of design artifact.

Narrow Definitions for !Architecture"

I consider many existing definitions for systems architecture are too narrow. They frequently focus on:

- **structure** (MIL-STD-498, Maier and Rechtin 2002 p285): This term is commonly used to define architecture. Even in civil architecture, it is, at best, one category of the architecture. In systems engineering, it is practically, but not totally, irrelevant. It hides the more central notion of a 'design artifact', which is something that determines system properties or enables them. This point is also made by IEEE Architecture Working Group (Maier and Rechtin 2002 p285-6).
- **components, interfaces & connections:** These terms fall into the same category as 'structure,' they describe specific, but narrow, classes of design artifacts. This in practice leads to the *exclusion* of the more general concept of 'anything, which satisfies the requirements'. It certainly does not include concepts like training, operator selection, motivation, human communication, contracts, policies and other 'non-hardware' designs, which can be every bit as dramatic in influencing the impact on the system requirements.

Good architects for building structures are much more than one-dimensional 'structural' architects. So it is strange that so many systems people take a narrow view, and corrupt the traditional understanding of architecture. 'Structural' must be understood as *only one subset* of systems architecture.

Architectural Distinctions

I want to take this opportunity to make some architecture distinctions. My reason is to make it quite clear that 'architecture' is not something necessarily 'explicitly created by an architect'. Interesting 'Architecture' distinctions include:

- **Perceivable Architecture:** the architecture, which is somehow directly or indirectly perceivable in a real system, as determining the range of performance and cost attributes possible. This applies regardless of who, if anyone, consciously specified the architecture design artifacts.
- **Inherited Architecture:** architecture, which was not consciously selected at a particular level of architecture activity, but was either:
 - . incidentally inherited from older systems,
 - . accidentally inherited from specified design artifacts, specified by architects, managers or engineers.
- **Specified Architecture:** the formally defined architecture specifications at a given level and lifecycle point, including stakeholder requirements interpretation, architecture specification, systems engineering specification done by this architecture level, certification criteria, cost estimates, models, prototypes, and any other artifact produced as a necessary consequence of fulfilling the architecting responsibility.

The design artifacts may be selected consciously, traditionally, accidentally or

unintentionally, - even foolishly, by anybody – including cultures, legal systems, political systems, and nature – even the formal ‘architect’. But the point is that they are in fact in existence in either a real system or a model of such a system. The selection is not necessarily a conscious act for formal engineering, but the design artifact is observably in place and in force – irrespective of its history.

The design artifacts, once chosen or already in place, constrain future decisions about other designs. In fact the conflict can get so severe that past design decisions might have to be reworked in order to accommodate essential new designs.

Conclusions

The benefits of adopting my definition for ‘Architecture’ include:

- we become more focused on the outcome of the architecture, and less on its technical content
- we are more flexible in meeting complex performance and cost requirements - we can more freely swap to better architecture in order to meet requirements - we are not so stuck in the design technology itself
- we become aware of the many competing levels of performance and cost, which we must respect, as we build the architecture solution

Too many of the existing definitions of ‘Architecture’ focus on the ‘means,’ (such as, ‘structure’), rather than the ‘ends,’ (the requirements). We shouldn’t make the mistake of using such narrow, one-dimensional ‘structural’ definitions for systems architecture. We must move towards a definition that recognizes the purpose of systems architecture: to impact the requirements by constraining design decisions.

References

Maier, Mark W., and Eberhardt Rechtin. 2002. *The Art of Systems Architecting*. 2nd Edition. ISBN 0-9493-0440-7. 303 pages.

Gilb05: Gilb, Tom, Competitive Engineering, [A Handbook For Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage](#), ISBN 0750665076, 2005, Publisher: Elsevier Butterworth-Heinemann.

Author Bio

Tom has been an independent consultant, teacher and author, since 1960. He mainly works with multinational clients; helping improve their organizations, and their systems engineering methods.

Tom's latest book is 'Competitive Engineering: A Handbook For Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage' (Summer 2005).

Other books are 'Software Inspection' (with Dorothy Graham, 1993), and 'Principles of Software Engineering Management' (1988). His 'Software Metrics' book (1976, OoP) has been cited as the initial foundation of what is now CMMI Level 4.

Tom's key interests include business metrics, evolutionary delivery, and further development of his planning language, 'Planguage'. He is a member of INCOSE and is an active member of the Norwegian chapter NORSEC. He participates in the INCOSE Requirements Working Group, and the Risk Management Group.

Email: Tom@Gilb.com

URL: <http://www.Gilb.com>

Version Nov 9 2005