Architecture description in-the-large*

Rich Hilliard
r.hilliard@computer.org

Abstract

Successful Architecting in-the-large, namely across various architecture genres and domains of application, will require Architecture Description in-the-large. This position paper addresses the issues of the workshop from insights of Architecture Description, as those ideas have evolved from Software Architecture. It is a heretical or minimalist approach maintaining that for the most part Architecting is Architecting, and that the differences across domains and genres derive from the key architectural drivers, namely: Stakeholders and Concerns. In that context, I further argue what is needed to better address stakeholder and concerns in context.
Keywords: *architecture genres, architecture viewpoints, architecture frameworks, system stakeholders and system concerns.*

1. Introduction

This paper presents the perhaps heretical ideas that:

(I) *architecture genres* such as Enterprise Architecture, System-of-Systems Architecture, Systems Architecture, Software Architecture, and others yet to be discovered and named, are pretty much the same stuff; and

(II) to the extent that architectures in these genres vary, the source of this variation is easy to isolate and express.

When I use the term *architecture*, I distinguish two senses of that term: **architecture**$_1$ as a discipline (the art or science of Architecting) and **architecture**$_2$ as an intellectual product resulting from the application of that discipline to a system or other entity of interest.

I come to these heretical ideas based on 30 years of experience with systems and starting from the perspective of *Architecture Description*: what and how one writes down things about architectures. Admittedly, Architecture Description is one narrow bit of Architecting, but the community now has years of experience there. Perhaps it can shed some light on these larger challenges?

2. On the Workshop Questions

This section does a walk-through of the Workshop Questions (taken verbatim from the CfP but re-ordered for this presentation), and provides short answers to them that sketch the main elements of my argument.

*Position paper presented at the workshop on *Exploring Enterprise, System of Systems, System, and Software Architecture* at WICSA/ECSA 2009 (Cambridge UK, September 2009).

What are enterprise, system of systems, system, and software architectures?

It seems like the systems and software communities have spent the last 20 years trying to define **architecture**$_2$. The SEI website offers us an extensive list from the Software Architecture community [see How Do You Define Software Architecture?]. Those definitions vary with regard to their specificity to software, the constituents involved, and many other parameters. The following definition seems equally applicable to software, systems, systems-of-systems, enterprises and other aggregations of interest:[1]

> **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. [6, 7]

The key ideas in the IEEE 1471 definition are these:

(1) **Architecture**$_2$ embodies the *fundamental* concepts about a system, including its components, their relationships and governing principles. More recently, this has also been expressed this in terms of "the significant decisions about" a system [P. Kruchten].

(2) **Architecture**$_2$ recognizes the role and influence on an architecture of the *environment* in which a system is embedded. Some even say this recognition of context is what distinguishes **Architecture**$_1$ from Design, but that is a debate for another time.

(3) **Architecture**$_2$ is not merely about the static arrangment of a system's "parts" but its *evolution*—whether we are talking about a constructed system (as in this workshop) or a natural system (e.g., a cell) and the principles governing that evolution.

Each of these key ideas about **Architecture**$_2$ has consequences for what we say about the discipline of **Architecture**$_1$. More about this below.

What do these architectures have in common and what makes them different?

This two-part question has one answer: *Stakeholders and Concerns*. The way we come to understand any complex system in its environment is to look at and understand the influences on that system. One way to "operationalize" this understanding it to identify, *Who cares about this system?* and *What exactly do they care about?* This is the basis for stakeholders and concerns. It is a truism to say that complex systems have one or more stakeholders—often with conflicting concerns. For concreteness, let me quote the current definitions [ISO/IEC WD4 42010]:

> **stakeholder (of a system)**: individual, team, organization, or classes thereof, having concerns with respect to a system
>
> **system concern**: area of interest in a system pertaining to developmental, technological, business, operational, organizational, political, regulatory, social, or other influences important to one or more of its stakeholders

[1]For the remainder of this position paper, I am going to use **system** as a shorthand for a long list of aggregations of interest to include the ones listed above and others such as product lines, product families, software applications, etc. The moral should be anything you are interested in can be a "system". I won't recite the list again, unless there are reasons to distinguish its members.

($\mathcal{I}$) The idea that complex systems have multiple stakeholders and concerns unifies the architecture genres we are discussing; the particular stakeholders and concerns which are relevant within each genre determines the ways in which they are different.

Diverse stakeholders and concerns contribute to what what makes a system complex, and it is also the basis for why we understand **Architecture**$_1$ as multi-disciplinary:

> "The ideal architect should be a man of letters, a skillful draftsman, a mathematician, familiar with historical studies, a diligent student of philosophy, aquainted with music, not ignorant of medicine, learned in the responses of jurisconsults, familiar with astronomy and astronomical calculations."
>
> — Vitruvius, *De Architectura* (25 BC)

WARNING: The next few answers promise to be pretty boring—they are merely applications of ($\mathcal{I}$) as stated above. Stay tuned!

Where does one type of architecture end and another begin (e.g., how do you know when you have moved from talking about enterprise architecture to system of systems architecture)?

The boundaries of the genres can be understood in terms of the stakeholders of systems within each genre and their concerns. Moving from one genre to another it is worth asking, *What stakeholders carry over between genres? How do stakeholders interests shift? What stakeholders are unique to each? What concerns carry over? What new concerns arise or are reinterpreted?*

What do we mean when we say "we're designing" one of these architectures (e.g., what activities are we performing, what types of requirements are we dealing with, what types of design decisions are we making)?

I would claim that the activities themselves are not very interesting: I admit to being a process cynic, for the reasons discussed in [4]—an eloquent argument against method in science and equally applicable to prescribing methods in Design and Architecture.

What commonalities exist among activities that are performed for each type of architecture (e.g., specification, evaluation)?

There are some basic activities of Architecting. It would be a mistake to prescribe a process (for the reasons alluded to in the previous), but there are some general characteristics I think any form of Architecting would take:

- Bound the problem
- Understand the context, environment in which problem and solution are situated
- Work the problem within those bounds in terms of:
  - Requirements analysis and specification
  - Investigate elements of solutions
  - Modeling and analysis

– Check and integrate the details

- *Iterate as needed*

Hofmeister et al, discuss a generic process for Software Architecting, aside from some terminology, there is no reason to believe the same archetype would not work for other domains and genres, or Architecting in general. I'm perfectly happy with a generic Architecting process along the lines of their generic model [5]. My version of that process shown below:

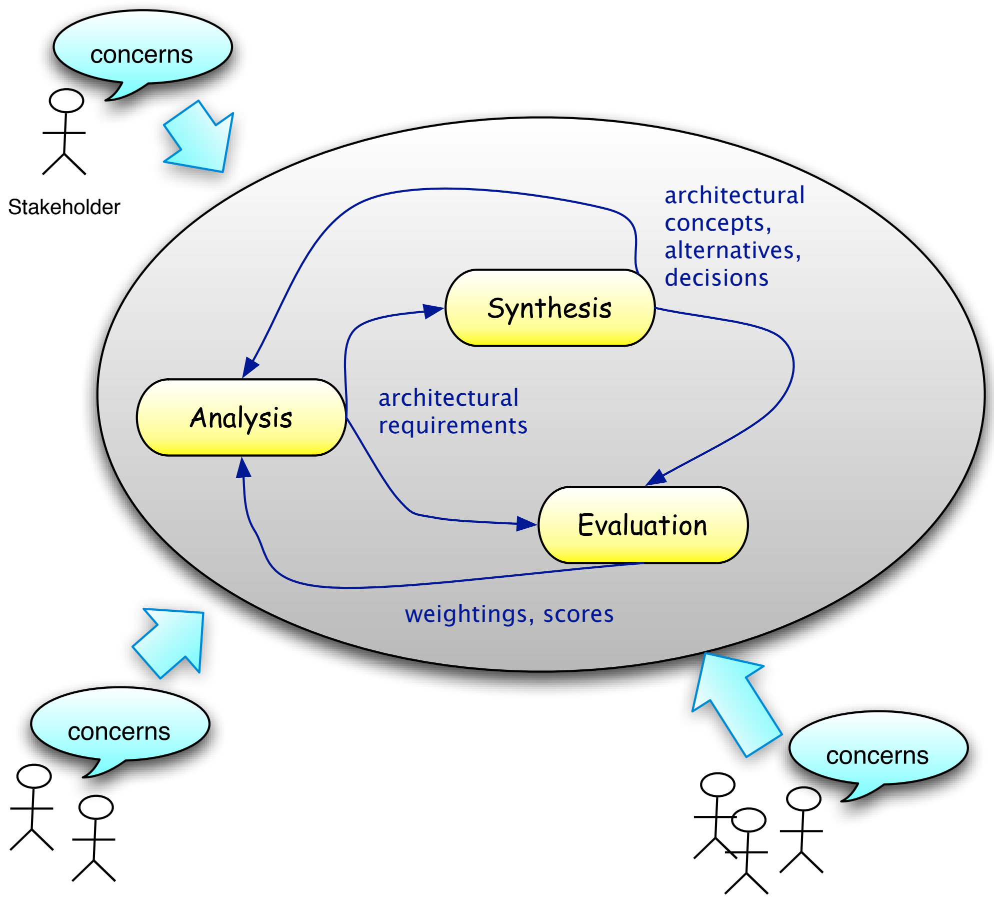

Figure 1. A generic model of Architecting

In this model the Stakeholders and Concerns *shape* both the problem space being Analyzed, and the solution space being synthesized. Each analysis, architectural decision or evaluation step potentially identifies new stakeholders and concerns. True differences in activities arise when you get into specialities of the domain of application, the technologies involved, etc., e.g., certifying a aircraft for flightworthiness.

What do architects for one type of architecture need to know about the other types of architecture?

As a rule of thumb, if one drew a picture of how these genres relate (perhaps cross-cut vertically by domains of application), one might say downward information flow (e.g., enterprise to system, system to software) *bounds the problem*, whereas upward information flow (e.g., system to system of systems or software to enterrprise) *constrains the solution*. Of course often who constrains whom is determined by a postmark—who got there first?

What are the relationships among the documentation artifacts of the different types of architectures?

There is probaly a small set of possible relationships, reflecting the types of needed knowledge posed in the preceding question, such as:

$system_A$ constrains $system_B$
$system_C$ buildsOn $system_D$
$system_E$ integratesWith $system_F$
$system_G$ uses $system_H$

What is the impact of design decisions made for one type of architecture on the other types of architectures?

See previous.

What are the challenges (e.g., requirements, design, management, funding, and authority) unique to each type of architecture?

It's not clear to me there are any challenges peculiar or particular to all the systems within a genre.

3. So What? or What Next?

The observation that Stakeholders and Concerns reflect basic variations (if any) across architecture genres is not particularly deep—I hope it is pretty obvious! Still, the observations above and in $\mathcal{I}$ suggest a way forward. This section makes some concrete recommendations in that direction.

If we take the above seriously, then we need good ways to identify, model and analyze stakeholders and their concerns.

Find the stakeholders and concerns within a genre, domain or system space. If Software Architecture has been any guide, there is no reason to assume the existence of or look for a single solution across a genre (or domain or space)—at best we may find some typical or recurring cases. During the development of IEEE 1471 there was no agreement on a single set of viewpoints suitable for all software development. Then, as now, there are viewpoints such as *4+1*'s Logical, Deployment, Process and Implementation; SEI's *Views & Beyond*: Component and Connector, and Module with wide applicability. Yet, these common viewpoints fail to adequate capture critical concerns which may arise in areas such as Reliability, Scalability, Safety or Information Assurance. For systems with these concerns added viewpoints are often necessary.

I would expect the same within Systems Architecture, Enterprise Architecture and other genres [8]. There may be some "typical" concerns such as Planning or Resource Management. At the same time, we may find that, e.g., hospital systems, systems of systems, and enterprises may have more in common – due to their shared domain of application than two arbitrary systems within the same genre.

If we truly believe there is commonality within and among genres, domains and system spaces, let's start a **Concerns Catalog**, and possibly a **Stakeholders Catalog** to capture recurring themes. Zachman is not a bad start for enterprises, but misses specialized concerns (again such as Safety, Security, IA). DODAF is particularly unhelpful in this regard because despite that fact that it enumerates many architecture products or models for use, it never "indexes" these products for use. It never tells us why: *What concerns can each of its products address? Do I need architecture product XV–5 for my current project?* A general solution would build on existing work on taxonomy of quality attributes and recent work in goal-oriented requirements specification and "concern-oriented modeling" from the Aspects community.

If we can get a handle on (typical) Stakeholders and Concerns within each genre (or domain or system space), we might be able to begin to understand the boundaries and information flows between genres. Finding the same stakeholder in both may tell us there will be common concerns in each (due to the nature of her job). Understanding the relations *between* Stakeholders (organizationally) may offer insights across genres and domains. This suggests **Architecture**$_1$ is a lot more sociological than how it is usually written about. This quote reminds us we have a ways to go, and probably always will:

> Unfortunately, in contrast to building architectures, we have yet to agree on what the appropriate software structures and views should be and how to represent them. One of the reasons for the lack of consensus on structures, views, and representations is that software quality attributes have matured (or are maturing) within separate communities, each with their own vernacular and points of view.
>
> — Mario Barbacci [1]

But the Barbacci quote also points out an opportunity: of capturing languages, notations and models suitable to diverse community when dealing with *their* concerns. Most technical communities have requirements and standard practices independent of and prior to the arrival of the Architects. Often, this means existing models and notations are in use with respect to certain concerns. (Safety is a classic example, here.)
Remember mega-programming? The evolution of Programming was from raw assembly code to routines to modules to components. One could aspire to a similar progression in Architecture Description: from individual models that worked for describing a particular system, to patterns (and the Rule of 3), to architecture viewpoints (precedented approaches to framing certain system concerns), and reasoning frameworks [2] (reusable approaches to addressing certain quality concerns) to architecture frameworks (coordinated sets of viewpoints). Viewpoints and frameworks can be repositories of such commonality.

> A viewpoint determines that (1) when addressing concerns like this, (2) for stakeholders like that, use (3) notations and models like this.

In addition to ordinary syntax and semantics of its models and notations, a viewpoint can include additional guidance, such as suggested tasks to perform, heuristics and patterns to help get the job done, and reasoning frameworks appropriate to its models. The viewpoint construct recognizes the multi-disciplinary nature of architecture discussed above and suggests a concrete container for packaging up knowledge from a discipline in a convenient form.

A further level of "reuse" is suggested by frameworks—coordinated sets of viewpoint and correspondence rules between them. In some future Nirvana, we could even imagine composition by mixing and matching framework elements (defined in a common format) to address new systems across genres, domains and system spaces [3].

References

[1] M. R. Barbacci. Analyzing quality attributes. Column in SEI newsletter, The Architect, March 1999.

[2] L. Bass, J. Ivers, M. Klein, and P. Merson. Reasoning frameworks. Technical Report CMU/SEI-2005-TR-007, Software Engineering Institute, Carnegie Mellon, 2005.

[3] D. Emery and R. Hilliard. Every architecture description needs a framework: Expressing architecture frameworks using ISO/IEC 42010. In R. Kazman, F. Oquendo, E. Poort, and J. Stafford, editors, *Proceedings of the 2009 Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture (WICSA/ECSA 2009)*, pages 31–40. IEEE Computer Society Press, 2009.

[4] P. Feyerabend. *Against Method: Outline of an Anarchistic Theory of Knowledge*. NLB, 1975.

[5] C. Hofmeister, P. Kruchten, R. L. Nord, H. Obbink, A. Ran, and P. America. A general model of software architecture design derived from five industrial approaches. *The Journal of Systems and Software*, 80(1):106–126, 2007.

[6] IEEE. *IEEE Std 1471–2000, IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*, October 2000.

[7] ISO. *ISO/IEC 42010:2007, Systems and software engineering — Recommended practice for architectural description of software-intensive systems*, July 2007.

[8] M. W. Maier, D. Emery, and R. Hilliard. ANSI/IEEE 1471 and systems engineering. *Systems Engineering*, 7(3):257–270, 2004.