Rich Hilliard’s Remarks

Thank you, Paris, for the good words. Thank you to the selection committee for choosing us. And thank you to the community for all those citations.

My co-author David could not be here but I’ll give him the last word.

I’m especially happy about this for a couple of reasons.

First, because of all the communities I’ve participated in, from Free Software to Software Engineering to Balkan folk dance, the Software Architecture community has a lot of great people, many of whom have become friends and colleagues, so this recognition is especially meaningful.

Second, the award signals the continuing impact of the Architecture Description standard that our paper was written about; originally as IEEE 1471 in 2000 and now as ISO/IEC/IEEE 42010.

I have to confess I was never comfortable with the title of the paper; it was David’s suggestion. Of course it is completely logical and good practice: every architecture description needs to be built on an integrated set of viewpoints to be meaningful and useful to architects and other stakeholders. In the standard, we ended up reserving the term architecture framework for reusable, stand-alone assets, but the point holds.

Looking back, I realize we brought a language designer perspective to this standard (both David and I were involved in the Ada programming language effort — we are
that old!). Instead of adding new features to the standard based on the popular trends and buzzwords, we tried to provide minimal constructs that allowed users to have those features. Sometimes this approach was too subtle for most users and has led to misunderstandings. I’ll give two examples:

1. The notion of **Concern** in the standard was introduced exactly and only in the spirit of Dijkstra’s separation of concerns—an area of interest that you might be able to articulate and study independently of others. We thought it was a simple idea but it has been misunderstood variously as Risks, Stakeholder Needs, Architecture Requirements, ..

2. In Enterprise Architecture, a number of architecture frameworks are depicted as 2-D grids. Suggestions have been made recently that we need to “add grids” to the standard to reflect current practices. But, as proposed in our paper, we provided a general mechanism for grids and any other arrangements of architecture frameworks. We generalized the notion of correspondences/rules not just to relations like traceability and refinement, but to arbitrary *AD elements* including Viewpoints, Model Kinds, Stakeholders and Concerns. With these correspondences and rules one can invent all sorts of frameworks, and model the ones that already exist.

Despite some misunderstandings and confusions, the Standard has stood up pretty well. The notion of viewpoint and data template for specifying a viewpoint has found its way into use.

Although we were not thinking about architecting processes, choosing to focus on organization and content of the product, Architecture Descriptions, the principles and requirements in the Standard work pretty well both for High Ceremony processes and Agile approaches — I think because we focused on minimal requirements.

As I think back on that period of time, I was lucky enough to have some great collaborations — or *collusions* as we say in the US now. These are reflected in our paper and more importantly in the Standard.
David Emery’s Statement
Thank you for recognizing the paper Rich and I wrote. My apologies for not attending. But Rich said he’d read some remarks from me, and I couldn’t resist the opportunity to preach to the faithful from my retirement home.

Since this is recognition of an accomplishment in the past, let me go back much further. 30 years ago, our group at MITRE was pondering the systems we were supporting. There were many failures, and a few successes. One thing we noted about successes was that they tended to have some representation of the entire system that you could read, understand and reason about. We started calling this “architecture”. In the early 1990s, the influential Perry and Wolf paper came out, CMU was talking about similar things, and Maier and Rechtin published their Art of Systems Architecting. Then came Philippe Kruchten’s “4+1” paper, which provided a concrete notion of software architecture framed by relevant concerns.

Thus there was an emerging notion of ‘architecture’ even without a consensus on how to define it or how to do it. Rich can talk to the history of IEEE 1471/ISO 42010 that built that consensus.

But my point is that the imperative for architecture still exists: Successful systems have a core. That core can and must be captured. That core (from the “architecture description”) needs to allow you to reason about the system internally, about both functional and non-functional requirements and behaviors. And it also needs to allow you to reason about the system in use, what the 1471/42010 definition calls “in its environment”. We need to keep this utility in mind, and resist the efforts by bureaucrats who say, “Here are a dozen mandatory architecture products that have no real user demand, answer no stakeholder concerns, and provide no utility in representing the system as a whole. But we insist you produce this.” Unfortunately, that describes much of what are called “architecture frameworks”. Architecting occurs early in the overall development of a system, where a small group of people can make a big contribution to the success – or the failure – of a system. Frameworks allow us to capture both best
practices and worst practices in architecting. Let’s learn from the mistakes, and resist attempts to impose frameworks that don’t actually help us build successful systems.