The Pragmatic Architect

Return of the Pragmatic Architect

Eoin Woods

Abstract: There are many types of architect working in the software industry, but when we consider the breadth of their work and their primary expertise, we find that they can be organised into three major groups, the enterprise architects, the application architects and the infrastructure architects. Knowing which group an architect falls into helps us understand their expertise and what to expect of them.

Welcome back to The Pragmatic Architect. Frank Buschmann founded this column in 2009 and it ran until the middle of 2013, when he had to step down as editor. My name is Eoin Woods and I'm taking over as column editor.

Before going further, I must congratulate Frank on developing such a thought-provoking series of articles. I'm excited by the opportunity to continue in the same direction but I'll need your help to do this—more on that later.

So let me introduce myself. I'm a software architect and have been working in the software engineering field since 1990. I spent most of the 1990s working on system software products, but since about 2003, I've worked as a software architect for financial services firms, mainly designing large transaction processing systems. I'm very much a practitioner rather than a researcher but I do have a keen interest in all things relating to software architecture, including its research field.

My goal for this column is to provide a series of practical, easily digestible articles on topics that will interest working software architects. I plan to include introductions to emerging technologies and architecture innovations, lessons learned about software architecture methods and techniques, and personal accounts of real projects and architectures, particularly those where hard lessons were learned.

Characterizing the Pragmatic Architect

Back in his first-ever Pragmatic Architect column, "Introducing the Pragmatic Architect," Frank provided a definition of a pragmatic architect. I'd urge you read it again, but in summary, he concluded that effective software architects balance three competing approaches: the *abstract*, the *technical* and the *deep* (or the roles of "architecture astronaut," "techno-geek," and "architecture dwarf," if taken to extremes!).[1] At times, an abstract approach is necessary to understand the overall picture and communicate its implications to wide groups of stakeholders. But to make those abstract ideas real, you need to zoom down to reality and do concrete design work using real technologies. In some cases, you must dive into the minutiae of a specific problem and understand all the technical detail to help a team address a problem or ensure that a critical technical point is really understood. It's balancing such roles and concerns that makes a software architect's job so challenging.

Another intriguing aspect of technology architecture is the sheer diversity of work that people with "architect" in their job titles perform. As a reader of *IEEE Software*, I suspect that you're involved in software development, but not all architects are, so it's interesting to understand the other types of architects we might meet.

What Makes an Architect Anyway?

I've stopped counting the architecture job titles I've encountered over the years: enterprise architects, network architects, storage architects, software architects, application architects, solution architects, Oracle architects, data architects, information architects, consultant architects, user-interface architects, and so on. The list seems to go on forever, which makes it difficult to define. Are all these people really architects?

As I considered this, I realized that I needed a clear definition of the characteristics of an architecture position. So I wrote one that identified six key characteristics of an architect.

- **Design-centric.** Architecture is a design-oriented activity. An architect might design something quite concrete like a network, or something less tangible like a process, but design is core to the activity .
- **Leadership**. Architects aren't just technical experts in their areas of specialization, they're technical leaders who shape and direct the technical work in their spheres of influence.
- **Stakeholder focus**. Architecture is inherently about serving a wide constituency of stakeholders, balancing their needs, communicating clearly, clarifying poorly defined problems, and identifying risks and opportunities.
- **System-wide concerns**. Architects are concerned about an entire system (or system of systems) not just one part, so they tend to focus on systemic qualities rather than detailed functions.
- **Lifecycle involvement**. An architect might be involved in all phases of a system's lifecycle, not just building it. Architectural involvement often spans from establishing the need for a system and continues throughout its lifecycle to eventual decommissioning and replacement.
- **Balancing concerns**. Finally, across all of these aspects of the job, there is rarely a right answer in architecture work (I've often joked that architecture normally involves selecting the least worst option!) Owing to the number of concerns that must be balanced, a truly optimal solution often does not exist.

By considering whether someone's job includes these attributes, I can mentally check whether they're really doing architecture work .

Classifying the Architectural Species

Even using these attributes, I still had too many different types of architects to form a classification, so I started to think about that. It seemed that a network architect and an enterprise architect have more in common than a network architect and a software architect but what made them more or less alike?

Of course, I wasn't the first person to ask this question. IBM Global Services has long used the following classifications: enterprise, application, information, infrastructure, integration, and operations architect (www.opengroup.org/architecture/0310wash/presents/Claudio_Cozzi-IBM_Architecture_Practice.pdf). IASA views the IT architecture profession as being made up of enterprise, software, infrastructure, information and business architects (www.iasaglobal.org/Document.asp?MODE=DOWNLOAD&DocID=489). And Martin Fowler famously divided architects into the species *Architectus Reloadus* and *Architectus Oryzus*; the former being an all-seeing architect surveying his domain from on high (a reference to the film *The Matrix*), while the latter works directly in and with teams in a much more collaborative manner.[2] While useful in their own ways, none of these groupings seem very definitive, so I thought about the characteristics of the different the architecture roles that I've encountered.

I soon realized that two aspects of an architect's job dictated how I dealt with them: whether they primarily had expertise in the problem domain (business) or solution domain (technology) and whether they were concerned about one or many systems. These characteristics shape how you interact with architects because they drive their primary concerns and level of abstraction (which shapes factors like their style of involvement and time horizons).

These characteristics allow me to classify architects into three groups: *enterprise*, *application*, and *infrastructure*. Each group has synonyms and subspecializations, but all architects seem to fall into one of the three groups. Figure 1 illustrates the groups and their relative positions.

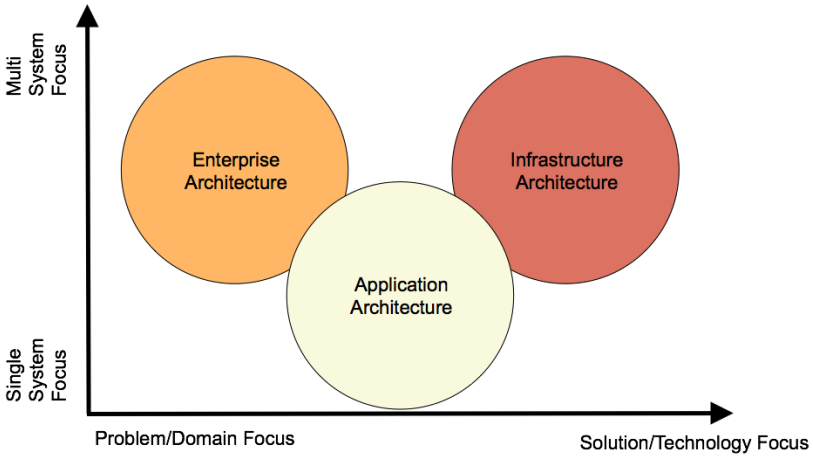

Figure 1. Architects in context. We can classify architects into three main groups: enterprise, application, and infrastructure.

Enterprise architects have a business domain specialization rather than a technical specialization and their primary concern tends to be aligning technology with business needs. They have a cross-system perspective, being responsible for all of an organization or business unit's systems, rather than just one or two systems. Subspecializations include enterprise application architecture, data architecture, and business architecture, but they all have a focus on the domain and a cross-system perspective. This perspective means that they tend to be interested in applications at the "black box" level (that is, their responsibilities, integration, and interfaces) rather than their internal details. And they tend to take quite a long-term strategic view of their work rather than being involved in day-to-day delivery timelines.

In contrast, *infrastructure architects* have a technical rather than a problem-domain focus, so although they often need a broad understanding of business area priorities, their deep expertise is in one or more of the technology domains. They also have a cross-system perspective, providing technologies to many applications and often have quite a formal service-based relationship with them. Infrastructure architects nearly always have a strong technical specialization such as network, storage, computer servers, or middleware technology. And they also have a long-term, strategic view because they're concerned with simplicity, standardization, stability, cost management, roadmaps, and vendor management.

Sitting in the middle are *application architects*, who need a balance of domain and technical knowledge to build effective systems. They tend to focus on a single system (or a small group of related systems) and are responsible for the function and internal design of "their" system, particularly system-wide concerns such as quality properties (performance, security, modifiability, and so on). Their interest in other systems tends to be as black boxes that they connect to or depend upon. They usually have a short- to medium-term view, compared to other architects, knowing where they think they're going but expecting lots of change along the way.

Of course in a healthy organization, these groups communicate regularly to align their work, as Figure 2 shows.

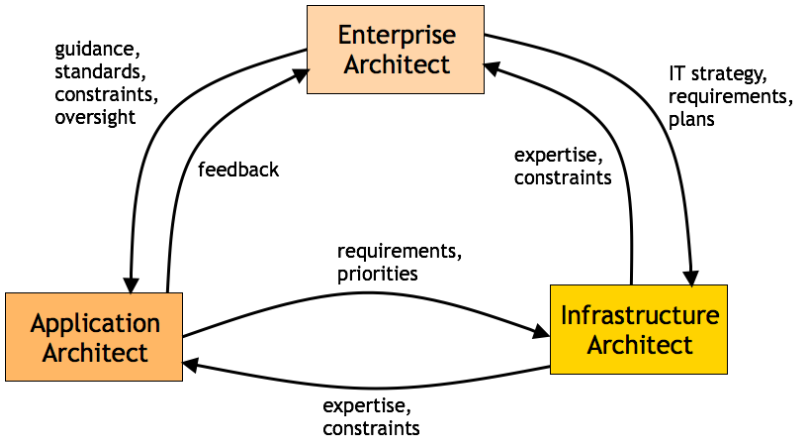

Figure 2. Interarchitect communication. In an effective organization, different types of architect communicate regularly to align their work.

Enterprise architects provide a context for the work of application and infrastructure architects via standards, strategy, roadmaps, and plans, while infrastructure architects in turn must meet the requirements of the application architects. Infrastructure architects provide technology expertise and an understanding of the technical constraints for application architects, and design the infrastructure platforms that the applications rely upon. Application architects are guided by the context enterprise architects provide and the infrastructure architect's technical expertise but ultimately, it's the applications architects who are responsible for delivering useful systems.

So we've set the scene and discussed the kinds of architects that we might meet in our work. In future columns, we'll explore a wide range of topics relevant to practicing architects, including the relationship between architecture and code, how risk drives architecture, pragmatic approaches for architecture documentation, handling architectural decisions, and much more.

Finally, I mentioned earlier that I need your help to ensure that the column covers topics of interest to you, the reader. Please do get in touch and let me know what you'd like to read and perhaps even consider authoring or coauthoring a column if you're an expert on a relevant topic. I hope to hear from many of you in the coming months and look forward to our journey through pragmatic architecture.

References

1. F. Buschmann, "Introducing the Pragmatic Architect," *IEEE Software*, vol. 26, no. 5, 2009, pp. 10–11.
2. M. Fowler, "Who needs an architect?" *IEEE Software*, vol. 20, no. 5, 2003, pp. 11–13.

Acknowledgements

Thank you to Chris Cooper-Bland, Andy Longshaw and Nick Rozanski for many helpful conversations and comments on earlier versions of this material.

***EOIN WOODS** is a software architect at a major European investment bank. Contact him at eoin.woods@artechra.com.*