Learn the discipline, pursue the art, and contribute ideas at
www.ArchitectureJournal.net
Resources you can build on.

THE ARCHITECTURE JOURNAL™
Input for Better Outcomes
Journal 15

The Role of an Architect

Microsoft®

Contents

THE ARCHITECTURE JOURNAL™
Input for Better Outcomes
Journal 15

Foreword

Dear Architect,

Welcome to Issue 15 of *The Architecture Journal*, where we take a quick breather from technology, turn the spotlight on ourselves and examine the role of the IT Architect. This has been a fascinating issue to put together, partly because of the different perspectives that many people in our profession bring to the table, but also because of the passion involved in defining what is still a very emerging profession.

We believe this issue of the journal goes beyond our normal boundaries of IT architecture and is applicable to anyone who would like to understand why architects exist, what architects do, why organizations need them, and most importantly, what one needs to know to be one. In short, this is an issue for both the architect and those aspiring to be architects, and as such, this issue should be shared with your colleagues.

The articles you will find in the following pages will talk about skills, responsibilities, experiences, and many other topics related to being or becoming an architect, so it seems appropriate in this introduction to attempt to answer a question that many aspiring architects have asked — *Why do I want to be an architect?*

The obvious answer, and one I hear at practically every aspiring architect event I attend, is quite simply — "if architect appears in my job title I will get paid more." While that is probably true, and in many cases could make the difference in allowing your family to eat prime rib instead of hamburger, fulfilling a monetary requirement does not necessarily address the less tangible goals we all have for ourselves.

If most people want more than just to be paid well, why is money the commonly mentioned benefit to becoming an architect? The answer is that this is probably the only common point. Just as every architect has their own perspective on good architecture, every architect has their own perspective on what makes a good architect and why they want to be one.

Whether you are an IT Architect by title, or someone that is heading that way in your career, we hope that you find the articles useful and insightful for the work that you do every day. We look forward to hearing your feedback about this and previous issues at editors@architecturejournal.net.

Simon Guest

We Don't Need No Architects!

by Joseph Hofstader

Summary

The role of an architect in software development has come under attack of late. On software projects, the title Architect is often ambiguously defined and the value provided by architects is not easily quantifiable. The perception that architects live in an "ivory tower" disassociated from the realities of delivering a software solution contributes to some of the animosity toward those of us with the title.

This article presents a defense of the practice of architecture in software development. It defines the qualities of an effective architect and describes the skills required to succeed in this profession. The article examines widely held perceptions of architects and some of the mistakes that architects make which contribute to negative perceptions. Ultimately, the intent is to show the value good architects bring to a software development effort.

Who Are Those Guys?

In the field of information technology, no title conjures up more raw emotion than Architect. I have witnessed and been involved in many debates over the definition of the term. When I introduce myself at meetings, reactions range from "we're not worthy" to "we do not need an architect"—the former, although friendly, reflecting the lofty image of architects, and the latter implying that an architect's knowledge and skills are irrelevant. Both responses demonstrate a lack of understanding of what architects really do.

At the OOPSLA conference in 2005, I attended a "Birds of a Feather" (BOF) hosted by Grady Booch. The topic of the BOF was his then upcoming "Handbook of Software Architecture." One of the attendees related some negative experiences he had had with architects, both in IT and in construction. One story was about the architect who drew plans for his house expansion. The attendee said that he viewed the drawings with engineering software and the plans were off by a few feet, and that the actual construction could not and did not follow the architect's specification. He was making the point, which I have heard echoed by many qualified individuals, that architects are detached from the reality of delivering tangible results and that their responsibilities should be relegated to the engineers and builders who are fully engaged in product development.

That meeting, and many subsequent conversations with others, led me to wonder what exactly the role of an architect is on a software product and what the characteristics of good architects are. The most concise definition I have come up with is: *The role of the IT architect is to solve a problem by defining a system that can be implemented using technology. Good architects define systems by applying abstract knowledge and proven methods to a set of technologies with the goal of creating an extendible and maintainable solution.*

From this concise definition, we can extrapolate that good architects draw upon a foundation of knowledge to be successful in their role. To "solve a problem," the architect must have a good understanding of the problem domain. To "define a system using technology," implies that the architect has technical acumen. "Abstract knowledge" requires the architect to be able to conceptualize the technical solution. "Proven methods" assumes an understanding of the patterns used to solve similar problems. Figure 1 depicts the key skills of an architect.

The key benefit an architect brings to a project is the ability to apply those skills in the definition and development of a robust software solution. An effective architect is part of the continuum of all phases of a software development project, with skills critical to defining the problem space, such as domain knowledge and the ability to conceptualize a software solution, as well as the ability to define the solution space using appropriate technologies and patterns. The risk of not having an architect actively involved in product development

Figure 1: The key skills of an architect

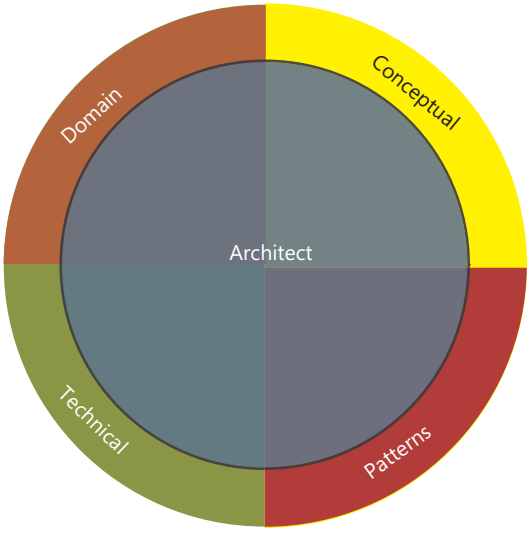
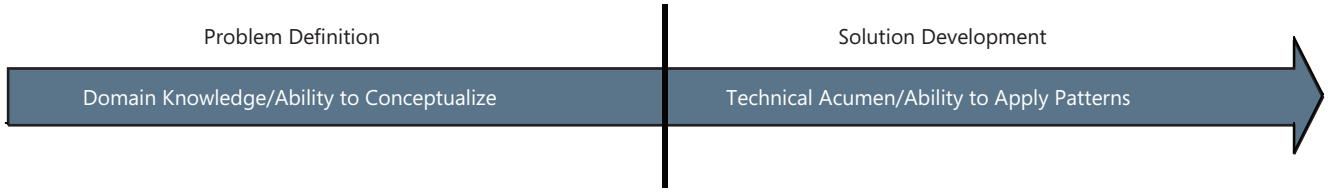


Figure 2: Architect skills in phases of software development



increases the likelihood that the product will take too long to develop or be developed incorrectly. Figure 2 illustrates the phases of a development project where the skills of an architect are applied.

Describing the architectural skills required for a successful project is not as straightforward as it may seem. Many areas, especially technical acumen and patterns, are often debated regarding the level of expertise necessary for an architect. The following sections, divided by problem space and solution space, offer an explanation of each of these skill sets and a rationalization of how these skills make an architect invaluable to a software development effort.

Problem Space

Defining the problem space and ultimately setting the scope of a software solution requires an understanding of what will be built, as well as domain knowledge and a conceptualization of how information will flow through the solution. As the following sections detail, these skills are essential to understanding the purpose of a software solution and communicating the proposed solution to all stakeholders.

Domain Knowledge

The problem domain for a software solution can be horizontal or vertical. A horizontal domain is applicable across industries, like workflow automation. Vertical domains are specific to a particular industry, like telecommunications. Problem domains can be further decomposed into subdomains, or aggregated into larger domains. An example of a subdomain within a vertical domain is network activation in telecommunications. An example of the aggregation of a subdomain into a larger horizontal domain is workflow in an enterprise application integration product.

There are many standards organizations and vertical industry groups that specify standards and protocols that need to be considered when defining a software solution. These organizations can be specific to a vertical industry domain or a horizontal industry domain. The TMForum is an example of a vertical organization that specifies management protocols for the telecommunications industry. The W3C specifies standards for the horizontal World Wide Web domain including technologies like Web services.

The value of domain knowledge is sometimes underestimated by IT managers. I once worked for a telecommunications company whose IT leadership wanted to change the organization from being structured around "centers of excellence" focused on a business domain to being structured with "pools" of resources based on technical skills without regard to domain knowledge. For some of the resources assigned to horizontal domains, like Web development, this model worked well. Many products require Web interfaces and the skills were applicable across verticals. Where the "resource pool" structure failed was in industry specific subverticals, like network activation. Understanding how to provision and activate services requires detailed knowledge of the provisioning process as well as the interfaces, protocols and standards of all network devices that comprise the telecommunications services.

"WHEN I INTRODUCE MYSELF AT MEETINGS, REACTIONS RANGE FROM 'WE'RE NOT WORTHY' TO 'WE DO NOT NEED AN ARCHITECT'—THE FORMER, ALTHOUGH FRIENDLY, REFLECTING THE LOFTY IMAGE OF ARCHITECTS, AND THE LATTER IMPLYING THAT AN ARCHITECT'S KNOWLEDGE AND SKILLS ARE IRRELEVANT. BOTH RESPONSES DEMONSTRATE A LACK OF UNDERSTANDING OF WHAT ARCHITECTS REALLY DO."

Deep domain knowledge often involves a steep learning curve. If the staff on every project is required to learn the intricacies of the domain for every release of the project, productivity is significantly reduced. Assuming features are sufficiently decomposed so that the amount of time to deliver the feature is constant, then productivity suffers proportional to the amount of time spent acquiring domain knowledge. Conversely, maintaining a "center of excellence" around each vertical domain supported by a business can also be an expensive proposition. Development work is seldom evenly balanced throughout a given timeframe and keeping a fixed number of resources assigned to a project that is not actively engaged in delivering solution features can drain productivity on other development efforts.

A balance that is common in technology companies is having an architect be a domain expert and a pool of resources available to different projects. This strategy increases productivity by minimizing the amount of time obtaining domain knowledge within a vertical. It also allows some preliminary research to be done prior to engaging development staff, helping to ensure that the development is consistently productive. This approach provides the company the added benefit of a flexible staffing model, in that development staff can be augmented with contractors without having valuable domain knowledge walk out the door at the end of the contract.

Figure 3: An architect thinking in bubbles

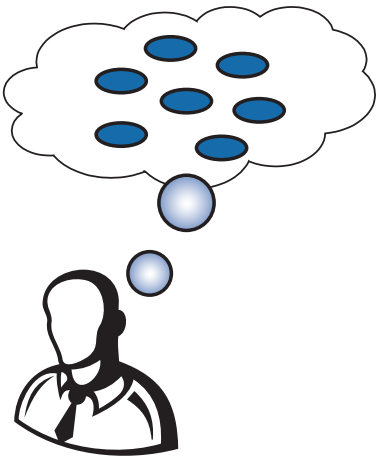
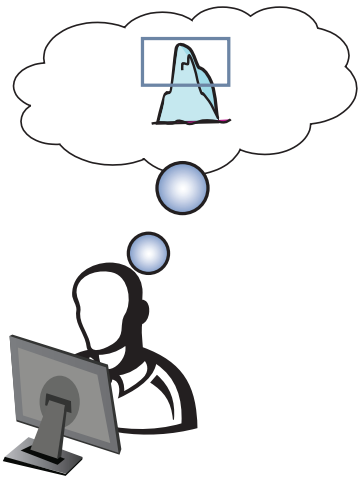


Figure 4: A developer seeing part of the picture



Conceptual Thinking

One of the main responsibilities of an architect is to communicate the solution to technical and nontechnical stakeholders. The ability to conceptualize a software solution is essential to communicating a software solution to stakeholders who care about delivery of functional requirements, and may not know or care about technical details. Defining a conceptual architecture prior to commencing the development of a software solution helps facilitate the feedback loop needed to define the scope of a product and can help determine an initial level of effort and cost estimate for a product.

A conceptual model is the artifact most associated with software architecture. The conceptual model typically shows the components of a software system that will fulfill the functional requirements and where they apply in a software solution (user interface, domain layer, and so forth). The conceptual model is often accompanied by a number of diagrams showing the flow of information through the proposed solution. In the case where the software system consists of components from other products or solutions, the conceptual architecture often contains the protocols that will be used to access different parts of the solution.

Applying the correct level of granularity is the main challenge in defining the conceptual model. The conceptual architecture should not contain any references to a particular platform or technology, other than protocols used to access subsystems. I once took over as architect of a project that was stalled in "analysis paralysis" for over a year. As I was reviewing documents to get up to speed, I noticed that the conceptual architecture consisted of a number of boxes containing the names of technologies with no reference to system functionality. After reviewing that document I understood why the system could not be developed: There was no mention of the required features, making it hard to understand what needed to be developed.

Solution Space

It is in the area of defining the solution space that opposition to architecture is most obvious. Developers will usually accept the architect working in the problem space, but may be resistant to having the architect define the solution space. In many instances, developers have valid arguments about architects meddling in the solution space, especially if the architects have not kept their technical knowledge up-to-date.

A colleague of mine illustrates the attitudes developers have toward architects when he says "architects talk in bubbles and developers talk in code" (Figure 3). The idea that code is the only artifact that matters in a software development project is so prevalent that it is one of the values listed in the Agile Manifesto: "We have come to value [...] working software over comprehensive documentation." A good architect understands that code is undeniably the most critical part of a software solution, and many of the modern development environments now produce code from "bubbles," including tools that support Model Driven Architecture (MDA) and Domain Specific Languages (DSL).

That being said, a good architect also understands that a software solution consists of more than the functional requirements implemented in custom code (Figure 4)—for example, development platforms, frameworks and infrastructure technologies. Consideration also needs to be given to the nonfunctional requirements of a software solution, like: deployment, security, scalability, and performance. Neglecting the nonfunctional requirements increases the likelihood that the system will fail.

Another critical piece of solution space knowledge is the patterns used to implement a software solution. Patterns allow a software solution to be developed with thought toward extendibility and reuse, which are critical to reducing the total cost of ownership of a software solution—especially in later phases of development or product maintenance.

Technical Acumen

Technology has been rapidly evolving for as long as anybody can remember. I've implemented countless technologies over the last dozen years, including technologies for process distribution, user experience programming languages, enterprise application integration, object-relational mapping, virtualization, and data persistence.

Understanding how a technology works is not enough to develop a robust software solution—understanding where the technology is applicable within a solution is essential to the development of a

quality product. On several occasions, I have reviewed architecture documentation consisting of little more than a number of boxes on a Visio diagram each representing a technology, with no mention of how the technology was intended to be used or any reference to the functional requirements to be fulfilled by the technology. Such practices give architects a bad name.

It is impossible for anybody to understand every detail of every technology. But an architect should understand, at a minimum, the intent and applicability of any technology prior to requiring its usage in the context of a software solution. The architect should also map the technology to the functional requirements, or to the applicable part of the conceptual architecture. Too often, I encounter the bad practice in which an architect makes a technical edict without explaining the intended application of the technology. To be honest, I made the same mistake on occasion earlier in my own career.

Architects sometimes allow their passion for technology to overshadow the problems that they need to solve. The art of choosing technology for a software solution is finding the minimum amount of technology required to meet the system requirements, both functional and nonfunctional. Delivering a software product that meets all quality standards, such as performance and scalability, will require a good amount of technical knowledge, and it is the job of the architect to define the platform for development and deployment.

With all of the advances in technology, keeping abreast of the latest trends can be daunting, but it is critical for an architect. One company that I worked with had a problem with the performance of a client/server reporting application. The application had been extended for years without a technology update. The architect responsible for the solution was adamant that building an object-layer over the relational model would solve the problems with his application. The proposed solution would have been status quo a decade ago, but database technologies have advanced greatly over the last decade and now contain optimized reporting services as part of the platform. The solution proposed by the architect would have increased the total cost of ownership of the solution (development, maintenance, licensing) and most likely would have adversely affected performance. Luckily, the architect was open to learning about the newer technologies and the product upgrade took advantage of the capabilities in the newer database platform.

Patterns

One critical skill possessed by all great architects is an understanding of how to leverage patterns in the definition of a software solution. Patterns have been a mainstay of software development for over two decades. Through the seminal *Design Patterns* by Gamma et al., the *Pattern Oriented Software Architecture* (POSA) series of books, various publications from industry luminaries, and the work of organizations like the Hillside Group, patterns have become a key measure of how software is judged. When reading about a new technology or a framework, I often try to list the different patterns that were used in the solution in order to assess the quality of the design.

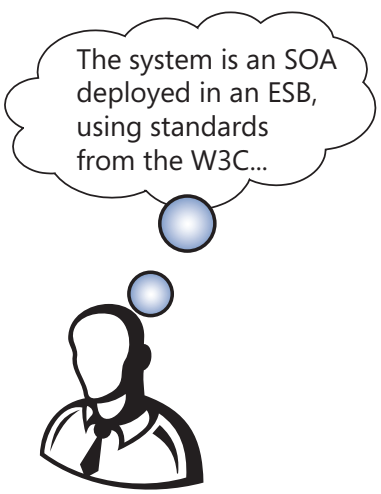
The mainstream usage of patterns in software development has made it a critical skill for architects. Learning how to leverage patterns takes time and effort. Some architects, like me, have had the good fortune of working with experts in patterns who provide mentoring in the discipline of architecture and design; others are left to acquire these skills on their own. With or without mentors, developing a proficiency in patterns does not happen by accident and requires dedication. There is a large learning curve in acquiring the patterns vocabulary and an even larger learning curve in understanding where patterns can be applied in a software solution. The effort put into learning patterns is paid back tenfold, giving an architect the necessary skills to intelligently discuss design with their peers and to create extendable software systems.

"THE ROLE OF THE IT ARCHITECT IS TO SOLVE A PROBLEM BY DEFINING A SYSTEM THAT CAN BE IMPLEMENTED USING TECHNOLOGY. GOOD ARCHITECTS DEFINE SYSTEMS BY APPLYING ABSTRACT KNOWLEDGE AND PROVEN METHODS TO A SET OF TECHNOLOGIES WITH THE GOAL OF CREATING AN EXTENDIBLE AND MAINTAINABLE SOLUTION."

One of the main motivations to leveraging patterns in a software solution is the ability to design frameworks that allow the solution to be easily extended. A software solution uses frameworks to define the solution's architecture in the context of the problem domain. Good software frameworks are usually defined using general-purpose patterns that are reusable over a number of domains. One of the main drivers behind domain-specific languages is to increase developer productivity by providing graphical tools to customize general frameworks. As mentioned above, patterns are an essential component of defining frameworks.

I can provide many examples in which understanding patterns has increased productivity, but the best example was in the project I mentioned earlier which I inherited after it was stalled for over a year. Having designed similar solutions in the past, I understood the patterns that were necessary to build an extendable domain model. While equipment was being ordered for the lab and development

Figure 5: An architect making alphabet soup



staff was being interviewed, I was able to define the domain model and frameworks with patterns that I had used on similar efforts in the past, accelerating the development phase. The initial frameworks I developed were a key factor in being able to deliver the product in a short time frame.

Don't Build 'The Homer'

With all of the skills possessed by good architects, it is often challenging to narrow the numerous options to be used in a software solution. Between the standards defined for specific domains, the alternatives for conceptualizing a software system, the plethora of technological options, and the numerous patterns to promote extendibility and reuse, it is easy to over-architect a solution that bears greater resemblance to alphabet soup than to a robust software system (Figure 5).

The first step in simplifying a software system is to understand that no matter how fun it is to try out new techniques and technologies, they must be applied in the context of a system requirement. It was no coincidence that the first system I designed after reading *Design Patterns* contained almost all of the patterns defined in that book.

"UNDERSTANDING HOW A TECHNOLOGY WORKS IS NOT ENOUGH TO DEVELOP A ROBUST SOFTWARE SOLUTION—UNDERSTANDING WHERE THE TECHNOLOGY IS APPLICABLE WITHIN A SOLUTION IS ESSENTIAL TO THE DEVELOPMENT OF A QUALITY PRODUCT."

Many years later, after a few painful experiences, I have learned that a minimalist approach, using the least amount of technology to fulfill the requirements, always yields the best results.

There is an episode of *The Simpsons* I often think about when defining a software solution. In the episode, Homer, the dim-witted father, was asked to design a car for an auto manufacturer. The design for Homer's car was full of nonessentials like multiple horns, cup holders, and shag carpeting—without thought to the overall cost or customer appeal of the car. The prototype, appropriately called "The Homer," was so costly and unappealing that the product bankrupted the company. Whenever I design a solution, or review the design of a solution, I frequently stop and ask myself if the resulting product is beginning to resemble "The Homer." If the answer is yes, I sharply refocus on the essential functionality and technology required to deliver the software system.

To Make a Short Story Long…

A couple of months ago, while waiting to make a presentation at a customer event, I was introduced to a developer attending the event as an architect from Microsoft. Expecting one of the usual greetings like "pleased to meet you," I was surprised that the first thing he said was "I think that architects are obsolete." Not wanting to be roped into an argument before my presentation, I responded with a grin and said, "I hope not."

Upon reflection, I have wondered why he felt he needed to make that statement from the outset: Was he having a bad day? Did he have a bad experience with an architect? Perhaps he is unaware of the contributions of architects in software development. Maybe his solution was so well architected that the development staff only needed to consider coding and testing functional requirements without needing to understand the considerations necessary to create a robust software solution.

A poorly architected project languishes without results, or is delivered with poor results. On the other hand, a successful software solution is well architected, by someone with domain knowledge, the ability to think conceptually with technical acumen, and the ability to leverage patterns to solve problems. Industry titles for this role have come to include architect, technical lead, and a number of others, but the importance of the role is clear. We should stop wasting time arguing the virtues of the role and continue solving problems with technology.

Resources

The Agile Manifesto
http://agilemanifesto.org/

Design Patterns: Elements of Reusable Object-Oriented Software, Erich Gamma et al. (Addison-Wesley Professional, November 10, 1994, ISBN: 978-0201633610)

Handbook of Software Architecture
http://www.booch.com/architecture/index.jsp.

The Hillside Group
http://hillside.net/

The Homer Wikipedia Entry
http://en.wikipedia.org/wiki/Oh_Brother%2C_Where_Art_Thou%3F#The_Homer

"Using Patterns to Define a Software Solution," Joseph Hofstader, November, 2006.
http://msdn2.microsoft.com/en-us/library/bb190165.aspx

"Using Patterns for Rapid Development," Joseph Hofstader, February, 2007. http://www.iasahome.org/c/portal/layout?p_l_id=PUB.1.366

About the Author

Joseph Hofstader is an Architect for Microsoft's Developer and Platform Evangelism organization. He is also an adjunct professor in the Department of Information Technology and Electronic Commerce at the University of Denver. Prior to joining Microsoft, he was a Distinguished Member of Technical Staff at Avaya. Joe has spent his career architecting, designing, and developing solutions in the telecommunications industry. He is happy to receive feedback at joe.architect@live.com.

Becoming an Architect in a System Integrator

by Amit Unde

Summary

I am currently involved in a program for grooming aspiring architects within L&T Infotech into full-fledged architects. As a result, I have extensively researched the role of an architect and talked to many architects across different industries to understand their role and the competencies that make them successful. This article is an attempt to crystallize the wisdom I've gathered from this work.

Being an architect is tough! What architects do is a mystery to much of the world—this is hardly surprising since an architect's work is intangible—"thought-ware," if you will—and it happens in the background. That makes many wonder about the architect's role in an organization. Architects interact with many stakeholders—CIOs, project managers, business users, and developers—and each expects them to work differently. While the CIO expects an architect to derive a solution roadmap for implementing the company's IT vision, the developer expects the architect to provide direction on the technical problem. The architect needs to have a bird's eye view in one scenario, while in some other scenarios, the architect needs to dive deep into the problem area. The architect is expected to be both a generalist and a specialist.

Many companies try to reduce the ambiguity by introducing different flavors of the role, such as enterprise architect or solution architect. Ironically, differentiation within the role can add to the confusion since there is no standardization of the designations across companies. Let's find the commonalities and define these different flavors of the role.

The Architect Role

Typically, there are three different variations of the roles (Figure 1):

Enterprise Architect/Chief Architect

The enterprise architect is responsible for implementing the CIO's vision and strategy for IT. It includes defining strategic programs (usually multiyear, multimillion dollars for large organizations), selecting the appropriate technology platforms, and providing guidance for implementations. The enterprise architect aids the CIO in making sure that the IT investments are aligned to the business strategy, and provide competitive edge for the organization. The person is also responsible for defining the standards and guidelines, and putting up a governance mechanism to align implementation to the defined standards and guidelines. In some organizations, this role is merged with that of the CIO and has the title "Chief Architect." This is especially true for many product and platform companies.

Solution Architect

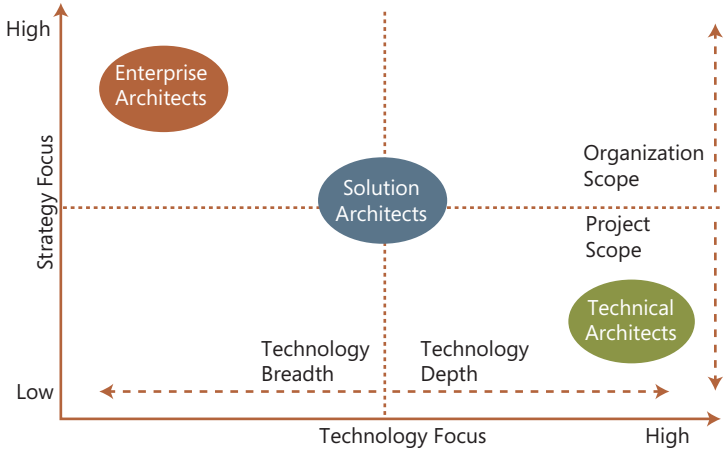
The solution architect is responsible for implementing a strategic IT program. This includes defining the architectural solution for the program (usually spanning multiple technologies), selecting technology platforms in adherence to corporate strategy, handling intergroup communication, and making decisions on technical issues in implementation. The solution architect usually needs to mediate between business and technology teams and various other groups. The solution architect is the "go-to" person for any technology conflicts, implementation issues, or decisions.

In some organizations, this role is defined just as "Architect." The senior position has the title "Lead Architect."

Technical Architect

The technical architect is usually a technology specialist in a particular technology. This person has expert knowledge of the underlying technology function, its integral components, and understands the strengths and limitations of the technology. This person is responsible for determining the applicability of the technology, for defining the best possible architecture using that particular technology, and also for guiding the team in implementing the solution. Generally, the technology architect is expected to know the various vendor tools in the technology area, the latest trends in the market, and various architectural alternatives for implementing the solution.

Figure 1: Architect Roles



There could be more flavors of this role—infrastructure architect, integration architect, BPM architect, .NET architect, J2EE architect, and so forth.

The Architect's Competencies

Now that we have defined roles and responsibilities, let's look at which competencies are required to perform these roles (Figure 2).

Leadership

Architect is a leadership designation. An architect is supposed to bring clarity to the requirements, define the foundation, and lay out the roadmap for execution. At each step, the architect has to make decisions and take ownership. Many times, the right decision will not be simple or clear-cut. The architect needs to find a solution that will work. It may not always be the best solution on technical merits, but it must be what will work best in the particular organization. To reach these decisions, the architect needs to have a very good understanding of the political environment, and should have the ability to generate "buy-in" from all the stakeholders to move the project forward. Architects must be confident enough to stand up to negative criticism, work their way through roadblocks, and shield the development team from political pressures. Hence, the most significant competency an architect must have is leadership.

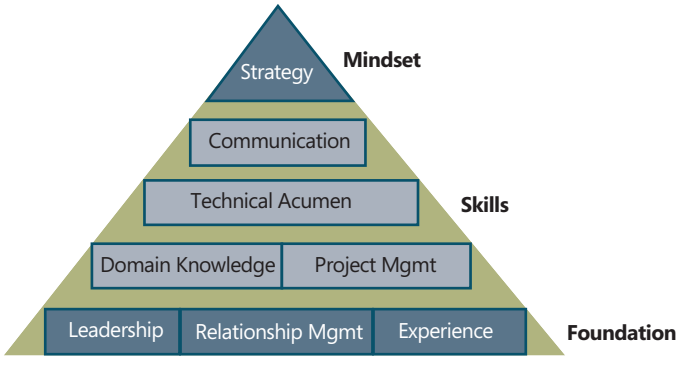
Strategic Mindset

This is an ability to look at things from 50,000 feet, at a strategic level, abstracting the operational complexities. It involves taking a larger vision such as "taking an organization into leader's quadrant by 2010" and dividing it into smaller, tangible steps to make it simpler for others to achieve it. Architects are often asked to choose a solution that provides the best return on investment to the organization and to create business cases to get the budgets. They often need to deal with top-level executives (CIO, CEO) where it is necessary to present a view at strategic level.

Human Relationship Management

Architects deal with many internal stakeholders as well as external stakeholders such as vendors and partners. Often, they need to get work out of people who do not report directly to them. They need to be connected to the organization's grapevine to understand the political implications. They should be approachable, to encourage developers to break bad news as soon as possible. Hence, relationship management at several levels is a necessary competency for the architects.

Figure 2: Architectural Competency Pyramid



Communication and Listening Skills

Listening skills are often considered part of communication skills, but I mention them explicitly to emphasize their importance. It is essential that the architects listen to the business users to understand their business problem, to the senior management to understand the most workable solution, and to the developers to understand the possible problems in the implementation. At the same time, it is important for the architects to effectively articulate the solution to the business users to generate buy-in, to the senior manager for funding and support, and to the developers so that they understand how to implement the defined architecture.

The architects need to adapt their communication style when interfacing with different stakeholders. For example, when they deal with the senior management, brevity is important, whereas when they deal with the developers, clarity is more important. The different stakeholders have different expectations—the executives require a business view of the solution explaining the investments, returns, and benefits, whereas the developers are interested in nitty-gritty of the technology implementation. The architect must understand the needs of these different stakeholders and change the articulation style and content of each interface accordingly.

Business Domain Knowledge

It is very important to understand the problem statement before defining a solution for it. It is also important to be aware of non-stated requirements, such as regulatory and legal requirements, competitive solutions, and so forth. The sound business knowledge not only helps in defining the appropriate solution, it is also necessary for understanding the requirements and articulating the solution. To have meaningful dialogue with the business users and to establish confidence with them, the architect must speak in their business vocabulary and draw examples from their domain.

Technical Acumen

This is a key competency since the architects are hired for their technical acumen. It is essential to have exposure to the breadth of technologies and vendor platforms to understand their relative strength and weaknesses, and make a best choice to suit the requirements. Even for a "specialist" role such as technical architect, it is desirable to have exposure to multiple tools and vendor platforms, and to be aware of technology trends within the industry.

A topic of debate is whether the architect needs to have hands-on experience in coding. Since I was a developer, I may be biased, but I think it's helpful to have a coding background to understand the possible issues and also to identify solutions to the problems. Nonprogramming architects often find themselves detached from the development teams and may be unable to help them with technology problems. This could seriously affect the team's productivity. (It is, of course, nevertheless possible for a team to deliver a good solution with the help of senior developers.)

Program/Project Management Skills

Why should an architect be required to have project management skills? If you take a close look at what architects are doing, you might see they are doing nothing but managing a project or a program, albeit largely from the technology standpoint. They often find themselves estimating, choosing development methodology, and

planning with the project managers. It is therefore beneficial to have project management experience or training.

Architects also need to guide their teams in following a process and maintaining discipline. An architect must be conversant in development methodologies (such as RUP, CMMI, and Agile) and architectural frameworks/methodologies (such as Zachman and TOGAF).

Variety of Experience

It is not just the gray hairs. Architects need exposure to projects of varying scope and scale on a range of technology platforms. The size of the project does matter in enhancing your architectural skills. For example, the architectural considerations for a small, local application for a limited number of users will be totally different than those for a large application being accessed by a large user base across the globe. I believe aspiring architects should deliberately try to get into the assignments that offer a range of experiences rather than sticking to the assignments of similar nature.

Does It Matter Where You Work?

The nature of your organization and its services surely influence your overall development as an architect. Generically speaking, if you are working for an IT services company serving multiple customers, you are likely to gain wider exposure to technologies and projects. If you work for a product or platform organization, you will get the opportunity to specialize in a particular business domain and technology suite. If you work for an end-user organization, you can get involved in strategic decisions and see the long term to know the effects of your decisions. On the whole, large companies provide more mentorship opportunities, whereas smaller companies provide more ownership. Of course, each organization is unique and generalizations are by their nature broad-brush. Aspiring architects should carefully evaluate the career opportunities available in their organizations and chart their own path for development.

Getting There

As the architect role has gained visibility in recent years, resources for aspiring architects have grown.

Education

Initiatives have been set forth to standardize the curriculum for educating architects. For example, the International Association of Software Architects (IASA) has defined a skill library for architects (http://www.iasahome.org/web/home/skillset) and a standard curriculum and certifications. Similarly, the Software Engineering Institute (SEI) has defined a curriculum and training program (http://www.sei.cmu.edu/architecture/arch_curriculum.html).

Many vendor companies provide educational resources for architects. Microsoft's MSDN Architecture Center is a one (http://msdn2.microsoft.com/en-us/architecture/default.aspx). IBM DeveloperWorks also provides a resource site (http://www-128.ibm.com/developerworks/architecture/).

Certifications

There are many certification programs. The value of these certifications is directly linked to the difficulty level in attaining those. For example, Microsoft Certified Architect programs (http://www.microsoft.com/learning/mcp/architect/default.mspx) are based on an expert panel interview during which the architect is evaluated on seven competencies, the technology knowledge being just one competency. Although provided by Microsoft, the MCA is actually a technology-independent certification. The Open Group has a similar certification program (http://www.opengroup.org/itac/).

There are other certification programs that are technology-knowledge-based programs, which do not involve any interview process. Often, these are technology-specific programs. For example, Sun Microsystems has a program for certifying on J2EE technology (http://www.sun.com/training/certification/java/scea.xml).

Groups and Forums

There are many blogs, groups, and forums available for architects to pick the brains of fellow architects and network within the architectural community. Here are some of the most notable ones:

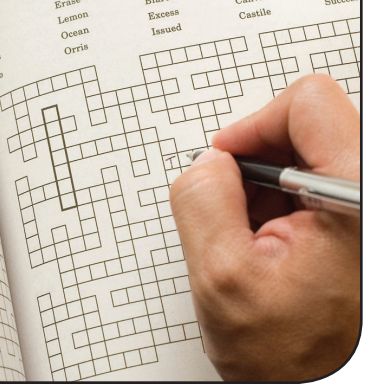
- International Association of Software Architect (IASA): http://www.iasahome.org
- Worldwide Institute of Software Architects (WWISA): http://www.wwisa.org
- MSDN Forum on Architecture: http://forums.microsoft.com/MSDN/default.aspx?ForumGroupID=58&SiteID=1
- Open Group Architecture Forum: http://www.opengroup.org/architecture/
- Grady Booch Blog: http://www.booch.com/architecture/blog.jsp
- Blogs by Microsoft Architects: http://msdn2.microsoft.com/en-us/architecture/aa699386.aspx

Conclusion

Experience and leadership qualities form the foundation of the architect role. You also need technical acumen, good communication skills, and domain and program management skills. Many educational resources and certifications are available. Experienced mentors are another important resource since training alone is inadequate for developing many necessary skills. Aspiring architects should consider many factors when making career choices, from types of projects to access to mentors. Architecture is a demanding but rewarding profession; it takes determination and good planning to fully develop your skills and mature into the role.

About the Author

Amit Unde is a lead architect at L&T Infotech. He is a Microsoft Certified Solutions Architect (MCA) and also a PMA-certified Qualified Project Management Professional. He has over 10 years of experience in Enterprise Architecting, Integration, and Application Development using .NET and J2EE Technology. Amit has worked with many large insurance and manufacturing organizations in the U.S., Europe, and Asia, implementing strategic programs like IT Strategy Roadmap definition, IT Rationalization, Application Reengineering and SOA Implementation. He works with L&T Infotech's Insurance Solution Office as a thought leader in conceptualizing innovative solutions for various contemporary business issues in Insurance domain.



Architecture Journal Profile: Paul Preiss

For this issue of *The Architecture Journal*, we met up with Paul Preiss, founder of a nonprofit group called IASA (International Association of Software Architects). We asked Paul about the goal of the organization, and some of his thoughts about the profession.

AJ: Paul, what do you do?

PP: I run the International Association of Software Architects (IASA). I spend most of my time trying to provide programs and services to practicing and aspiring architects.

AJ: Can you tell the readers about IASA? Where and how did it get started?

PP: IASA was founded about five years ago as a user group in Austin, Texas. We've grown to become the largest IT architect association in the world, with about 7,000 members and 50 chapters across 25 countries. Our focus is on professional growth and support for individual architects. We also aim to empower the architects to own their profession the way that other professionals do, such as doctors and lawyers.

I started the IASA to help stabilize my own career. I originally founded the user group because I wanted to help others and get help in my own career path as an architect. I had been practicing for about 10 years, working on some of the biggest and smallest architecture problems out there. I had run into a handful of major issues: the lack of resources targeted at the architect in the daily role; the lack of peers and the inability to find like-minded and similarly skilled people to interact with on a peer basis; the real lack of common definition for fundamental skill sets and the variability of the role across organizations; the overall difficulty of categorizing types of architects and of evaluating competence. I've done everything from seeking jobs as an architect to hiring and managing architects. So much uncertainty makes it very difficult for the individual architect to set a career path and follow that career path across organizations in a way that other professions may take for granted.

AJ: Sometimes IT Architecture is compared to other, more mature professional fields such as medicine and law. Do you agree with these comparisons?

PP: The profession that we are most closely modeling in terms of professional infrastructure is medicine, and I tend to model the organization mostly after the American Medical Association. The medical profession is arguably the most technically complex, mission-critical profession in the world today, with a tremendous volume of technical changes on a regular basis and growth in knowledge bases—and yet we graduate and grow doctors in a stable and regular way, through structures like clinical rotations and certification. The reach of IT architecture is broadening. Architects have become integral components of industry and business, in corporate fiscal policy and execution. Architects of healthcare and space shuttle systems are specifically entrusted with human safety. We impact the financial health of organizations and individuals everywhere through commerce-enabled systems. We can also have a direct impact on entire societies through innovations like YouTube, Web 2.0, and social networking. If we can prepare and support a doctor for everything that they have to go through, creating the professional infrastructure to support an architect can't be as hard!

AJ: Do you believe that our industry should follow the kind of specialization that we see in the medical field?

PP: Perhaps, but in the end a doctor's a doctor. If you're out having dinner and someone starts choking, you don't stand up and say, "Is there an ear, nose, and throat specialist in the house?" You say, "Is there a doctor in the house?" The general professional title has to be meaningful before specializations can be meaningful.

A key objective in the IASA is to identify the common differentiator that sets our profession apart from the others. If we don't do that—I will be honest with you here—we will be tuned out because business owners I talk to don't have the bandwidth to parse software vs. infrastructure vs. solution vs. business vs. application vs. enterprise. They want to know why they should hire an architect. If you want to do the profession a favor, help differentiate the profession first, and then work on specialization. Remember that although lawyers and doctors go through a process of specialization, they first go through a generalized education.

AJ: Can you elaborate more on this specialization aspect?

PP: Specialization can have long-term positive and negative impacts that we need to consider. I really urge everybody reading this article to think carefully about this because it's our job to define for ourselves what our future will look like. If we don't do this, then someday, somebody else will define our profession for us. Specialization in medicine has important insurance implications—in fact, if an oncologist or podiatrist delivers your baby and does it incorrectly, they will be protected from litigation by their insurance. On the other

hand, doctors are generally not covered if they practice outside of their specialization.

Given our direct impact on human safety, financial security, and society, I happen to know we are facing increasing degrees of scrutiny around the world as a group of practitioners. The impact of future regulation and regulatory activities should be of tremendous importance to each one of us, and working ahead of regulatory trends to define our profession for ourselves ought to be an immediate priority for each of us. We need to think more about our profession and less about specific individual jobs whether we work for Microsoft, Sun, American Express, Bank of America, or another company from the Americas to Europe, to Australia, to Malaysia or anywhere else in Asia. If we consider our profession first, then we can help stabilize future regulatory activities by guiding regulators to optimal decisions instead of what could be more knee-jerk, politically guided ones should any of their activities be triggered in haste. Personally, I feel I have a responsibility to help control my own professional destiny. After five years growing IASA, I have come to realize that what I do impacts how architects are perceived around the world.

AJ: What advice would you give to someone who wants to become an IT Architect today?

PP: Well, there are at least two important issues you need to understand. I call the first one the, "Where Developers-Go-to-Die Syndrome." The major symptom of this syndrome is, "I've been a developer for 15 years, so I guess I have to become an architect now because that's the next natural progression." This is similar to, "I've been a business analyst for 15 years, so I am going to become a business architect" or "I've been in operations and infrastructure for 15 years so I'm going to become an infrastructure architect." There's a notion that you can (or even ought to) become an architect by virtue of tenure or pay scale alone. Architecture is commonly seen as a land where other roles go to die. This is an utter fallacy. Architecture is an orthogonal profession distinct from development, business analysis, and system administration. Going back to the medical analogy: If you had been a nurse for 15 years, could you now become a doctor on grounds of tenure alone? You may have some advantages in terms of practical experience over any intern, but you've still got to start at the beginning of the medical profession; you have to finish medical school, qualify for your license, and complete internships—you've got to go do all those things.

AJ: So where do you think these perceptions have come from? Who's to blame?

PP: Well, I think it is a pretty natural progression, so in a sense, there's no one to blame. What has happened has been sort of organic in the sense of its original format, or the process of formation of the IT industry as a whole. It is natural that IT architecture is seen as specializing along multiple lines based on existing roles and other activities such as development, infrastructure management, and business strategy alignment. I think that in fact, the industry is mature enough to where those fulfilling the other roles have become comfortable investing their sense of identity in them. Architecture if often understood merely as a matter of extending what it is we already do, or perhaps even a role granted to those with enhanced innate abilities.

On the other hand, the shape of the profession going forward is up to us—I think we have an opportunity now to be proactive in defining our profession.


Paul Preiss

Paul received a bachelor's degree in Japanese from the University of Texas at Austin. He went on to become a project manager for Human Code. Later, Paul was the applications manager and architect for Dell Pan Asia based in Kawasaki, Japan. He then became the senior architect for a software consulting firm in St. Paul, Minn., where he provided architecture consulting for numerous government and private enterprise clients. Paul went on to become the director of engineering and chief architect of a digital asset management company. More recently, Paul has been spending most of his time creating and managing the International Association of Software Architects.

I recently blogged about the magician's apprentice, trying to dispel the common notion of, "If I work for an architect, if I put this on my title, if I study and happen to have the right sort of magical quality about me, I'll be a great architect." But in fact, "profession" is a rigorous concept. Professionals are groups of people that clearly define their skill sets, their value proposition, that which differentiates them as communities from other professionals and groups, and the hoops that they and their peers must jump through to be part of the club. That is all any profession really is. As long as the role in question is valuable to society, as we have seen IT architecture become over the years, then at some point the associated skill set splits off and becomes completely educable, that is you don't have to become something else first. Go to the American Institute of Architects Web site and look at their history (see Resources). You will read that the 13 founding members of the AIA gathered in 1857 with the aim to "elevate the standing of the profession" and out of frustration that "anyone who wished to call him- or herself an architect could do so…masons, carpenters, bricklayers….No schools of architecture or architectural licensing laws existed to shape the calling." That sounds an awful lot like the IT architecture profession today. So they put a stake in the ground, and they said that is no longer acceptable; 150 years later we have the building architecture profession in its current form.

AJ: Let's hope it doesn't take us 150 years to get there. In a previous comment, you mentioned hoops that you need to jump through to join the architect club. What are those hoops? Is it certification?

PP: The progression of medical knowledge and learning—what physicians have come to understand about their profession and how they practice their skill set, and so on—has allowed doctors to greatly improve the quality of care since ancient times. Keeping pace with the growth of the medical field, the professional bodies have continually raised the quality bar by creating bigger, broader, and more sophisticated hoops for people to jump through. The hoops right now for IT architects are being defined inside the IASA, and in other organizations, from a skills perspective. We have laid out 250+ skills in our taxonomy that defines a rigorous foundation body of knowledge and a rigorous specialization body of knowledge that any individual must possess to be a part of the club. What we call

the Skills Taxonomy Project resulted in a body of articles published in collaboration with Microsoft and our members. So the first thing that an architect or aspiring architect can do is look at our skills taxonomy, at our foundation body of knowledge. Regarding professional infrastructure, the profession will decide, for example, as most professions have, whether the first hoop that you have to jump through is a college degree. Generally speaking, most professionals must begin their career with a college education. You are forced to get a medical degree, a law degree, an accounting degree, a finance degree, a marketing degree, or whatever. So sometime in the future, if IT architecture truly maintains its status as a profession that will likely be what someone will have to go through first.

Now, all degrees are primarily knowledge-based, and they hinge upon tests. With that in mind, one of the things that we are working on now is effectively an associate certification, which will require a junior knowledge-based test that covers all 250+ skills in our taxonomy. We then have to decide whether the profession needs a significant amount of practical experience, commonly called internships. Those internships could be provided in a very rigorous fashion or a sort of lightweight fashion: A teaching internship is quite rigorous; a marketing internship is perhaps not as rigorous; a medical internship is very rigorous. We need to decide as a profession, how one progresses from the knowledge-based test to the next hoop which will be a professional certification that simply says: "This person has both the knowledge and the experience to practice architecture without oversight on a certain sized project." However, a professional certification as compared to associate certification will be the hardest to obtain.

A third hoop could be Master Certification, such as the Microsoft Certified Architect programs. And that basically says that anyone above this line represents the top five to 10 percent of the entire professional body globally. I am not going to dig into all of the details of the infrastructure necessary to move between the major hoops, because the ones that are of most interest are the first four, because they represent what it would take for a bus driver to become an architect. The first four hoops are: effective training in the conceptual and practical application of the body of knowledge—a knowledge certification, a really difficult test that certifies that you've properly assimilated that knowledge; an experience quotient often called an internship; and finally, a professional certification that differentiates you from what IASA terms the associate or junior architect as a mature individual professional who may now go out into the world and practice without a mentor or direct oversight. So those are the hoops that IASA members have identified, and those are the primary components of the comprehensive education plan that our members are in the process of building.

AJ: What would be your one take-home message for the people who, after reading this article, are saying, "Yes, I want to be on that path"?

PP: Becoming an architect is a challenge, and the process depends on where you are starting. In general, I would recommend taking a deep look at the skills taxonomy project on the IASA site. Really dig deep into that, even if you don't join. Many aspiring architects should be using that as their real decision-making point. Because when you look at those articles, you'll see the depth and the breadth—I mean, I have to tell you from my own perspective, when we first did the taxonomy I was in shock because I didn't realize it was that big. I was really surprised at how deep and far the expectations for architects are. I recommend first reading the articles in the IASA online skills library (see Resources), before deciding if architecture is really your path. Because most people today make their decision about becoming an architect based on what they think an architect is rather than what the overall skills and maturity model look like. So I would say that is their first step. The second step, if you make the decision to become an architect, is to join your local chapter. If there is no chapter in your area, help found one, and get involved with the IASA training program, which will allow chapter members to get those skills.

"TAKE A DEEP LOOK AT THE SKILLS TAXONOMY PROJECT ON THE IASA SITE. MANY ASPIRING ARCHITECTS SHOULD BE USING THAT AS THEIR REAL DECISION MAKING POINT. BECAUSE WHEN YOU LOOK AT THOSE ARTICLES, YOU'LL SEE THE DEPTH AND THE BREADTH—I MEAN, I HAVE TO TELL YOU FROM MY OWN PERSPECTIVE, WHEN WE FIRST DID THE TAXONOMY I WAS IN SHOCK BECAUSE I DIDN'T REALIZE IT WAS THAT BIG."

AJ: Can you find active chapters through the Web site?

PP: Yes, chapters, training program, and events are accessible from the IASA home page.

AJ: How about your own career—where do you see yourself in five years time?

PP: Well, I tell you, this has been a wild ride; an eye-opening experience for me. I have the fortunate job of being able to talk to really smart people around the world, including aspiring, professional and master architects, about really interesting challenges facing our profession. I don't see myself giving that up any time soon. It's my passion.

In five years, I want to be doing exactly what I am doing now—which is helping architects control their own careers, their own profession, building infrastructure and programs to help architects in their daily jobs, helping organizations best utilize architects to execute their technology strategies and get financial or other types of values. Like I said, you are going to have to pry my hands off of the grid because it is such a fun job. And if there's any measure of success that I can see, it's in the emails and discussions I receive saying that the programs that we're putting in place—the education, the community, and so forth—are actually helping people do a better job, understand their jobs better, plan their own personal career paths and really feel they have a chance to achieve their goals. That's the measure of success and it's gratifying—I believe that I really could do this forever.

The Open Group's Architect Certification Programs

by Leonard Fehskens

Summary

How do you know if someone is really an architect? This has become an increasingly important question as the context and nature of information systems have evolved into their present forms. Information systems have become mission-critical resources, essential to the routine functions of modern society, and IT projects need to "get it right the first time." "Do more with less" is a recurring mandate, while the requirements grow broader and more complex. At the same time, the fabric of information systems has changed; the long-term trends of commoditization and consolidation have pushed opportunities for competitive differentiation—and the necessary skills to take advantage of them—to higher levels of abstraction.

Many people have come to believe that the discipline of architecture is a powerful tool to address this daunting challenge.

The Open Group Architecture Framework

The Open Group, a consortium of IT vendors and users, was formed in 1996 by the merger of X/Open and the Open Software Foundation (OSF). Multiple forums allow members to contribute to open standards in a variety of technology domains. One of the most active forums is the Architecture Forum, with 176 members from all over the world and representing a wide variety of industry sectors. In 1994, the membership decided that a standard enterprise architecture framework was needed. This decision led to the development of The Open Group Architecture Framework (TOGAF) and a TOGAF certification program.

The certification of IT architects benefits three constituencies:

- Individual practicing IT architects, and thus the profession as a whole
- The employers of IT architects, both as in-house architects and as professional services architectural consultants
- The consumers of IT architects' services and work products.

Based on its extensive experience certifying UNIX implementations, The Open Group believed that the certification process needed to be demonstrably objective—that is, the same results would be achieved regardless of who executed the process. So, in addition to the publication of the TOGAF framework, The Open Group membership defined a policy for certifying TOGAF products (specifically tools and training), services (consulting), and individuals (practitioners). The requirements for certifying TOGAF tools, training courses, professional services, and individual architects are defined by four TOGAF product standards. TOGAF-certified training courses and TOGAF-certified professional services must be delivered by TOGAF-certified architects.

There are two ways an architect can become TOGAF certified: by taking TOGAF certified training, or by passing a TOGAF-certified examination. The training must address, and the examination will test, knowledge and awareness of TOGAF, and a thorough and complete knowledge of the elements of TOGAF listed in the TOGAF 8 Core Definition. This includes the phases and deliverables of the TOGAF Architecture Development Method (ADM); the TOGAF Technical Reference Model (TRM), which defines the substance of the framework; the resources available to a TOGAF practitioner (the Standards Information Base, or SIB); the Enterprise Continuum (a model for organizing and relating reusable architecture and solution building blocks); and finally, the relationship of TOGAF to other architectures and architecture frameworks.

A New IT Architect Certification

As TOGAF went through several successive revisions, members of the Architecture Forum asked the question posed above—how do you know if someone is really an architect?—in practice, not just in theory, and considered the problem of IT Architect Certification (ITAC) independent of TOGAF. Several of the Forum's members operated architecture profession programs, and certification was often part of the professional development and career path of profession members. These programs had comparable criteria and processes, but differed in many details and were essentially proprietary. The Architecture Forum recognized the value of industry-wide, vendor-independent standard certification criteria, and asked that The Open Group initiate a project to define such a standard.

In early 2004, IBM and HP began collaborating on a detailed proposal to The Open Group. The proposal was approved in October 2004, and a working group comprising volunteers from Capgemini, CLARS, EDS, HP, and IBM developed IT architect certification requirements and policies over the next year. These were approved by The Open Group membership and the program went public in July 2005.

The goal that certified individuals be actually, not merely potentially, successful practitioners led to the realization that IT architect certification did not lend itself well to traditional certification methods such as examinations. As a result, board review

of demonstrated skills and experience by certified peers was agreed upon as the evaluation method.

From its inception, the program was envisioned as offering three levels of certification: Certified, Master, and Distinguished, as shown in Table 1.

The initial focus was on level 2, as that was the membership's primary need. The working group also felt that it would be straightforward, after establishing level 2, to relax and strengthen the certification requirements, respectively, to address levels 1 and 3.

"THE GOAL THAT CERTIFIED INDIVIDUALS BE ACTUALLY, NOT MERELY POTENTIALLY, SUCCESSFUL PRACTITIONERS LED TO THE REALIZATION THAT IT ARCHITECT CERTIFICATION DID NOT LEND ITSELF WELL TO TRADITIONAL CERTIFICATION METHODS SUCH AS EXAMINATIONS. AS A RESULT, BOARD REVIEW OF DEMONSTRATED SKILLS AND EXPERIENCE BY CERTIFIED PEERS WAS AGREED UPON AS THE EVALUATION METHOD. HOWEVER, THIS METHOD MADE THE REQUIREMENT FOR A DEMONSTRABLY OBJECTIVE PROCESS PARTICULARLY CHALLENGING, ESPECIALLY CONSIDERING THE ADDITIONAL REQUIREMENT THAT THE PROCESS BE SCALABLE TO MANY HUNDREDS OF CERTIFICATIONS PER YEAR AND THOUSANDS OF CERTIFICATIONS IN TOTAL."

Using board review rather than examination to decide certification made the requirement for a demonstrably objective process particularly challenging, especially considering the additional requirement that the process be scalable to many hundreds of certifications per year and thousands of certifications in total.

Accredited Certification Programs

Because many member companies already had large architectural practices and internal certification programs, an obvious strategy was to leverage these existing programs. This led to the idea of "indirect" certification by an Accredited Certification Program (ACP), by which a company could certify its own architects using an internal process that had been accredited to conform to The Open Group standard for IT architect certification, and that was periodically audited by The Open Group for continued conformance and quality control. In addition, The Open Group would directly certify architects whose employers, for whatever reason, chose not to set up an ACP.

The certification process is depicted in Figure 1.

Board Review Certification Process

Candidates for certification prepare a submission package comprising a document of no more than 50 pages, based on a template provided by The Open Group, and letters of reference. If the package is judged complete and the references are confirmed, it is passed on to a three-member review board, and a board interview with the candidate is scheduled. The board members are themselves certified architects. The review board examines the package in detail, to confirm that the evidence the candidate has provided adequately demonstrates the skills and experience specified in the IT Architect Certification Conformance Requirements. The board's interview (three separate one-hour interviews with each board member) serves two purposes: to resolve any uncertainties about the evidence provided in the submission package and to confirm the candidate's ability to authoritatively discuss the work the evidence is derived from.

The three board members then meet to discuss their conclusions based on the review of the submission package and the candidate interview. While the goal is for a board to reach a unanimous agreement to approve or reject a candidate, a two out of three vote is required. Each board member's conclusions about the candidate's satisfaction of certification requirements are captured and preserved by an online candidate assessment tool. For each certification requirement judged not satisfied, the board member must provide a specific explanation for why the evidence provided fails to demonstrate the skill or experience required, and this feedback is provided to the candidate. Candidates approved for certification are also provided with career development suggestions from the board members.

Board interviews for direct certification are held in conjunction with The Open Group's quarterly Enterprise Architecture Practitioners Conference, and additional boards are scheduled at The Open Group's offices or elsewhere as needed. If a company has more than a few candidates for certification, it may be more economical for the board to travel to a company site at the company's expense.

Requirements

The Certification Conformance Requirements require that, for each of the following skills, the candidate cite three examples demonstrating mastery of the skill to the degree appropriate for the certification

Table 1: ITAC certification levels

Level	Role in Practice	Scope of Responsibility	Business Impact
1 – "Certified"	Supervised	Project	Some
2 – "Master"	Independent	Business Unit	Significant
3 – "Distinguished"	Supervisory	Enterprise-wide	Major

level (certified, master, distinguished) applied for:

- Apply communication skills
- Lead individuals and teams
- Perform conflict resolution
- Manage architectural elements of an IT project plan
- Understand business aspects
- Develop IT architecture
- Use modeling techniques
- Perform technical solution assessments
- Apply IT standards
- Establish technical vision
- Use of techniques
- Apply methods
- Define solution to functional and nonfunctional requirements
- Manage stakeholder requirements
- Establish architectural decisions
- Validate conformance of the solution to the architecture
- Perform as technology advisor.

Similarly, the candidate is asked to provide three examples demonstrating:

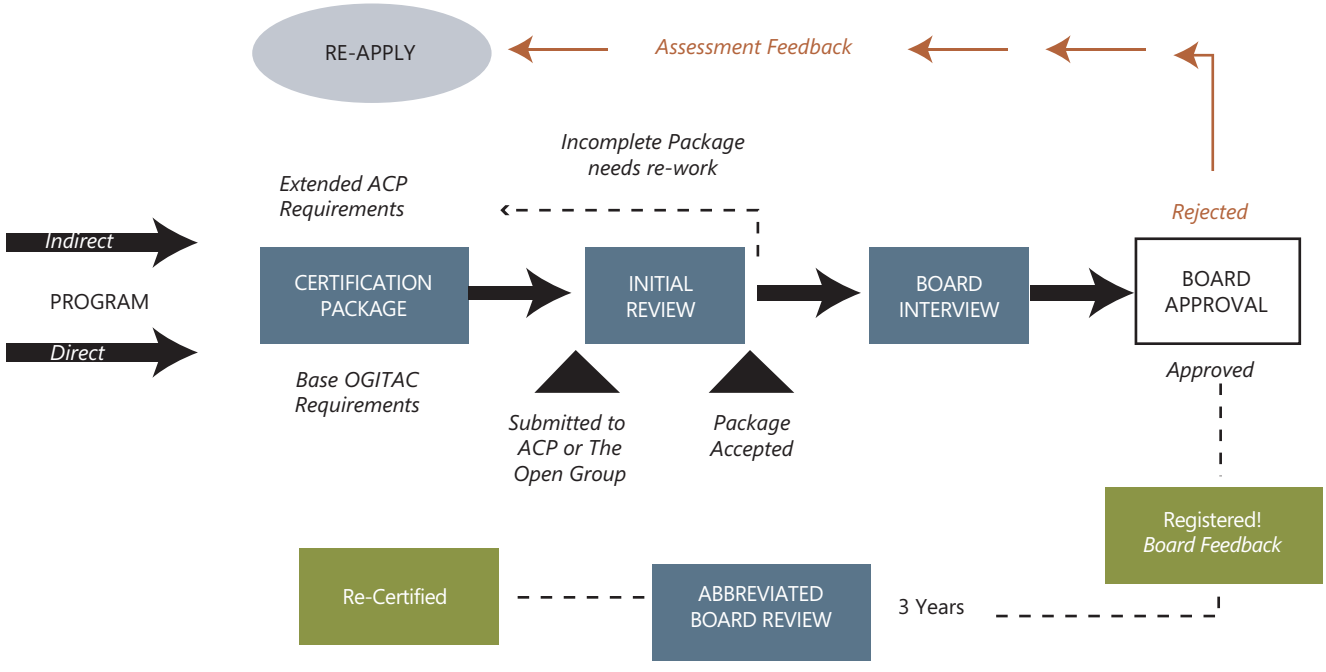
- Experience producing architectures
- Breadth of architectural experience
- Experience with different types of technologies and architectures
- Application of methods
- Full life-cycle involvement
- Industry knowledge
- Knowledge of IT trends.

"BECAUSE MANY MEMBER COMPANIES ALREADY HAD LARGE ARCHITECTURAL PRACTICES AND INTERNAL CERTIFICATION PROGRAMS, AN OBVIOUS STRATEGY WAS TO LEVERAGE THESE EXISTING PROGRAMS. THIS LED TO THE IDEA OF 'INDIRECT' CERTIFICATION BY AN ACCREDITED CERTIFICATION PROGRAM (ACP), BY WHICH A COMPANY COULD CERTIFY ITS OWN ARCHITECTS USING AN INTERNAL PROCESS THAT HAD BEEN ACCREDITED TO CONFORM TO THE OPEN GROUP STANDARD FOR IT ARCHITECT CERTIFICATION, AND THAT WAS PERIODICALLY AUDITED BY THE OPEN GROUP FOR CONTINUED CONFORMANCE AND QUALITY CONTROL."

In addition, the certification candidate is required to provide three experience profiles, each of which provides an overview of an architectural engagement the candidate participated in. The candidate may cite these profiles as providing the evidence asked for in the skills and experience sections above. Each profile specifies:

- Experience with strategy/design/implementation aspects of solution
- Key decisions made
- Demonstrated architectural capability

Figure 1: Direct and indirect certification process



- Broad technical experience
- Application of tools and methods
- Demonstrated success
- Perform as a lead IT architect.

Finally the candidate is asked to provide evidence of professional development and community activities:

- Training in the design and engineering of IT architectures
- Knowledge of the technology, trends, and techniques in the IT industry
- Vertical industry knowledge (telecoms, financial, and so forth)
- Skills and knowledge in IT architecture
- Contributions to the IT architecture profession
- Contribution to the IT architecture community.

Recertification

Certifications are valid for three years, after which recertification is required. Recertification entails a simplified application and interview process intended to validate that the architect has continued to practice and has continued with professional development and community contribution activities.

Benefits of Certification

The total number of certified IT architects to date is 2112. Three companies (IBM, EDS, and CA) are currently operating Accredited Certification Programs. Certified architects come from companies as diverse as Accenture, Adnovate BV, Allstate Insurance, Armstrong Process Group, ASC, BearingPoint, BK Larsson Consulting LTD, Capgemini, Carlson Companies, Cisco Systems, Codecentric GmbH, Credit Suisse, Computer Sciences Corporation, Datamail, Deutsche Post AG, EDS, First Canadian Title, Fortis, Ganz, GTECH Corporation, Gulf Business Machines, Hewlett-Packard, IBM, IntegrityOne Partners, Intel, ISM Canada, ITA Consulting, ITSC Bonn, Microsoft, QR Systems Inc., Rapier Solutions Consulting Ltd., Riosoft Consulting, and Rogers.

TOGAF or ITAC certification entitles one to membership in the Association of Open Group Enterprise Architects (AOGEA).

The Open Group's TOGAF and ITAC certifications provide multiple benefits to the IT architecture community:

- Standards developed via an open, multinational process represent a consensus as to the industry's best practices.
- Internationally recognized standards for IT architect certification promote the development and recognition of the IT Architect profession and, thereby, raise the bar for qualifications across the entire industry.
- Certification provides professionals with a portable vendor-independent credential verifying their experience and competence, a credential which, by acknowledging their value and contributions, can aid in career advancement.
- Internationally recognized standards of architectural competence provide employers with a useful filter for potential hires, and supplementary criteria for selecting the most qualified individuals for critical roles and responsibilities, as well as provide a clear career path for employees.

"THE OPEN GROUP'S TOGAF AND ITAC CERTIFICATIONS PROVIDE MULTIPLE BENEFITS TO THE IT ARCHITECTURE COMMUNITY: STANDARDS DEVELOPED VIA AN OPEN, MULTINATIONAL PROCESS REPRESENT A CONSENSUS AS TO THE INDUSTRY'S BEST PRACTICES; INTERNATIONALLY RECOGNIZED STANDARDS FOR IT ARCHITECT CERTIFICATION PROMOTE THE DEVELOPMENT AND RECOGNITION OF THE IT ARCHITECT PROFESSION AND, THEREBY, RAISE THE BAR FOR QUALIFICATIONS ACROSS THE ENTIRE INDUSTRY; AND CERTIFICATION PROVIDES PROFESSIONALS WITH A PORTABLE VENDORINDEPENDENT CREDENTIAL VERIFYING THEIR EXPERIENCE AND COMPETENCE, A CREDENTIAL WHICH, BY ACKNOWLEDGING THEIR VALUE AND CONTRIBUTIONS, CAN AID IN CAREER ADVANCEMENT."

- To assure quality of service, clients can require staffing by certified IT architects in requests for project proposals, procurement specifications, and service-level agreements.
- Solutions providers deploying certified IT architects through their service organizations will hold a competitive advantage as procurements increasingly specify certified practitioners as a requirement. This is happening to the project management profession and can be expected to happen to the IT architecture profession as well.
- All parties benefit from the ease with which the credential can readily be verified via The Open Group Certification Directory.
- Organizations with Accredited Certification Programs gain credibility and increased stature with clients, partners, and employees.

More information on The Open Group's architecture-related activities and its certification programs can be found at The Open Group's Web site: http://www.opengroup.org.

Information about the Association of Open Group Enterprise Architects can be found at the AOGEA's website: http://www.aogea.org.

About the Author

Len Fehskens is the VP, Skills and Capabilities for The Open Group. Len joined The Open Group in September 2007 after 23 years with Digital Equipment Corporation, Compaq Computer Company and Hewlett-Packard, where he led the worldwide Architecture Profession Office for HP Services. Len majored in computer science at MIT, and has over 40 years of experience in the IT business as both an individual contributor and a manager, within both product engineering and services business units. He is the lead inventor for six software patents on the object-oriented management of distributed systems.

The Need for an Architectural Body of Knowledge

by Miha Kralj

Summary

An important step toward defining IT architecture as a stand-alone profession is a clear definition of knowledge areas of the new discipline. A well-articulated body of knowledge will drive the recognition and growth of the discipline, and helps ensure that the title of IT Architect is used only after the necessary competence is acquired and verified through formal qualifications which could be regulated by professional bodies. This article covers why an Architectural Body of Knowledge is an important building block in professionalization of IT Architecture and how the Microsoft Certified Architect (MCA) community drives the creation of Architectural Body of Knowledge (ArcBOK) through its Special Interest Group (SIG).

Architecture in IT

Systems in IT are becoming more and more complex, so it is no surprise that we are witnessing the rise of a new profession in IT, loosely called IT Architecture. Let's ignore the name for the moment and focus on the problems this profession tries to solve.

Defining and designing complex structures is a common activity performed by almost every discipline, profession, and artisanship throughout the centuries. All the disciplines of old discovered that skills and knowledge required for the composition of large complex systems don't match the skills that are required for small bottom-up assembly activities. In IT, the same problem became noticeable about 10 years ago, and the gap between core engineering and high-level system design has grown ever since. Grady Booch's aphorism, you can't build a sky-rise the way you build a doghouse, encapsulates the common dilemma facing high complexity, high interdependency, and low transparency projects: The sheer amount of detail required in complex compositions is so overwhelming that a function of analysis, decomposition, and abstraction becomes vital for the success of such endeavors.

In the structural construction business, architects branched away from civil engineers and construction workers many centuries ago. They were (and still are) groomed, educated, and taught quite different skill sets than their engineering counterparts. If you would ask a civil engineer what a building is, the definition will focus on thickness of the walls, angle of the roof, sturdiness of beams and type of concrete required for house fundaments. Architects on the other side will describe the house as a wrapper around the living space, nested into the environment that allows the inhabitants to do whatever they intend to do in the house.

Below is a sample curriculum for a four-year structural architecture program of study:

- History and Theory of Architecture
- Building Design and Construction
- Materials and Methods
- Architecture Design
- Theory and Method in Architecture
- Structural Systems
- Site and Urban Design
- Space and Composition
- Types of Structures
- Preservation and Restoration
- Heating, Cooling, and Lighting
- Human Settlement Patterns
- Construction Estimating
- Project Planning and Feasibility
- Environmental Systems
- Architectural Internship.

Obviously, the knowledge acquired throughout the study of architecture is diverse and often overlaps other professions or arts. It is understood and accepted that one architect doesn't have and doesn't need to have a total knowledge of architecture–interior designers, for example, will use a different subset of knowledge than urban or landscape architects.

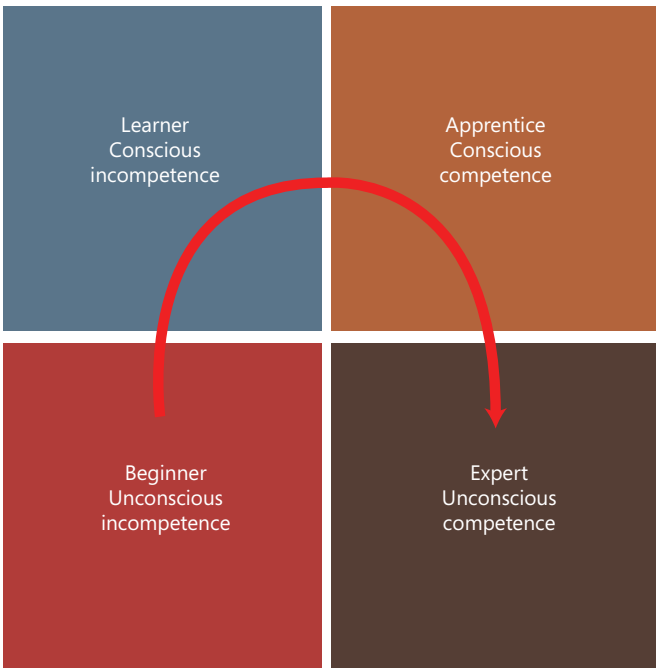
How does that translate to IT, where we have borrowed the name and title of architecture? Our modern profession has not had centuries to diversify and evolve naturally. It seems that every high-complexity IT endeavor is now called architecture instead of engineering. In the words of Alan Cooper: "[nowadays] Web designers are called programmers, programmers are called engineers, engineers are called architects, and [true] architects are never called."

The Importance of Knowledge

According to Scott B. Parry, a competency is defined by four characteristics:

1. A cluster of related knowledge, attitudes, skills, and other personal characteristics that affect a major part of one's job,

Figure 1: Stages of competence



2. *Correlates with performance on the job,*
3. *Can be measured against well-accepted standards,*
4. *Can be improved via training and development.*

From the perspective of knowledge growth, the most important is the fourth characteristic—the ability to learn and improve. Let's look at the stages of competence a person typically goes through as knowledge is internalized and put to use during daily work (Figure 1):

Beginner

A person that is not aware of the existence or relevance of a certain skill area, or even denies the relevance or usefulness of the skill, is called a beginner. Until of the beginner recognizes a deficit, it is not possible to improve the skill, so we can say that person is incompetent without knowing it.

A cohesive collection of available knowledge areas for a profession would help beginners identify deficits so that they can determine how to acquire the skills and become learners.

Learner

A learner is aware of the existence and relevance of the skill; the deficiency in this area is often exposed through trying and failing to perform a missing skill and generates a thirst for knowledge. Ideally, a learner makes a commitment to learn and practice the new skill until the adequate proficiency level is met.

People at this stage urgently need to sources of relevant knowledge and training. A reference index of learning resources could direct them to the best sources.

Apprentice

When a certain skill can be performed reliably and at will, the stage of conscious competence is reached. Apprentices need to concentrate to perform the skill deliberately; the apprentice still lacks intuitive command of the skill. The knowledge is gathered but requires practice to become "second nature."

Concentrating and thinking about the skill requires frequent reminders and guidelines. A single source of information to help apprentices follow the steps, it would shorten the time required to develop unconscious competence.

Expert

This is the stage when a skill is used without a second thought, just like driving, swimming, or skiing. It becomes so natural that the decision to use it is not conscious; this is the mastery stage when a skill starts to turn into art and the expert can turn into a teacher.

Teaching something that has become second nature can be difficult. People who have been experts in specialized domains for a long time sometimes have difficulty explaining basic concepts. A coherent study guide with rationale behind each skill would also serve experts as a useful teaching aid.

IT Profession and Specialized Knowledge

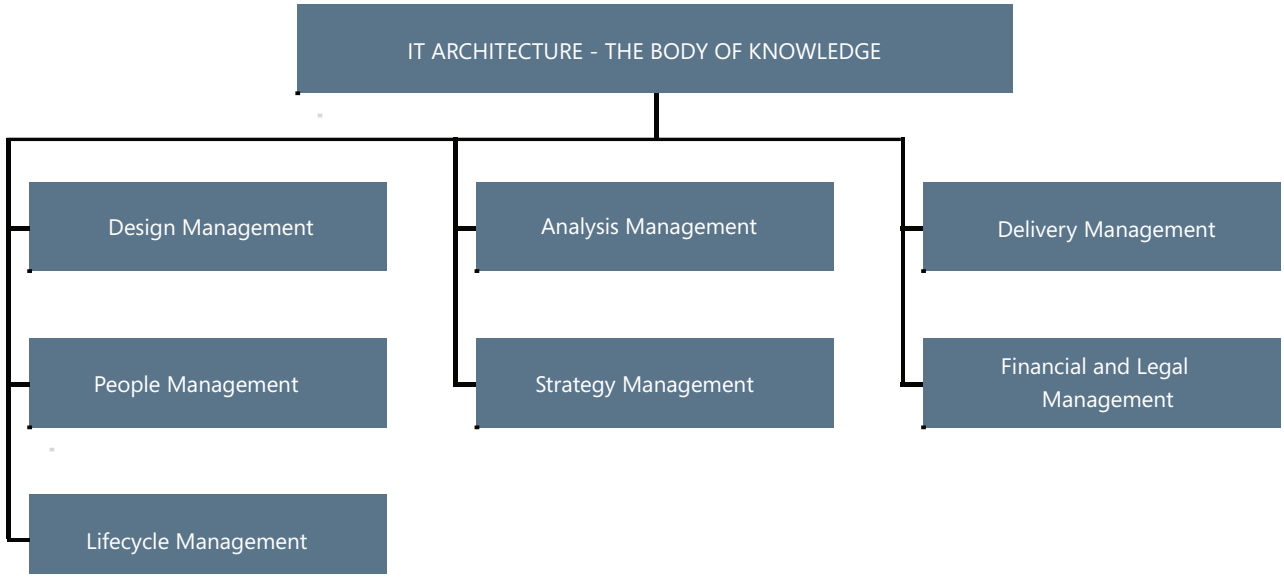
IT architecture is gradually becoming a stand-alone profession, branching away from engineering and software development. As a vocation and a prospective career, it has its own specialized body of knowledge that will make it different from other professions.

"HOW DOES THAT TRANSLATE TO IT, WHERE WE HAVE BORROWED THE NAME AND TITLE OF ARCHITECTURE? OUR MODERN PROFESSION HAS NOT HAD CENTURIES TO DIVERSIFY AND EVOLVE NATURALLY. IT SEEMS THAT EVERY HIGH-COMPLEXITY IT ENDEAVOR IS NOW CALLED ARCHITECTURE INSTEAD OF ENGINEERING. IN THE WORDS OF ALAN COOPER: '[NOWADAYS] WEB DESIGNERS ARE CALLED PROGRAMMERS, PROGRAMMERS ARE CALLED ENGINEERS, ENGINEERS ARE CALLED ARCHITECTS, AND [TRUE] ARCHITECTS ARE NEVER CALLED.'

But having specialized knowledge is not enough if IT architecture is to become a respected and sought-after discipline, on par with the other disciplines in computer sciences. The title IT architect should be acquired after achieving a defined level of competence through practice and experience, proven through some sort of formal qualification, and perhaps regulated by professional bodies, which would then protect the reputation and code of practice.

Why such rigor you may ask? With the every nascent profession, there is a risk of using the new terms—names, titles, or accreditations—without controls and verification. Currently, the title architect is used to describe everything from distinguished engineers to developers, from senior consultants to experienced sales specialists.

Figure 2: Facile model of the ArcBOK knowledge areas



Generally Accepted Knowledge

IT architecture is an emerging and quickly evolving profession, so there are many areas that are not yet accepted as mainstream. An ArcBOK should focus on identifying and describing all the knowledge and only the knowledge that is generally accepted in the architectural community.

What is "generally accepted" knowledge? The Project Management Institute in its Guide to the Project Management Body of Knowledge (PMBOK) defines generally accepted knowledge for project management in the following manner:

"Generally accepted" means that the knowledge and practices described are applicable to most projects most of the time, and that there is widespread consensus about their value and usefulness. "Generally accepted" does not mean that the knowledge and practices described are or should be applied uniformly on all projects; the project management team is always responsible for determining what is appropriate for any given project.

In the IT architecture we have another degree of complexity: There are many flavors of architects, and more architectural subdisciplines sprout each year. The generally accepted knowledge of a typical solution architect is quite different from the generally accepted knowledge of an enterprise architect or security architect. The ArcBOK should encompass all of these yet make a clear distinction of which area is core and which area is supportive, depending on the architectural subdiscipline.

The Body of Knowledge

We need to be very precise with the definition of the ArcBOK: It should be the total sum of all available knowledge in the area of IT architecture, classified by the appropriate taxonomy of knowledge areas. Development and recognition of a core body of knowledge is essential to the development of the profession, accreditations, and university curricula.

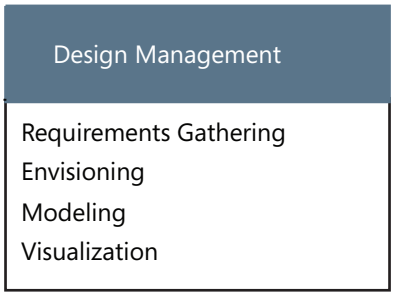
IT evolves so fast that capturing the architectural knowledge itself would make the ArcBOK obsolete even before it would be consolidated, reviewed, and published. Instead of capturing and republishing the knowledge itself, the ArcBOK should become a metaknowledge reference base, with a complete 360-degree view of the reference material required to adequately perform the job of IT architect.

The process of building the ArcBOK should follow the consensus-building process, asking the community and professional bodies for feedback and comments. It should be divided and subdivided into knowledge areas, the major components of a discipline, or subfields of study.

The following example is a facile model of knowledge areas of the ArcBOK (Figure 2):

- **Design Management**—activities related to requirements gathering, modeling, visualization, and communication of IT designs
- **Analysis Management**—activities related to analysis, deduction, innovation, creativity, and problem solving
- **Delivery Management**—activities related to project, engagement, transformation, development, planning, coordinating, and quality management

Figure 3: Design management knowledge area



"WHAT IS GENERALLY ACCEPTED KNOWLEDGE FOR IT ARCHITECTURE? THERE ARE MANY FLAVORS OF ARCHITECTS, AND MORE ARCHITECTURAL SUBDISCIPLINES SPROUT EACH YEAR. THE GENERALLY ACCEPTED KNOWLEDGE OF A TYPICAL SOLUTION ARCHITECT IS QUITE DIFFERENT FROM THE GENERALLY ACCEPTED KNOWLEDGE OF AN ENTERPRISE ARCHITECT OR SECURITY ARCHITECT. THE ARCBOK SHOULD ENCOMPASS ALL OF THESE YET MAKE A CLEAR DISTINCTION OF WHICH AREA IS CORE AND WHICH AREA IS SUPPORTIVE, DEPENDING ON THE ARCHITECTURAL SUBDISCIPLINE."

- **People Management**—activities related to leadership, organizational politics, stakeholder, and relationship management
- **Strategy Management**—activities related to defining the business intent, enterprise strategy, and roadmaps
- **Financial and Legal Management**—activities related to billing, sourcing, legislation, and procurement
- **Life-cycle Management**—activities that focus on various stages of the IT life cycle, including envisioning, SLA management, change management, and IT decommissioning

Each knowledge area should be divided into knowledge competencies, specific to that area, and each competency should get the list of resources available.

For example, one core knowledge area for architects is design management. Our example divides the area into four competencies and lists the various techniques, frameworks, tools, and skills for each competency (Figure 3, page 19).

This sample model is by no means verified or accepted by the community; it is just a teaser to gather momentum and invite the participation.

Potential Misuses

We have looked into the benefits of having the ArcBOK as the daily reference for architects; it is also worthwhile to discuss potential misuses of such knowledge collection.

The most obvious misuse of ArcBOK would be the idea that someone must know everything that is in the book in order to use the title architect. You can imagine abuse scenarios, such as being denied a promotion by a small-minded manager because you didn't demonstrate a competency from a remote subdiscipline, or being required to cite whole passages from the ArcBOK during an interview as the proof of architectural knowledge. Such misuse is happening with PMBOK, so it would be naïve to think it couldn't happen with ArcBOK.

Another foreseeable misuse is the premise that knowing ArcBOK would make someone an architect. We all know how common cramming for MCSE exams is, where candidates memorize useless information by heart just to pass the MCP test. The potential pitfall of ArcBOK could be that candidates for MCA or other architectural certifications would cram the ArcBOK in hope that this would be enough to pass the review board. The ArcBOK should not become an "MCA for Dummies" guide and should be very explicit about that.

Putting any sort of measuring scale on top of ArcBOK would be another potential misuse. The knowledge areas are diverse and non-related, it would be wrong to evaluate and average competencies against a unified scale—"I'm 4 in Modeling and 2 in Trade-off Analysis, so my average architecture index is 3" would not be a useful measure of anything.

Call for Action

As it would be with any body of knowledge, building the ArcBOK must be a group endeavor, requiring the consensus of many practicing professionals, in IT architecture and related professions. The MCA community has formed a SIG to work on ArcBOK. If you are interested in participating, have an idea or would just like to know more about the project, please register your interest by emailing me at: miha.kralj@microsoft.com. You don't have to be a certified MCA yourself; as long as you have personal and professional interest in IT architecture, your participation is more than welcome.

Why am I asking for the registration of your interest? There are several ways that work on the ArcBOK could progress. The level of interest based on your feedback will determine which course we take:

- If nobody is really interested, aside from a few architects inside MCA community, we'll continue our slowly progressing work by emailing the drafts to each other.
- If there is interest to read and use the ArcBOK but not to participate in its creation, a blog or some other form of publishing the work-in-progress will be considered.
- If there is indication that many enthusiasts would like to add their opinions and gathered knowledge, a wiki or similar collaborative tool will be launched to support the effort.

Summary

The Architectural Body of Knowledge is a big piece of work and requires strong community support both to build and endorse it. The profession of IT architecture must compose such work sooner or later to raise the quality bar. When the time is right, all pieces of ArcBOK should come together with very little effort.

Resources

"Just what is a competency?" Scott B. Parry, training material, 1998.

About the Author

Miha Kralj is an architect in the Industry Solutions Group, part of Microsoft Enterprise Services organization. His consultancy tenure started in Europe and extended to South Pacific where he worked as a solution architect and enterprise strategy consultant. He has infrastructure background and is a certified MCA architect. Email Miha about the ArcBOK at miha.kralj@microsoft.com.

Microsoft®
tech·ed
North America | 2008

Be a part of the experience.

Microsoft Tech·Ed North America 2008 will kick off the Worldwide Tech·Ed 2008 series, and you're invited.

For the first time, **Tech·Ed North America is evolving from a one-week conference to two separate back-to-back conferences:**

Tech·Ed North America 2008 Developers, June 3–6, in Orlando, FL

Tech·Ed North America 2008 IT Professionals, June 10–13, in Orlando, FL

The two-week format will continue to provide the same experience you expect but with more learning opportunities:

1. Increase your knowledge of Microsoft technology through **16 Technical Tracks for Developers** and **20 Technical Tracks for IT Professionals**
2. Hear the **Bill Gates Keynote** for Developers and the **Bob Muglia Keynote** for IT Professionals
3. Get **valuable hands-on training** in the Technical Learning Center
4. **Connect with over 250 industry partners** in the Partner Expo

Don't miss this opportunity to learn how Microsoft can help you advance the goals of your organization – and your career.

Register today at microsoft.com/teched2008

Additional worldwide conference dates will be available soon via **microsoft.com/teched**

Developer Tracks
- Architecture
- Business Intelligence
- Database Platform
- Developer Tools and Languages
- Development Practices
- Dynamics
- Infrastructure for Developers
- Microsoft IT
- Office and SharePoint®
- SOA and Business Processes
- Software-Plus-Services
- Unified Communications
- Web and User Experience
- Windows® and Frameworks
- Windows Embedded
- Windows Mobile®

IT Professional Tracks
- Application Development for IT Professionals
- Business Intelligence
- Database Platform
- Dynamics
- Identity and Access
- Integrated Product Solutions
- IT Managers
- Management
- Microsoft IT
- Office and SharePoint®
- Security
- SOA and Web Infrastructure
- Software-Plus-Services
- Specialized Servers
- Unified Communications
- Virtualization
- Windows Client
- Windows Embedded
- Windows Mobile®
- Windows Server® Infrastructure

A Study of Architect Roles by IASA Sweden

by Daniel Akenine

Summary

In this article, we first examine why there is a need for IT architects. We then describe a study undertaken by IASA Sweden to better understand IT architecture, which entailed a process of mapping the artifacts produced by architects at different levels in an organization. Finally, we discuss four architect roles that IASA Sweden, as a result of the study, recommends for a typical organization.

Why we need architects

As you are well aware, IT roles in general are relatively new compared to other professions and the associated responsibilities have been evolving over the past decades. This is not surprising as the use of IT and software has changed continuously since computers and IT systems were introduced. IT and software have been integrated into many people's lives, and technology is easier to consume and use than ever before. Does that mean IT systems are less complex than they used to be? There are really two trends going on in the industry:

- IT gets easier and easier.
- IT gets more and more complex.

With increasingly powerful tooling and modeling, producing new software solutions gets easier and faster. You can create an advanced SOAP-based service in less than a minute. On the opposite side, however, the fundamental architecture of these solutions is more complex than it used to be. Why? Well the simple answer is because we expect much more from today's software — it has to do more complex things.

Many years ago, I got a telephone call from a friend. Suddenly, he told me he was strolling around in the city center while talking to me. I was amazed. Obviously I had heard of mobile phones before but I had never actually spoken to somebody using one. Not so many years later, a mobile phone is an integrated part of my life. The technology does not amaze me anymore; on the contrary I get very annoyed when the technology does not perform as I expect it to. Traveling by train in a tunnel, I still expect the mobile phone to work even though I know this is a big technology challenge. My expectations just get higher and higher. This is the exact same way software works. Twenty years ago, we accepted a simple editor for text processing; today we expect a sophisticated software solution that checks spelling, formats, visualizes, collaborates, and so on. Think about it, a simple modern distributed application probably has a more complex architecture than the high-end COBOL application running in the heart of your bank of choice. This is one of the reasons why IT architecture has a great future; to control and master these modern, complex, and integrated solutions.

In this context, let's consider some of the reasons often given for why architects are important.

We need architects to deal with complexity

Modern distributed solutions are more complex today than they used to be. There are really two types of complexity in software solutions: fundamental complexity and accidental complexity, as described by Fred Brooks in 1986. Fundamental complexity is embedded into the business problem that the software is going to support. Some business problems presented as complex can actually be simplified; in other cases, the business complexity is inescapable. Software solutions for fundamentally complex business problems need a skilled architect to understand and model the problem correctly.

Accidental complexity is different; it springs from the technological and architectural choices made while solving the problem—for example, how much flexibility for change should the solution have? Decisions on architectural qualities, such as manageability, scalability, security, and reusability, must be made with input from multiple stakeholders whose needs may be at odds. These decisions determine the accidental complexity, and balancing them correctly is one of the most important tasks for an architect.

We need architects to deliver business agility

The idea that IT can deliver business agility is frequently heard, but has the correlation been proven? Certainly, bad architecture can make a business inflexible and slow to act. Many large organizations have invested in IT for many decades now. In the last 10 years, investments in IT have been driven largely by the shockwaves from the Internet revolution, a revolution that has changed the way organizations interact with their customers; generating new types of information in documents, emails, portals, and so on. However, many of these investments were made with little care to any enterprise-wide strategy or architecture and some organizations now refer to the last decade of IT investments as a "Software Crises." We are experiencing the cumulative effects of large numbers of short-term and isolated decisions. The cost of changing the software solutions to support new innovative processes or business capabilities is getting out of proportion. To make small changes in the business model or process takes so long or costs so much that they are simply not done. This is an example when IT is bad for business agility and in the end we need to fix that. Who is better suited to fix that than an architect?

We need architects to reduce the cost of IT

As others such as David Anderson have written, the cost of managing your existing IT is often as high as 70-80 percent of your total IT budget. More often than not, projects under time pressure result in decisions which multiply life-cycle costs during operations and maintenance. Disparate solution designs and technologies from different projects drive total costs and dilute skill sets. Complex and unknown interdependencies lead to unexpected consequences and costs. Architects can make a real difference in cost reduction; by architecting quality solutions for manageability and supportability.

We need architects for better business alignment

In the early days, we used IT primarily to automate manual processes. We used IT to lower costs on information transactions and to store and query data in ways that had not been possible before. However, those quick-wins will someday come to an end. Some organizations have already used IT very effectively to automate and support their business processes, and now they want to move on. It is time to do more sophisticated IT investments aligned with the business strategy and actually use IT to differentiate a company from its competitors.

We need architects to create long-term and short-term strategies for how to use IT in the most effective way for the business.

A Process to Map Artifacts to Levels

Clearly, we need architects. However, architects do not play the same role in every project or company. What kinds of roles should they play, and how can they work with other architects and team members effectively? What we need is a framework for delivering new IT capabilities that supports and improves the current business model, in which different kinds of architects can work together in a consistent way.

The first problem you encounter when you start discussing different roles is the number of perspectives and roles out there. IASA found more than 50 different roles: Although many roles were more or less equivalent, many organizations have created their own roles with unique deliverables. This causes a lot of problems. For instance:

- The same architect role has different deliverables between organizations.
- Organizations may have different roles but they produce the same deliverables.

How can an architect determine best practice, get training, and have a career in the profession when the roles are so murky?

In June 2007, IASA Sweden put together a working committee with different types of architects from different sectors of the industry with the goal to create general recommendations for more consistent architect roles. To avoid being locked into specific roles too early in the process the committee decided to focus on something that turned out to be very similar between organizations: the artifacts that architects deliver, such as use cases, IT strategies, security strategies, deployment models, and so forth.

The group found approximately 40 different artifacts that could be agreed upon. The next step was to map those artifacts into three different levels in an organization to see if they would form some kind of natural cluster. The levels were defined as:

"ARCHITECTURE IS BY NO MEANS THE EXCLUSIVE DOMAIN OF ARCHITECTS. EVERYONE INVOLVED IN THE DEVELOPMENT OF A SYSTEM CONTRIBUTES TO ITS ARCHITECTURE AND MAY POSSESS SOME OR EVEN ALL OF THE ARCHITECTURAL SKILLS AND MINDSET. WHERE THE ARCHITECT ROLE CONTRIBUTES IS IN IMPROVING THE CONSISTENCY AND EFFICACY OF THE RESULTING DE FACTO ARCHITECTURE THROUGH TAKING DIRECT RESPONSIBILITY FOR ITS QUALITIES."

—PONTUS GAGGE, SANDVIK CORP.

- Level 1: Architects create strategies together with the business on how to use IT in the business in smart way. Policies and principles are created here that influence the whole organization. Examples of level 1 artifacts include city plans and strategies.
- Level 2: Architects create artifacts which support the mapping between business and technology. At this level, architects try to understand the processes of the organization and how they can be improved using IT capabilities. Examples of level 2 artifacts include process maps and service maps.
- Level 3: Architects produce artifacts that model the technical architecture, trying to create good solutions, which are cost-effective, scalable, flexible, and so forth. Examples of level 3 artifacts include application models and data models.

These levels do not necessarily map to architect roles in a simple way — an architect role can operate on several levels. As an example, an enterprise architect can work on both high-level policies as well as technology polices if they are considered to be important for the whole organization.

After six months of workshops and reviews, the committee released its recommendation for four architect roles, based on the three levels and 40 artifacts:

- Enterprise Architect
- Business Architect
- Solution Architect
- Software Architect

The four roles overlap but do not necessarily report to each other (see Figure 1, page 24).

Architect Roles

Let's explore the definitions and significance of the committee's recommendations.

Enterprise Architect

Typical artifacts include: IT strategies, capability maps, city plans, integration strategies, as-is/to-be analysis, architectural principles, gap

analysis, life-cycle analysis, application portfolio strategies.

Description: The mission for an enterprise architect (with IT focus) is to support the business strategy of the organization with IT solutions and information. The enterprise architect, or a group of enterprise architects, should be responsible for the overall strategy regarding IT capabilities as well as to ensure that the IT architecture is cost effective. In some organizations, it may be appropriate to use enterprise architects in governance functions and to regulate enterprise-wide standards for communication and messaging. In other organizations these responsibilities may be delegated to specific governance or integration centers, where the enterprise architect is a representative. The role often reports to a CIO or to a chief architect. An enterprise architect owns strategies at several different levels in the organization — from standards that have a global impact on the organization or strategies regarding things like security or infrastructure.

A classic analogy is to compare the enterprise architect to a city planner who, using strategy, planning, and regulations, is responsible for different functions in a city that must work together effectively.

The use of enterprise architecture is still immature in most organizations. As IT departments evolve and become more closely aligned with the business departments, we expect the profession to mature and become clearer over the years to come. We believe that growth in tooling and evidenced-based best practice from the scientific community will also influence this profession.

Competence: Deep knowledge in both business and IT; leadership and negotiation skills; experience in governance, project management and economy; knowledge in enterprise architecture and business modeling.

Business Architect

Typical artifacts include: process maps, use cases, information models.

Description: Business architects work very close to the business and understand in detail how the organization works. They are active in modeling processes in the organization and support solution architects with analysis and requirements on new or existing solutions. They understand how the IT systems support the business and suggest improvements together with enterprise architects. Business architects are active in ongoing projects in the organization using their influence to ensure that projects deliver benefits to the business in an optimal way.

Business architects are often involved in areas related to general business and process improvements. The business architect is also a very important resource in every IT project in the organization.

Competence: Deep knowledge in the business; process modeling; requirement analysis; workshop leader skills.

Solution Architect

Typical artifacts include: application diagrams, system maps, service interfaces, technical interfaces, integration strategies.

Description: A solution architect works with the design of IT solutions based on requirements from the business, making use of existing IT capabilities and services in the organization.

Solution architects have a special responsibility to reuse existing functions and services. They align new solutions to the current architectural principles regarding standards and integration in the organization. They balance the functional and nonfunctional requirements with necessary prioritizations and compromises. The goal for the solution architect is the success of the current project, in addition to how well the project aligns to the architectural principles and how well it reuses existing capabilities.

When organizations move from traditional applications to integrated solutions and services, the role of the solution architect becomes more and more important. The role of the solution architect is clearer in larger projects, particularly when many systems are involved. If the project is small or the application is isolated, this role may not be necessary in the particular project.

One could also argue that the solution architect is a natural evolution of the traditional system architect role. Moving away from systems to solutions generates new competencies and responsibilities.

Competence: broad and general technical knowledge, as well as deep competences in things like infrastructure, data models, service orientation; good understanding of enterprise architecture.

Software Architect

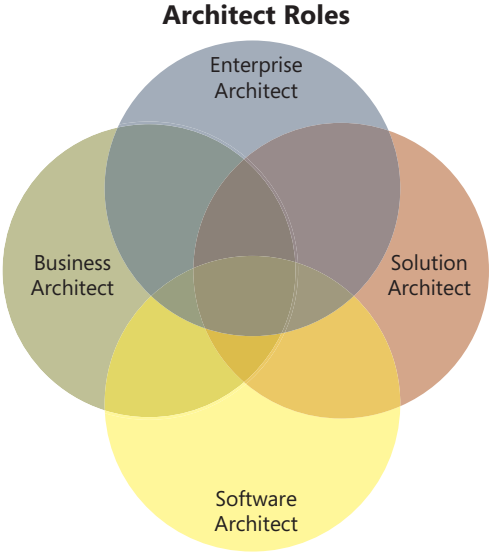
Typical artifacts include: frameworks, class models, patterns, aspects.

Description: A software architect works with the structure and design of software systems. Software architects work with both functional requirements as well as different architectural quality attributes such as flexibility, performance, reusability, testability, and usability. Some quality attributes obviously may be shared with the solution architect. They prioritize and optimize the different quality attributes with respect to cost and other constraints. The focus for the software architect is primarily the current project, whereas the solution architect has a wider focus to reuse existing assets, policies and regulations, although this could apply to the software architect as well if the organization finds it necessary to have strict guidance in place for software development.

The role of the software architect becomes more and more important as the complexity of systems continues to increase.

Competence: deep knowledge in programming, frameworks, standards and technical modeling.

Figure 1: Architect Roles; note that the roles do not necessarily report to each other.



Biggest Challenges

To get a broader perspective, we also want to bring forward some of the challenges concerning the roles for architects. For this, we turned to Sten Sundblad, chief architect at Sundblad & Sundblad, and Pontus Gagge, an enterprise architect at Sandvik Corp., to ask them what they think are the biggest challenges.

Sten: *I believe that the most important challenge is about helping achieve business agility.*

When a business is challenged from the outside, it often needs to change parts of its business strategy. Almost without exception, this means that it has to change its business architecture (meaning its business processes), too.

One of the most often mentioned reasons for delaying the change of a business process is that the needed software changes take too much time. And this is almost inevitable unless the architecture of the software involved is well aligned to the business architecture. So, business agility requires close alignment of software architecture to business architecture.

Achieving this close alignment between software architecture and business architecture is what service-oriented architecture (SOA) is about. SOA is not a way to structure an application; it's a way to successively — over a number of years — structure the entire portfolio of business software.

Achieving SOA is not possible without enterprise architecture, business architecture, and software architecture (which is my preferred term for what IASA Sweden talks about as solution architecture). A vision of the electronically serviced enterprise is needed, and this vision should guide business architecture and software architecture so that every new solution becomes a piece in the enterprise architecture's puzzle.

To succeed with this alignment between software and business architecture, a lot of cooperation between the different roles are needed. This is especially true about the business and software architects. Business architecture is mostly about architecting the business, making it perform better than before. Software architecture is a lot about mapping business architecture to a business-oriented software solution. This requires, for most business architects, a higher level of software savvy, and for the software architect, a higher level of business savvy, than is often the case.

Anyway, this is where the challenge lies: How could we make business architects and software architects work better together, and how could we convince the agile development community that architecture is not just okay but a requirement for the development of an agile and competitive business? that there's no conflict between agile development and having an agile development project be restricted by architecture? and that there's no reason that software architecture can't be established in agile projects well before any coding takes place. If we who believe in business and software architecture can succeed in this, then we have met this challenge, and then there's a great future for business and software architecture. If we can't, then I'm afraid many businesses will be less agile, and less competitive, than they could have been.

Pontus: *Sten addresses the perennial goal of aligning business and IT. Whether we believe components, services, events, DSLs, MDAs, lean development or whatever paradigm currently espoused — indeed, even architecture and enterprise architecture itself — are essential to the future, pragmatic architects must be able to recognize the kernel in each approach and make them work toward the ultimate objective within the context of their own organizations. This emphasizes how important the ability to shift perspectives and to communicate always is to architects.*

I would say the direct challenge to defining architect roles is the architects themselves. As a group, we tend to a wider scope and interest than is common, and as generalists we can find ourselves in any number of supporting roles that, while worthwhile from a business perspective, strictly speaking do not contribute to architecture in the sense we are after. However, we need to keep in mind that a role is a 'hat, not a head' — the role describes our current engagement, not the totality of our interests and skills, and that generally, we will have areas where we are comparatively stronger. The roles provide focal points both for our engagements and employments, and for our career development.

While the role of the narrowly focused software architect may be a reality, in the long run I believe all software architects should strive to encompass the objectives of the solution architect, blurring the line between the roles as described. Architectural guidance should come from below as well as from above, and who better to say in what ways technology is to be employed than the architect with the most direct hands-on experience?

I would also emphasize that architecture is by no means the exclusive domain of architects. Everyone involved in the development of a system or an organization contributes to its architecture and may possess some or even all of the architectural skills and mindset. Where the architect role contributes is in improving the consistency and efficacy of the resulting de facto architecture through taking direct responsibility for its qualities.

Conclusion

As we mentioned before, not all architect roles work only with IT. The final goal for an architect is to help improve the business, support its mission, and if possible innovate the business using IT to make the organization more competitive as a result.

IASA Sweden here presents one way of aligning business to IT by collaboration in distinct and clear architect roles, all working together with the same mission.

Resources

"No Silver Bullet - essence and accident in software engineering," F.P. Brooks, Proceedings of the IFIP Tenth World Computing Conference, pp. 1069-1076, 1986

Design for Manufacturability: Optimizing Cost, Quality and Time-to-Market, David M. Anderson (Second ed., CIM Press, 2001)

Enterprise Architecture as Strategy, Jeanne W. Ross, Peter Weill, and David C. Robertson (Harvard Business School Press, ISBN: 1-59139-839-4)

IASA
http:// www.iasahome.org

2xSundblad Online Architect Academy
http://www.2xsundblad.com

About the Author

Daniel Akenine is a chapter president at IASA and an architect evangelist at Microsoft. He has a history as a physicist and researcher in neuroscience before he joined the IT industry 10 years ago. He is the cofounder of two R&D companies and holds patent applications in applied cryptography in the U.S. and Europe. During the last 10 years, Daniel has been working as a developer, senior architect, and CIO before he joined Microsoft in 2004.

The Softer Side of the Architect

by Joe Shirey

Summary

Most of us would agree that having strong technology skills is a key ingredient to being a competent architect. However, the most successful architects that I have met possess more than just great technical skills. They also have qualities that enable them to work well with people.

Developing and refining these "soft skills" can take ordinary architects to new levels of effectiveness in their careers. This article outlines a framework I developed for defining these soft skills and strategies for the architect based on my experiences and interactions with architects I admire.

Why Soft Skills are Relevant

We have all worked with one of them at some point in our careers—brilliant technologists who can solve just about any technical problem, but are so arrogant and insufferable that people despise working with them. Some of these übertechs will reach a point in their careers where they no longer get promoted and their value is marginalized from a business perspective. The reason they hit this ceiling is due to their lack of the "people" and "business" skills that we often call soft skills.

If you look at the Microsoft Certified Architect program, you will notice that there is a set of competencies that go well beyond technology skills. These competencies were based on focus groups from companies large and small. One common theme from these focus groups was that the soft skills really matter. In fact, they identified more soft competencies than technical competencies. In their view, the soft skills are what separate the highly skilled technologist from the true architect.

The International Association of Software Architects (IASA) has also gone through a detailed analysis and polled its members to determine the skills necessary to be a successful architect. Of the five foundational skill areas the IASA identified, two are based in these soft skills.

It is apparent that attempts to define the critical skills of an architect contain some measure of soft skills. In my experience this is quite true; the most successful architects I know are able to increase their effectiveness by combining their technical and nontechnical skills.

In my mind, the successful technical solution also requires three distinct soft skills: business alignment, perspective awareness, and communication (Figure 1). Most architects acknowledge the importance of these areas yet fail to make them real priorities during the project life cycle. In this article, I'll take a look at each of these areas and offer strategies for ensuring that they remain priorities during the project without significant additional effort.

Business Alignment

I doubt anyone would deny that business alignment is a critical success factor for any project. Most projects begin with some type of requirements document that drives most of the technical decisions or at least an architecture document that demonstrates how the architecture meets business needs. The issue generally isn't lack of intent but with the alignment at the strategic level.

In my different roles, I have had the opportunity to review many projects and discuss them in detail with the architects. Usually, the architect can discuss the business requirements, but it is surprising how often the architect cannot explain the project in terms that the CFO would understand. There is a lack of understanding of the real business drivers and the detailed financial implications versus the business requirements. It is the critical factor that drives the real project decisions.

When I mention business drivers, I mean tangible and measurable items. Often, an architect will describe the project drivers as "increase customer satisfaction" which is quite nebulous. I believe that business drivers should boil down to financial terms when possible. In the above example, I would recommend a goal such as "increase repeat customer revenue by 10 percent and reduce call center support costs by 5 percent over the next year" with detail to support why these goals are attainable based on the project. These types of objectives will drive the true project benefits and can frame what a project should cost.

There are some projects that are not driven by purely financial terms, but there is always an underlying business reason for the project. For example, compliance projects often are not done based on a pure return on investment model. However, it is important to understand the business risks associated with not doing the project and use those risks to develop the appropriate solution.

By understanding the actual business drivers, the architect is able to make rational technical decisions during the project life cycle. Because many projects are based on business requirements, we treat them as a "contract." We sometimes become inflexible about changing requirements because we have based all of our plans on that "contract," but business situations and drivers can change during the course of a project. If we possess a true understanding of the business aspects, then we can help the business side of our organizations understand the implications of those changes in our solution development.

I often think back upon a project that I worked on for a niche software vendor a couple of years ago in which the customer was pushing us to complete a fairly large and complex project in a timeframe that was unrealistic for building a high quality and maintainable solution. After many heated discussions about the situation, we started to understand that the date was not arbitrary — it was about the company's ability to survive. If the solution wasn't done by a certain date, they would likely go out of business. That sense of urgency certainly drove the architecture of the solution (or some might say "lack of architecture"). We had to sacrifice quality and maintainability for project schedule. The CEO was aware that he would have to pay the price down the road, but that was better than having to shut down the business.

In that example, I certainly wasn't very proud of the technical solution, but we solved the customer's problem and kept the business afloat, which was really a loftier goal. We also took the appropriate steps to ensure the customer understood the trade-offs made during the project and the future implications. Because we took the time to fully understand the issues that drove the business, we moved into a strategic rather than a tactical role.

In my opinion, agile methodologies are a strategy for aligning with the business. The methodology enforces regular and scheduled stakeholder interactions, resulting in better understanding of and alignment with the underlying business needs. Not to say that agile projects will always achieve this alignment, but the agile methodology does make it more likely. Nor is it true that non-agile methodologies cannot be aligned, but many of these methodologies do not enforce the regular involvement of the business.

Summary Strategies for Business Alignment

- Think about your project like a CEO and CFO. Invest the time up front to dissect the business drivers for the project, and if possible, determine the true financial impact of the costs and benefits of the project.
- Use business drivers instead of requirements as your guide for developing the solution architecture. Keep a finger on the pulse of the business throughout the project life cycle to maintain the appropriate flexibility in the project.
- Evaluate how your methodology maintains business alignment during the project life cycle. If needed, inject some regular touch points to keep the business close to the project.
- After your solution is put into production, look for ways to measure its ability to meet the defined objectives.

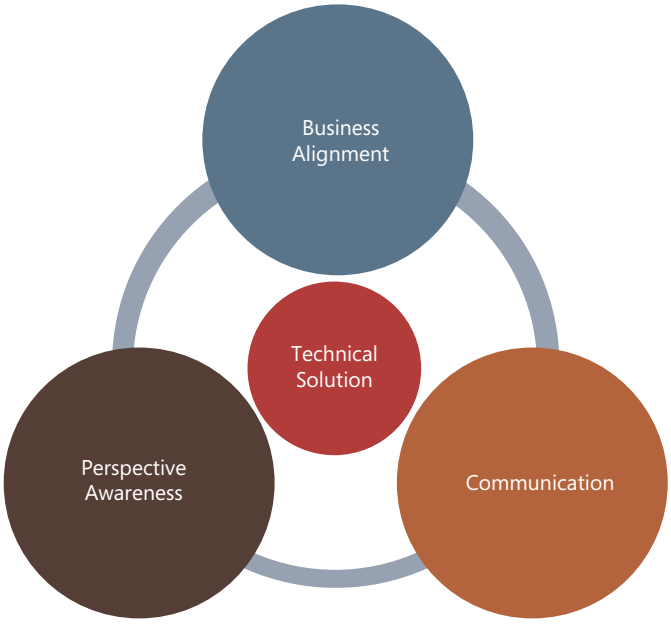
Perspective Awareness

Each member of the team, from the project stakeholders to the developers and testers, has views and motivations that have the potential to create conflict, posing problems for the architect that can even derail the project.

It is fairly common for the development team to dismiss views from outside of the team. This attitude is dangerous not only because the team might miss some important business aspects of the solution, but they also risk alienating others and gaining a reputation for being difficult. It is the role of the architect to bring these groups together to ensure that the team is building the solution that best meets the needs of the business.

Understanding others' often diverse needs is key to ensuring that the solution is the optimal fit. The best way to understand their needs is to view the business issues from their perspectives. An end user of the system has concerns about usability while an operations manager wants to ensure the system performs properly in a production environment. Every stakeholder's perspective should inform the shape of the solution architecture. Having a mindset that every individual has something to contribute facilitates genuine interactions and meaningful communication.

Figure 1: Successful technical solutions require three distinct soft skills.



The architect should be cognizant that not everyone involved may understand or believe in the business drivers of the project. Personal motivations or organizational politics can sometimes work against fruitful interactions. As the architect, you cannot control all factors, but if you take the time to frame these interactions within the scope of the business drivers, then it becomes more difficult for others to inject non-important or divisive factors.

Many architects are in leadership roles with regard to the development team. Managing the attitude of the development team in some environments can be a challenge. Some development teams rarely interact with anyone outside their own teams; in their isolation, these teams become disconnected from the organization. Often it takes someone on the project team to continually remind people that others have a job to accomplish and everyone is on the same team. It is amazing to watch teams transform when leaders take the initiative to help them see other views and motivations instead of joining in when the group starts complaining.

Summary Strategies for Perspective Awareness

- Take the time to understand the perspectives and motivations of the individual.
- Ensure that multiple perspectives are taken into account.
- Frame the conversations with the guiding business drivers to alleviate personal agendas.
- Lead by helping others see alternate perspectives.

"OPEN AND HONEST COMMUNICATIONS ARE OF PARTICULAR IMPORTANCE WHEN COMMUNICATING WITH THE PROJECT SPONSOR. PROJECTS SHOULD BE DEVELOPED WITH A STRONG COMMUNICATION PLAN BETWEEN THE ARCHITECT AND THE PROJECT SPONSOR—WHEN SPONSORS ARE THE FIRST TO KNOW ABOUT POSITIVE OR NEGATIVE SITUATIONS, THEY CAN BE IN CONTROL OF THE PROJECT AND REACT WHEN NECESSARY."

Communication

Most people would probably agree that communication is a key skill of the architect, but few architects employ deliberate strategies for successful communication as part of their roles. Since many of us have technical roots, we tend to revert to our technical strengths when problems occur.

Many problems stem from the inability to communicate with people in the terminology they use on a day-to-day basis. Some of the best architects I've known have the ability to meet with executives and discuss topics in business terms, then walk down the hall to the development team and dive deep into technology discussions. I believe this skill can be learned, but it takes discipline and listening skills.

If you have ever traveled to a foreign country where you don't speak the language, you probably understand how daunting it can be to interact with others. Sometimes you are distrusted just because you don't know the language. But if you learn a few basic words and seek to understand the culture and customs, you may even be welcomed. In general, people who are fluent in the language are much more likely to be trusted and embraced.

You have probably experienced a similar phenomenon when you have taken a job at a new company. You have to learn a new language, culture, and customs. For example, the term "architect" at one company can mean a radically different role from an architect at another company. Over time, you pick up on the language nuances and begin to use them in your everyday communications. You start to incorporate acronyms in your daily language that meant nothing to you before.

This kind of subgroup acculturation also happens within an enterprise. Each group develops its own "native" language. As an architect it is important to listen carefully for these language clues and understand their usage across the enterprise. You need to become multilingual, using the appropriate language and terminology based on the audience.

The skill to develop is the ability to pick up on the language in a group rapidly and to use it effectively, even within the course of a meeting. This means learning to listen effectively during the course of your time with a group and to start cataloging the terms. By communicating with someone in their "native tongue," you get over a trust barrier and increase the possibility for open communication.

After becoming multilingual within an organization, you might want to restructure your meetings to bridge languages and needs. I have attended many meetings among business and technical groups in which one group will dominate the meeting and the other will lose interest and stop communicating. When the situation calls for bringing together disparate groups, you may need to play translator to ensure that each group is communicating effectively.

In all of your communications, one of your primary goals should be to build trust with your audience. The best way in my opinion is to be tactfully open and honest. In general, if people feel that they can trust what you have to say, they are more likely to give you their true thoughts even when they are unpopular.

Open and honest communications are of particular importance when communicating with the project sponsor. I have observed many projects developed with a strong communication plan between the architect and the project sponsor. In these cases, because the sponsors were always the first to know about positive or negative situations, they were in control of the project and could react when necessary.

These types of communications need to be very efficient. Sending a 20-page status report to the sponsor on a weekly basis with the minutiae of the project is not an efficient use of anyone's time. I believe that a written status report on one side of one sheet of paper with the macro-level project statistics and the primary issues and risks delivered in person in 15 minutes a week sets up an effective communication channel and builds rapport. When using this process, I have had project sponsors comment that they felt more engaged with the project than any other in the past.

When communicating with others, it is important to avoid absolutes. Terms like "can't" and "won't" convey inflexibility. The alternative is to present options and implications of decisions in an open and honest manner. This approach also endows the conversation with a peer quality, opening the lines of communication to a discussion rather than a debate. If the business drivers are also a part of the discussion, then there is a framework for making joint decisions.

Summary Strategies for Communication

- Communicate with others in their "native" language.
- Foster an open and honest environment.
- Provide efficient communications to the project sponsor.
- Present alternatives and implications when possible.

Conclusion

Most of us could benefit from further developing our soft skills to improve our effectiveness in our day-to-day role. The framework I've described can help you develop strategies to improve these important skills.

About the Author

Joe Shirey is a senior architect evangelist for Microsoft based in Denver, Colo. Prior to joining Microsoft, Joe was a vice president at Interlink Group where he was responsible for services delivery for their largest market. In the past, Joe was a Microsoft Regional Director; he has served on the Microsoft Architect Advisory Board and the .NET Partner Advisory Council. He has more than eighteen years of hands-on technical and functional experience in project management, systems analysis, design, development, and implementation. Joe attained his Microsoft Certified Architect in Solutions in 2005.

An A-Z Guide to Being an Architect

by Mark Bloodworth and Marc Holmes

Being an architect isn't just about baffling people with unusual diagrams that only make sense when the author is in the same room. No longer can an architect wave a hand judgmentally and dismiss an idea as being "inconsistent with the prescribed architecture." These days, an architect has a lot of diverse responsibilities: to the business, the team, the vision, the technology, and even the wider world.

In this article, Mark Bloodworth and Marc Holmes provide a handy A-Z guide to being an architect. Good luck, and may all your architectures be "n-tier," which, given that 'n' can be any value from 1 and "tier" is just a metaphor for lumps of similar code, seems quite likely...

A is for Advocate

"I think you'll find that you really don't want to do it like that."

Architects have to explain and advise on technical issues to business stakeholders. They also have to be able to advise delivery teams on how to build. This advice is the currency of an architect; invested wisely, it will return goodwill and trust. The architect is asked for advice because it is the architect's job to "see the whole."

See also: Abstraction, Agile, Acrobat, Availability, Analysis, Applications

B is for Balance

"A little more to the left. Keep going. A bit more. Not that far. Sorry."

All decisions involve trade-offs — for example, adding a security measure may hurt performance. It is the architect's lot to make the right trade-off. Architecture may be a zero-sum game, but knowing what the system is intended to achieve enables the architect to choose the trade-offs that make the system successful. Of course, where there are competing objectives, it falls to the architect to explain the issue and seek resolution through prioritization of the objectives.

See also: Best Practice, Benchmarks, Building Blocks

C is for Coach

"Work through the pain!"

With so many choices for the implementation of a solution, architects cannot simply dictate to development teams their notion of the "architecture." They are now called upon to coach development teams. Softer skills are needed: asking how and why rather than instructing "do this" and "do that." This is a Good Thing. Development teams who understand the reasons for the architecture are more likely to commit to it and are likely to do a better job of implementing it. Architects can also begin to spot talent within development teams and offer useful career progression opportunities.

See also: Communication, Champion, Context, Collaboration, C#

D is for Dependencies

"What happens if I unplug this? Oh!"

The relationships among the components that make up an architecture are of fundamental importance. Dependencies are inevitable but should be as few and as manageable as possible. Draw a diagram and map the dependencies. Circles, whether direct (A depends on B and vice versa) or indirect (A depends on B which depends on C which depends on A) are a Bad Thing. If many things depend on D, then D needs to be stable because changing it will have a significant effect.

See also: Design, Development, Delivery, Domain, Documentation

E is for Evangelist

"Let me show you something really cool."

Architects need to be advocates for the choices they have made; others need to believe in the ideas, frameworks, and guiding values of an architecture. Evangelism is about telling stories to different people. A simple segmentation may be a technical versus business audience, but there are really many differing audiences within that. The architecture needs to have a compelling story for each. An evangelist is able to synthesize and simplify complex scenarios for the benefit of common understanding.

See also: Enterprise, Engineer, Enthusiast

F is for Frameworks

"How do I get there?"

Creating the architecture for a solution may be difficult. Creating the architecture for multiple solutions is harder—especially given time pressures and the integration between solutions. An architecture framework is a structure that removes some of the wheel reinvention that would otherwise occur. It provides tools, methods, and a common vocabulary for the process of creating an architecture. An architecture framework can be considered to address the how of architecture.

See also: Facts, Functionality, F#, Firewall

G is for Governance

"It is the opinion of the subcommittee..."

There comes a time, as they say, when you have to put on

the suit if you're serious about doing business. Control is an important part of realizing an architectural vision. Regardless of the model of IT—centralized, decentralized, or federated—there will be competing requirements of equal value. A good architecture needs to be able to flex to differing needs, but not so much that the values of the architecture are lost to the immediate, possibly short-term, must-haves of the business. Equally, good governance can give a positive, dashboard-style view on technology for the business. Common understanding is always a Good Thing.
See also: Generative Programming, Generalist

"ARCHITECTS CAN BE THE MOUTHPIECE FOR THE TECHNICAL TEAMS, AND THE EARS OF THE BUSINESS FOR INNOVATION."

H is for Human Dynamics

"The system would have been a success if it hadn't been for those pesky users!"

Understanding how people interact with each other and the systems that support them is crucial to delivering successful solutions. The dynamics of each project and team will be different; the stakeholders' relationships and motivations may be unique to a given project. Knowing how to navigate human relationships is a key skill of good architects and good leaders.
See also: Heterogeneous Environments, Heated Debates, High Performance Computing

I is for Innovation

"The lifeblood of any organization"

Most products can be viewed as a cycle of invention, innovation, commoditization, and redundancy. Invention is costly, slow, and can require luck and big leaps in thinking. By commoditization time, the game is up, and harnessing the work of others is probably the best option. Typically, therefore, it is the innovation space where advantages—efficiency, competitive differentiation, and so forth—can be achieved through perhaps smaller, but no less valuable evolutions and revolutions of existing ideas and solutions. Small teams can push for innovation constantly and take chances to make their name. Larger groups and organizations may not be able to move as quickly, but they need to enable innovation to percolate from individuals and teams and develop mechanisms for making the best of this inspiration and imagination. Architects can be the mouthpiece for the technical teams, and the ears of the business for innovation.
See also: Integrity, Inspiration, Infrastructure

J is for Judgment

"With great power comes great responsibility."

When the discussion is done and a technical decision must be made, then an architect is going to have to exercise judgment. The team entrusts the architect to make these judgment calls, and the architect's good judgment gives confidence to the team. For better or for worse, a series of good or bad calls may be seen to characterize the architect: conservative or wise, impulsive or prescient, biased or brave. Exercising good judgment is vital, but in practice, even good judgment will sometimes turn out to be wrong. Don't worry about making a mistake; worry more about not doing anything.
See also: Java, Just In Time

K is for Knowledge

"If only I knew then what I know now."

Knowledge is a key architectural tool. Of course, being aware of the boundaries of your knowledge is a Good Thing. Areas that are known unknowns are ripe for proof-of-concepts and other knowledge-building exercises. Unknown unknowns, on the other hand, are Bad Things: They are the architectural equivalent of gremlins. Knowledge of technology is only one, albeit important, domain that an architect needs to command. An architect also needs to know about the nontechnical factors that will be in play, such as organizational structures, enterprise strategy, business processes, and development methodologies.
See also: Kernel, Keyboard

L is for Leadership

"I'm behind you all the way."

Leadership is vital for an architect and typically takes two forms: thought leadership and team leadership. As guardian of the architecture and the values behind the architecture, the architect is thought leader: The architect continually reevaluates the vision and re-presents the "newer, shinier" vision, with comment on competing visions and emerging technology. As team leader, an architect may not be required to perform line management duties, but may be called upon to be an icon for the rest of the team, providing confidence, insight, motivation, and inspiration.
See also: Lean, Linux, Latency, Load Balancing

M is for Modeling

"So, to help us visualize how this might work, I made this model using nothing but twigs and guitar strings."

A model is a representation of something—for example, a business process or computer system. Views of a model provide a way to communicate and understand ideas about the problem and the solution. Different views address different concerns—overloading one view in an attempt to address multiple concerns will either lead to an overcomplicated view or an oversimplified understanding. Having a shared notation for representing these views of a model can simplify conversations about the model—although if the notation becomes too complex, this benefit is soon lost.
See also: Management, Maintainability, Messaging

N is for 'N-tier'

"A house of cards"

Data Layer, Business Logic Layer, User Interface Layer. Job done. Well, not quite. N-tier is a vague term at best, and doesn't really say anything more than pointing to the idea that there should be some kind of separation of concerns between various chunks of code. With Service-Oriented Architecture (SOA), and most recently, the explosion in cloud services—either as back end (cloud storage for example) or front end (such as Facebook)— actually describing the

architecture can be harder than building it. We're fans of the "Petri dish" approach: concentric circles usually containing square boxes as if suspended in agar jelly.
See also: Needs, Networks, Nonfunctional Requirements, .NET

O is for Object Orientation

"Encapsulate this!"

Object Orientation (OO) is a programming paradigm that rose to prominence in the 1990s. It can be thought of as a way to conquer complexity by dividing a big problem or program into bite-size, digestible and, of course, logically coherent chunks. Object orientation is often referred to as OO and sometimes as OOP (Object Oriented Programming), which is only a letter away from sounding like a mistake. To ensure acronym coverage, we must also mention OOAD (Object Oriented Analysis and Design.) Object orientation is as much an analysis technique as it is a programming paradigm, although translation is often required from an analysis, or conceptual, model to a design, or logical, model—just as there is between the design model and the implementation, or physical, model. An object —the entity at the core of OO — has both behavior and data, and the functionality of a system is achieved through the interaction of objects. Objects, which are instances of Classes, expose their abilities via Methods. There are, predictably, some key concepts and terms to be learned in order to grasp OO—the most important being: Inheritance, Polymorphism, Encapsulation, and Abstraction.
See also: Operations, Object-Relational Mapping, Operating System, OLAP

P is for Patterns

"I think I see something emerging from the chaos. Is it a zebra?"

Patterns are everywhere it seems. Where there was the Gang of Four and their original *Design Patterns*, now there are many resources and books dedicated to patterns across many disciplines. Some are stronger than others and probably some judicious pattern-weeding is necessary for a well-maintained architectural garden. Patterns provide both a template for the implementation of a particular concept but also a common language to discuss abstract and complex concepts without the need to resort to a full description, or a diagram—although we'd probably do that, too.
See also: Principles, Platforms, Politics, Performance, Process

Q is for Quality

"Good enough isn't good enough."

Quality is often understood as a synonym for good. Good is hard to define and measure. Quality should be defined and measurable. What quality is really about is ensuring that the solution meets the requirements and all the applicable standards (as defined by the enterprise, industry, statutory authority, and so forth). By defining and specifying the metrics and standards, a solution can be judged — and, if necessary, improved.
See also: Qualifications, Queries, Quantification, Quantum Computing

R is for Roadmaps

"You take the high road, and I'll take the low road."

Where architecture and real life sometimes come unstuck is in the difference in times between the production of concepts and the subsequent realization of the vision. Many obstacles stand in the way of a beautiful architecture: differing views, changing product strategy, short-term tactical needs. A roadmap can help to maintain the original vision, providing a view on the now, the soon and the later of the implementation. A roadmap can provide the business with a view on the plans and targets of the technology teams. A roadmap can sometimes help you remember just what it was that you were trying to do in the first place.
See also: Requirements, Realization

"AS GUARDIAN OF THE ARCHITECTURE AND THE VALUES BEHIND THE ARCHITECTURE, THE ARCHITECT IS THOUGHT LEADER. AS TEAM LEADER, AN ARCHITECT MAY NOT BE REQUIRED TO PERFORM LINE MANAGEMENT DUTIES, BUT MAY BE CALLED UPON TO BE AN ICON FOR THE REST OF THE TEAM, PROVIDING CONFIDENCE, INSIGHT, MOTIVATION, AND INSPIRATION."

S is for Strategy

"What are we trying to achieve?"

Strategy sets out how to achieve your goals. Architectural strategy is derived from the enterprise strategy—it should enable the enterprise to achieve its goals. The word "strategic" should be used with care and caution—many before you have used it to justify costly, long-term investments with ill-defined benefits. A strategy, like a good military plan, should be adaptable—otherwise it will collapse upon contact with reality. Strategy is often confused with—and sometimes mistakenly thought to be in opposition to—tactics. Tactics are the specific actions that, by achieving objectives, are the implementation of your strategy.
See also: Services, Software, Standards, Security, Scalability

T is for Thinking

"I think, therefore I clearly have too much time on my hands."

As a skill, thinking is typically not a problem for developers and architects. Finding the space and time to think is a little harder. In these days of a constant bombardment of information from the blogosphere—good, bad, and ugly—it can sometimes be hard to find the inclination to think for oneself. Such a crucial activity needs to be given focus and an architect should be prepared to make the space and time and defend it. Think about thinking: What works for you? Long train journeys? Music? A hot bubble bath? It might be hard to install a bathroom suite in the office, but you never know.
See also: Technology, Transparency

U is for Understanding

"I do believe you've got it."

Understanding is complementary to knowledge. Understanding people, systems, and processes makes a significant difference to the outcome of a solution. It is the antithesis of assumption. Some nefarious types will present assumption as understanding—this is undoubtedly a Bad Thing and will not lead to the Promised Land of Good Architecture. Questioning is a key technique for reaching understanding—used well it can puncture assumption, myth, and other forces that could derail a project.

See also: UML, Unix

V is for Values

"Explain to me again why we're doing this..."

The values of an architecture are best expressed as principles—the value system that guides decision-making and architectural practice is made up of these values. Principles are, therefore, the foundation that underlies architecture. To be effective there should be no more than a handful of enterprise-level principles and they must have the support of senior leaders. A good principle is clear, consistent, relevant, appropriately focused, adaptable, and stable.

See also: Virtualization, Visualization, Views

W is for Whiteboard

"It's probably easier if I just draw a picture."

Good whiteboarding skills are a true art—it is easy to become an apprentice, but achieving mastery is always elusive. On the evidence of our own careers, we suspect that many great ideas have never been implemented simply because of a "bad gig" on the whiteboard. In the future, if the original pioneers of computer technology are to be remembered (that's you, by the way) then the most fitting monument would be a huge statue of a whiteboard in pristine white marble, with just a few tell-tale signs of the accidental use of a permanent marker.

See also: Workflow, Wikis, Windows, Web

X is for XML

"<xs:element name='quote' type='xs:string' />"

XML has become a universal markup language— thus providing a nonproprietary format for data storage and a means to integrate systems and applications. While it has its detractors and there are rival markup languages (such as JSON and YAML), there is, as yet, nothing that can rival the reach of XML. While some may think of XML as the Esperanto of the Web, it is really nothing more than the basis for a shared language. Think of XML as providing the letters and the punctuation, but not the words or grammar. XML Schema (XSD) provides a means of defining XML documents that can be shared and used to validate documents. And while there are alternatives, such as RelaxNG, XSD, like XML, has sufficiently broad reach that it is likely to be understood by partners and customers.

See also: XSD, XPath, XQuery, XAML, XOML

Y is for YAGNI

"Stay on target! Stay on target!"

Great designs are often not grand designs. Using good judgment to decide when to build new features, or reuse prior work, or skip the features is all part of the architectural game. Still, it can be appealing to just keep building new stuff just in case it's needed in the future, because "you never know." Of course, you do know—not much software lasts for all that long these days owing to new techniques, channels, and even languages that can be exploited. If you're not sure, then more than likely, You Ain't Gonna Need It.

See also: YAML, Yottabyte

Z is for Zeitgeist

"All the cool people are doing it."

Zeitgeist or "spirit of the age" is an important aspect of thinking and values and leadership. It's magnified with the rate at which "ages" manifest themselves. We're already on Web 2.0 after all. Understanding how to react to the zeitgeist ensures that the right steps are taken to respond to changing circumstance: "Let's reinvent ourselves as Facebook tomorrow." Typically, for an architect, it is not so much the manifestation of new thought—those are just implementations—as the underlying memes and their importance in the technology landscape. Everyone else sees '"social networking" where an architect sees "the semantic Web."

See also: Zeal, Zettabyte, Zero Day Exploit

In Closing

When we set out to compile this A-Z, we wondered how much of a challenge it would be to construct. In fact, we were inundated with possibilities, and spent a lot of time debating the merits of any given entry.

For us, the list has been an affirmation that architecture is as much about softer skills: good judgment, balance, and other wisdom, as it is about understanding the broad technical landscape, or the skills required to design and implement an architecture.

We've had a lot of fun writing our version of this A-Z, but would love to hear of your own alternatives. We wouldn't be architects if we all agreed on the same list!

About the Authors

Mark Bloodworth is an Architect Evangelist in the Developer and Platform Evangelism Team at Microsoft. Prior to joining Microsoft, Mark was the Chief Solutions Architect at BBC Worldwide, where he led a team responsible for architecture and systems analysis. This role encompassed technical strategy, architecture, and team management. Mark's background is in software development and design, specializing in Web applications and integration. Most of his career has been focused on using Microsoft technologies, especially .NET, with a little Java thrown in for good measure. Mark keeps a blog at remark.wordpress.com.

Marc Holmes is an Architect Evangelist for the Developer and Platform Evangelism Team at Microsoft in the U.K., where he specializes in architecture, design, and proof-of-concept work with a variety of customers, partners, and ISVs. Prior to Microsoft, Marc most recently led a significant development team as Head of Applications and Web at BBC Worldwide. Marc is the author of *Expert .NET Delivery Using NAnt and CruiseControl.NET* (Berkeley, CA: APress, 2005) and maintains a blog at http://www.marcmywords.org.

ARC

Microsoft®

THE ARCHITECTURE JOURNAL™
Input for Better Outcomes