Characteristics of a software architect

Level: Introductory
Peter Eeles, Senior IT Architect, IBM
15 Mar 2006

from The Rational Edge: If, in movie-making terms, the software project manager is the producer, since they make sure that things get done, then the software architect is the director, who makes sure that things are done correctly and, ultimately, satisfy stakeholder needs. As the second of a four-part series, this article describes the role of software architect.

This is the second article in a four-part series on software architecture. Last month, the first article in this series defined what we mean by architecture. We can now turn our attention to the role that is responsible for the creation of the architecture -- the architect. The role of the architect is arguably the most challenging within any software development project. The architect is the technical lead on the project and, from a technical perspective, ultimately carries the responsibility for the success or failure of the project.

Here's how the IEEE defines the term "architect":
[An architect is] the person, team, or organization responsible for systems architecture.[1]

As the technical lead on the project, the characteristics and skills of the architect are typically broad, rather than deep (although architects should have deep skills in particular areas).

The architect is a technical leader

First and foremost, the architect is a technical leader, which means that, as well as having technical skills, the architect exhibits leadership qualities. Leadership can be characterized in terms of both position in the organization and also in terms of the qualities that the architect exhibits.

In terms of position in the organization, the architect is the technical lead on the project and should have the authority to make technical decisions. The project manager, on the other hand, is more concerned with managing the project plan in terms of resources, schedule, and cost. Using the film industry as an analogy, the project manager is the producer (making sure things get done), whereas the architect is the director (making sure things get done correctly). As a result of their positions, the architect and project manager represent the public persona of the project and, as a team, are the main contact points as far as people outside the project are concerned. The architect, in particular, should be an advocate of the investment made in creating an architecture and the value it brings to the organization.

The architect is also involved in organizing the team around the architecture and should actively contribute to planning activities as a result, since dependencies in the architecture translate to the sequencing of tasks and therefore the skills required at particular points in time. On a related note, since the success of the architect is closely linked to the quality of the team, participation in interviewing new team members is also highly appropriate.

In terms of the qualities that the architect exhibits, leadership can also be characterized in terms of interactions with other team members. Specifically, the architect should lead by example and show confidence in setting direction. Successful architects are people-oriented, and every architect takes time to act as a mentor and coach to the members of their team. This benefits the team members requiring help, as well as the project and, ultimately, the organization itself, since one of its most valuable assets (the organization's people) becomes better skilled.

Also, the architect must be focused on the delivery of tangible results and must act as the driving force for the project from a technical perspective. An architect must be able to make decisions (often under pressure), and make sure that those decisions are communicated, understood, and, ultimately, implemented.

The architect role may be fulfilled by a team

There is a difference between a role and a person. One person may fulfill many roles (for example, Mary is a developer and a tester), and a role may be fulfilled by many people (for example, Mary and John fulfill the role of tester). Given that the role of architect requires a very broad set of skills, it is often the case that the architect role is fulfilled by more than one person. This allows the skills to be spread across a number of individuals, each bringing his or her own experiences to the role. In particular, the skills required to understand both the business domain and also various aspects of technology are often best spread across a number of individuals. The resulting team does, however, need to be "balanced." Throughout this article, the term "architect" refers to the role, which may be fulfilled by either an individual or a team.

[A team is] a small number of people with complementary skills who are committed to a common purpose, performance goals, and approach for which they hold themselves mutually accountable.[2]

If the architect role is to be fulfilled by a team, then it is important to have one individual who is considered the lead architect, who is responsible for owning the vision and can act as a single point of coordination across the architecture team. Without this point of coordination, there is a danger that members of the architecture team will not produce a cohesive architecture or that decisions won't get made.

For teams that are new to the concept of architecture, it has been suggested that, in order to achieve this common purpose, goals, and approach, the team create and publish a charter for the architecture team.[3]

Good architects know their strengths and weaknesses. Irrespective of whether or not the architect role is fulfilled by a team, it is often the case that an architect is supported by a number of "trusted advisors." Such architects acknowledge where they are weak and compensate for these weaknesses by either obtaining the necessary skills or working with other people to fill the gaps in their knowledge. The best architectures are usually created by a team, rather than an individual, simply because there is a greater breadth and depth of knowledge when more than one person is involved.

One pitfall with the concept of an architecture team is that it is sometimes perceived by others in the organization as an "ivory tower," whose output is intellectual rather than useful. This misconception can be minimized from the outset by 1) ensuring that all stakeholders are actively consulted, 2) continually communicating the architecture and its value, and 3) being conscious of the organizational politics at play.

The architect understands the software development process

The architect should have an appreciation of the software development process, since this process ensures that all of the members of the team work in a coordinated manner. A good process defines the roles involved, the activities undertaken, the work products created, and the handoff points between the different roles. Since the architect is involved on a daily basis with many of the team members, it is important for the architect to understand their roles and responsibilities. Day-to-day, the development team will often look to the architect to tell them what to do and often how to do it. There is therefore a clear overlap between the role of the architect and the role of the project manager.

The architect has knowledge of the business domain

As well as having a grasp of software development, it is also highly desirable (some would say necessary) for the architect to have an understanding of the business domain.
[A domain is] an area of knowledge or activity characterized by a set of concepts and terminology understood by practitioners in that area. [4]

Such knowledge will allow the architect to better understand and contribute to the requirements of the system and be in a position to ensure that "likely" requirements -- i.e., requirements that, based on the architect's domain knowledge, will probably have to be considered -- are captured. Also, it is often the case that a particular domain is associated with a particular set of architectural patterns that can be applied. Knowing this mapping can also greatly assist the architect.

Therefore, a good architect will have a balance of software development knowledge and business domain knowledge. When architects understand software development but not the business domain, a solution may be developed that does not fit the problem, but merely reflects what the architect is comfortable or familiar with. Another reason the architect needs familiarity with the business domain is that the architect needs to anticipate likely changes to the architecture. Given that the architecture is heavily influenced by the environment in which it will be deployed, having an appreciation of the business domain will allow the architect to make better-informed decisions about the likely areas of change, and the areas of stability, from an architectural perspective. For example, if the architect is aware that new regulatory standards may need to be adhered to at some point in the future, then this should be accommodated in the architecture should such standards be mandated during the life of the system.

The architect has technology knowledge

Certain aspects of architecting clearly require a knowledge of technology; an architect therefore must maintain a certain level of technology skills. However, architects do not need to be technology experts. This relates to the idea in Part 1 of this article series -- that an architecture focuses on significant elements. Correspondingly, the architect need only be concerned with the significant elements of a technology and not the detail. Since technology changes fairly frequently, it is essential that the architect keep abreast of these changes.

The architect has design skills

Although architecting is not confined to design, design is clearly an important aspect of architecting. The architect should therefore have good design skills since the architecture embodies key design decisions. Such decisions could represent key structural design decisions, the selection of particular patterns, the specification of guidelines, and so on. In order to ensure the architectural integrity of the system, these elements are typically applied "across the board" and can have far reaching effects in terms of the success of the system. Such elements therefore need to be identified by someone with appropriate design skills.

The architect has programming skills

The developers on the project represent one of the most important groups that the architect must interact with. After all, it is their work products that ultimately deliver the working executable software. The communication between the architect and the developers can only be effective if the architect is appreciative of the work of developers. Therefore, architects should have a certain level of programming skills, even if they do not necessarily write code.

Most successful architects have, at some stage, been hard-core programmers, where typically they have learned certain aspects of their trade. Even as technologies evolve and new programming languages are introduced, good architects can abstract the concepts in any programming language and then apply this knowledge to learning a new programming language to the depth required. Without this knowledge, the architect will be unable to make decisions with respect to the architecturally significant elements of the implementation, such as the organization of the implementation and the adoption of programming standards, and a communication barrier will emerge between the architect and the developers.

The architect is a good communicator

Of all of the "soft skills" associated with the architect, communication is the most important. There are a number of dimensions to effective communication, and the architect needs to be proficient in all of them. Specifically, the architect should have effective language skills, including speaking, writing, and presentation abilities. Also, the communication is two-way. The architect should be a good listener and observer, as well as a good talker.

Being able to communicate effectively is a skill that is fundamental to the success of a project for many reasons. Clearly, communication with stakeholders is particularly important in order to understand their needs and also to communicate the architecture in a way that secures (and maintains) agreement with all stakeholders. Communication with the project team is particularly important, since the architect is not simply responsible for conveying information to the team, but also for motivating them. Specifically, the architect is responsible for communicating (and reinforcing the communication of) the vision for the system, so that the vision becomes shared and not something that is only understood and believed in by the architect.

The architect makes decisions

An architect who is unable to make decisions in an environment where much is unknown, where there is insufficient time to explore all alternatives, and where there is pressure to deliver is unlikely to succeed. Such an environment is to be expected, and successful architects acknowledge the situation, rather than try to change it. Thus, the architect needs to be "thick-skinned" since they may need to correct their decisions and backtrack at times during a project. As Philippe Kruchten puts it, "The life of a software architect is a long and rapid succession of suboptimal design decisions taken partly in the dark."[5]

An inability to make decisions will slowly undermine the project. The project team will lose confidence in the architect, and the project manager will be concerned because those waiting on the architecture cannot make the required progress. Here's the greatest danger: If the architect does not make and document decisions about the architecture, then team members will start to make their own, possibly incorrect, decisions.

The architect is aware of organizational politics

Successful architects are not geeks only concerned with technology. They are also politically astute and are conscious of where the power in an organization resides. This knowledge is used to ensure that the right people are communicated with and that support for the project is aired in the right circles. To ignore organizational politics is, quite simply, naïve. The reality is that there are many forces at work in organizations that lie beyond the project team delivering the system, and these need to be accounted for.

The architect is a negotiator

Given the many dimensions of architecting, the architect interacts with many stakeholders. Some of these interactions require negotiation skills. For example, a particular focus for the architect is to minimize risk as early as possible in the project, since this has a direct correspondence to the time it takes to stabilize the architecture. Since risks are associated with requirements, one way to remove them is to remove or diminish the requirement with which the risk is associated. Hence the need to "push back" on such requirements so that a mutually-agreeable position -- between stakeholders and architect -- can be reached. This requires that the architect be an effective negotiator, able to articulate the consequences of different tradeoffs.

Summary

This article has focused on defining the characteristics of a software architect. The remaining articles in this series will focus on the characteristics of the process of architecting and the benefits of treating architecture as a fundamental IT asset.

Acknowledgements

The contents of this article have been derived from a forthcoming book, provisionally entitled "The Process of Software Architecting." As a result, the content has been commented upon by many individuals that I would like to thank, who are Grady Booch, Dave Braines, Alan Brown, Mark Dickson, Luan Doan-Minh, Holger Heuss, Kelli Houston, Philippe Kruchten, Nick Rozanski, Dave Williams, and Eoin Woods.

Notes

[1] IEEE Computer Society, IEEE Recommended Practice for Architectural Description of Software-Intensive Systems: IEEE Std 1471-2000.
[2] Jon R. Katzenbach and Douglas K. Smith, *The Wisdom of Teams*. Harvard Business School Press 1993.
[3] Philippe Kruchten, "The Architects -- The Software Architecture Team," Proceedings of the First Working IFIP Conference on Software Architecture (WICSA1). Patrick Donohoe (editor). Kluwer Academic Publishing 1999.
[4] Grady Booch, James Rumbaugh, and Ivar Jacobson, *The Unified Modeling Language User Guide*. Addison Wesley 1999.
[5] Philippe Kruchten, Op. cit.

About the author

Peter Eeles works for IBM Rational, IBM Software Group, and has spent much of his career architecting and implementing large-scale, distributed systems. Based in the UK, he assists organizations in their adoption of the Rational Unified Process and the IBM Software Development Platform. He is coauthor of *Building J2EE Applications with the Rational Unified Process* (Addison-Wesley, 2002), coauthor of *Building Business Objects* (John Wiley and Sons, 1998), and a contributing author to *Software Architectures* (Springer-Verlag, 1999).