Architect Soft Skills

By Arnon Rotem-Gal-Oz (http://www.rgoarchitects.com/nblog)

Introduction

There's a lot of discussion about the hard skills software architects needs to have; for example, see one example at the Software Engineering Institute (SEI). Architects need to be familiar with a wide range of technologies, methodologies, understand the software lifecycle, have design experience, and some say an architect must write code, and so on and so forth. Indeed, the hard skills are important, very important. However it doesn't stop there. There are also several soft skills that you need to master if you are to be a good architect.

I believe that the minimal skill-set for an architect should include capabilities from the following areas:

- Leadership. Influencing others to accomplish tasks and following your guidance
- System thinking. Understand decisions and constrains in the wide scope pertaining to whole of the solution at hand. This includes the ability to abstract problems.
- Strategic thinking. Understanding decisions and constrains and their alignments to the overall business of the company.
- Organizational politics. Understand the environment you operate in and how it influences you.
- Communications. Making sure you get your point across.
- Human relations. Understand the "people" aspects or human factors and dynamics. This includes things like pragmatism, understanding team dynamics and personal dynamics

Let’s take a look at them one by one

Leadership

Solutions architects are the "technical managers" of projects. This means they are responsible that all the designs and code are aligned to the functional requirements and that the quality attributes are kept.

However, architects are seldom the direct managers of a development team - and even if the architect is the manager, you still need to inspire your workers. Tyranny? Well, that just doesn't work. You might think that establishing yourself as a technical authority will be enough (that's

why they made you the architect in the first place, right?) but it isn't. You need to cultivate your leadership skills as well.

What is leadership anyway? Leadership is about exerting influence on people and increasing the chance that people will follow your vision and decisions. To do that you need to gain the respect of the teams you work with, communicate your vision and designs clearly and are trustworthy.

So how do you do that? Unfortunately, I don't have a definitive answer to that but here are a few things to think about which I think are useful:

- Provide direction. To lead you need to know where you are going and make the decisions that will get you there.
- Explain your decisions. Deus ex machina doesn't count. You work with intelligent people, they may not have your experience or the same depth of knowledge but they want to know why they are doing something.
- Listen to what others have to say. They may actually say something valuable you know :)
- Don't postpone decisions and don't avoid conflict. This will not make them go away. Do try to manage your conflict though.
- Motivate people. This can be done by things like mentoring and teaching, allowing people design freedom. (Letting others make the decisions in their relative fields/areas even if the solution they propose is not perfect.)
- Set an example. E.g., you can sit (pair with) other developers in the team to design/code important things together

Leadership is one of the most important soft skills. As I mentioned earlier, architects are usually not the managers but they do need to lead if they want to ensure the stakeholders' needs (the system quality attributes) will indeed make it into the solution. To better grasp the quality attributes you'd need "System Thinking"

System Thinking

If you managed to make yourself a leader, it only increases your responsibility to actually know where to go. Two soft skills that can help you with that are System Thinking and Strategic thinking (which is described in the next section)

Microsoft Encarta defines system as "any collection of component elements that work together to perform a task." (You may want to take a look at some of the characteristics of systems by Donald E. Gray). When we get a problem, that is, a software solution that needs to be built, we tend to think about breaking it down into manageable "parts" (the subsystems/components/services/objects) that makes that solution. This, however, will only take us so far if we don't also employ "Systems Thinking".

System Thinking originated as a way to think about social systems but has emerged as a way for problem solving problems for systems in other practices. In essence it is about understanding that system parts that work together behave differently then each part alone ("The whole is greater than the sum of its parts"). Which is, by the way, the reason "loose coupling" is such a holy grail--as it helps reduce the parts interdependence and interactions, and thus simplify the system.

One important trait for understanding systems behavior and component interaction is the ability to create abstractions and models; that is, simplifications of the reality which contain enough detail to be useful (Another thing that is needed is the ability to communicate that to the different stakeholders which is another soft skill I'll expand upon). It is important to remember that mental models limit the perspective we have which is why we need to have several models and why it is beneficial to have more than one person working on a problem

As it happens this is also aligned with my definition of software architecture:

Software architecture is the collection of the fundamental decisions about a software product/solution designed to meet the project's quality attribute requirements. The architecture includes the main components, their main attributes, and their collaboration (i.e. interactions and behavior) to meet the quality attributes. Architecture can and usually should be expressed in several levels of abstraction (depending on the project's size).

This definition gives interactions and the environmental implications on the system as a whole the same weight as designing the parts themselves. If the architect doesn't (or can't) understand the effects of the components playing together the quality attributes (performance, availability etc.) of the system will suffer and the system will not operate as planned.

For an introduction to the subject, I recommend Gerald M. Weinberg's *Introduction to General Systems Thinking*. It lives up to its name.

The second part of understanding “where to go” is strategic thinking.

Strategic Thinking

While system thinking takes care of the system in its environment, strategic thinking is about understanding where the organization is heading and its long term goals so that the solution being developed is in sync with them. While I guess it is easy to see why this is important for internal projects, I think it is also important for product/solution companies.

Strategic thinking involves the techniques and thinking processes essential to setting and

achieving the business's long term priorities and goals. Strategic thinking (understanding the problems) is the preamble to strategic planning -- but we'll leave planning to management and focus on the understanding which is important for the architects.

Ruth Malan and Dana Bredemeyer define the Strategic Thinking soft skill (which they call "Strategic Perspective"):

> "[An architect who has strategic perspective (ARGO)] understands the industry, market, customers, competitors, suppliers, partners and capabilities of the business. Identifies opportunities and threats, and actively identifies trends and future scenarios. "

This is a good definition because it really explains why this skill is so important for architects. One of the architect's core roles is to understand what the different stakeholders' need, then balance these needs to create a usable, robust solution. Their needs expressed as quality attributes are the driving force of the architecture.

Furthermore, if you think about all this "align business and IT" stuff we constantly hear about these days (especially in regard to SOA), it is evident that all the careful planning of how technology and software can help in getting that alignment is useless unless we really have a good understanding of where the business is going and what this alignment really is. Thus, the architect should really "get it". The architect should first understand what the business is about and where it is going. Then, armed with this understandings and insights, she can translate them to technological and architectural decisions that ensure these needs are met.

In an ideal world, gaining these insights would be enough. However, the architects do not operate in a vacuum. Architects should also understand the organizational forces that can make the solution work (or break)

Organizational Politics

If strategic thinking helps you understand where the organization is going, Understanding Organizational politics helps you understand how e organization is working.

Consider the following anecdote; A while ago I co-architected a Naval Command and control system. One of the key elements of that system was a service bus component which we wanted to base on a commercial messaging middleware. We thoroughly explained why choosing messaging was the best choice for the to the project's management. Nevertheless, another solution based on a proprietary (and fundamentally flawed) distributed objects middleware was constantly suggested and eventually made a constraint we must follow. It took us several iterations (and a lot of rework later) to prove that a messaging middle-ware was indeed the (much) better solution for that project). What happened here? Two experienced architects gave

ton of good reasons justifying a technical decision, but somehow that decision was overruled, why?

Decision making, especially technical decision making, seems like such a logical process. You just look at the alternatives; analyze the merits vs. the problem at hand, and may the best option win. This works out well if you are the king (or work alone which makes you the king by default) -- otherwise there are other people and they won't necessarily agree with you. One reason for that may be they really have another solid opinion, in which case you need to negotiate with them , but that has to do with the leadership skill (we already mentioned) or communications skill (discussed later). The other reason for people to disagree is that they may have other interests and agendas, which run much deeper than the positions they externalize (i.e. their disagreement with you).

Organizations (and the larger they are, the more complex they get) tend to get to decisions by employing a system of rules which encompasses a lot of interests on top of the rational reasons to agree or disagree. Understanding Organizational Politics is about understanding these non-rational influences on the decision making processes.

To return to the anecdote above, it didn't take us too long to understand the real motivation there. It turns out both the project leader boss and a few others recommended buying that flawed component which also happened to cost a small fortune. As that component was already bought, it had to justify itself by being used everywhere. In this particular case the only way we found to reverse the decision was to prove that it was flawed and to minimize its infiltration into the project (so it will be relatively easy to remove it later). In other cases there might be more cost effective ways to do achieve the desired result.

The first thing to do is understand where you are. This is one of the reasons the first step in the SPAMMED framework is to understand the stakeholders. If you manage to uncover the agendas and interests of the different stakeholders as well as the influence they may have on your project, it will at least help you pin-point your problems.

The tricky part in knowing to how to navigate and influence the organization. Tools you can use are interpersonal skills, networking capabilities, schmoozing, and you also need excellent communication skills.

The point I am trying to make here is that even though technical people tend to regard organizational politics as dirty, you cannot afford to dismiss them. Organizational politics can have a severe influence on your project and actions. I didn't talk a lot about how to become more judicious political animal, I am probably not qualified enough to do that (you may want to check some of the resources below though)

More resources:

- David Ing mentions this in his "overly long guide to being a Software Architect"
- Ruth Malan and Dana Bredemeyer talked about it as well (see for example their paper on the architect's role or their competency card on organizational politics).
- David Dikel, David Kane and James Wilson cover some related organizational patterns in their "Software Architecture: Organizational Principles and Patterns"
- Mark Maier and Eberhardt Rechtin have a chapter on politics in their "Art of System Architecting, Second Edition"

As, I mentioned, one of the essentials to actually getting your points across is good communications skills.

Communications Skills

"What we've got here, is failure to communicate, some man you just can't reach…" - this can work for a warden in a movie or an opening to a song[1], but it is definitely not an excuse for an architect. The architect is a hub of communication between management, users, developers and what not - A failure to communicate in the case of an architect can mean a conceptual bug down the line if talking to a developer, increased costs and animosity if talking to a project manager or even cancellation of the project if talking to upper management.

As a senior technical person you already know how to solve problem and translate your ideas into working code – However as an Architect working with teams you have to solve a few more soft problems. One barrier to cross is conveying your ideas to others – or presentations skills.

Presentations skills start with the way you create your slides (e.g. bullet points vs. telling a story) and go well beyond that into how you present, your stance, your interaction with the audience etc. Note that presentation doesn't have to be a keynote address in OOPSLA it can be a by-the-white-board standup with a college. There are different focuses for each type of presentation but the principles are the same.

So assuming we covered presentation skills is that enough? – No, since explaining yourself will only set you off on a journey, you also need to engage in a dialog with the "others" so that you'd reach an agreement (That you are right, negotiate some middle ground or understand your errors). Thus the next component of communication skills is negotiation skills. Negotiation skills will let you defuse situation that would otherwise deteriorate into a bunch or raving lunatics shouting at each other and instead have a collaborative common problem solving experience. That may sound like BS but if you want to move things in positive directions you need to be prepared. If anything you should at least know when to stop (a.k.a. BATNA – Best Alternative to a Negotiated Agreement) but there are many more things you can do before that (see more resources below)

[1] Cool Hand Luke (1967) and opening line for Gun's and Roses "Civil War"

More resource

- Beyond bullet point – A refreshing approach to building presentations
- Presentation Zen – more presentation tips
- Business Storytelling resources
- Negotiation/Persuasion - Anything based on "Getting to yes","Getting past no" and "The power of a positive no" should be fine.

Human Relations

Negotiations skills, Leadership and dealing with organizational politics are all, in a sense, aspects of Human Relations. Nevertheless Human Relations have a few additional aspects which I think we, as architects, should be aware of. Basically there are three main components I want to discuss–team dynamics and personal dynamics and pragmatism.

The first aspect of human relations is team dynamics. There are several models explaining the various stages teams go through as they grow. I think the best knows is Bruce Tuckman's model Forming-Storming-Norming-Performing. The team members act and interact differently during these stages as a (technical) leader of a team or teams; you should be aware of these dynamics and behave accordingly. For instance mentoring and guidance are usually more welcomed during the Forming stage, while these efforts might be rejected during the storming one.

Looking at the team as a whole is one thing. However the developers and the rest of the stakeholders are all individuals. Each with his/her own personality, motivations and what not. Again there are many theories related to individuals from Maslow's pyramid of needs (which talks about what motivate people) to Jung's personality types which affects how people interact with each other's e.g. software developers are likely to be Introverts, Sensing, Thinking, Judging/Perceiving (ISTJ/ISTP) so, for instance, they would not like to be told to do things that don't make sense to them. In any event, I guess my main advice here is to be aware of some of these theories and pay attention to how they manifest themselves in situations you encounter. In a company I once worked for, I got a small tip from my direct manager. It turned out that when I was talking to people I didn't hold in high regard – it well, er.. showed. This made them feel uncomfortable working with me but unfortunately none of us was going away. Being aware of the situation made me improve myself. For instance, I wouldn't just cut away their speech when they tried to make a point or tried not to talk down at them, and hey, even listen sometimes ☺- This brings me to the last point I want to make on the subject – Pragmatism.

Pragmatism - the art of the possible. As a leading technical figure (I assume that how you got to be an architect) you should be wary of architectural tyranny. Letting others have their way even

if it that way is not the great (I am sure) way you've managed to come up with. There's a whole lot of clichés that can be thrown here like "there's more than one way to skin a cat", "better be smart than right" etc. so what? Sometimes even clichés are right☺. As project span over a long time and you have to maintain good relations with most if not all the involved parties you should be watchful for absolute truths. Sometimes a little bit of pragmatism can go a long way towards making the atmosphere of the project calmer and nicer. If you take into account the person you are taking to or

Further reading

- Team dynamics what to expect and do - Abhishek Agrawal talks about some of the models for team dynamics
- Collaboration Explained - A book by Jean Tabaka on team collaboration

Summary

The architect's role goes well beyond the technical skills. I hope that this short paper helped highlight some of the softer skills that are required (in my opinion) for an architect to be successful.

The goal of this paper was not to teach all the soft skills but rather to highlight tem and increase the awareness for them. I'd be happy to hear if you find any of the stuff here useful