The Journey Continues: Discovering My Role as an Architect in an Agile Environment

Avraham Poupko(✉)

Cisco, SPVSS, Shlomo Halevi 5, Jerusalem, Israel
apoupko@cisco.com

Abstract. This paper continues telling the story begun in "It has been a long journey, and it is not over yet" (published in *Agile Process in Software Engineering and Extreme Programming XP2015*, Helsinki – 2015). This experience report tells the tale of the quest to define the role of the architect and of architecture in an agile environment. The primary observation here is that people skills are a key factor in that role.

Keywords: Experience · Journey · Extreme programming · XP · Architecture

1 Introduction

"When you come to a fork in the road…Take it."
Yogi Berra

In my last experience report I described my long journey along the agile road. I ended the report saying that the journey is far from over, and that it should be exciting to see how things evolve. I was right. It is very exciting, with surprises, twists and turns along the way. This second chapter of the story goes on to tell of an Architect trying to find his place within the agile environment.

2 Background

Since 1994 I have been working for NDS (acquired in 2012 by Cisco). I write code and design systems. I take great pride in a job well done. Having fun is a major objective. My current role is defined as Senior Systems Architect. I am expected to be deeply familiar with the core products, to understand the customers' needs, and to lead the task of building something that meets or exceeds expectations, while remaining in line with the company's technical and business objectives.

I work directly with customers, developers and development leads. In addition to being an architect, I manage a large team of architects, and as such I am expected to provide them with leadership and technical guidance.

The company I work for (Cisco) is committed to agile, and continues to adopt many of its practices and values. The Agile Transformation includes a deep look at the current roles in the organization and an attempt to understand how they will adapt to an agile world. As one that has been with the organization from early on, I am deeply involved in that

H. Sharp and T. Hall (Eds.): XP 2016, LNBIP 251, pp. 226–234, 2016.
DOI: 10.1007/978-3-319-33515-5_20

transformation. So it is natural that people often ask me, "What *exactly* does an architect do in an agile environment?" (Often, when asking the question, they will emphasize the word *exactly* and accompany that emphasis with tone and gesture that express more skepticism than genuine curiosity). And the honest answer is that I do not know *exactly* what an architect does or should do, but I can tell them what *I* do, and by way of generalization, deduce what an architect does.

3 The Scary Problem

Often people in the agile community will say something like, "The architecture is emergent." They go on to say that we do not really need architects any more, as agile does not believe in upfront planning. This really has me quite challenged or even worried. I am an architect, I do architecture, and as such, I like to feel that I bring real value to the organization I work for. If the architecture is emergent and we don't need an architect, then what does the architect do? How can the role of the architect be justified?

Put a bit more formally the question can be formulated thus: Agile means responding to change. The Agile Manifesto item that most strongly represents the agile spirit is *we value responding to change over following a plan*. Looking up the word "agile" in a dictionary, I came across the definition, "able to move quickly and easily". On the other hand, architecture is about long term planning; the architect has a long term vision of the domain and of the software system, and engages in technical planning. One of the definitions of architecture is "a unifying or coherent form or structure". At first glance, long term planning and responding to change seem to be at odds with each other. Hence the question. Why do we need an architect in an agile environment?

Historically, one of the drivers behind the agile movement was frustration with the architects. Architects design a system based on requirements. Developers implement a system as described by the specification. However, for many reasons requirements might not really carry the true intent of what the customer wants. Furthermore, the requirements even when clear and well understood, are in constant flux, and as a result architects are often in the business of predicting the future. Not only do we define what requirements the system needs to fulfill, but we also define what the future requirements are likely to be. In that environment, an architect makes decisions based on predictions, many of which are wrong to some extent or another. As Yogi Berra quipped, "It's tough to make predictions, especially about the future." If the predictions are wrong, then an implementation based on the design stemming from those inaccurate (or just plain wrong) predictions will not meet the customer expectations.

Frustrated with these wrong predictions, the agile community adopted the practice whereby all decision-making is deferred to a time of greater certainty. Now we can restate the question. If architecture is emergent, and it is difficult to say anything meaningful about the future, what is the role of the architect? It was this question that forced me to better define what exactly is architecture, and more importantly, what an architect does. I raised this question with many of my colleagues. All of them agreed that agile projects need architecture and architects. They differed on what exactly those architects and architectures are needed for, and quite a few of them were not able to clearly

articulate that need. But through the aggregation of all those answers, I was able to come up with some meaningful insights.

4 The Weekly Meeting

About two years ago, I started a weekly meeting with other architects and technical leaders. Since many of us live and work in different countries, these meetings were often phone calls. We would hang around for an hour or so, and discuss the contributions we have made as architects, any interesting issues we faced that week, and what about our implementation of agile needs to be fixed. This turned out to be a great platform for discussion and idea sharing. I discovered that some architects are often great at retrospection and introspection, and others are quite good at understanding people and how they interact with each other. Agile is about people more than it is about technology. So these insights on how people work turned out to be extremely valuable.

One key insight that we kept returning to is that whether or not it is defined in the role of the architect, the agile architect deals with people, and a large part of the architect's role is a people role. For example, we were discussing the responsibility of the architect to communicate design decisions. We asked the question, to what extent should the architect feel obligated to *convince* the team of the wisdom of his decisions? We all agreed that not only must the architect explain the rationale behind his technical decision; he must convince the team of that rationale. Why? Because a team that understands and agrees with what they are doing, have a better chance of doing the right thing. If they identify with the work they are doing, they will do high quality work.

5 No Design Phase

I think that part of the question regarding the role of an architect emerges when agile is compared to waterfall. In a waterfall environment, there is an explicit stage called the *design* phase. This phase is **led by the architect**. He designs and documents the solutions. The design is prescriptive. The assumption is that if the developers follow the design, then the software will be OK. This design phase is the most demanding and solemn of the entire software phase. If you get the design phase wrong, you will end up paying a high price for years to come.

On the other hand, in an agile environment, there often is no design phase. In consequence, the reasoning often goes: "If in an agile environment, there is no design phase, so there is no designer hence no architect." So we must clarify. No design phase does not mean "no design". No design phase means that there is not an explicit phase during which design happens exclusively. Rather, **design happens all the time**. Likewise, when we say that the architecture is *emergent,* that should not mean that we do not need architects because architecture just happens. That is not true. It means something much more subtle. When we say that the architecture is emergent, we are saying that as the system evolves, a certain structure emerges. Someone needs to keep an eye on that emerging structure. When it is emerging according to plan, the architect will strengthen it, and when it is starts diverging, the architect will act accordingly. Maybe adjusting

the plan, or maybe pushing the development in a particular direction – or both. Personally, this was the most dramatic change for me as an architect when transitioning to an agile environment. The fact that I am designing all the time, and not just during the design phase, has a significant impact.

So in my current role, I am always designing. The domain is changing, the requirements are changing, and our understanding of the world is changing. Agile realizes and accommodates these changes. As a result, I am always designing.

5.1 Reservation

I'll be honest. Not because I have to, but because I choose to. Making a decision at the "most responsible moment" is a great slogan. In reality it is somewhere between very hard and impossible. We often do not know what the most responsible moment is until that moment is long gone. That has often happened to me. I defer a decision to a later moment, because I feel that it would be irresponsible to decide now. At that later moment, I realize that I missed the opportune moment. This happens much more often than I care for. Every time it happens, I realize (once again) that my responsibility as an architect is not only to make and communicate technical decisions, but to make them at the right time.

It is safe to say that as an architect I am always designing. I am always trying to find the right time to make design decisions, and I am often fixing mistakes when I missed the right time for a particular decision[1].

6 Retrospection

Looking inward, I notice more and more that most of what I do as an architect is to interact with people. As an architect, even when I write technical documents, I am dealing with people. When a developer writes code or script his primary audience is the compiler or interpreter. If he writes something that is clear and unambiguous, but is not clear to the compiler he has not delivered. His secondary audience are human beings. The developer writes clear code with useful names for classes and functions, he writes comments that are concise and do not contain too many lies.

On the other hand, when the architect designs something, or creates an artifact of any sort (diagram interface and so forth), his primary audience is the human being reading that artifact. The architect is constantly learning and explaining. On further inspection, architecture itself is a very human thing. Structure or lack of structure are how humans perceive the system. This is captured beautifully by Christopher Alexander in the introduction to *The Nature of Orde*r. Order and symmetry are perceived by humans and used by humans. Even when humans cannot find formal

[1] For an insightful discussion of last possible moment as opposed to most possible moment see http://wirfs-brock.com/blog/2011/01/18/agile-architecture-myths-2-architecture-decisions-should-be-made-at-the-last-responsible-moment/.

words to express what it is they are feeling, they share a sense of *orderliness* that can be communicated through aesthetic design.

"The structure I identify as the foundation of all order is also *personal.* As we learn to understand it, we shall see that our own feeling, the feeling of what it is to be a person, rooted, happy, alive in oneself, straightforward and ordinary, is itself inextricably connected with order. This order is not remote from our humanity. It is the stuff which goes to the very heart of human experience".

Wow! Does that mean that when I, as an architect, deal with order, I am dealing with the stuff that "goes to the very heart of human experience"? That is way more than I bargained for when taking this job.

7 Domain Knowledge

Even when the architecture is emergent, the domain certainly is not. The domain is the real world. It is the business need that the software is there to solve. The business needs and the rates of change of those business needs are not emergent. What might be emergent is the understanding of the domain, and the software structures that support that understanding.

Domain understanding is critical for the success of the software. Good architecture will not emerge from software being developed, unless the developer of the software has a good understanding of the problem she is solving.

Someone needs to understand the domain, and be capable of organizing that understanding and explaining it, and then, by observing how that understanding is reflected in software, confirm that indeed the understanding was correct. And in cases where the understanding was not correct, that someone needs to fix things.

That someone is the architect.

8 Dependencies

One of the most complicated things that the architect deals with is dependencies. It is easy to state, "A is dependent on B", as though dependency was a binary value that either exists or does not exist. In reality, it is the nature of the dependency that matters. The architect understands the dependencies between the domain elements (human and otherwise) as well as the dependency between the solution elements (human and otherwise). She is able to offer guidance based on her understanding of those dependencies.

An example can illustrate this point. In one particular project, we recorded dependencies in Rally. Each user story had a list of other user stories it was dependent on. Those user stories had to be implemented first before the current user story could be implemented. One time, after we put all the dependencies in place, we realized that we have a cycle or gridlock where A (Optimize Bandwidth Usage) was dependent on B (Configure Channel), and B (Configure Channel) was dependent on A (Optimize Bandwidth Usage). That caused a few moments of panic until I pointed out that the dependencies have different meanings. When A was dependent on B, we meant to say that we cannot implement A until we have a proper understanding of B, because B is the primary

client of A, and thus if we do not understand B we cannot implement A. On the other hand, when B was dependent on A, we did not mean to say that B is of zero value without A, all we meant to say was that B cannot provide its full business value without A.

The nature of these dependencies were clear to me as an architect, and were quite clear to many on the team that understood the domain and the project. However, the graph of the features in Rally seemed to indicate a circular dependency. Once I pointed out the nature of the dependencies, the way forward was obvious, and we decoupled A's implementation from A's interface in a way that allowed development of B to progress.

9 Metaphors

Rebecca Wirfs-Brock elaborates on the various types of architects ("Why we Need Architects and Architecture on Agile Projects" – Most recently presented at ILTam Conference 2015). I would like to propose some additional metaphors that describe the architect. These are not distinct. Rather in an agile world each architect has all of them. Personally, I have acted in all these roles, often in many of them simultaneously.

9.1 The Tribal Elder

In this role, the architect acts as the guardian of the collective memory. He has seen the domain evolve and the architecture evolve. He understands why decisions were made. He understands the deeper meaning of the various dependencies. He is also very experienced beyond the scope of the particular problem. He remembers mistakes that were made, and how they can be avoided. He remembers useful lessons. He has seen how many predictions that were made with certainty do not materialize, and he has seen many surprises. He has a wealth of stories that he is happy to share to make his point. He might not be as "hands on" or as quick on his feet as some of the younger ones, but he more than compensates for that with maturity, understanding of human nature and experience.

One of the challenges with retrospectives and with lessons learned, is how do we preserve these insights over time? How do we maintain a record of these lessons? In a lean or agile environment, this tribal elder is constantly making sure that lessons that were painfully learned are not forgotten.

The challenges for a tribal elder are many. Too often he expects that people will accept his word on authority alone, rather than on the merit of his idea. On occasion we might find that the tribal elder has lost touch with the times, as lessons learned a long time ago might not be relevant any more, and the tribe elder will try to keep us in line with dogmas that are no longer relevant. So this architect needs to be humble and to have a very good sense as to when his experience is of value, and when it is outdated.

In my group, we are constantly caught up in the tension between a generic product and a customized product tailored to meet the customer. This tension has existed for over twenty years, and we periodically cycle from one extreme to another. Now, when someone suggests a shift from a generic product to a customized product or vice versa, the tribal elder (me) will tell stories of what worked and what failed in the past. The tribe elder will warn of pitfalls, while taking care not to dampen any enthusiasm.

9.2 The Architect as the Human Document

A document is a source of information and of agreement. If you need information consult the document. If you are in an argument, the document might be able to arbitrate. The architect can fulfil some of that need but even better.

Sometimes code is referred to as the "Living Document" – meaning that the code is always up-to-date in communicating what the system does. (This is as opposed to a "dead document" that at best was correct at some point in the past.) However, code will only tell you what the system *does* and *how* it does it. It will never tell you *why* the system does what it does, or what the system *should* be doing.

The architect is able to actually explain all the relevant knowledge that is not captured in code. Relevance is a matter of context. Because the architect is contextually aware, he can give the information that is relevant in a particular context.

There are several dangers that the architect as a human document must be aware of.

First and foremost, the architect must not think that he can forgo clear, concise and accurate documents, just because he is smart and articulate. Good documents have the advantage of being unambiguous, context free, and they never change their minds. Interfaces must be documented. This is especially true for teams that work over long periods of time or across great geographical distances.

In the role of the human document, the architect must take care not to become a bottleneck. If everyone is waiting in line to get the time and attention of the architect, then the work gets held up. It is the responsibility of the architect to ensure that does not happen.

9.3 The Architect as the Potter

Looking at a potter working, you will notice that the interaction between the potter and the clay is very light and very accurate. The illusion is so strong that the clay might actually think that it is molding itself into a vase or bowl. The clay does not fully understand the role of the potter, and might see him as redundant. But that is not true. Without the potter's continued presence and intervention, the clay will find itself all over the walls of the workshop. On the other hand, the potter needs to be gentle. If the potter is too aggressive, the clay will resist, and the result will be a shapeless mass. The "potter architect" does the same. He **allows** the architecture to emerge on its own, and only makes featherlight touches here and there when he sees that things need a little fixing. If he is really good, the people will say that the architecture is emergent. He might be so effective the team will start asking what they need him for.

10 Qualities

As I mentioned in the opening of this paper, it is becoming apparent to me that the role of the architect in an agile environment has a great deal to do with people skills. In my own personal journey I am discovering that quite often I need to call on those skills in order to be an effective architect.

I would like to outline some traits that I find particularly useful. While these are good qualities in any person, they are critical for the architect. I am not including the standard qualities such as a quick understanding and good memory. I am focusing on social skills.

Teacher at Heart – The architect must be able to explain complex material at the depth that is appropriate to the audience. He must be able to tell if the audience understands what he is saying, and if not, he must explain it again. He will use all tools that a teacher uses. Metaphor, drama, humor, multisensory, example, tonality and so forth.

Empathy – The architect must be able to see things from someone else's perspective. That will allow the architect to understand the domain and to be able to explain the domain as well as the solution to people that have experiences other than his own. The empathetic architect understands that his perspective is not the only perspective, and that the underlying assumptions of the audience, are very different from his own.

Sense of Humor – The architect must take himself seriously, but not too seriously. A sense of humor will allow the architect to see new and surprising angles on things. A sense of humor will allow the architect to break patterns in interesting ways and find a solution to an elusive problem.

Humility – The architect holds power, authority, and a great deal of respect. His word is often the final word on technical matters. He often gets to set the technical direction of a product or a project. However he is not immune to mistakes, and his mistakes can often have far reaching consequences. Agile is about recognizing and admitting mistakes. An architect that sticks to a decision just because it was his decision will fail. The architect must be prepared to admit that he made an error in judgment, or was not diligent enough in his research.

11 Summary

Agile is about responding to change. Architecture is about structure, uniformity and stability. The architect has a high awareness of how to balance the two. As the system undergoes change, the architect will understand and communicate what parts of the system need to stay the same and what parts should change.

The architect is the bridge between the people and the technology. He not only understands what the software does and knows how to get there. He is deeply aware of the human aspect, and uses his human understanding to lead the project forward.

Acknowledgements. The ideas in this article have been evolving over a long time. I thank all my friends and colleagues that patiently listened to my musings and provided feedback and adjustment. Thanks to my colleagues from Cisco, David Russ and Warren Pratten for many hours of fruitful discussion. I look forward to more such discussions in the future.

Special thanks to Rebecca Wirfs-Brock who encouraged me to write this paper and who provided a great deal of valuable insight and feedback.