Table of Contents

How Does The Role of Architect Change in Agile? - SSA 3
What is the Difference Between Emergent and Intentional Design? 3
Why Should an Architect Collaborate with Teams? 5
The Enterprise Asset 5
Interactions 6
Collaboration 7
How Do We Introduce a New Technical Approach? 8
How Do We Work on the “Big” Architectural Initiative? 9
How Do Architects Ensure Consistency? 10
How Does the Architect Interact With the Product Management? 10
What Are the Benefits of a Collaborative Approach to Architecture? 11
Want To Know More? 11

How Does The Role of Architect Change in Agile? - SSA

The role of “Architect” in an agile transformation sometimes causes confusion. Traditionally there has been a need for someone to not just worry about the current features to be delivered but to also look ahead and understand how they can position the Teams so they can work coming requirements. There has also been a need for consistency in an overall enterprise offering so that customers of that offering feel like the solution could have come from one person, to reduce the amount of re-invention we have to do, and to ensure we have clean implementation regarding non functional issues such as security, performance, and so on. That need does not go away just because we now have empowered Teams. In fact, it could be argued these issues are even more critical in the new world.

> Note: This page does not try to define the role of “architect” in its various forms. This is more a discussion on the changes in thinking we need to have to be a successful agile architect.

What is the Difference Between Emergent and Intentional Design?

Early implementations of agile often seem to hide the need for good design. Most of the time the assumption was that “if you have good people, with a good understanding of design principles then good design will follow.” The Agile Manifesto argued that

> “The best architectures, requirements, and designs emerge from self-organizing teams.”

This has proven to be true - there is a huge benefit in having the Teams do design as they do the work (assuming Teams with strong technical skills). And it has worked for many situations. But as you have more and more Teams contributing to the single solution, or as people started to apply agile into more and more existing IT situations, it became increasingly clear that something more was required. The question then became

> “How do we leverage this emergent property of design, while also keeping overall guidelines in place and doing some level of look ahead planning?”

SAFe in particular recognized the problem and developed the notion of intentional architecture in addition to emergent architecture. One of the key ideas was that for a Train program that the System Architect works with the Teams to build an “architectural runway” of new “enablers” that enables the Train to develop new capabilities better, faster, and cheaper.

The problem with this distinction is that, for many, the idea that we need “intentional” architecture is an excuse to say “the role of the architect does not really change”. This misses the point. The discussion must be more nuanced than this. A couple of examples:

- Lets say we are about to embark on a significant re-design / re-factor of an existing legacy, “hairball” application that is worked by multiple Teams. In this situation if we just have each Team go off and refactor at will, the application will probably not work for a long period. Some level of alignment across Teams, with driving principles and approaches needs to happen to reduce the risk to the application. One the other hand, an architect deciding that “here are all the interfaces” to develop to might miss key issues and functionality that only the people working the code can see. What approach might we take. A first step might be simply to have all Teams adopt a common function / variable naming convention across the application and use the IDE to enforce this so the real structure of the application become more clear. So the first principle might be to establish this approach, and to work with the Teams on making it happen. The Architect would be the person driving the consistency of approach based on a solid understanding of the current state of the system as well as what is possible through collaboration with the Teams involved.
- Lets say we are about to select a foundation technology to build our new capability on. In this situation, if we don't have some degree of commonality, then we could up with an application that does not talk to itself. The traditional approach is to have the architects determine the best foundation right at the beginning of the project, and ensure that the Teams comply with this approach. There is no doubt that a single solution is required, but agile approaches use the collective knowledge of people to generate data to come up with the “best” solution. The architectural approach in this case might be to work with each of the Teams on a different foundation technology (so multiple experiments) and then use the data generated from those experiments to make a decision as late as possible (“the last responsible moment”)

From these example you can see that architects need to adopt a different approach. Part of the problem is caused by the traditional way architects try to address these issues. The mindset of command and control applies in this space as it does in many others. An architect will issue guidelines and demand reviews of designs, for example. From the perspective of the Teams, Architects are often perceived as providing guidelines that do not work in reality and as the place where innovation stops as architectural approval is required, and that is often slow in coming. It’s not that Architects want to be in this position; they are senior people working hard. But that is how it works out.

What we are trying to get to a more collaborative approach for this role, just like we are with any leadership role. Like all agile, there is a mindset shift required from Architects as they a work with agile Teams:

1. Collaboration: According to James Coplien, the mindset required is the same as all agile and lean - “Everybody, all together, from early on.“ The idea is that instead of treating the job as if it can be independent of the work, Architects need to be involved from the beginning all the way to end. This is interesting in that again, per James Coplien “Maybe half of software development is about nerd stuff happening at the whiteboard and about typing at the keyboard. But the other half is about people and relationships.” This is a significant change for many Architects.
2. Architecture as a Product or Service: We need to position Architects so that they are seen to provide a valuable service to trains and their “product” is the service of helping with good design etc. We have to encourage architects to ensure that their “products” are attractive to customers (Teams on Trains) so they want to use it, not required to use it “because we said so”.

And we are leverage key agile ideas to help us improve the result. From the examples above:

- De-centralized decision making: Understand when decisions need to be centralized versus de-centralized.
- Keeping options open: To maximize the data we have to make a decision and so reduce the risk of that decision.

So how does this work practically?

Why Should an Architect Collaborate with Teams?

One role of the Architect is is help the Team get features out the door, better, faster, and cheaper. Another role is the work to improve the Enterprise Asset.

The Enterprise Asset

I first came across the concept of the Enterprise Asset while reading “A Seat at the Table” by Mark Schwartz. The idea is that our IT infrastructure is pretty much the embodiment of how work gets done in an organization. “When we add all of our current IT capabilities together, we arrive at an asset that enables the enterprise to earn future revenues and reduce future costs — that is, an asset in the classic economic sense. … The asset does not appear on the company’s balance sheet. … Much of its value is hidden; its ability to support Agility, for example. But it is an economic asset nonetheles.“

To me the idea of an “Enterprise Asset” helps make IT more like a product development shop. There is, in product development shops, an obvious asset that we are working to improve. This allows us to make investment decisions based on the total life-cycle value of the product.

Similarly, while the Enterprise Asset in an IT shop is made up of our components, our products, our solutions (whatever it is we deploy) the real value is when you consider all these items together holistically as this is where you will be able to, for example, make trade-off decisions between competing investments.

> Architects apply lean and agile values and principles to support the long term viability of the Enterprise Asset or the Product.

Note: from here on in I will treat the notion of “Enterprise Asset” and “Product” as interchangeable.

One of the reasons I really like the notion of an Enterprise Asset is that it allows us to think about how we spend our IT budget. We have a tendency to think in terms of the new project budget being different to the keep-the-lights-on budget although it is the same group of people doing the work and both budgets result in improvements to the overall Enterprise Asset. The different budgets result from different accounting view of capital vs operational tracking and, while valid for accounting practice, actually get in

the way of thinking about how we work to improve the Enterprise Asset.

Why is this important? Most people will report that they are spending too much of their limited budget on keep the light on activities, that they would like to spend more of new capabilities. In fact, a recent Computerworld article noted:

> In a recent Forrester Research survey of IT leaders at more than 3,700 companies, respondents estimated that they spend an average 72% of the money in their budgets on such keep-the-lights-on functions as replacing or expanding capacity and supporting ongoing operations and maintenance, while only 28% of the money goes toward new projects.
>
> Another recent study yielded similar findings. When AlixPartners and CFO Research surveyed 150 CIOs about their IT spending and their feelings about IT spending, 63% of the respondents said their spending was too heavily weighted toward keeping the lights on.

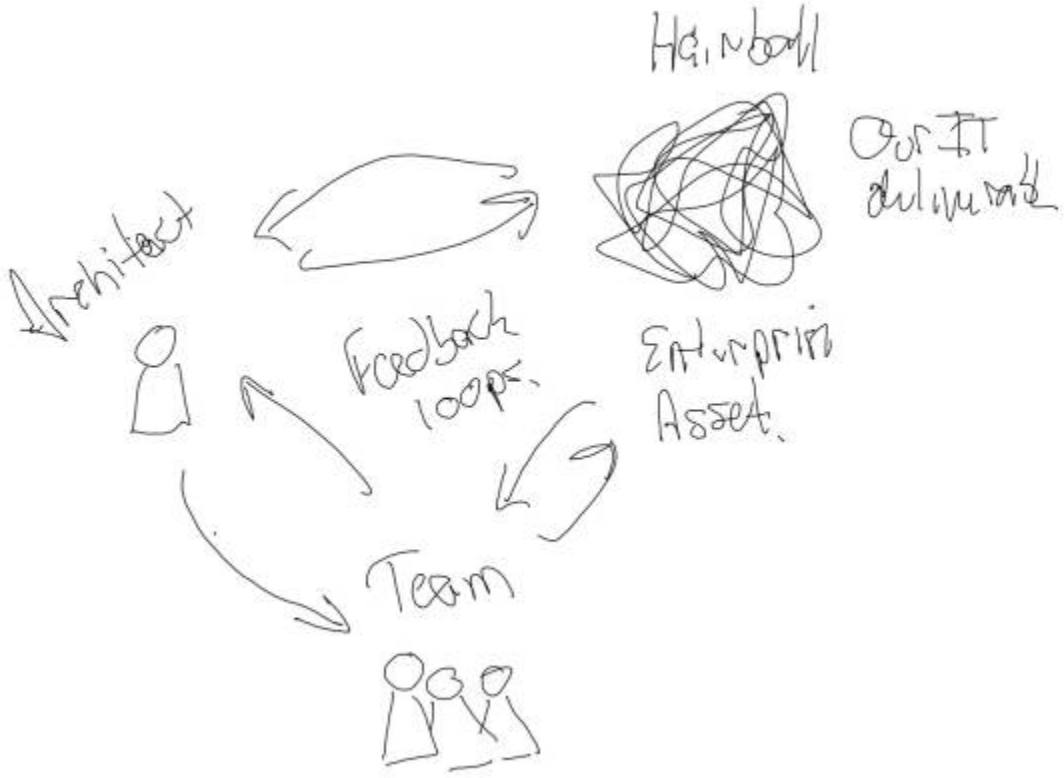
This is probably less of an issue in a Product development shop, but there are probably similar numbers. By thinking more holistically improving our understanding of where we should be investing rather than worrying so much about budget allocations, we can make investment decisions aimed at improving the overall asset. This is not to say we should worry about the budget. Just that we should capture budget information as needed and not let it necessarily constrain good decisions.

The traditional Architecture approach is to provide guidelines and guardrails, documentation, that Teams are expected to follow. Often this does not work out. To help address, Lean and agile suggests Architects collaborate directly with the Team. By having the notion of an “Enterprise Asset” we can frame what form that collaboration should take.

In many ways the terminology of “architect” and “architecture” misrepresents what Architects should be concerned with. Unlike buildings, the Enterprise Asset is not static. It is constantly in a state of change, with all your technical people adding to, changing it, it all the time. A better metaphor for what the architect does is more of a gardening metaphor. Like the Enterprise Asset, plants grow in a landscape and require constant attention by hands-on masters mindful of the original or evolving vision. This means that, unlike building Architects, IT Architects have to be involved in the day-to-day work of the Enterprise Asset to be effective. This is not just an “agile” observation but rather a more general commentary on the effectiveness of the role of architect. The thinking becomes even more important as we expect Teams to take responsibility for all their work.

Interactions

There are 3 potential interactions between the Architect, the Team, and the Enterprise Asset:



The Architect has an understanding of the Enterprise Asset, and it is probably a wider view of the system than the Team has. With this understanding, the Architect wants to help the Team do things the right way.

In the meantime, the Team is also working directly on the Enterprise Asset, and in many situations, the Team will have a more detailed understanding of their portion of the system. When the Team looks at what the Architect is suggesting in terms of guardrails / guidelines the Team will sometimes find that these are in conflict with what the Team knows is the best, or most practical, approach.

The reality of the situation might be that there is in fact a problem with either the guideline, or the system. Or it might be there is just a misunderstanding on either side. Irrespective, if the Architect is working directly with the Team, collaborating, then this becomes an issue that can be worked, thus improving both the overall Enterprise Asset, as well as the work of the Team.

If the Team is working directly on the asset and the Architect is telling the Team to do something that does not make sense, the interests of the Architect will be overridden as the Team is tasked with delivering to the customers.

Collaboration

The reason that Architects need to collaborate directly with Team is that the Enterprise Asset is a

complex system. When working with the Enterprise Asset, a simplistic view of it does not help. Understanding is improved through working directly on the Enterprise Asset. This sets up feedback loops - what we thought was versus what is actually happening.

The reason Teams need to work with Architects is that Architects have a wider understanding of the Enterprise Asset. This means they have heuristics for understanding wider impacts on the Enterprise Asset as we make changes. Again, this is a feedback loop.

When Architects work directly with the Team on the Enterprise Asset it sets up a positive reinforcement loop. Architects working with Teams see their ideas implemented and increasing understand the Enterprise Asset. Teams working with Architects will be able to leverage the wider knowledge of the Architect and will see their impact on Architectural decisions.

One note on the word “collaboration”. For many traditional organizations “collaboration” means “we have a meeting.” Meetings are certainly part of collaboration, but really we need something more than this. When a Team is working on something that needs input from an Architect, the Architect should be pairing / swarming with the Team, potentially even working directly on the keyboard. In this way the Architect develops real understanding of the effect of changes being made the Enterprise Asset. Trade-off decisions as a result of the implementation will inform level and type of guardrails needed, and when guardrails should be ignored. The result is an improved understanding in the trade-off between emergent and intentional architecture that will lead to better guardrails with a higher probability of acceptance, and therefore improve the Enterprise Asset.

You might hear people refer to “going to the Gemba”. The Gemba is the place of work and this is the general notion that you can only have a real direct understanding of the work if you go to the place of work - sitting in your office doesn’t cut it.

In reality, an Architect’s view of what should be standardized, what we need in terms of guardrails, will also have an understanding of when that standard should not apply. There needs to be variability. The Teams are probably the best source of good variability. In fact Architects are often in danger of standardization that does not make sense to the people doing the work. The role of the Architect in this area is to help teams to understand good variability versus good standardization, with everyone involved understanding the dynamic.

In many ways, this dynamic is also a difference between “intentional” vs “emergent” architecture.

How Do We Introduce a New Technical Approach?

One of the frustrating aspects of being an architect is knowing that there is a particular approach or technology that will help the Teams but also facing resistance from the teams in actually implementing that idea. Through a collaborative approach you will find there is help in getting some of these ideas to

move. This is, in fact, one of those places where there is a significant benefit to having a large scale organization to work with.

In my early career as an agilist for a product development shop, we knew there was a benefit to doing test driven development but for a number of reasons (feature pressure, concern of the unknown, disbelief in the approach, ...) we were unable to get teams to really bring the practice into their world. No amount of presentation, cajoling, threats made any kind of impact. Then I found a team that had actually done it, and were very pleased with the results. We had that team talk about their experiences and suddenly all kinds of teams were now adopting the practice.

It turns out that, no surprise, people listen more to the peers than anyone else. And in a large organization there is probably someone who has tried whatever practice or technology you are interested in working. Architects who are open to these experiences and work directly with teams are best positioned to leverage these peer effects.

How Do We Work on the “Big” Architectural Initiative?

The base recommendation is that, where possible, you split large chunks of work into smaller ones but focus on still providing end user value even when you do the split. In other words, as you divide epics / feature / stories up ensure there is some valuable functionality with each as you deliver. This contrasts with the approach a typical technical organization would take when given a large amount of work to do. Traditional organization have the instinct is to split the big effort along architectural boundaries (data access layer, middle tier, client tier) or components and then build all the real functionality over once the base layers are in place. The problem with the traditonal architectural approach is that:

- Architectural Development needs to continue for a long time before anything is provided to an end user.
- Architectural Development will often proceed in isolation, so that integrating all layers happen late, adding risk.
- There is no proof that the architecture will work in the context of the end user requirement until the end when end user functionality is exposed.
- There is a tendency to do “technology for technology sake” resulting in wasted effort that does not provide value to end users (“but we might need it”) which means we add to the future maintenance and support burden.

In other words we need to resist dividing the epics / features / stories up along “technical” boundaries and components as this will result in iterations of development with nothing complete (and valuable) to the end user, waste, and added risk to the schedule.

Rather we need to deliver new architecture in a useable context. So instead of building all the architectural components and then combining them together to get something done for the end user, you build a thin sliver of functionality that passes through all the layers and components and determine how to pull together all the systems required to make this happen as you go. Even if the end user cannot

actually do any real work at this stage, you can get feedback from the customers. The value here is "learning" but that learning is not "this is how the widget works" but rather "this is how all these widgets work together to deliver value and how do can use / deploy it".

When building in a DevOps world, this approach has been extended and labeled as the Walking Skelton approach - see Kickstart Your Next Project with a Walking Skeleton. And then there are various "riffs" on this walking skeleton approach.

As these implementations are put in place, the people doing the work should keep certain principles in mind. Principles include things like "single responsibility", common naming / set up for interfaces, rule for interfaces (eg you cannot delete a public interface, only add to), etc, etc. This is where Architect role adds additional value. This also why Architects need to collaborate through the whole implementation process.

How Do Architects Ensure Consistency?

Let's take the example. One of the roles of an Architect is to establish the guardrails (or standards) by which the organization (enterprise, Train, Team) operates. However, we need to be careful that we say role is to just "establish guardrails". This is a problem in that:

1. It is a negative job from the perspective of a Team or Train (ie architects say "no").
2. It does not allow for creativity from the Teams (ie "we want to try this")
3. It means Architects are a potential bottleneck (eg everything needs approval)
4. Its not too far from what people think of as Archtects do today (ie an ivory tower where commandments come from)

Instead of this approach, part of the new job of the Architect is to work directly with Teams to help them understand why the guardrails are in place and, as Teams work solutions, to work those new ideas into the guardrails and distribute those ideas to others.

How Does the Architect Interact With the Product Management?

Traditionally Architects develop an independent understanding of what the customer needs. This is best developed by direct conversation with customers, but in many instances is developed independent of that discussion, and also independent of the overall product strategy.

This approach could be more effective. To improve this needs to be a more collaborative effort.

Architects should be involved directly with customers through as Product Management / Product Owners discussions happen (not separate).

> If there is a customer meeting, then the Architect should be involved.

And there will be a lot of customer meetings, won't there:-O

One way to think about the establishment of the "architectural runway" is that the Architect, having heard what customers are thinking of longer term asks themselves the question "What do we need to put in place so that our teams can deliver these upcoming requirements better, faster, cheaper?" and / or "How will this effect the Enterprise Asset?"

What Are the Benefits of a Collaborative Approach to Architecture?

From the above, and to paraphrase Scott Ambler, the benefits of a collaborative approach to architecture can be summed up using these three words:

- Better. An agile architecture enables agile teams to produce better quality solutions for their stakeholders by providing a more reliable ecosystem with which to work.
- Faster. An agile architecture enables agile teams to deliver solutions to market faster due to improved reuse and infrastructure quality.
- Cheaper. An agile architecture enables agile teams to deliver solutions to their stakeholders at a lower cost due to improved reuse, greater quality, and greater platform consistency.

This is all about enhancing the Enterprise Asset. Perhaps a more succinct way of saying this is that the architect's job is to focus on "design for deployability" based on doing the work aimed at increasing the flow of value.

These benefits need both intentional and emergent to fully leverage the combine intellect of the people doing the work.

Want To Know More?

- System and Solution Architect
- Kickstart Your Next Project with a Walking Skeleton
- "Riffs" on this walking skeleton approach
- "Lean Architecture: for Agile Software Development" - James O. Coplien and Gertrud Bjørnvig
- LESS Architecture Design
- Torbjörn Gyllebring : The Reverse Conway - organizational hacking for techies: Interesting

discussion on the relationship between system and organizational design.
- Scaling the Practice of Architecture, Conversationally: Really thought-provoking article on how we can have more architecture done by teams than from an ivory tower. Makes a specific proposal of a practice based on one core element - the Advice Process - and four supporting elements - Architecture Advisory Forum, Lightweight ADRs, Team-sourced Principles, and Your own Tech Radar.

FAQ, Management, Culture, Mindset, SAFe, Architect, SubjectSpecificArticle