Xebia

E-Guide

Thinking Forward About Agile Architecture

IT Architecture is the central foundation of your Digital business as well as the enabler of innovation and competitive advantage. But how can we approach IT Architecture to achieve our business goals in a sustainable way?

This e-guide provides insights about the value of Agile Architecture to your organization and how to bring this value into reality.

by **Edo Poll, Kenny Baas-Schwegler, Paul de Raaij and João Rosa**

Index

The value of Agile Architecture in a modern organization

Talking about the added value of applying Agile Architecture in your organization, we see fewer and fewer “IT architects” in organizations. Is that because we do not need Architects anymore?

by **Edo Poll**

Do We Need Agile Architects Or Do We Need To Do Agile Architecture?

In fact, nowadays, Architecture has shifted from a job title to a role. And this role or this function can also be assigned to a whole team. Even different people working in product teams can fulfill the role together.

Mostly, this is due to the fact, that modern Agile Architecture requires a broad skill set to be done right as a business function.

Significant Decisions About Your IT Landscape

In times, when many organizations are facing or find themselves in the mid of a major (Digital) transformation, Enterprise Architecture helps you to align your organization, your initiatives, and the products you are developing with your goals. These goals can be business drivers or transformation goals.

It also helps you to manage dependencies that occur once you have started with lots of changes in an organization and keep them aligned with your strategic goals.

So, do we only need Architecture when we are going through major changes?

– Architecture helps you to make significant decisions about the key design of your IT landscape: components, systems or, domain-related. It can refer to all applications or systems you need to align with the business strategy.

The main value of Agile Architecture is to ensure that you are making the right significant decisions in all layers of your organization. That implies that you cannot limit the role of architecture to one single person within a growing organization. Different types of architecture roles add a different kind of value to the organization.

An Enterprise Architect adds a different kind of value than a Tech Lead or Lead Engineer, who does architecture as well.

Portfolio Management, Program Management And The “Business Wish List”

But what about Portfolio managers or Program managers?
- Agreeing that you cannot load the burden of all significant decisions on a very limited number of shoulders, it is obvious that cooperation is key, especially with Product management and Program management.

However, there are some differences to bear in mind:

Portfolio management and Program management are responsible for gathering all business requirements, the so-called “business wish list”. But when it comes to implementation of the wish list, please have an architect enter the stage:

Let’s take the example of a specific domain with a dedicated domain portfolio manager and domain program manager. They are responsible for developing different products within the domain. The domain architect on the other hand is responsible that everything which needs to be developed and what needs to be implemented, complies to the overall architectural guidelines, and is implemented in the right order.

For example, if there is a specific business requirement, the architect makes sure that all the right platforms and enablers are in place first instead of quickly realizing a specific feature request.

The architect watches over the broader perspective of the IT landscape, manages complexity, takes care of security, ensuring non-functionals are addressed in a proper manner.

Of course, Portfolio management can also take care of this aspect, but in practice, we must focus on the skills required. It is not about the name of the job, but about the skill set to take care of architecture. In the end, people from different functions can attribute their specific skillsets to the common effort of “doing architecture”.

Skill requirements in Agile Architecture

While the traditional training frameworks for architecture, e.g., TOGAF address processes and models, they leave out skills like stakeholder management and modeling, or analytic skills. Additionally, we are also noticing that DDD and strategic DDD become important skills in Agile Architecture

Another one to consider is technical leadership, experience in the technical domain which can be described as “know your stuff”. Also, do not underestimate communication skills. Even if you know your stuff very well, it won’t help you if you cannot get your message across.

Driving technical change or creating a solution design, your results must get out there and make impact. That requires a certain level of communication skills: verbal abilities and the abilities to communicate your architecture in an efficient way.

Finally, let’s talk about modelling. If you want to take the architectural responsibility, you should be capable of developing the right models at the right moment for the right audience.

The value of Agile Architecture in a modern organization

When companies transform towards an agile and DevOps way of working, they sometimes ask how to proceed with architects. Some companies ignore architects in their transformation, some will upskill their architects, and some will make the DevOps teams responsible for the architecture.

by **Kenny Baas-Schwegler**

A core problem we see is that those responsible for the transformation have little experience dealing with architecture in an agile way. The agile coaches are concerned with making the organization agile on a process level. The scrum masters are concerned with the agile process on a team level. And in some transformations, the team also has an agile technical coach who teaches the team new practices and techniques from Continuous Delivery, DevOps, and SRE.

In contrast, we observe that architects tend only to get training, if any, and no coaching. This article will lay out why it is crucial to rethink how organizations deal with architects and to start explicitly coaching architects when going towards agile architecture.

How The Role Of Architects Is Changing In Agile And DevOps

The considerable debate in the industry is whether we still require architects when we can make the high-performing software teams responsible for the architecture. That is an ideal situation that is very hard to bring to reality, and irrespective of how you organize yourself, you still need to practice architecture outside the development teams. In this post, we will describe why:

- Software teams have different levels of work, focusing on their purpose within the system. They focus less on and are not incentivized to think about a higher level of work.
- Software teams are parts of your organization's system design; we still need to ensure they fit and align with your business strategy and value streams.
- All those parts should be made responsible for their own architecture decisions; It is the responsibility of the architects to facilitate it and that requires architects to learn new social capabilities, which are hard to upskill by training itself and involves coaching.

So, where does that urge come from wanting to remove architects? Well before, architects did their job in a waterfall organization outside the teams. So the feeling development teams get about architects is that they reside in their ivory tower, isolated from the teams, making decisions for the teams from outside their context and that were disconnected from what was possible in practice. Making decisions for the teams made (some) sense in a waterfall approach because you needed to design all the architecture before moving it to the development teams.

And then, the organization goes through an agile transformation, primarily using the Scrum framework. Nothing is said about architecture; it says '(we have come to value) working software over comprehensive documentation over processes and tools.' And that statement usually gave teams the idea that the old way of writing extensive architecture documents is no longer needed.

> **Questions about whether design is necessary or affordable are quite beside the point: design is inevitable. The alternative to a good design is a bad design, not no design at all.**
>
> - Douglas Martin

But the problem we see is that the teams also do not know how to do software architecture correctly because they were never tasked and able to do so in the past. And as Douglas Martin said, architecture and design are inevitable; you can either have a good one or a bad one.

Along came the DevOps transformation focusing on merging Development and Operations practices so that the teams could take full responsibility for what they built. And as a bonus, it also brought in the architecture capabilities, which helped move the teams in the right direction. And it also removed much responsibility from the architects to the teams. Like what we observed in an agile transformation, architects are left out, which does not mean we do not need them.

We Still Require Architects

Software teams are moving towards autonomous high-performing teams, owning the products or services they build for and getting closer to their customers or stakeholders. In addition, by continuously improving their team practices and capabilities, they get to know the business value they are delivering better—an excellent thing. And because of this, we see fewer and fewer “IT architects” in organizations. Something Edo Poll already wrote about in his article ‘The value of Agile Architecture in a modern organization’.

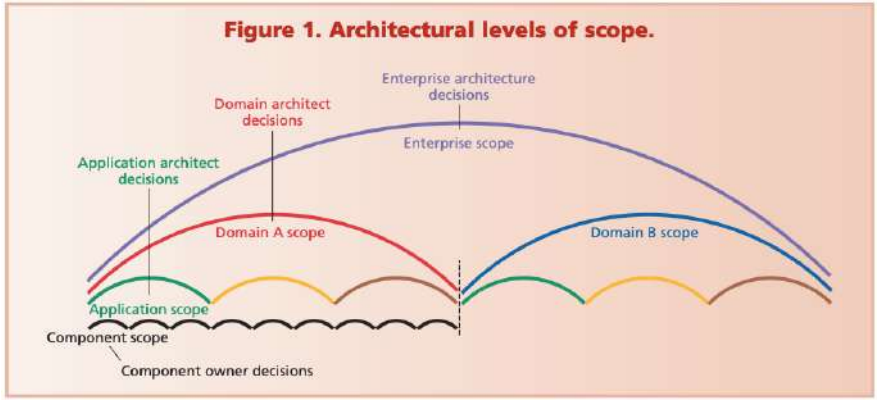
And organizations also need to take care that these high-performing autonomous teams stay aligned to the business strategy.
We observe two major approaches:

1. Organizations are implementing an agile scaling framework focused on implementing bureaucratic processes to manage the coupling between the teams to keep them aligned, lowering the team’s autonomy.

2. Or the teams get full autonomy and focus entirely on their own goal but do not know their impact on the organization’s business value stream.

And this is precisely why we still need architects, organizing business and technology teams for fast flow and either facilitating and making decisions with these teams or making decisions with them to make them aware of their impact.

Architects work on different architectural levels of scopes in an organization. In the whitepaper’ Architecture as Business Competency,‘ Ruth Malan and Dana Bredemeyer describe the relationship of an architect with a team flawlessly. “It cannot be up to developers and business analysts to decide what parts of the architecture to apply and what parts to ignore. If you treat your architects as consultants rather than decision-makers … all you can expect to get is an assemblage of parts with unpredictable properties.”
The parts of the architecture on different levels are not abstract from the other levels; they are different parts of the system. Teams can still make decisions on their scope, but not on the scope of another level.



So different levels of scope require other skills and a different way of thinking. That concept of architecture scopes also aligns with the work of Elliott Jaques about levels of work. Jaques makes a great analogy in that different levels of work have as different a nature in terms of modes of thinking as, by analogy, water is different to steam. Similar to how application scope is different from domain scope. Each requires other aspects and has other properties.

> **Different levels of work have as different a nature in terms of modes of thinking as, by analogy, water is different to steam**
>
> - Elliott Jaques

Architects should still be dealing with all the different levels of scope as they did pre-agile, to quote Gregor Hope, ‘ride the architecture elevator.’
The difference in agile Architecture is that architects now need to take the lead in new opportunities and then continuously shift the responsibilities to teams and managers. We can do this through participatory design.
We collaboratively model the solutions with stakeholders, users, and teams.
However, this requires a new set of capabilities of an architect.
These capabilities are more focused on the social domain than the technical.

Start Coaching Your Architects

Instead of getting rid of your architects during the transformation, train your architects. Teach architects how to make collaborative architecture decision-making and do collaborative modeling workshops. Teach them how to facilitate these workshops between domain experts, stakeholders, and software engineering teams—designing a shared sense of reality with a shared language and understanding. Facilitating these sessions requires architects to invest in people and group skills, like holding space, dealing with resistance, seeing the power of ranking, dealing with cognitive biases, creating buy-in, and managing conflicts. And learning all these new things takes practice and, most of all, coaching.

Coaching is vital because everyone will be insecure when dealing with group challenges for the first time. And understandably so, because we are entering a new domain of capabilities. Technical problems are relatively simple to solve, given enough time and knowledge. Dealing with social issues is a lot harder because they are complex. They require experience and reflection to solve. Every situation is different, and most problems architects are dealing with involve a person's inner emotional issues. For instance, if someone doesn't feel welcome in a group, that is a challenge that someone needs to be aware of during these sessions. Is it just that person's feeling in particular? they are dealing with? Or are they really not welcome in that session? No wonder architects then fall back on what they already know, making decisions in isolation and then getting stereotyped as ivory tower architects.

When this happens, we get consultants in to solve the problem. Their job is to facilitate these sessions and consult the organization into transforming and shifting the responsibilities and changing the team topologies by leading collaborative modeling sessions. And sometimes they help the architect upskilling. But as soon as they leave and do the job, the underlying issues with the current architects are not solved. Of course, part of consultants' job is coaching architects during their engagement; however, it is never the prime focus.

That is why we find it essential to make coaching architects more explicit in such a transformation. First, upskill architects via training and then define individual goals and growth paths to close gaps in skills and experience by applying the learned knowledge in practice. Next, let architects start co-facilitating these sessions to help them reflect on what they found hard and what issues they faced. Then coach them to facilitate sessions to face these issues and solve these for themselves.

Thoughts On Organizing Architecture

Talking about the added value of applying Agile Architecture in your organization, we see fewer and fewer “IT architects” in organizations. Is that because we do not need Architects anymore?

by **Paul de Raaij, João Rosa**

Do We Need Architects Anyway?

Looking at the current technological and organizational paradigm, we can only recognize the world is massively different from 10 or 20 years ago. We live in a world where we can instantly make use of infrastructure via cloud providers. Using common software, functionalities can be purchased and integrated with the click of a button and the availability of a credit card.

Gone are the days of creating large project plans and business cases. No need to negotiate the proposed solution for any given problem with the budget holder. No long debates with other engineers about the envisioned solution. Great for our agility, but it also has consequences.

Value-stream teams have been given more autonomy and possibilities to select, purchase and integrate hardware and software. Albeit via cloud providers where you can autoscale your infrastructure, or via Software-as-a-Service providers who offer you functionality out of the box. Gone are the days of making well-thought documents who are reviewed and tested by colleagues in the organization.

Clearly this benefits the speed of delivery and flexibility in choosing solutions. Consequently, however, it requires more maturity from a team. The solution proposed and decisions made not only have to fit the context of the team, but as well as the organization. The complexity and pressure to guard consistency and best interest for the organization, as a whole, now relies on the value-stream teams. The tension of choosing between different options and stakeholders now solely falls on the burden of the team.

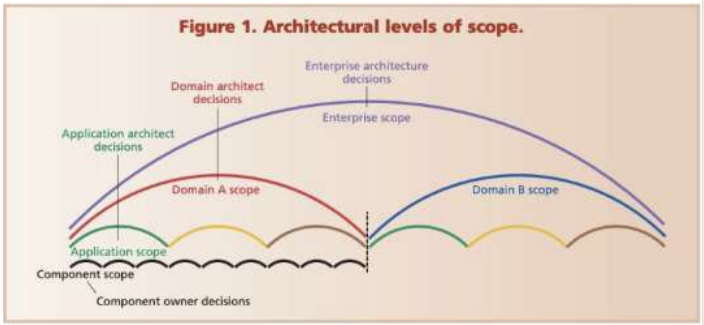
The architect function is key in getting to conscious decisions that are beneficial for the customers and the organization as a whole. Either by limiting the number of options that are available for a team, or by refining and improving the reasoning and acceptance of trade-offs between proposed solutions.

Organizing Architecture Guided By Two Perspectives

First-of-all, architectural scopes are not to be seen as static elements. Architects should be dynamic, understanding the purpose of management with the organization and the engineering challenges of the development teams. Gregor Hohpe describes this as riding the elevator. One way to determine scope is to look at complexity. The scope of complexity for a value-stream team is different from the complexity of a whole organization.

Inspired by Ruth Malan and Dana Bredemeyer, we can use the model where the architecture function can be organized in two axes: on the vertical axis is the locality of decision-making, and the horizontal axis the complexity and detail of the challenge:

(Image Credits Ruth Malan and Dana Bredemeyer)

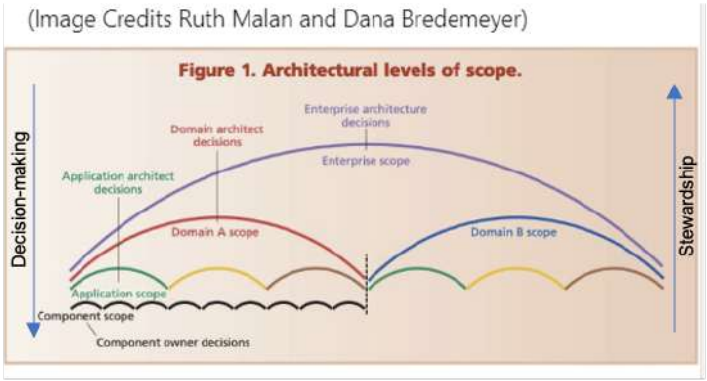


Let's start with the axis of complexity. The scope of a team often concerns a limited number of components, microservice or other functionalities. The main objective for the team here is to ensure functionality of the component is above or up to par of customer's expectations.

As we move up to domain-oriented units, business lines and enterprises, we see an increase in the complexity. On business-line and enterprise level, for example, the concerns are different from a team. One has to balance the best interest of multiple teams, commercial interests and organizational concerns. Simply put, the world is a bit bigger and increasingly complex.

Decision-making is the other axis to consider, especially the impact of a decision. The decisions made on organizational level typically offer boundaries and guidelines towards the organization. These are meant to bound the options available to the smaller units of the organization and aid in better decision-making for those units in the best interest of that unit and the organization. We can frame as the stewardship increases with the complexity, and the decision-making increases with the locality:

(Image Credits Ruth Malan and Dana Bredemeyer)



What About The Architecture Role?

As a starter, we see architecture as a function. A function that can be delegated to a day-to-day role, or a function that is attributed to an existing role. This could be a principal engineer or assigned to a group of people, e.g. the product engineering team.
A senior engineer or team lead can be appointed as the tie-breaker.

For an organization, it is important to be conscious about the attitude of the architects it hires and puts in charge. The character and way of working of the architect function has a huge impact on the engineering culture. Putting a benevolent dictator in charge has a different impact than an architect that coaches and supports the team in their architecture decisions.

However, context matters here. An architect for an in-shop product engineering department requires different capabilities and attitudes compared to an architect that has to work with vendors and ensure successful integration. The latter architect needs to be stronger in vendor management and the corresponding negotiation.

Do I need an architect?

Again, context matters and generic answers can't apply here. However, we have some considerations to think about.

- Do you have the capabilities and time to guide the technical decisions? Especially as a manager of a small technical department, can you support the time in guiding the decision-making process? From the perspective of time and capabilities.
- Can the complexity of an organizational unit and the impact of decisions managed via a shared responsibility by a group of people? For example, in the case of a group of mature principal engineers and a clear decision framework, it could work better than a dedicated domain architect.

Whilst we are talking here about the architecture function, we also propose that these principles and guidance apply to a broader set of disciplines. Design and user experience is a prime example of this. A design system in a way is a decision-framework that enables local decision-making and ensures compliance to the corporate brand and experience.

Xebia – Architects of IT transformation

Xebia explores and creates new frontiers.
Always one step ahead of what businesses need,
we turn the latest technology trends into advantages
for our customers. As a mainstream frontrunner,
we create new solutions and build the future
with our clients.

Let’s work together to enable your IT
to deliver more business value.

Learn more

Xebia