Table of Contents

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.

Want To Know More?

- Is Design Dead? - Martin Fowler's excellent discussion on more upfront versus emergent design.

FAQ, Architect, Intentional, Emergent