Table of Contents

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.

FAQ, Architect, Initiative