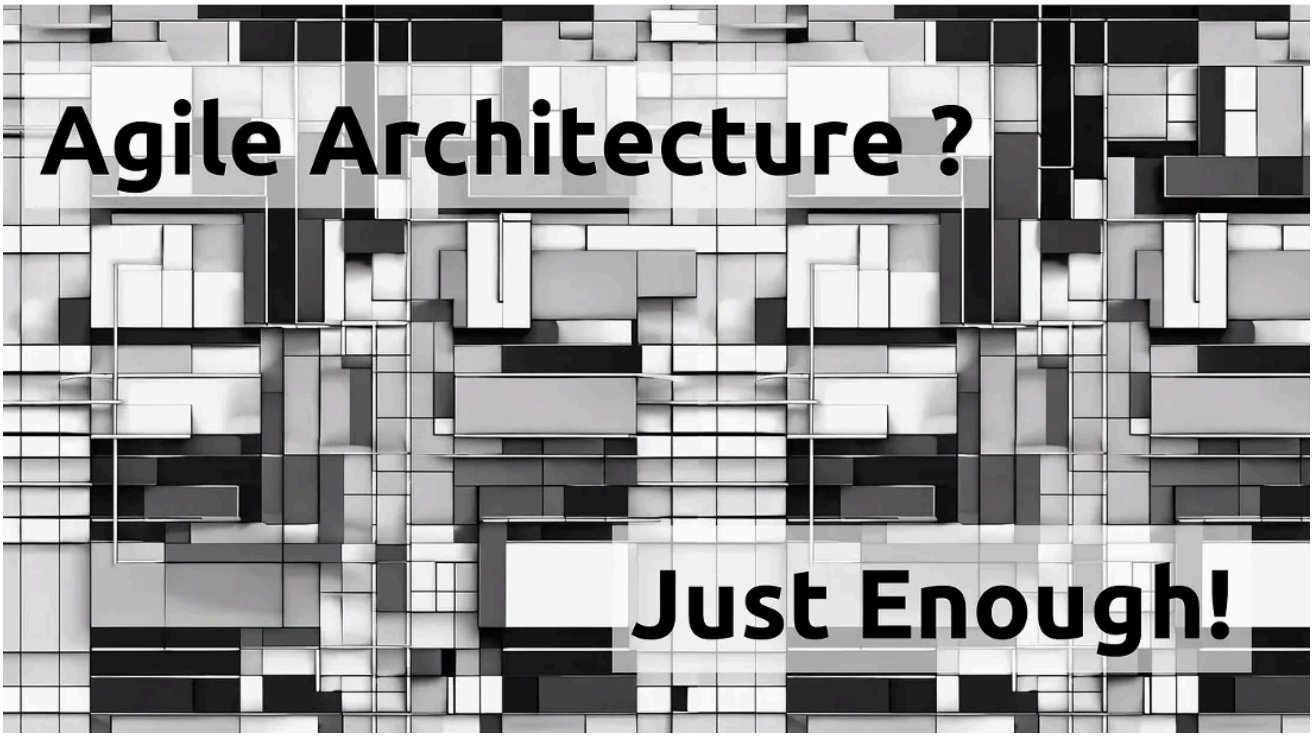
Agile Architecture — Just Enough!



When I started my career waterfall, V-Cycle were common software development lifecycles. It was clear that after the requirement phase the design phase starts. Typically, the architect in an engagement was responsible for this. Often it ended up in a Big Design Up Front (BDUF) what nowadays is seen as an anti-pattern. The result was a word document or in a model driven approach a formal model (e.g. UML). Architecture Trade-off Analysis Method was used to safeguard the compliance of the design to the given requirements. The next phase: implementation was based on the stable design and the requirements, at least in theory. Change request came all the time and sometimes had an impact on design and architecture, but a well-

defined Architecture could last for long time (sometimes even centuries).

With the rise of Agile this changed … and in the beginning it wasn't clear at all how architecture and design shall be handled. In SCRUM even the role of an architect wasn't defined. So being an Architect in an agile engagement was quite challenging. A lot of agile methodologies where defined and evolved starting from the Agile Manifesto. SCRUM, DAD, LeSS, NEXUS, SAFe to name some. The way they propose to handle design and architecture differs and the level of detail.

Still an Architects needs to define how to safeguard that the proposed solution will fulfil the functional and non-functional requirements. Emergent Architecture was the new kid on the block and while easy to write quite tricky to find the right balance.

Another concept is "Just In Time — Just Enough Architecture" (JIT-JEA). It is a concept to guide architects working in an agile context. The JIT-JEA approach emphasizes the delivery of just enough architecture, documentation, and governance to support agile development. It focuses on providing architectural guidance and information in iteration-size chunks, aligning intentional, emerging, and evolving architectures, and integrating architecture practices within the agile framework. The concept aims to balance the need for architecture with the agile principles of flexibility, collaboration, and responsiveness to change (see Link).

Let's have a look at different levels of JIT-JEA …

“Just Enough Architecture”

The approach balances intentional architecture (high-level planning) with emerging architecture (team-level design decisions), emphasising simplicity, continuous feedback and collaboration with development teams. It supports agile principles by enabling rapid iterations and adjustments to meet evolving business needs.

In this area I really like the approach of a Minimum Viable Architecture (MVA) based on the following principles:

- **Starting from the MVP:** The architecture should align with the immediate needs of the MVP, ensuring it can be developed and delivered efficiently.
- **Simplicity and Sufficiency:** MVA aims to balance simplicity with sufficiency, providing only the essential architectural elements required for the MVP.
- **Future Considerations:** While focusing on the present, MVA should also consider potential future expansions and integrations within a larger system landscape.

It may look easy, but it’s a real challenge to find the right balance! The balance will be influenced by several factors: seniority of architect, team mix and seniority, client demand, complexity of application.

“Just Enough Documentation”

It’s about producing the necessary amount of documentation to support the development process effectively, without overburdening it with excessive detail. The concept focuses on efficiency, relevance

and usefulness, ensuring that the documentation serves its purpose and meets the needs of its audience.

- **Audience and Purpose**: Tailor documentation to the specific needs of the target audience, ensuring it supports design activities, technical decisions, guidelines, service contracts, stakeholder communication, or onboarding.
- **Efficiency**: Documentation should be concise and useful. Avoid creating documents that do not add value or are not needed by the reader.
- **Clarity and Visuals**: Use diagrams, pictures, and visual aids over text-only documents to enhance understanding and communication.
- **Single Source of Truth**: Avoid duplication and maintain a single, easily accessible point of truth for documentation.
- **Documentation as Code**: Adopt the “Docs as code” approach, using the same tools and versioning methods as code to ensure consistency and ease of maintenance.

The mantra is: It is better to produce less, but useful, working documentation that is read by the team, than old-fashioned, voluminous documentation that is read by no-one!

“Just Enough Governance”

Integrating architectural governance seamlessly into existing agile rituals and minimizing the bureaucratic overhead. The goal is to maintain control and ensure alignment without stifling agility and innovation.

- **Integration with Agile Rituals:** Governance activities should be part of regular agile ceremonies such as sprint planning, demos, and retrospectives to ensure continuous collaboration and alignment.
- **Distributed Decision-Making:** Empower architects and agile teams to make decisions within clear boundaries, allowing autonomy while maintaining alignment with the overall architectural vision. Please note that there may be additional layers of architectural guidance that need to be considered.
- **Alignment of Architectures:** Ensure that team-level (emergent) architectures align with enterprise-level (intentional) architectures, maintaining a cohesive and scalable architectural landscape.

This approach ensures governance is effective and efficient, supporting agile teams without imposing unnecessary constraints.

"Just In Time"

In agile architecture one must focus on delivering and documenting architecture iteratively, rather than completing a full design upfront (Big Design Up Front or BDUF). This approach allows for more flexibility and responsiveness to change. Be aware that on the other side of the road you'll find the anti-pattern of spaghetti architecture. Again, the right balance is needed, and an architectural vision can be used to guide all team members.

- **Team Level:** Architects provide guidance needed for current and next sprints, connecting regularly with teams through both formal and informal channels to anticipate their needs.

- **Enterprise Level:** Architects align with product management plans to determine long-term enabler epics and necessary guidance for delivering business needs over six to twelve months.
- **Program/Solution Level:** Architects ensure timely information flow to team-level architects for planning the next three to six months, integrating feedback from both team and enterprise levels.

Effective “Just In Time” architecture relies on good communication and being connected across different organizational levels, ensuring that architects and also the agile teams provide the right information at the right time.

This was only a brief overview of JIT-JEA and you’ll find much more information in the series on the internet:

- **Part 1:** Scope & Definition
- **Part 2:** Real-life examples
- **Part 3:** Practicing JIT (10 building blocks)

You may ask yourself the question: What’s next?

It’s actually an easy question to answer: With the rise of GenAI, we are facing the next (r)evolution.

- How will the role of an Architect evolve?
- How to use GenAI in design and architecture?

This my blog on this topic: https://medium.com/@thilo-hermann/software-architect-assistant-augmented-architect-with-gen-ai-1b17a354d0ad

👉 **Original Post: Medium.com**