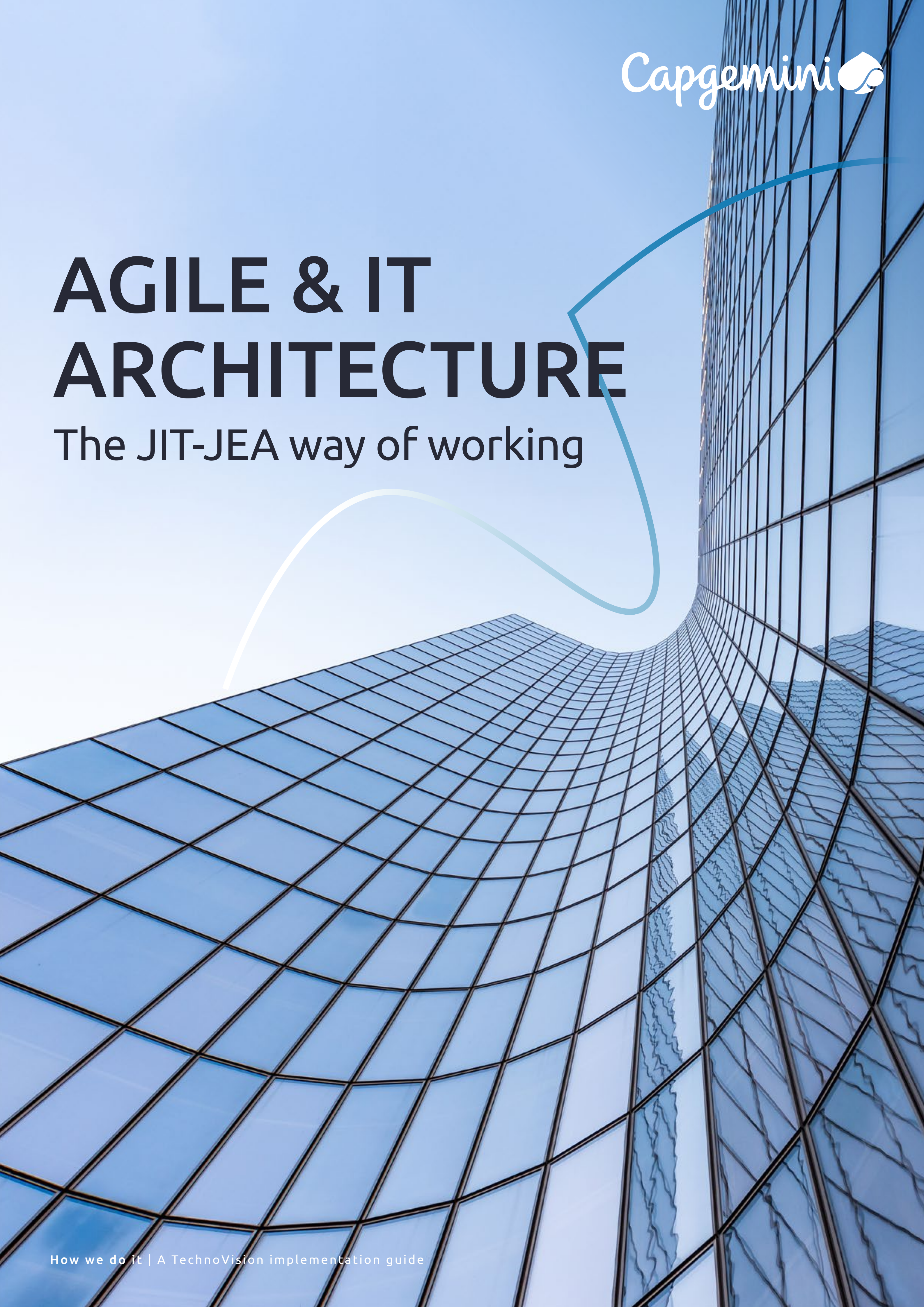


AGILE & IT ARCHITECTURE

The JIT-JEA way of working

How we do it | A TechnoVision implementation guide

WELCOME TO THE JIT-JEA WAY OF WORKING

Agility is a central requirement for many organizations. For organizations looking for a digital future, the question is no longer whether an agile innovation should be used, but which innovation, when, and how. Questions we should ask ourselves include: How to incorporate agility into the 2022 digital development plan and operating landscape? What steps would accelerate digital transformation? How best to architect in an agile way?

At its core, the term "agile" refers to an iterative, incremental method of managing design and building activities with an aim of developing new products in a highly flexible and interactive manner. As noted in the 2019 Capgemini "Agile at Scale" report, companies are starting to adopt and scale agile ways of working, changing the way they work.

And this change is also impacting the way we, as (IT) architects work. We are living in a complex and fast-moving world, ever gathering pace. What was acceptable yesterday, will not be accepted tomorrow. Our world is shifting, client expectations are changing and so is the market. Now more than ever, our clients expect their IT function to seamlessly support the business as we see the lines of business and technology blurring beyond recognition. As a result, we need to change and adopt new ways of working. We need to respond to change, at an ever increasing frequency.

There is a common misconception in the IT industry that architecture must be created "top-down;" where architecture-related artifacts are developed over two or three months - in one go - proving that "architecture" and "agile" are not compatible. This is not true. Working as a team, following a more agile architecture approach and designing a solution can be done, in one day. Of course, the level of detail will not be as deep as with a solution that takes months to produce, but it may be sufficient to take any necessary decisions to move forward.

Working in an agile way can drive change creating business opportunities through technological innovation. Architects can shape and translate business and IT strategy into realizable and sustainable technology solutions, while moving end-to-end solution delivery ownership from idea to benefits delivery. How to do this successfully in an agile fashion is the subject for debate in this paper.

Working with a dedicated team of IT Architects from across the Capgemini Group, we hope to provide a detailed and comprehensive overview on not only what we in Capgemini mean by "agile IT architecture," but also to outline how we architect in an agile way. Our thanks goes out to everyone who has been involved in creating this paper, our experts leading the agile charge in architecture.

[illegible] [illegible] [illegible]

GUNNAR MENZEL,
Master Certified Architect
Manchester, UK

KAI SCHROEDER,
Global Architects Community Lead
München, DE

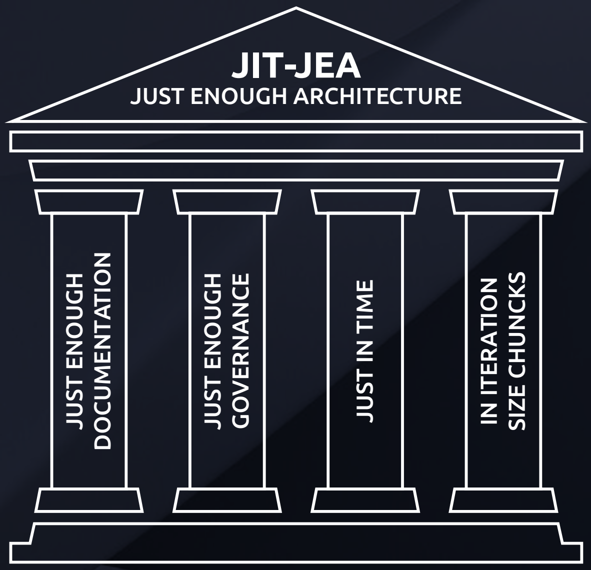
STEFANO ROSSINI,
Italy Industrialization Lead
Milan, IT

CONTENTS

INTRODUCTION

This section provides the context and key definitions to frame the next chapters of this document. We do not intend to redefine concepts, but merely ensure that the scope of this agile IT architecture point of view is clear. Throughout this paper we will use the terms **"architecture"** and **"architect."** To avoid confusion with other professions, whenever we use the term, we are referring to Business and IT (Information Technology) architecture and Business and IT architects.

Today, we as architects are often faced with the challenge of developing just enough. Agile development demands faster turnaround, expecting quick iterations and delivering just about the right amount of documentation; not too much and not too little. But what is actually good enough? When have we as architects developed enough material? What do we mean by **"just enough?"**



Developing "just enough architecture" is a common challenge. It is all too easy for the statement to become an end-goal in itself, rather than something that is there to help the business and development process. As a result, it can become difficult to determine when exactly to stop designing the architecture.

To assist architects who are working in an agile context, Capgemini encourages architects to **connect** with Agile and DevOps communities all around the world to share their work and help each other. This POV paper will explain the proposed Capgemini method of designing agile architectures known as JIT-JEA: Just in Time - Just Enough Architecture.

But, before we introduce JIT-JEA, what is an Agile Architecture?

1.1 WHAT IS IT ARCHITECTURE?

Architecture can be defined as:

- The fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution.[1]
- The structure of components, their inter-relationships, and the principles and guidelines governing their design and evolution over time.[2]

IT Architecture is not only about technology, infrastructure, or software; it is mainly about managing complexity to reduce risk and costs. In other words, if there is no complexity an IT architecture is not needed.

IT Architecture is an art form. It is the art of designing and delivering the right solution, providing:

- structure, where otherwise there would be chaos,
- alignment, where otherwise there would be none,
- certainty, where otherwise would be unpredictability.

1.2 What is Agile?

Agile, according to the Agile Alliance[3], is the ability to create and respond to change both adequately and in due time. It is a way of dealing with, and ultimately succeeding in, an uncertain and turbulent environment. Agile is a mindset that drives certain behaviors, centred around the four values and twelve principles of the Agile Manifesto[4].

In IT, "Agile" refers to an **iterative** and **incremental** method of managing design and building activities with an aim of carrying out and developing new products in a highly flexible and interactive manner.

"Agility" is our ability to sense and respond to change both adequately and in due time, while "Agile" is the myriad of tools and techniques to help us achieve agility.

It is important to understand how architecture fits within agile methodologies (see chapter two), as well as how agility can be approached in architecture frameworks (chapter three).

1.3 What is Agile Architecture and why JIT-JEA?

Agile architecture is the art of designing and delivering the "right" solution - meeting the requirements, expectations and demands of the client - while being able to respond to change in any uncertain environment. Responding to change means that we no longer create a "Big Design Up Front" (BDUF). Instead, the architecture designed in an agile context:

- provides the vision **(intentional architecture)** where the teams fit in with their (development) work.
- gives clear boundaries for the agile teams to make their own design decisions **(emerging architecture).**
- evolves with the cadence of iterative and incremental development along the agile journey **(evolving architecture),** and ensures a proper alignment between the intentional architecture (top-down) and the architecture emerging form the agile teams (bottom-up).
- must be fit for purpose; do as much architecture work as needed to build it **just in time!**
- breaks up the solution in pieces of work **(iteration size chunks)** that can be taken further by different teams.

Ideally, agile architecture should also enable designing for testability, deployability, and releaseability.

The notion of JIT-JEA (Just In Time - Just Enough Architecture) aims to assist all the thousands of architects working on agile architectures across the Capgemini Group. Working with the agile team(s), the architect can deliver:

- just good enough architecture,
- just good enough documentation,
- just good enough governance,
- just in time, and
- in iteration-size chunks.

To further define what is "good enough," outline and articulate the level of detail and amount of architecture material required, ten key principles outline the JIT-JEA approach:

1. **Understand the As-Is:** Ensuring there is a holistic view across business, data, applications, and infrastructure of what is currently installed and what will most likely change (not needed for complete green field).
2. **Understand the context:** Discerning the context across both business and IT, including any external factors (regulation, etc.) that may affect the results.
3. **Define the principles:** Formalizing traceable business objectives and principles, driven by the business mission and vision.
4. **Know the requirements:** Understanding and/or delivering the functional and - in particular - non-functional requirements.
5. **Record decisions:** Documenting the rationale for all architectural and design decisions, ideally reflecting the principles and business needs.
6. **Ensure traceability:** Providing clear traceability back to the business objectives within the architecture.
7. **Develop the solution:** Documenting the solution(s) including investigating alternatives to ensure that decisions are not made in isolation.
8. **Assumptions and constraints:** Capturing, validating, and managing any assumptions and constraints that affect the architecture.
9. **Risks and issues:** Proactively documenting and managing risks and issues - both processes as well as results.
10. **Plan:** Creating a clear and sensible plan/roadmap to achieve the desired business outcome(s).

The reason why we have (or need) an architecture is to effectively manage risks and costs. Using architecture frameworks like the Open Group's Architecture Framework[5], and Capgemini's Integrated Architecture Framework (IAF)[14], as well as receiving help and support from various communities, ensures that our architects can deliver JIT-JEA.

ARCHITECTURE IN AGILE

At first glance, architecture design methods and agile software development might seem incompatible, but this is not quite the case. Both can work seemlessly together, ensuring that custom build applications are delivering value for money, not just today but also for tomorrow. In the following section we will quickly explore how agile methods and frameworks either directly address architecture, or at least leave room for it.

The core question remains: Is architecture addressed by agile methodologies?

We will start by looking at the twelve principles within the agile manifesto, before we cover the main agile frameworks Crystal Clear[6], Extreme Programming[7], Disciplined Agile Delivery (DaD)[8] and Scrum[9], followed by the scaled agile frameworks Less[10], Nexus[11] and Scaled Agile Framework (SAFe)[12].

2 ARCHITECTURE IN AGILE METHODOLOGIES

Agile Manifesto

The eleventh principle of the agile manifesto refers to architecture:

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

This is an important principle exploring the concept of emerging architecture within agile teams.

Crystal Clear

The architecture core concept of Crystal Clear is the "Walking Skeleton:"

"A Walking Skeleton is a tiny implementation of the system that performs a small end-to-end function."

The "walking skeleton" is based on an initial solution architecture, where its completion is incremental over time.

eXtreme Programming

In eXtreme Programming the design is addressed through two concepts of system metaphor and spikes.

System metaphor shapes the system to explain the system design to people, without need for lengthy and detailed documents.

A spike solution is a very simple program to explore potential resolutions in order to reduce the technical risk.

Scrum

"Scrum does not explicitly identify an architect role."

Scrum has 3 defined roles: Product Owner, Scrum Master, and Team.

Architecture emerges as a result of the collaboration of all team members, as noted within the eleventh principle of the agile manifesto.

However, Scrum does not forbid the architect role.

An architect can be:

1. part of the team, or
2. can be outside the team, or
3. a joined effort within the team.

DAD

Compared to Scrum, **DAD** puts more emphasis on architecture, in order to reduce technical risk through a role:

- Architecture owner: is the person who owns the architecture decisions for the team

and through 2 practices:

- Architecture envisioning: initial high-level design
- Proven architecture: working code to justify the architecture

LeSS

In essence, the Large Scale Scrum (LeSS) is a scrum applied across many teams.

It is believed that the agile architecture comes from the behavior of agile architecting.

It is primarily about mindset and actions, not the use of a particular design pattern or tool. And part of that mindset is thinking “growing,” or “gardening,” over “architecting.”

NEXUS

As with Scrum, in Nexus architecture and the role of architects are not defined in the Nexus guide.

The assumption is that Nexus places a heavy emphasis on emerging architecture from the team.

However, the Nexus integration team, which is responsible for the overall integration into a product increment should provide at least some architectural guidance to align the teams.

SAFe

In the Scaled Agile Framework (SAFe), founded in 2011, architecture is strongly presented both as roles (Enterprise Architect, Solution Architect and Systems Architect) and also with artifacts (e.g. enablers, the architectural runaway, and even new events like Architect Sync).

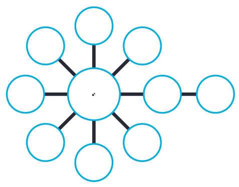
AGILE IN ARCHITECTURE

Agility per se, is not strictly mentioned in frameworks like TOGAF[13], IAF[14], or even later frameworks like IT4IT[15]. However, architecture frameworks and methods are clearly compatible with agile.

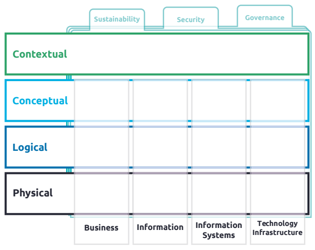
3 AGILE IN ARCHITECTURE FRAMEWORKS

Architecture frameworks are built to be adapted to be tailored to a specific context. You can pick and choose, they are not rigid methods. As a specific example, the first step in the ADM of TOGAF specifically describes the tailoring of the framework to the specific needs of the organization.

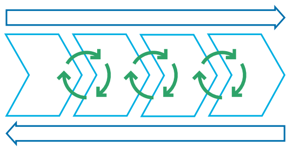
Moreover, they are iterative, with key concepts necessary to manage evolution over time such as the "feedback loop," or "requirement management."



TOGAF™
Adaptability, partitioning, evolution



IAF
Business goal alignment structure



IT4IT™
Flow based process and methodology agnostic

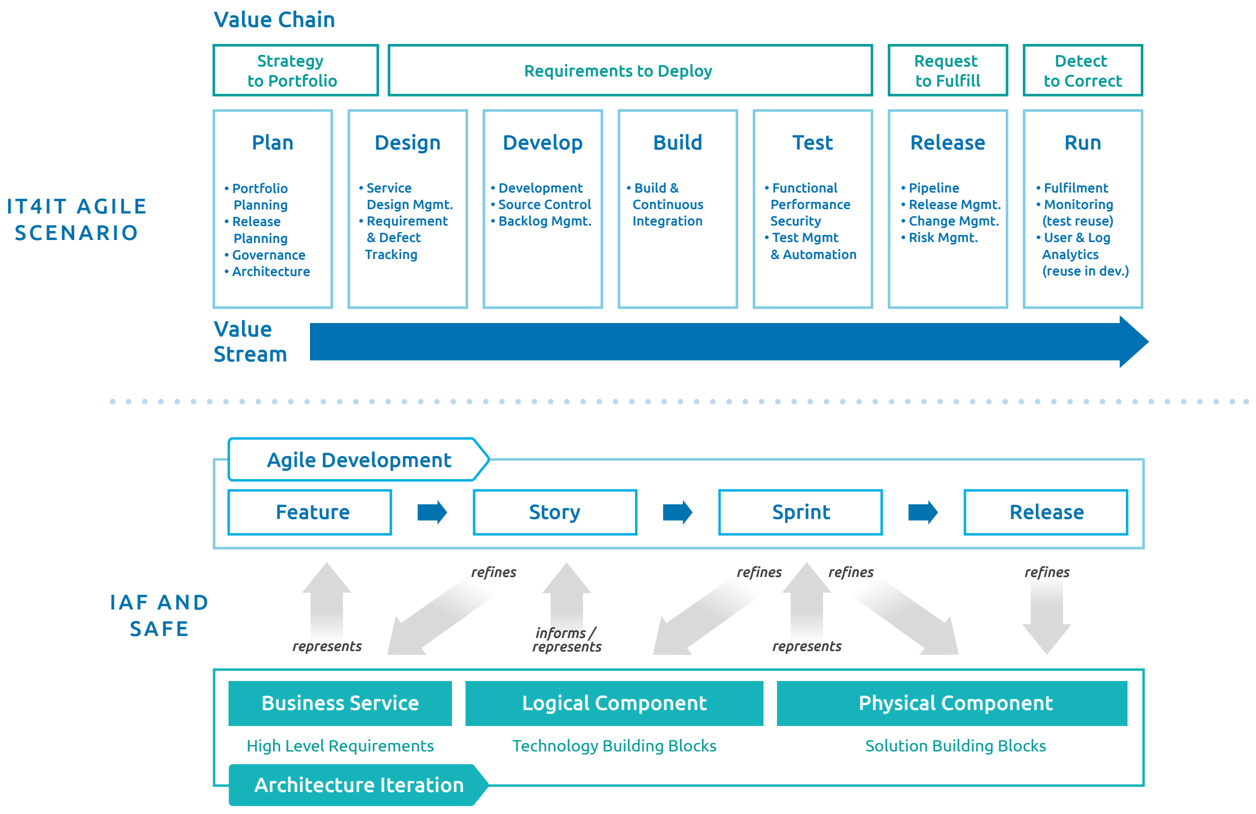
Architecture concepts are generic and compatible with agility

As IT architects, we have been implementing these frameworks since their creation, building best practices that are fundamentally different from the current context of today. They resulted in critical successes of enterprise and IT transformation, but these best practices are no longer aligned with digital expectations.

Recent updates of these frameworks provide clearer guidance related to agility. With the adoption of agility, architecture practices are evolving through two main approaches:

Approach #1: translation effort

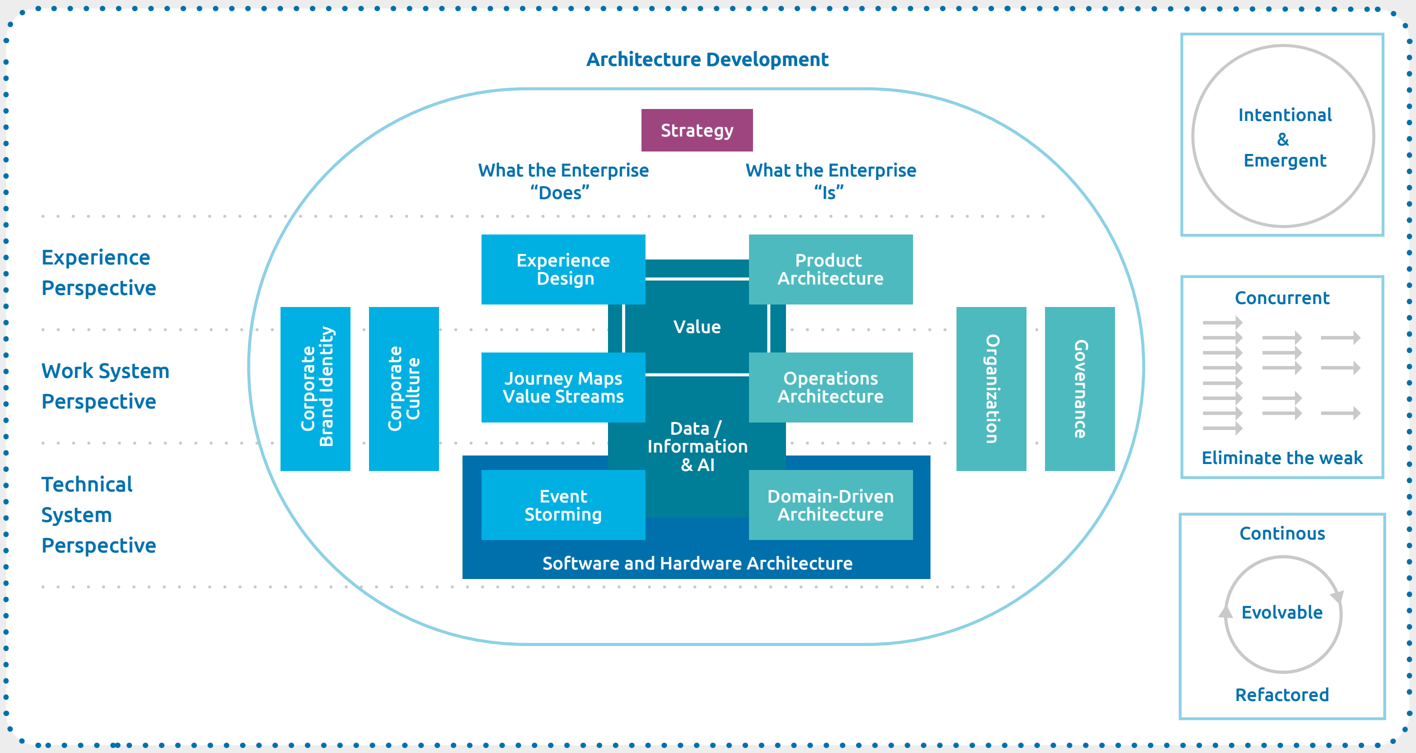
Mapping the effort between architecture frameworks and practices alongside agile methodologies.



Apporach #2: new framework

Another approach is to build a new framework like the recent Open Agile Architecture Framework (O-AA)[17] created by the Open Group, taking agile, lean, and design thinking concepts as foundations to create a new framework.

OPEN AGILE ARCHITECTURE FRAMEWORK



#AGILEBYDESIGN

To enable us to further assist our architects across Capgemini, we developed the JIT-JEA way of working.

4 THE JIT-JEA CONCEPT

An agile environment requires three main architectural stages:

- **Intentional architecture** at enterprise level outlining the direction of the organization
- **Emerging architecture** at team level outlining how teams are building towards the next stable situation
- **Evolving architecture** where enterprise and operational levels meet, and where emerging and intentional architectures become aligned

4.1 THE JIT-JEA CONCEPT

When considering the manifesto for agile software development, it is clear that architects can apply the twelve principles of the manifesto to architecture as well. Thus, the approach to developing an architecture, and creation of content within the architecture can also follow an agile approach.

In Capgemini, we refer to "Just In Time - Just Enough Architecture." Or simply put, "JIT-JEA."

Developing "Just Enough Architecture" can be characterized by the following:

OBJECTIVE OF THE ENGAGEMENT

Architecture practices should not repeat "more of the same," avoiding decisions that are centered around architectural guidance. Instead, architects should focus on working with new business activities or technologies that need to be integrated into a given environment, project, process or solution.

MATURITY OF THE TARGET AUDIENCE

At team level, "Just Enough Architecture" forms the blueprint of what needs to be done in the next sprint (or set of sprints). In agile, this blueprint should leave as much room possible for design decisions to be made by the teams. The more experienced and knowledgeable on the subject and on agile practices, the more room can be left to the team, so long as the architecture provides the **guidelines** to keep the team on track.

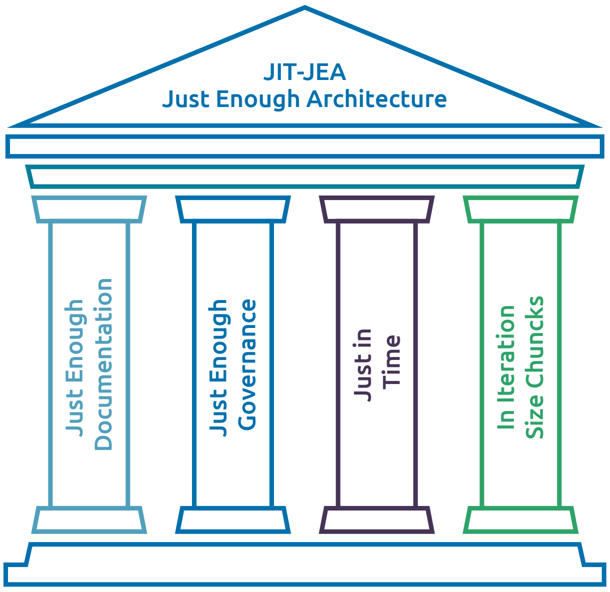
AMOUNT OF TECHNICAL RISK AND UNCERTAINTY

The more technical risk, the more up-front architectural design and/or research required. To reduce risk, architects could define **proof of concepts** to validate assumptions or explore technology choices.

COMPANY CULTURE

A highly agile team in a non-agile company culture will not fit well. In an **organization** working with fixed price contracts, with fixed scopes and delivery dates, the amount of upfront architecture is significantly higher to form a better view on how much work is required.

Besides the above aspects, the main principles of "Just Enough Architecture" include: simplicity, continuous architecting, and structural feedback from and towards the teams.



4.1.1 Emerging and Intentional Architecture

When it comes to "Just Enough Architecture," an organization should rely on the main principle of "think big, act small, fail, and learn fast:"

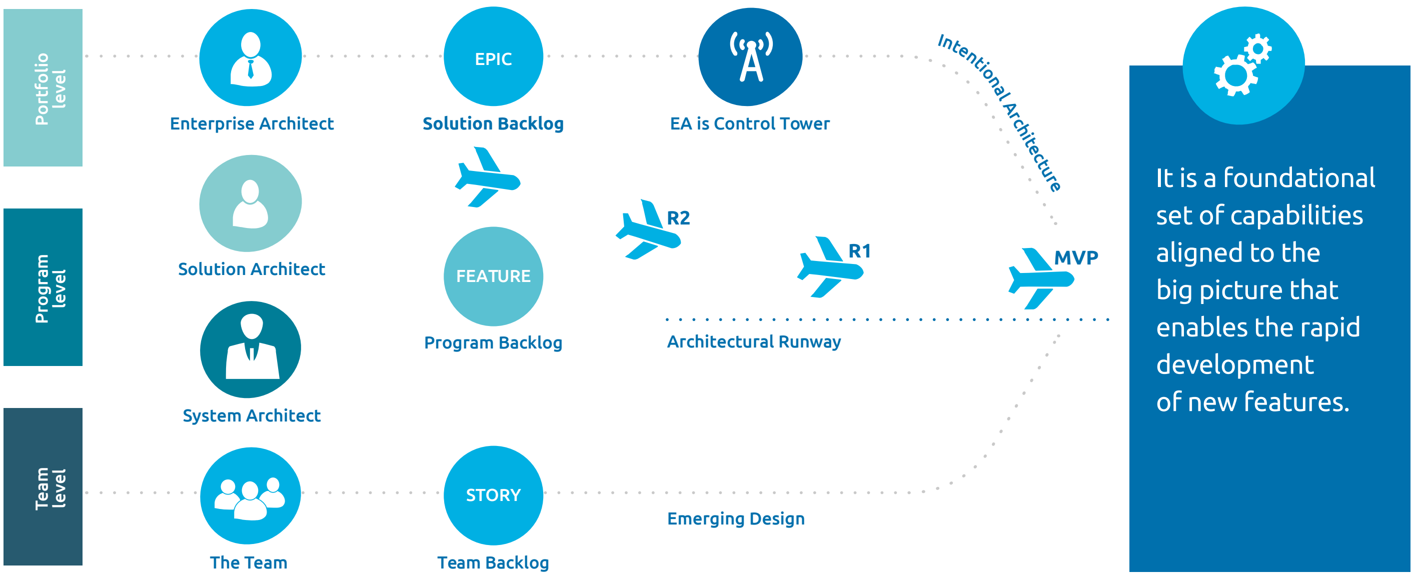
- **Think big:** the primary objective is to build a high standard, competitive solution, with a clear, high-level target architecture in mind
- **Act small:** the teams must deliver small operational pieces of software, clearly demonstrating the value of enabler work
- **Fail and learn fast:** accept that there will always be some form of failure - and the sooner the better - to refactor or rebuild the architecture.

The keywords to capturing an emerging architecture are: collaboration and continuous integration, both of which are the foundation of any application lifecycle management. Indeed, "Just Enough Architecture" is not possible when staying in the "ivory tower." Interaction with the development teams is fundamental and mandatory.

4.1.2 Architectural Runway

Organizations need to respond simultaneously to new business challenges with larger-scale architectural initiatives that require intentionality as well as planning. Emerging architecture alone cannot handle the complexity of large-scale system development.

Instead, we must balance both an intentional and an emerging architecture. The SAFe concept of architectural runway provides the technical foundation for smooth development and implementation of future business value.



The aim is to reduce technical debt and time-consuming re-work over time. For this purpose, the enablers - and therefore the epics on the architectural runway[18] - fall into one of four categories:

1. Exploration enablers supporting research, prototyping, and/or spiking to better understand customer needs and what solutions can be applied.
2. Architectural enablers covering guidelines, features, and stories where the architectural runway for the teams is created.
3. Infrastructure enablers are created to build, enhance and automate the development, as well as test and deployment environments.
4. Compliance enablers facilitating specific compliance activities such as documentation,privacy and security, and industry specific regulations.

4.1.3 Minimum Viable Architecture

Good practices show that Minimum Viable Architecture (MVA)[19] can be defined as the least possible set of principles to support the Minimum Viable Product to be released. An MVA should be defined in accordance with all aspects of the architecture.

However, an MVP might become part of a larger landscape and in which case, the MVA should also consider this future situation.

4.1.4 Data Driven Architecture

Most developments in architecture take place in an existing situation. It is therefore good practice to use data analysis on any existing information to support the architecture for new developments. Gathering data can be completed to differing levels of detail and in different ways. Examples include:

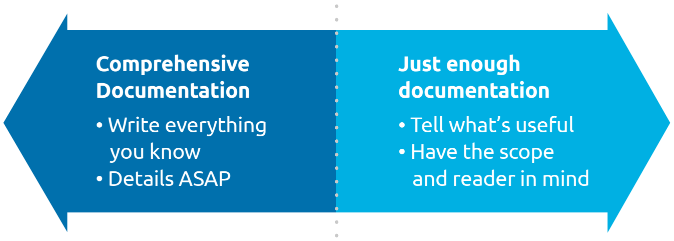
- **Network performance monitoring** to help identify bottle necks in infrastructure or specific applications
- **Application portfolio management** to provide clues on what systems may be loaded with technical debt or are in need of replacement
- **Day in the life of** might be used as a tool to analyze the work done by a person in a specific role indicating waste in processes

Some techniques that might help gather and analyse data and then decide and implement the decision are the OODA loop[20], PDCA cycle[21], and the Lean A3 method[22].

4.2 Just Enough Documentation

What "Just Enough Documentation" means mainly depends on the purpose of the documentation needed, the target audience, and the level of detail required at a given time.

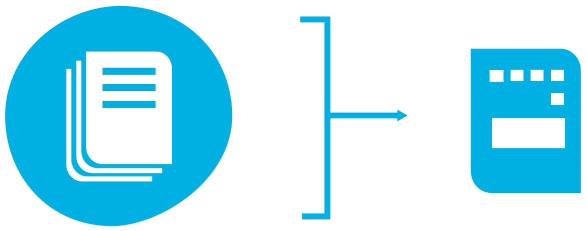
When creating architecture documentation, it should never be done for the sake of documentation alone, or to follow a specific process or framework, but solely with the reader in mind.


Comprehensive Documentation
• Write everything you know
• Details ASAP
Just enough documentation
• Tell what's useful
• Have the scope and reader in mind

Before deciding on what documentation to produce, it is important to have a clear view of the target audience and their needs in terms of timing (e.g. iteration or program) and content (e.g. drive design activities, support technical decision, guidelines, service contracts, communicate to stakeholders or onboard newcomers).

In addition to defining the stakeholders and their needs, the following main principles should be used:

- **Avoid any text-only documents:** diagrams, pictures or spikes, and walking skeletons are worth a thousand words
- **Avoid writing documents that are not useful:** if the reader does not need it, do not write it!
- **Write efficiently:** never spend more time in writing the documentation than the time you – or others – will gain from it
- **Do not document any publicly available information:** instead use references to information already exists
- **Avoid duplication of documentation across multiple documents or locations:** hold a single point of truth to be easily maintained
- **Use a centralized platform** with powerful search and update facilities



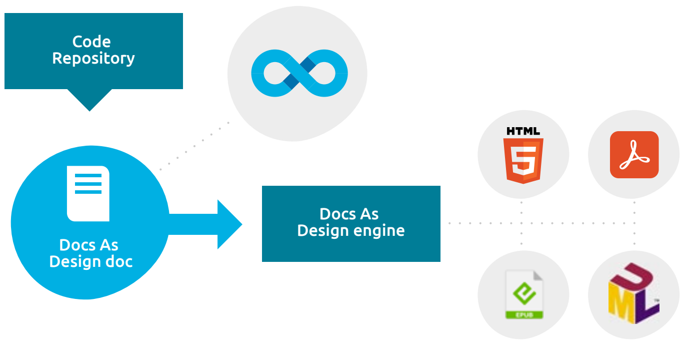
The Just Enough Documentation mantra is light but efficient: "better less, but useful working documentation that is read by the team, than old-school, extensive documentation read by none."

But how can we do that? Visual management is; a picture is worth a thousand words. Diagrams using ArchiMate[23], UML[24] and other techniques, enable clear alignment and shared understanding across all stakeholders (including architects and teams).

4.2.1 Documentation as Code

Documentation as code (Docs as code)[25] can be a useful way to make just enough documentation. It refers to a philosophy that you should be writing documentation with the same tools as code and apply versioning in the same manner as code is versioned.

However, architecture documentation is not only meant for developers. As a result, documentation for other stakeholders - the **Architecture Decision Record (ADR)** as just one example - should be made available in other formats as well.



4.3 Just enough governance

Ideally architecture governance within an agile context should be:

- **integrated in existing agile rituals** to minimize the number of meetings and maximize collaboration and sharing. Rather than controlling agile teams as a kind of authority, the architect should be a part-time team member (participating to sprint planning sessions, demo sessions, retrospectives etc.) while acting as a concierge efficiently guiding the teams
- **distributed by design** to empower architects at the right level and within clear boundaries in which they have the autonomy to make decisions
- **data-driven,** to support decision making and share information across the organization in a transparent and visual way

Architecture governance aims to:

- ensure alignment between emerging (team-level) and intentional architecture (enterprise-level)
- provide mechanisms to own the technology roadmap and identify and feed (transversal) enablers[27] into the journey
- define the boundaries in which teams and architects have the autonomy to make architectural and design decisions
- support architects in managing the architectural landscape

4.3.1 Faster Architectural Decision-Making

When regarding agile architecture, slow decision-making is often one of the main challenges. To accelerate the decision-making process, empowerment of the right people at the right level is needed, as easily reversable decisions require less formality than irreversible decisions:

- Decisions of low strategic importance impacting a single agile team - should be made by the team itself
- Decisions of low strategic importance, but impacting multiple teams should be taken by an empowered community
- Decisions of high strategic importance should be taken by a team of architects at strategic to enable the decision-making process, it is important to define clear boundaries where each level has the autonomy to make the appropriate decisions.

To enable the decision making process, it is important to define clear boundaries where each level has the autonomy to make the appropriate decisions.

It is good practice to also provide clear examples on the governance required for different decisions, such as changing cloud provider, changing public APIs, experimenting with a new framework, or adopting a new programming language to name a few.

4.3.2 Validating Solutions Against Architectural Compliance

A typical challenge in IT architecture is the deviation between the initially defined architecture (intentional architecture / top-down) and the solution built by the delivery teams (emerging architecture / bottom-up). While teams often make such deviations for good reason, architects may not be aware, leading to differences between the "predefined architecture" and the emerging "architecture as built."

Good practices to ensure architectural compliance include:

- Rather than a simple reference architecture on paper, a "walking skeleton" or a "working reference architecture" offers a technical template which is fully in line with the required architectural principles. The reference architecture to adhere to is to be decided upon by the architects in close cooperation with the teams developing the products.
- Have an architectural compliance check integrated in the "Definition of Done" (DoD) [28] to ensure that principles and guidelines are covered in each user story.
- Embed "automated architectural compliance checks" in the CI/CD (continuous integration/ continuous deployment) pipeline to automatically highlight potential deviations from predefined architectural principles.

4.3.3 Structural Collaboration between Architects and Delivery Teams

Often, architects can be perceived to be "forcing decision from their ivory tower," without considering the realities on "the floor." In a successful and collaborative agile model:

- architects respect experienced professionals in the development teams and listen carefully to their suggestions
- architects are visible to the agile teams, ensuring their input and guidance is recognized, valued, and directly usable by the teams
- architects keep sight of the bigger picture to ensure alignment between the intentional architecture (top-down) and the architecture emerging form the agile team (bottom-up)
- architects identify and support prioritization of architectural topics in the product backlog, based on inputs and continuous feedback loops with the agile teams
- architects bring in their experience to coach, advise, and technically support agile teams, based on the real product rather than using documents as the way to communicate
- architects understand agile development practices and have frequent interaction points with the development teams by participating in agile ceremonies (sprint planning, sprint review, retrospective, sprint demos, etc.)

For more information on the changing role of the architect in agile and delivery teams, read: the new role of the architect[29].

4.3.4 Architecture Community of Practice

To maximize learning, sharing of information, and thoughts across architects, we recommend setting up a Community of Practice (CoP) for Architecture. Such a group provides a platform for identifying and discussing common challenges, allowing people to participate in architectural decisions, supporting teams in staying up to date with latest technologies, and sharing learnings from architectural spikes etc.

Other good practices include:

- GEMBA walks[30] offer opportunities for people to stand back from their day-to-day work, consider where value is created, and listen to employees providing perfect opportunities to receive and offer feedback on how teams are working
- The Architect Sync[31] event as defined in SAFe ensures architects stay aligned and share progress at the large solution level
- A technology radar[32] is a strong visual way to communicate and align across teams about existing or new techniques, tools, platforms, languages and frameworks

4.3.5 Architecture Decision Record (ADR)

The ADR covers all decisions related to the architecture and is therefore an important tool in the architectural governance, even more in an agile context than in waterfall. The main reason is that some architecture decisions might be left to the teams, and therefore need to be documented by the teams. As already stated under Just Enough Documentation, the ADR needs to be versioned as well to ensure decisions can be traced back and changes in decisions are documented.

4.4 Just in Time

With Agile Architecture we need to transform from a **Big Design Up Front (BDUF)** approach - where the architecture design is to be completed and perfected before the implementation starts - to a **just-in-time approach** implementing and documenting the target architecture in an iterative way, bit-by-bit, step-by-step.

The key question to answer is **"who needs what, by when?"**

To be able to answer this question, architects must get closer to the developments, providing more visibility on the details without losing the broader view, vision and strategy. Doing so allows the architects to learn from every sprint, and receive feedback from development team.

Some examples of good practices:

JUST IN TIME ARCHITECTING
Do not spend too much time on architecting non-functional requirements when implementing a prototype or an XP Spike, but be prepared for the question, "when can we put this prototype into production?"

JUST IN TIME DOCUMENTATION
Know what information each stakeholder needs, and when (to prepare for a next iteration).

JUST IN TIME GOVERNANCE
Know what architecture decisions must be taken and when; what information decision makers need and when will they need it? Apply the principle of the **latest responsible moment**, which advises to keep important and irreversible decisions open until the impact of not deciding exceeds the impact of deciding. This allows decisions to be made with the maximum possible information, but also avoids wasting time revisiting the same discussions. Until this last possible moment, you are learning and collecting information.

Just in Time has different meanings depending on the level of architecture being worked on:

TEAM LEVEL
The architect focuses on what the agile teams need for their current and next sprints. To do so, architects should connect regularly with the agile teams to anticipate what guidance and information they will need at any given point in time. A good practice is to use both formal (e.g. attending agile ceremonies) and informal channels to connect.

ENTERPRISE LEVEL
The architect understands what product management plans will be delivered by the teams in a certain timeline (for instance six to twelve months). From that perspective, the architect determines the enabler epics that prepare for delivering the business needs and what guidance will be necessary.

PROGRAM / SOLUTION LEVEL
The architect reconciles the needs at both team and enterprise level. As a rule of thumb, the solution architect ensures that the information flowing to the architects at team level is in time to plan their work for the coming three to six months, bearing in mind that the backlog is also filled with work coming from the architects at team level.

In short: JIT (Just in Time) is critically about good communication and being connected on different levels from team, to enterprise.

4.5 In iteration size chunks

The agile architecture is to be delivered "in iteration size chunks," meaning that the scope of what an architect delivers should:

- fit within the amount of time that the receiving team (or architect) has before the next delivery
- consider what preparations are required for the coming iteration of program plan
- limit information and documentation to the team, depending on iteration context and scope
- attend the agile events on a regular basis (daily meeting and visit sprint review sessions)

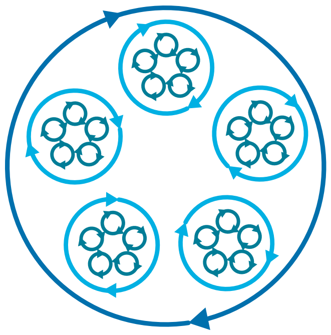
The shorter the product delivery cycles, the smaller the chunk of architectural information needs to be. As a result, more teams of architects can start working with a backlog and coordinate their work in sprints as well.

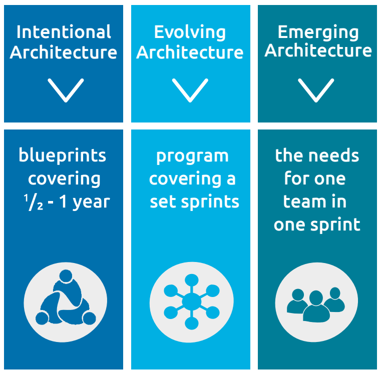
In those situations, one of the architects takes on the role of Technical Product Owner (TPO) for the architecture work to be done. This TPO regularly interacts with the POs responsible for the business functionality to understand size and priorities in the architecture work.

Other aspects determining iteration size chunks are the domain and the boundaries of the context. The clearer the domain and the context, the smaller the amount of information necessary, since architecture gives the guardrails within which the teams operate.

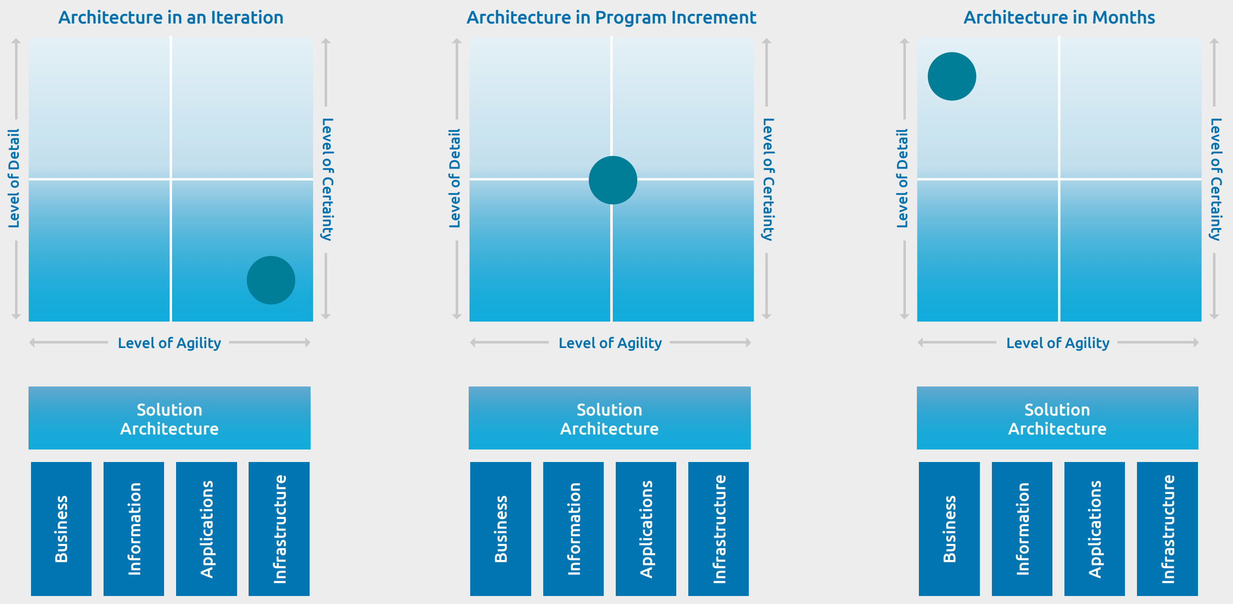
More uncertainty on boundaries leads to the need for more information.

As a rule of thumb, the mechanism of architecture in a day, in a week, or in a month might be helpful.




Intentional Architecture
Evolving Architecture
Emerging Architecture
blueprints covering ½ - 1 year
program covering a set sprints
the needs for one team in one sprint

The higher the amount of agility (team level), the faster the architecture documentation can be prepared. At enterprise level, the direction is usually more stable and thus more architecture documentation (relative to the team level) can be prepared for future use by the teams.



4.6 Concluding remarks on JIT-JEA

Agile architecture is not a unicorn - a physically impossible role without definition. Architecture can fit well within agile methodologies just as agility fits within architecture frameworks. Agile architecture is possible, but sharing and implementing good practice is fundamental, and Just In Time - Just Enough Architecture (JIT-JEA) is key.

However, this is not the end state for agile architecture. It is merely a starting point (MVP if you wish) to be further elaborated over time. Future versions of this POV might cover aspects such as "how IT architecture can help to create business agility," "how architects behave in an agile environment," or "the benefits of agile architecture as seen from a CIO perspective."

Every IT system has an architecture, whether you are conscious about it or not.

In a similar thought you could say that every IT person directly or indirectly contributes to IT architecture.

So, agile or not, be conscious about it.

There is a role for an Agile Architect whether it's a Platform Concierge, a Product Owner for Enablers, an Architecture Coach to Agile Teams, or Architect as a Service.

BIBLIOGRAPHY

REFERENCES

1. ISO 22010:2017. ISO.org. [Online] https://www.iso.org/standard/50508.html.
2. Definitions in TOGAF. TOGAF. [Online] https://pubs.opengroup.org/architecture/togaf9-doc/arch/chap03.html.
3. Agile Alliance. [Online] https://www.agilealliance.org/agile101/.
4. Agile Manifesto. [Online] https://agilemanifesto.org/.
5. Group, Open. [Online] https://pubs.opengroup.org/architecture/o-aa-standard/.
6. IAF-Integrated Architecture Framework (Wikipedia). [Online] https://en.wikipedia.org/wiki/Integrated_Architecture_Framework.
7. Crystal Clear. Crystal Software Development. [Online] https://newline.tech/crystal-clear-methodology/.
8. XP-eXtreme Programming. [Online] http://www.extremeprogramming.org/.
9. DaD. [Online] https://en.wikipedia.org/wiki/Disciplined_agile_delivery.
10. Scrum. [Online] https://www.scrum.org/.
11. LESS. [Online] https://less.works/.
12. Nexus. [Online] https://www.scrum.org/resources/scaling-scrum.
13. SAFe. [Online] https://www.scaledagileframework.com/.
14. TOGAF. [Online] https://www.opengroup.org/togaf.
15. IT4IT (The Open Group). [Online] https://www.opengroup.org/it4it.
16. The Open Group. Introduction to the ADM. TOGAF Standard version 9.2. [Online] https://pubs.opengroup.org/architecture/togaf9-doc/arch/chap04.html#tag_04_03.
17. Open Agile Architecture. The Open Group. [Online] https://pubs.opengroup.org/architecture/o-aa-standard/.
18. Enablers in SAFe. Scaled Agile Framework. [Online] https://www.scaledagileframework.com/enablers/.
19. MVA (Open Group). [Online] https://pubs.opengroup.org/architecture/o-aaf/snapshot/Agile_Architecture_Framework.html.
20. OODA model. Value Based Management. [Online] https://www.valuebasedmanagement.net/methods_boyd_ooda_loop.html.
21. Wiki. PDCA Cycle. [Online] https://en.wikipedia.org/wiki/PDCA.
22. Kanbanize. LEAN A3. [Online] https://kanbanize.com/lean-management/improvement/a3-problem-solving.
23. Archimate. The Open Group. [Online] https://www.opengroup.org/archimate-forum/archimate-overview.
24. UML. Wiki. [Online] https://en.wikipedia.org/wiki/Unified_Modeling_Language.
25. docs-as-code. [Online] https://www.writethedocs.org/guide/docs-as-code/.
26. Enterprise Architecture Docops. [Online] https://capgemini.github.io/architecture/enterprise-architecture-docops/.
27. SAFe. Enablers. [Online] https://www.scaledagileframework.com/enablers/.
28. Definition of Done. Scrum. [Online] https://www.scrum.org/resources/blog/done-understanding-definition-done.
29. Menzel, Gunnar. [Online] 2017. https://www.capgemini.com/wp-content/uploads/2017/07/the_new_role_of_the_architect_-_central_to_growing_your_business_in_todays_digital_world.pdf.
30. Gemba Walk. [Online] https://kanbanize.com/lean-management/improvement/gemba-walk.
31. SAFe Agile Architecture. [Online] https://www.scaledagileframework.com/agile-architecture/.
32. Technology Radar. GitHub. [Online] https://github.com/bdargan/techradar.

AUTHORS

STEFANO ROSSINI

Italy BU Industrialization Leader, Capgemini Chief Architect and Agile Coach.

Stefano is the Italy BU Industrialization leader.

He is a Capgemini Chief Architect expert in services architecture SOA and MSA and he is also an Agile evangelist and coach.

Stefano loves the topic of Agile Architecture since he really loves both of them: Agile and Architecture.

He leads the DevOps global community and the Italian communities about Agile and Architect.

WIGER LEVERING

Dutch Architects Community Lead.

Wiger is Account Chief Architect for the department of Education, Culture & Science in the Netherlands, including the education sector.

Next to his activities in developing architectures, he works with architects on finding their role in an agile context.

Over the past thirty years, he has successfully lead teams in developing architectures that guided a transformation towards developing agile application landscapes and agile ways of working at scale.

JEAN-PHILIPPE DEFRANCE

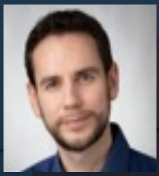
Group Portfolio Enterprise Architect, Capgemini Senior Architect, Agile Community Leader.

Over the last 15 years, Jean-Philippe combined his expertise in architecture, design thinking and agile to lead successfully large IT transformations. He now defines how Capgemini manage its portfolio of service priorities worldwide and drive the development of new offerings.

GERT HELSEN

Gert is a Chief Architect in Capgemini working in the Financial Services sector.

Being passionate about people and IT technology, Gert is active as coach, mentor and certified trainer for many architects across the Capgemini group. In addition, he leads an Innovation Service and acts as Chief Account Architect for a strategic Financial Services client based in Europe.



PASCAL ESPINOUSE

Digital & Innovation Architect, Capgemini Chief Architect.

Pascal is a Chief Architect in Capgemini, specialized in Digital Transformation and Innovation. Led by passion, Pascal is a trainer and mentor within the Capgemini Global Architects Community. Leader of his practice's Architects Community, he is also part of the Core Team leading the 1000 plus Architects of Capgemini France.

GUNNAR MENZEL

Capgemini Master Architect and CTIO for Europe North and Central.

Gunnar is the Chief Technology & Innovation Officer for Capgemini Europe North and Central. As a Certified Master Architect and Certified IAF Master Architect he is a member of Capgemini's Global Architecture Board, and as a senior leader within Capgemini's Global Architecture Community, he plays a key role in setting the direction of the architecture profession across the Group. Over the past thirty years, Gunnar has successfully worked with many organizations to either transform to an agile model and/or apply agile mindsets and tools.

A special thanks to the other Capgemini Colleagues:

Monika Demichowicz (Poland), Greg Holliday (US), Jorge Rodriguez Oporto (Spain), Marien Krouwel (Netherlands), Andreas Lutz (Germany), Bjorn Gronquist (France) and Yann Ducrocq (France).

About Capgemini

Capgemini is a global leader in partnering with companies to transform and manage their business by harnessing the power of technology. The Group is guided everyday by its purpose of unleashing human energy through technology for an inclusive and sustainable future. It is a responsible and diverse organization of 290,000 team members in nearly 50 countries. With its strong 50 year heritage and deep industry expertise, Capgemini is trusted by its clients to address the entire breadth of their business needs, from strategy and design to operations, fuelled by the fast evolving and innovative world of cloud, data, AI, connectivity, software, digital engineering and platforms. The Group reported in 2020 global revenues of €16 billion.

Get the Future You Want | www.capgemini.com