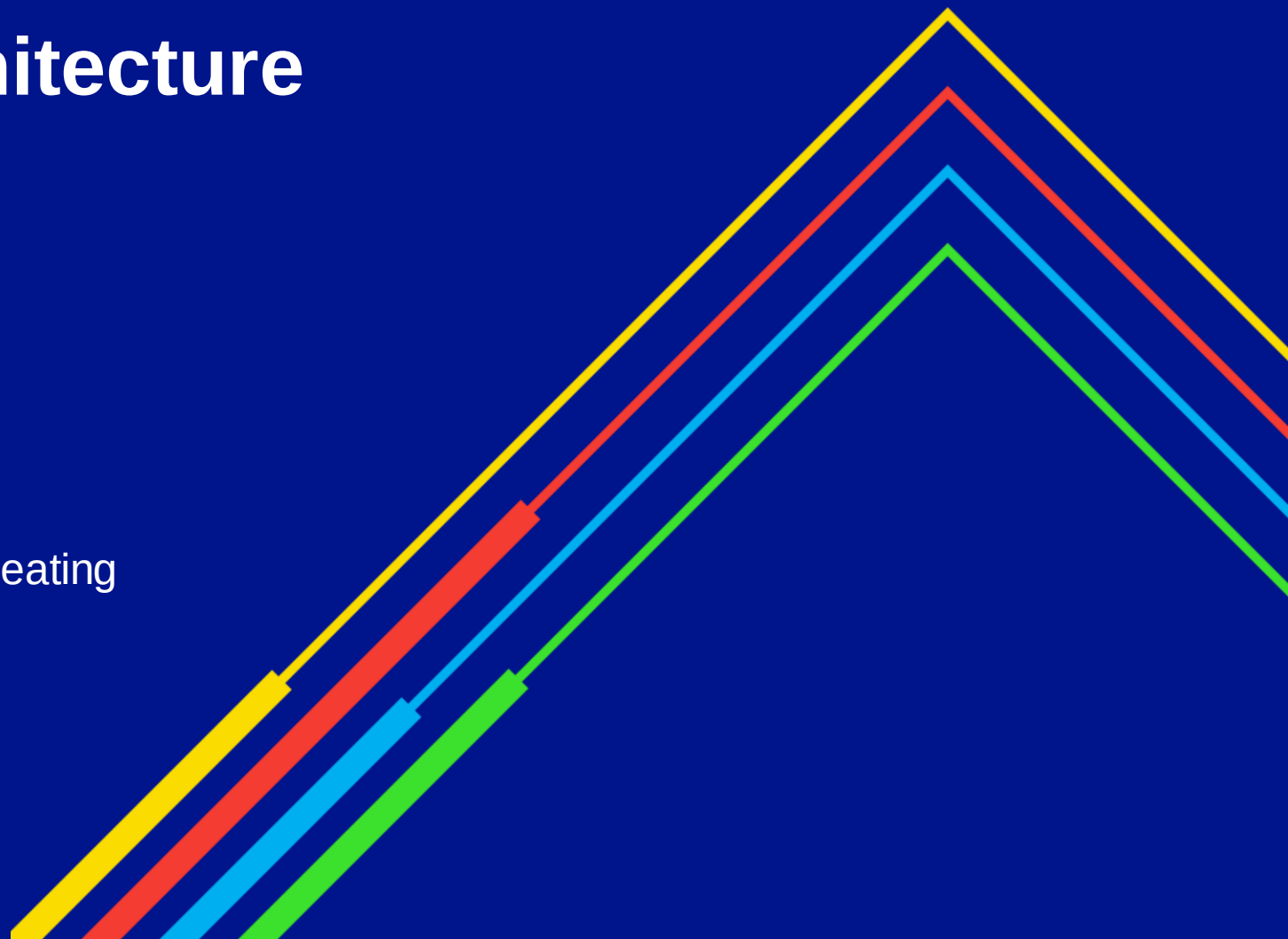
Find better ways of working

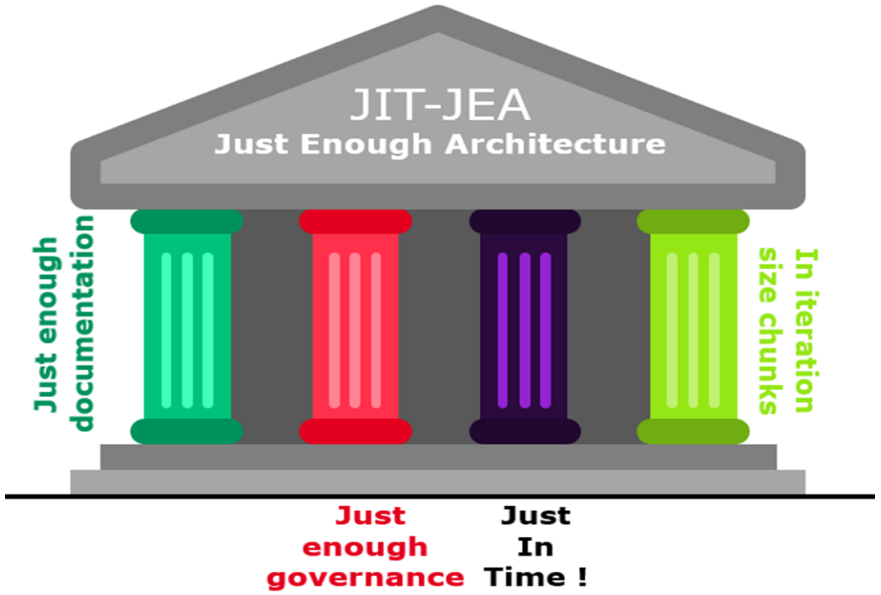
JIT – Just In Time

JEA - Just Enough Architecture



Will I benefit from Agile ways of working if I am creating BIG UPFRONT plans and architectures ????

nationalgrid



The idea is that the architect will work with the agile team(s) to deliver:

- ***just good enough architecture***
- ***just good enough documentation***
- ***just good enough governance***
- ***just in time***

What is Agile Architecture and Why JIT-JEA?

Agile architecture is the art of designing and delivering the "right" solution in such a way that it is able to respond to change in an uncertain environment' in a turbulent context. Responding to change means that we do no longer create a Big Design Up Front (BDUF). Instead, the agile architecture design in an agile context:

- Provides the vision (**intentional architecture**) where the teams fit in with their (development) work
- Provides the guard rails between which the agile teams 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 and built it **just in time**!
- Breaks up the solution in pieces of work (**iteration size chunks**) that can be taken further by different teams

Agile architecture should break up the solution into pieces of work that different teams can take further. Ideally, agile architecture also enables designing for testability, deployability, and releaseability

What is “Good enough”

To define “what is good enough” and to outline and articulate the level of detail and therefore the amount of related architectural material necessary, 10 principles that relate to JIT-JEA:

- ✓ **Understand the As-Is**; if there is an As-Is make sure you have a view across business, data, application, and infrastructure of what is currently installed and what will most likely change (not needed for complete green field)
- ✓ **Understand context**; Understanding of the Business (and IT) Context, including external facts (regulation, etc.) that may affect the results.
- ✓ **Define principles**; Formalized and traceable Business Objectives and Principles, driven by the Business Mission and Vision.
- ✓ **Know the requirements**: Understanding and/or delivering the functional and in particular the key non-functional requirements.
- ✓ **Record Decisions**; Documenting the rationale for all Architectural and Design Decisions, ideally reflecting the Principles/Business Needs.
- ✓ **Ensure traceability**; Providing clear traceability back to the Business Objectives within the Architecture
- ✓ **Develop solution**; Document the Solution(s) including Investigate Solution Alternatives to ensure that decisions are made holistically and not in isolation.
- ✓ **Assumptions and constraints**; Capturing, validating, and managing any assumptions and constraints that affect the Architecture.
- ✓ **Risks and issues;** Proactively document and manage risks and issues, both processes as well as the results.
- ✓ **Plan**; Have a clear and sensible plan/roadmap to achieve the desired business outcome(s).

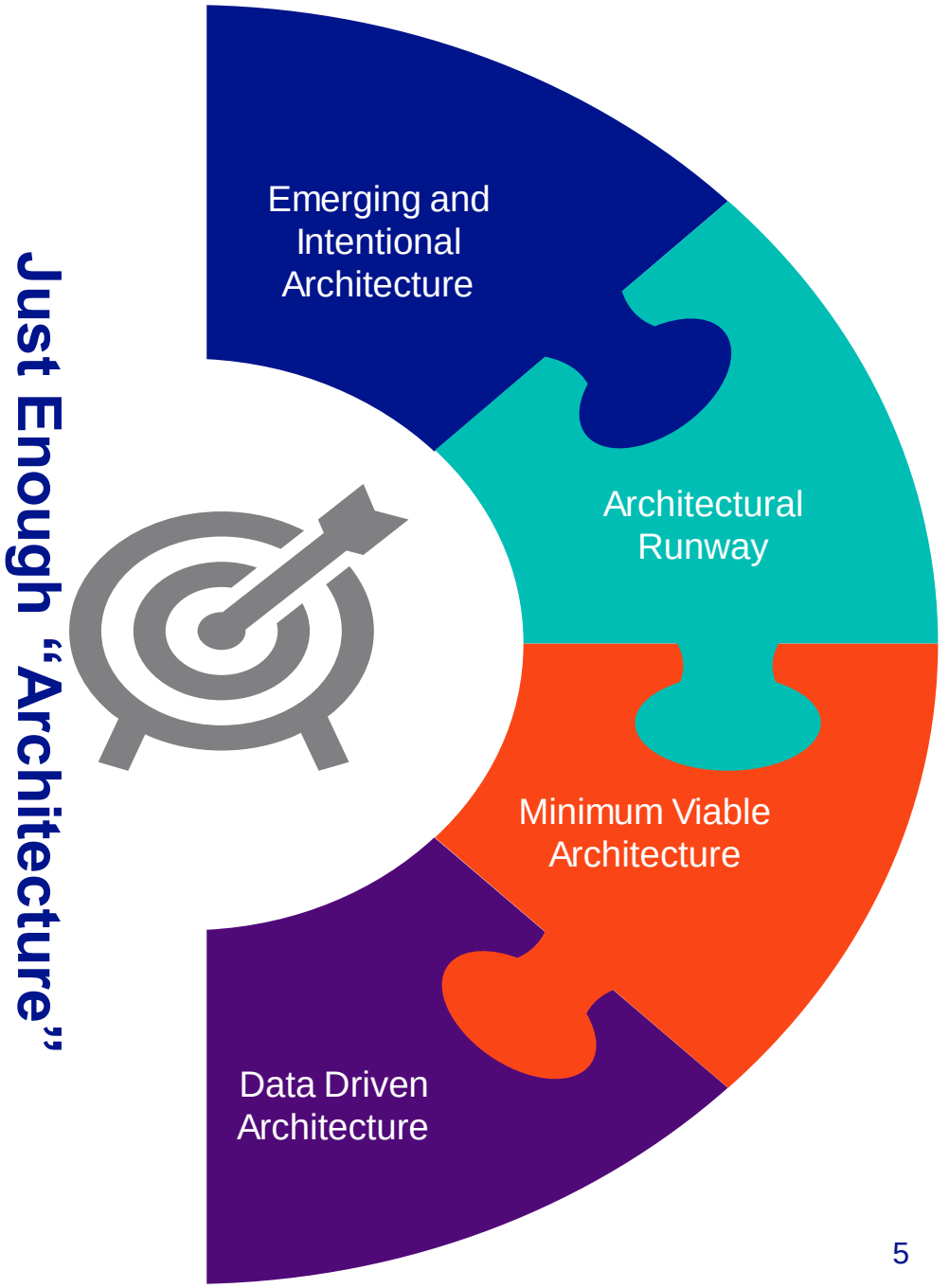
Just enough Architecture

nationalgrid | information technology

Just Enough “Architecture”

Just Enough Architecture - Developing ‘Just Enough Architecture’ can be characterized in the following:

- ✓ Objective of the Engagement - The architecture for ‘more of the same’ can (or even should) do with less architectural guidance than starting to work with new business activities or technology that needs to be integrated in an existing environment.
- ✓ Maturity of the Target Audience - Looking at the team level, Just Enough Architecture is the blueprint for what needs to be done in the next sprint (or set of sprints). In agile, this blueprint should leave the maximum room for design decisions to the teams. The more experienced and knowledgeable on the subject and on agile practices, the more room can be left to the team, provided the architecture gives the guardrails that keep the team on track.
- ✓ Amount of Technical Risk & Uncertainty - The more technical risk, the more up-front architectural design and/or research might be 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 & delivery dates, the amount of up-front architecture is significantly higher to get a better view on how much work is required.



Just Enough "Architecture" - Emerging and Intentional Architecture

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

- ✓ Think big, as the primary objective is still to build a high standard and competitive solution and having a clear high-level target architecture in mind
- ✓ Act small, team must deliver small operational pieces of software which demonstrate the value of enabler work
- ✓ Fail and learn fast, because there will be some failure and the sooner the better, to refactor or rebuild

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.

Just Enough "Architecture" – Architecture Runway

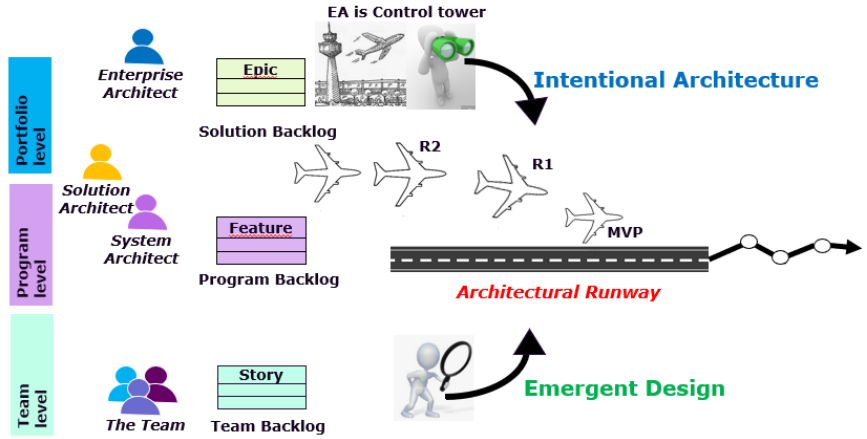
Organizations need to respond simultaneously to new business challenges with larger-scale architectural initiatives that require intentionality as well as planning. As a result, emerging architecture alone cannot handle the complexity of large-scale system development.

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

It is a foundational set of capabilities aligned to the big picture, that enable rapid development of new features.

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

- Exploration enablers which support research, prototyping (or spiking), et cetera to better understand customer needs and what solutions can be applied
- Architectural enablers covering guidelines, features, and stories from which the architectural runway for the teams is created
- Infrastructure enablers are created to build, enhance and automate the development, test and deployment environments
- Compliance enablers facilitating specific compliance activities like documentation, privacy and security and industry specific regulations



Just Enough “Architecture” – Minimum Viable Architecture

Good practices in the field show that Minimum Viable Architecture (MVA) can be defined as: the minimal set of principles to support the Minimum Viable Product to be released. An MVA should be defined in accordance with the aspects as discussed before.

However, an MVP might become part of a larger landscape. In that case, the MVA could also take this future situation into account.

Just Enough “Architecture” – Data Driven Architecture

Most developments in architecture take place in an existing situation. It is therefore good practice to use data on that existing information to support the architecture for new developments. Gathering data can be done on different levels and in different ways. Some examples are:

- Network Performance Monitoring might help identifying bottle necks in infrastructure or specific applications
- Application Portfolio Management might give clues on what systems need replacements or are loaded with technical debt
- 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

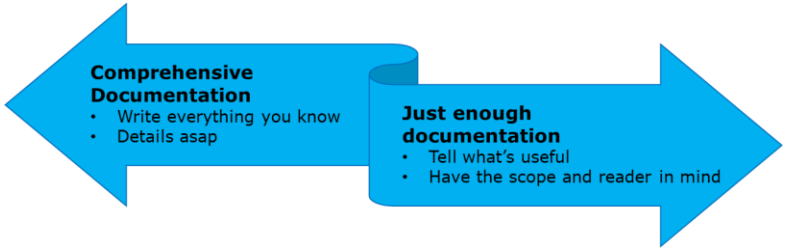
Techniques that might help gather and analyse data and then decide and implement the decision are the OODA loop , PDCA cycle, Lean A3 method and so on.

Just enough Documentation

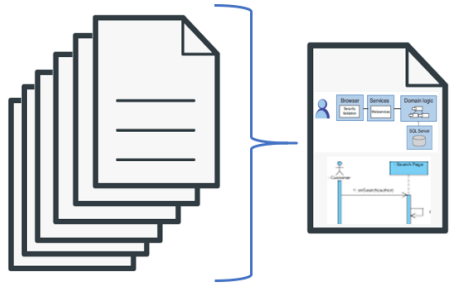
nationalgrid | information technology

Just Enough “Documentation”

What is Just Enough Documentation mainly depends on the purpose of the documentation, the target audience and the level of detail required at a certain time.



When creating architecture documentation, we should not document for the purpose of documentation or completion of a framework, but with the reader in mind. So before deciding what documentation to produce, it is important to have a clear view on 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 stakeholder or onboard newcomers).



In addition to defining the stakeholders and their needs, the following main principles should be used:
•Avoid any write-only documents: diagrams, pictures or spikes and walking skeleton worth a thousand words
•Avoid writing documents that are not useful: if the reader does not need it, do not write it!
•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 on the net
•Avoid duplication of documentation across multiple documents (single point of truth and maintenance)
•Use a centralized platform with powerful search and update facilities

The Just Enough Documentation mantra is “light but sufficient: better less but useful working documentation that is read by the Team, than old school extensive documentation read by none”.

But “with what” can we do that? Visual management is the key of just enough documentation: a picture is worth a thousand words. Diagrams using ArchiMate, UML and other techniques, enable for alignment and create shared understanding across all stakeholders (including architects and teams).

Just enough Governance

nationalgrid | information technology

Just Enough “Governance”

Ideally architecture governance in an agile context should be:

- integrated in existing agile rituals to minimize the number of dedicated meetings and maximize the 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 that is leading and guiding the teams
- distributed by design to empower architects at the right level and within clear boundaries in which they have the autonomy to take decisions
- data-driven to support decision making & 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 into this roadmap
- define the boundaries in which teams and architects have the autonomy to take their own architectural and design decisions (for example Architectural Runway seen before)
- support architects in managing the architectural landscape

Just Enough “Governance” - Faster Architectural Decision-Making

Slow decision making is one of the main issues when it comes to developing and agile architecture. To accelerate the decision-making process, empowerment of the right people at the right level is needed as easily reversable decisions require less formalism 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) architect(s) at strategic level

To enable such decision-making, it is important to define clear boundaries in which each level has the autonomy to take their own decisions.

A good practice is also to provide clear examples of decisions for each quadrant like changing cloud provider, changing public APIs, experimenting with a new open-source framework, adopting a new programming language, to name but a few.

Just Enough “Governance” - 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 actual solution built by the delivery teams (emerging architecture / bottom-up). While teams often make such deviations for good reasons, architects are not (kept) aware, which can lead to big differences between the “predefined architecture” (usually on paper) and the emerged “architecture as built”.

Some good practices to ensure architectural compliance:

- Rather than just a reference architecture on paper, a “walking skeleton” or a “working reference architecture” offers a kind of technical template that is fully in line with the required architectural principles. Which 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 DoD “Definition of Done” to ensure that principles and guideline 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

Just Enough "Governance" - Structural Collaboration between Architects and Delivery Teams

Architects are often perceived as "pushing down decisions from their ivory tower", not considering the realities on "the floor".

In a successful agile collaboration model:

- architects respect experienced professionals in the development teams and listen carefully to their suggestions
- architects are visible to the agile teams, so that their input and guidance is recognized, valued, and directly usable by the teams
- architects keep an overview on the bigger picture to ensure proper 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, advice, 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 to agile ceremonies (sprint planning, sprint review, retrospective, sprint demos, etc.).

Just Enough “Governance” - Architecture Community of Practice

To maximize learning and sharing of information and thoughts across architects, Recommendation is to set up a Community of Practice for Architecture (CoP). Such guild provides as 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, sharing learnings from architectural spikes etc.

Here Some other good practices:

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

Just Enough “Governance” - Architecture Decision Record (ADR)

The ADR covers all decisions related to the architecture and is therefore an important tool in the architecture governance, even more in an agile context then in waterfall.

The main reason is that some architecture decisions might be left to the teams and thus 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 as well.

Just in TIME

nationalgrid | information technology

Just in Time (1 / 2)

With Agile Architecture we need to transform from a ***Big Design up-front (BDUF)*** approach, in which 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 "descend" from their Ivory Tower to be closer to the developments, providing more visibility on the details without losing the broad view, vision and strategy. Doing so, allows the architects to learn by from every sprint and take 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 still 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 by when (e.g. to prepare for a next iteration)

Just in time governance: know what architecture decisions must be taken by when, what information decision makers need and by when they need it. A good practice is to 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 to make decisions with the maximum possible information, but also avoids losing time in redoing the same discussion. Until this latest responsible moment, you are learning & collecting information.

Just in Time (2 / 2)

Just in Time has different meanings depending on the level of architecture one is working on:

- Team Level: the architect focuses on what the agile teams need for their current and next sprint. To do so, architects should have regular connects with the agile teams to be able to anticipate what guidance and information they need at what point in time. A good practice is to have both formal (e.g. attending agile ceremonies) and informal connects

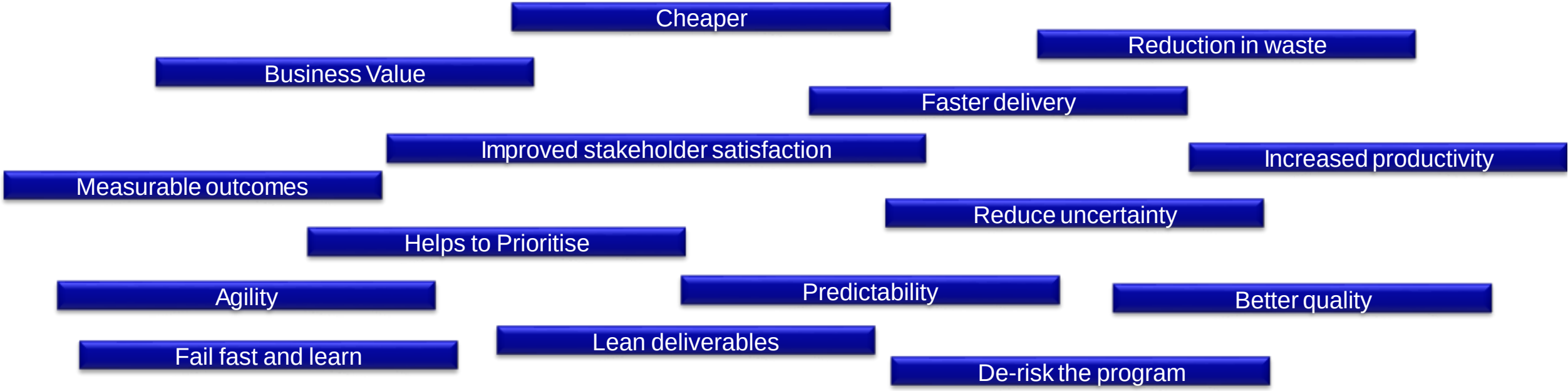
- Enterprise Level: the architect understands what product management plans to have delivered by the teams in a certain timeline (for instance six to twelve months). From that perspective, the architect derives the Enabler Epics that prepare for delivering those business needs and what guidance is expected to be necessary

- Program / Solution Level: the architect reconciles the needs from the team and enterprise level. As a rule of thumb, this means that 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 all about good communication and being connected on different levels from team to enterprise.

How does these “Just in” benefit National Grid ?

“Just in xxxxxx” provides



Thanks and questions