Capgemini

AGILE & IT ARCHITECTURE PART. 2

The JIT-JEA in Action

WELCOME TO THE JIT-JEA IN ACTION

After we issued the first JIT-JEA (Just In time-Just Enough Architecture) external POV paper in January 2022, we wanted to provide some working examples illustrating not just the definition of JIT-JEA but also showcasing how the 'art' of Agile Architecture is practised.

This paper entails 10 real-life examples covering 'Just Enough Architecture', 'Just Enough Documentation', 'Just Enough Governance', 'Just in Time' and 'In-iteration Size Chunks'.

[illegible]

KAI SCHROEDER

Global Architects Community Lead München, DE

[illegible]

GUNNAR MENZEL

Master Certified Architect Manchester, UK

[illegible]

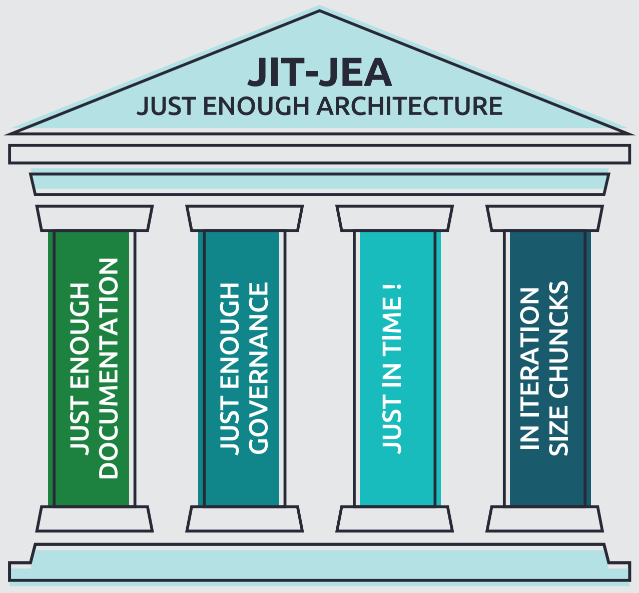
STEFANO ROSSINI

Chief Architect Italy TPO Lead Milan, IT

TABLE OF CONTENTS

INTRODUCTION

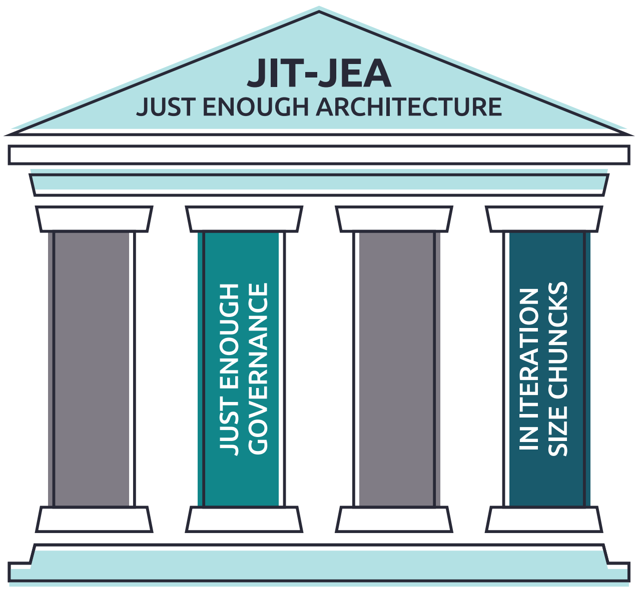
We will share in this document 10 concrete examples that you may leverage to accelerate your practical way to do Architecture in an agile context. We have mapped them, in the table below, with the 5 agile pillars from our JIT-JEA model:



N.	Sample	Just Enough Architecture	Just Enough Documentation	Just Enough Governance	Just in Time!	In iteration size chunks
1	Architecture Enablers	√		√		√
2	Documentation as Code	√	√			√
3	Lightweight ADR	√	√	√		√
4	Architecture Styles	√			√	
5	Walking Skeleton	√	√	√		
6	Architectural Runway	√		√		√
7	DevOps Guardrails	√		√		√
8	Low Code	√		√		
9	Design Systems		√	√	√	
10	Technology Radar		√	√	√	

Disclaimer: The examples provided are for illustrative purposes only and may not fully capture the specific context of the client's situation.

SAMPLE 1: ARCHITECTURE ENABLERS



Scope: Replatforming of a legacy application

Sector: Insurance

Tools: JIRA

A practical way to introduce 'Just Enough Architecture' topics into the Agile way of working is to introduce a new kind of task into the agile backlog to describe architectural and technical topics.

SAFe *names these kind of activities* **Enablers** *and describes four types of them: exploration, architecture, infrastructure, and compliance [SAFe Enablers].*

Agile aims to deliver value through business features however, architectural topics must also be addressed to guarantee mid-term and long-term results.

Product Owners and Agile Architects must work together to **decide on the right balance** between User Stories and Enablers into the Product and Sprint Backlogs.

Having Enablers displayed on the task board the way user stories are visualized makes this task easier and gives the team the **capability to spend the right amount of effort** on them.

Enablers bring the following key benefits:

- Enablers make the architecture and technical debt topics visible in the backlog
- Enablers allow the team and Product Owner to prioritize architectural topics together with business features and develop them in iteration size chunks
- With enablers, the Agile Architecture is built-in and integrated in the standard Agile Way-of-Working

Enablers in action:

For an important insurance project, we extended the configuration of the JIRA Tool that was used to track business user stories.

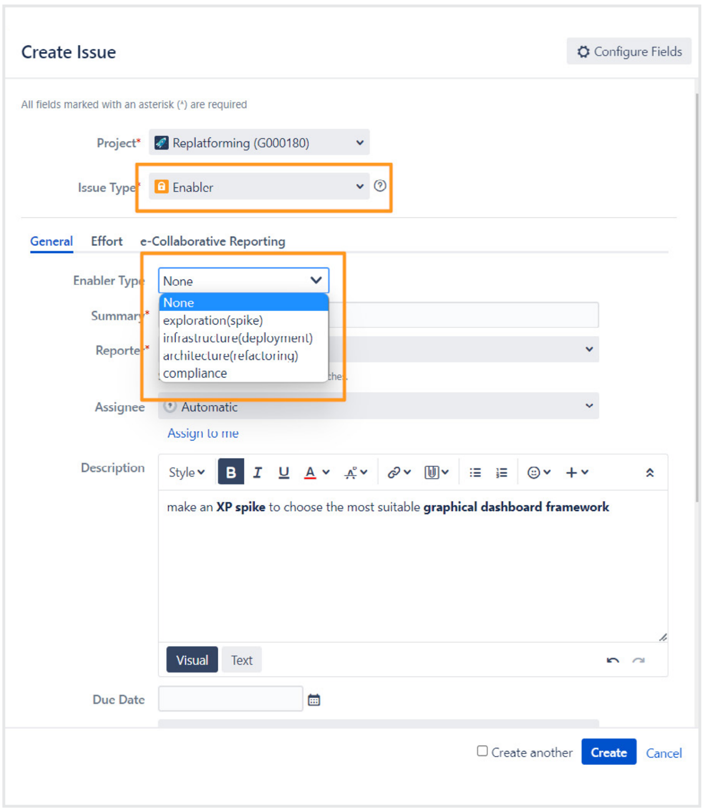
We created a new Jira Issue Type representing Enablers and a new custom field to track its different types—Exploration, Architecture, Infrastructure and Compliance.

We decided to manage the new Enablers stories in the same way (with the same workflow and same task boards) we did for business stories, avoiding the creation of a 'dedicated' task board and integrating the development of Enabler tasks with the rest of activities.

Using different card colours helped to visualize – starting from the iteration planning activity – the balance between enablers and user stories.

Activities like 'create a nightly build of DevOps Toolchain' or 'need for a Spike for a cache component' were, therefore, clearly visible in the task board as Infrastructure and Exploration enablers with priority and estimation, and their progress has been tracked during the iteration.

During the iteration review we demonstrated the results achieved through Enablers and we tracked the percentage of time spent on Enabler stories compared to business user stories.



Create Issue

All fields marked with an asterisk (*) are required

Project*: Replatforming (G000180)

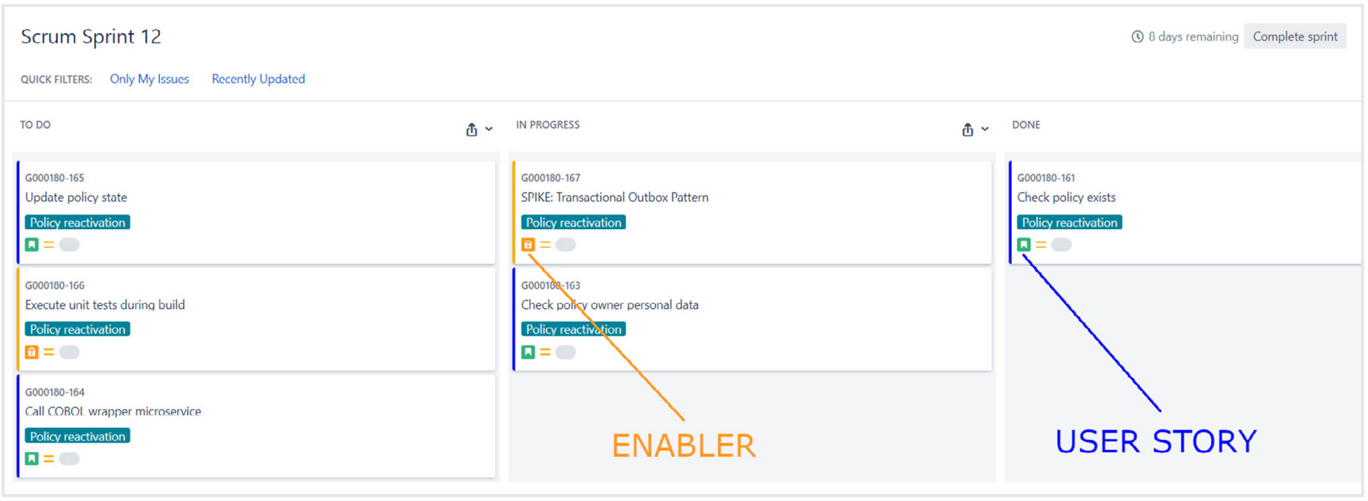
Issue Type*: Enabler

Enabler Type: None

- None
- exploration(spike)
- infrastructure(deployment)
- architecture(refactoring)
- compliance

Assignee: Automatic

Description: make an **XP spike** to choose the most suitable **graphical dashboard framework**



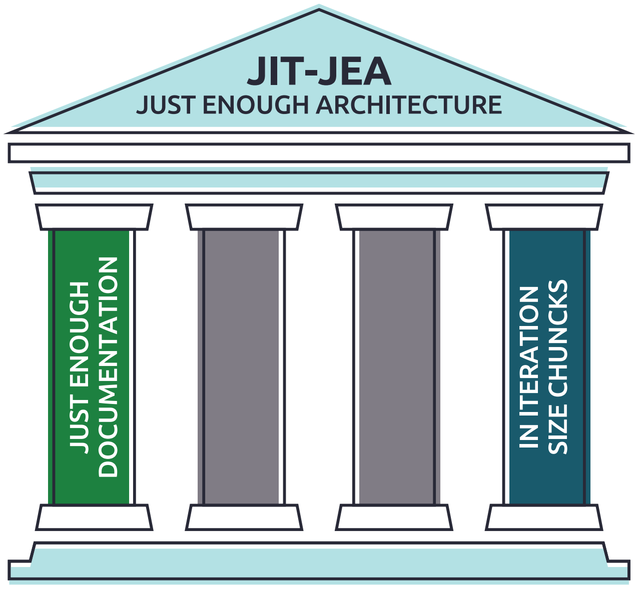
Scrum Sprint 12

QUICK FILTERS: Only My Issues Recently Updated

8 days remaining Complete sprint

TO DO	IN PROGRESS	DONE
G000180-165 Update policy state — Policy reactivation	G000180-167 SPIKE: Transactional Outbox Pattern — Policy reactivation	G000180-161 Check policy exists — Policy reactivation
G000180-166 Execute unit tests during build — Policy reactivation	G000180-163 Check policy owner personal data — Policy reactivation	
G000180-164 Call COBOL wrapper microservice — Policy reactivation		

SAMPLE 2: DOCUMENTATION AS CODE (DAC)



Scope: Web Application

Sector: Internal Capgemini Product

Tools: Gitlab, MkDocs, HTML website, gollum

The mantra of Agile Architecture is that the architecture must always be in sync with the code.

'Everything as code' is a commonly adopted principle by Agile and DevOps practitioners and useful for Agile Architecture. It consists in applying the same techniques used to manage source code (reviewing, versioning, branching, configuration control) to all the other aspects related to software development: infrastructure, environment provisioning, architecture itself, DevOps pipelines and also **Documentation** [Documentation as Code].

Using a plain-text format for documents, such as AsciiDoc, Markdown, or LaTex, which are among the most common ones, we can leverage the powerful features of source-code management, including versioning, diff, merge, etc.

The source code of documents is processed to build human-readable formats such as Microsoft Word and PDF or even HTML websites.

Documentation as Code brings the following key benefits:

- The simple markdown format helps to focus on the content and readability of the document more than on its format and beauty.

- The document can be produced in an easy, effective, and collaborative way and can be easily updated and maintained over time. For example: producing a new version for every agile iteration and following the same lifecycle of the software development.
- The version history helps to understand who has applied the changes, when, and how. The work done by all the team members is merged into a single document with the possibility to review, integrate, and validate. Publishing workflows can be managed as well.

Documentation as Code in action:

For a Capgemini product, we have created and maintained technical documentation to keep track of all the configurations related to different environments.

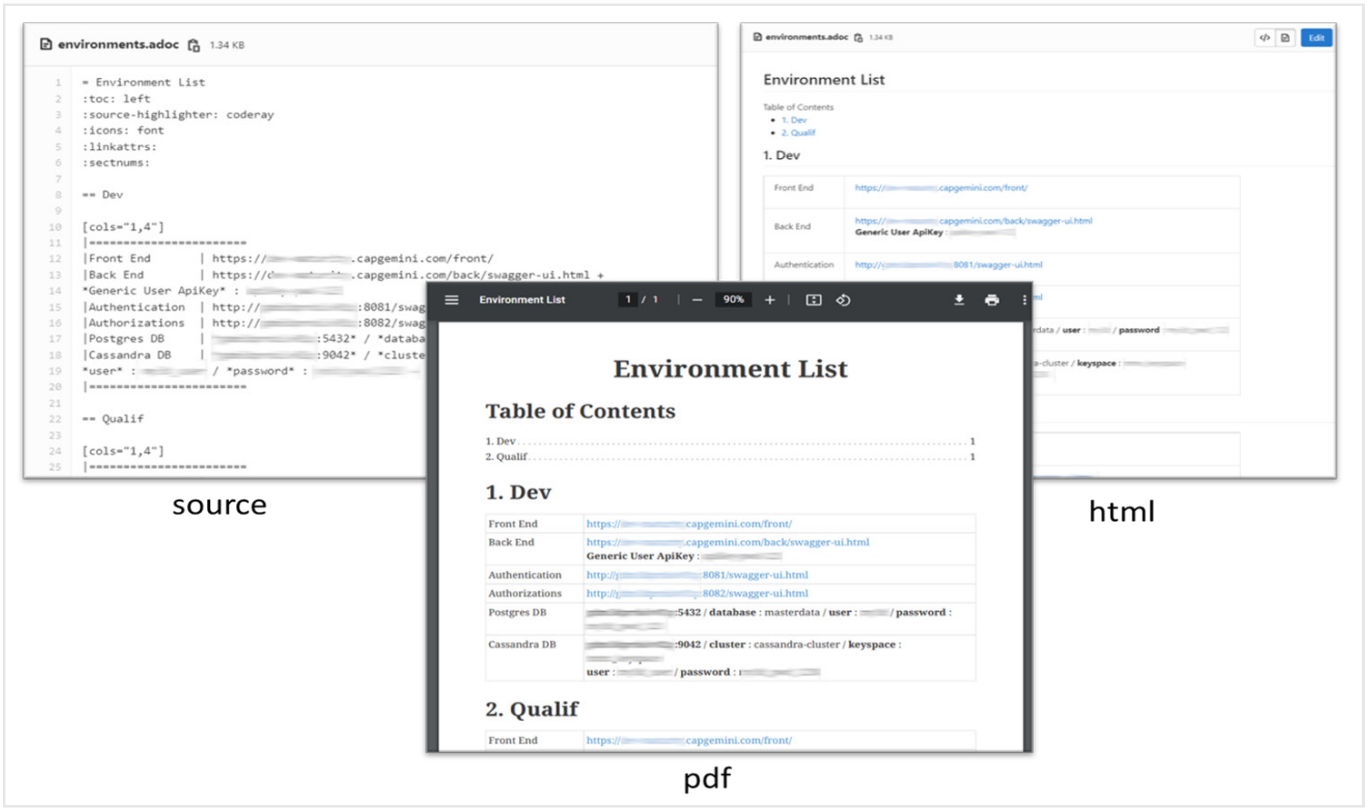
We have also produced a user manual and technical section to explain details of the user interface and calculation rules applied in the software.

This documentation was periodically shared during workshops to describe the product features to the users.

We decided to use the Markdown language [Markdown Language] to format the document, leveraging the built-in capability of Gitlab to automatically render the document in HTML.

For the developers to be able to locally modify the document and immediately see the effect of their changes, we chose Gollum [Gollum] as the markdown editor and renderer, fully integrated with GIT.

We were also able to compile PDF documents from markdown for major releases of each document.



```
= Environment List
:toc: left
:source-highlighter: coderay
:icons: font
:linkattrs:
:sectnums:

== Dev

[cols="1,4"]
|======================
|Front End        | https://[redacted].capgemini.com/front/
|Back End         | https://[redacted].capgemini.com/back/swagger-ui.html +
*Generic User ApiKey* : [redacted]
|Authentication   | http://[redacted]:8081/swag
|Authorizations   | http://[redacted]:8082/swag
|Postgres DB      | [redacted]:5432* / *databa
|Cassandra DB     | [redacted]:9042* / *cluste
*user* : [redacted] / *password* : [redacted]
|======================

== Qualif

[cols="1,4"]
|======================
```

Environment List

Table of Contents

- 1. Dev
- 2. Qualif

1. Dev

Front End	https://[redacted].capgemini.com/front/
Back End	https://[redacted]capgemini.com/back/swagger-ui.html **Generic User ApiKey** : [redacted]
Authentication	http://[redacted]8081/swagger-ui.html
	ml
	rdata / **user** : [redacted] / **password** [redacted]
	a-cluster / **keyspace** : [redacted]

Environment List

Table of Contents

1. Dev 1
2. Qualif 1

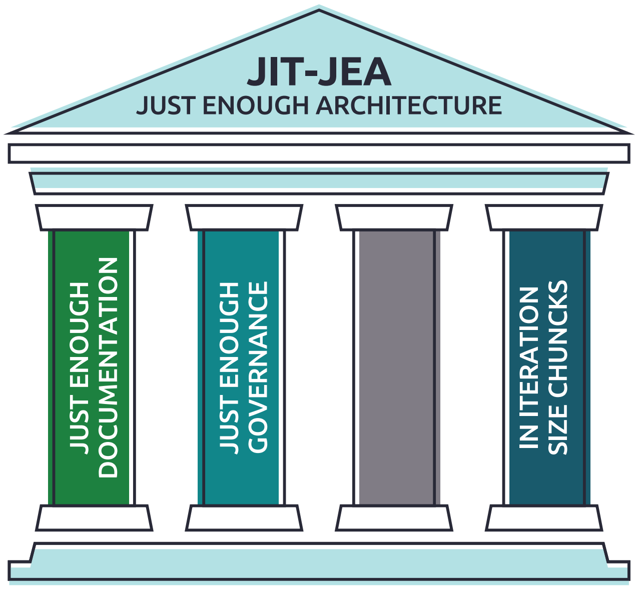
1. Dev

Front End	https://[redacted]capgemini.com/front/
Back End	https://[redacted].capgemini.com/back/swagger-ui.html **Generic User ApiKey** : [redacted]
Authentication	http://[redacted]8081/swagger-ui.html
Authorizations	http://[redacted]8082/swagger-ui.html
Postgres DB	[redacted]:5432 / **database** : masterdata / **user** : [redacted] / **password** : [redacted]
Cassandra DB	[redacted]:9042 / **cluster** : cassandra-cluster / **keyspace** : [redacted] **user** : [redacted] / **password** : [redacted]

2. Qualif

Front End	https://[redacted]capgemini.com/front/

SAMPLE 3: LIGHTWEIGHT ARCHITECTURE DECISION RECORD (LADR)



Scope: Enterprise

Sector: Automotive and Insurance

Tools: Sharepoint

Any architectural decision must be tracked and versioned in a simple and effective way. It is better to have lightweight information updated frequently than heavy comprehensive documents that can quickly become outdated. Furthermore, those decisions need to be shared not only with IT but with the business.

For any architectural decision, we just describe the context of how it was taken with possible scenarios envisioned before reaching there and the foreseen implications and consequences. Another field is used to track the status, including proposed, accepted, and rejected.

We recommend adding tags to be able to categorize the decisions, e.g. by product, organization, type of decision, etc. These tags act as filters to help take the decision easily: a search engine is key, to ensure the success of LADR usage within the teams.

Because a decision can be revised and changed over time, it is important to keep track of the changes to the record itself so that it is possible to see how it looked like at a certain point in time.

Using the right tool is key to achieve ease of access both to track new records and read the past decisions. Documentation as Code, as described earlier, is a great technique to implement a lightweight decision registry, ensuring both a quick and easy access and the capability to track changes and compare revisions.

A sample Lightweight Architect Decision Record (LADR) can be found here [LADR Example].

LADR brings the following key benefits:

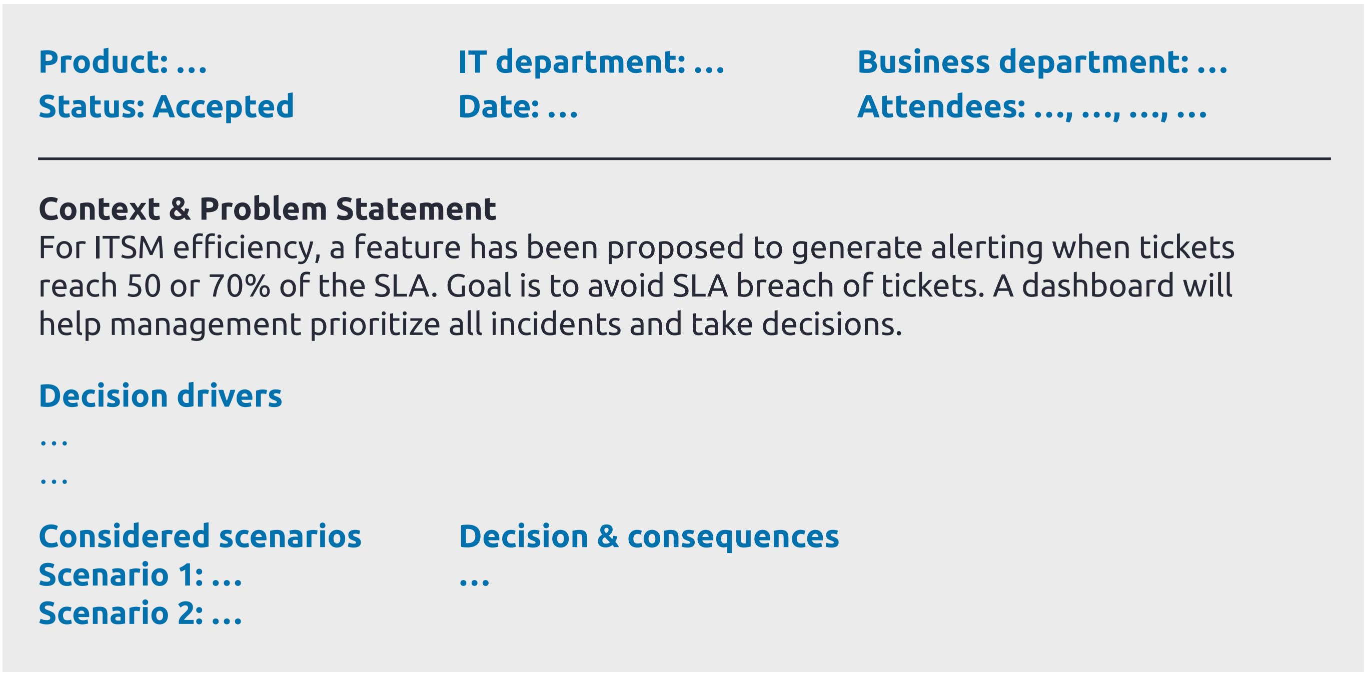
- Architecture decisions are easily shared and communicated through the organization. They can be easily found through a search engine.
- LADR speeds up the decision process by bringing focus and alignment across all roles in the organization.
- LADR gives you a clear Architecture history log, which otherwise would have been hidden in big architecture documents.

LADR in action:

For two important customers from the automotive and insurance sectors, it was decided to keep track of all the Architectural decisions in a lightweight flavour.

To be sure that the documents were always updated to track every key decision taken by the team, we enhanced the 'Definition of Done' of the Agile Team to include the updated ADR as an integrated and mandatory step of the software development lifecycle. If during the analysis or the development of a user story the team takes an architectural decision, the story itself cannot be considered as 'done' until the related Architect Decision Record is properly created or updated.

The ADR registry is periodically checked before every code release.



Product: …	**IT department: …**	**Business department: …**
Status: Accepted	**Date: …**	**Attendees: …, …, …, …**

Context & Problem Statement
For ITSM efficiency, a feature has been proposed to generate alerting when tickets reach 50 or 70% of the SLA. Goal is to avoid SLA breach of tickets. A dashboard will help management prioritize all incidents and take decisions.

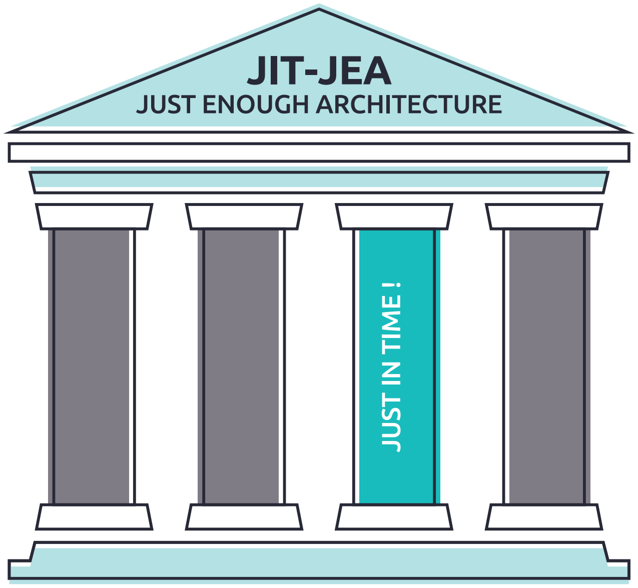
Decision drivers
…
…

Considered scenarios
Scenario 1: …
Scenario 2: …

Decision & consequences
…

LADR sample for automotive client (over a Sharepoint website)

SAMPLE 4: ARCHITECTURE STYLES ENABLING DESIGN FLEXIBILITY



Scope: e-Filing System

Sector: Public Sector

Tools: Archi and Power point

An emerging architecture encourages teams to make decisions 'Just in Time'. Consequently, an architectural style used in an agile environment should enable the ability to respond to changes quickly. One principle that supports agility is a clear separation of concerns between the domain and other parts of a system. While decoupling the domain part from other parts, the domain logic is kept independent of any technical aspects, making it easier to change or delay technical decisions without impacting the domain model.

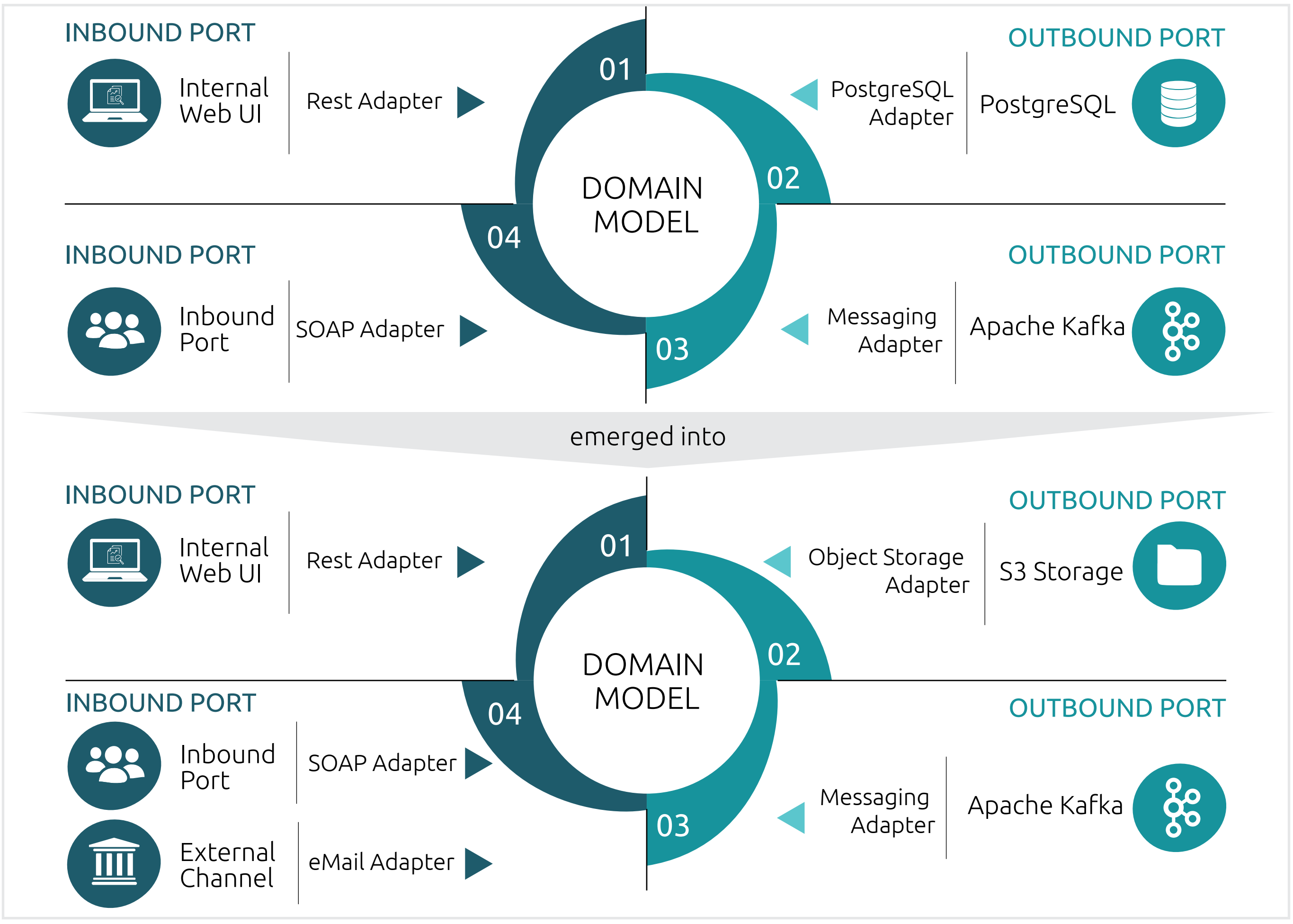
There are different architectural styles that follow the idea of a clear separation of concerns. Examples are the hexagonal architecture style promoted by Alistair Cockburn [Hexagonal Architecture] and Clean Architecture promoted by Robert C. Martin [Clean Code]. Both architectural styles suggest that the domain model is forming the core of the system with no dependencies on the other parts of the system. All dependencies are directed towards the domain core, which provides interfaces to be invoked by the surrounding components (Inbound) as well as interfaces that are invoked by the domain core and implemented by the surrounding components (Outbound). These kinds of architectural styles lead to some considerable benefits with regards to an emerging architecture:

- **Delaying technical decisions:** Since the domain part of the system is well encapsulated, the team can evolve the domain logic without depending on the technical decisions. That makes it easier to postpone decisions to a more responsible moment. For example, you can kick off a project with no database at all and decide later if a relational or NoSQL database suits you better.
- **Changing infrastructure with less impact:** With the inward-facing dependencies, you do not rely on adapter implementations. That makes it easier to change or replace them. For example, you can easily replace a REST adapter with a gRPC adapter.
- **Testability of business logic:** Since the domain part of the system is independent of any surroundings, a high testability without the use of infrastructure components can be achieved.

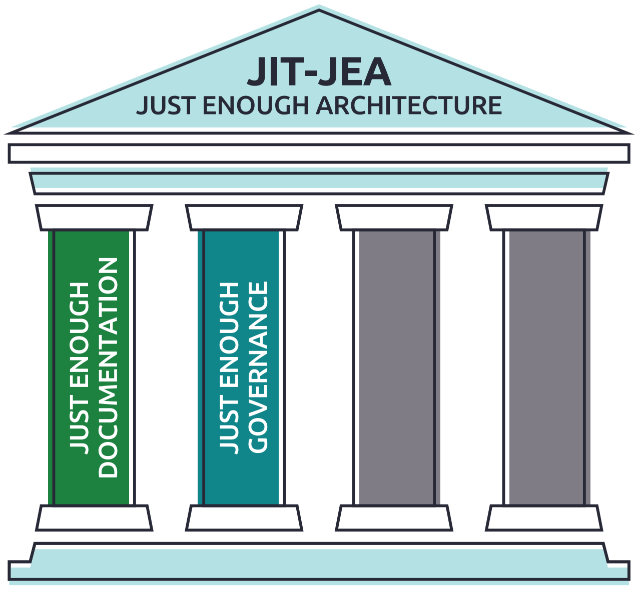
Postponed decisions can be tracked as part of an architecture decision log.

Architecture styles enabling design flexibility in action:

In a public-sector project with high uncertainty and technological complexity, the hexagonal architecture style was used to keep the domain logic independent of database technology. Initially unsure about which database technology to use, the team used a simple and well-known database while delaying the decision. Eventually, the team gained enough confidence to make the infrastructural change with minimal effort. The clean domain core allowed for effective management of different technologies and easy evolution of the solution by adding specific adapters. The testability of the business logic was completely independent of technological enhancements, ensuring validity of the domain logic at any time.



SAMPLE 5: WALKING SKELETON AS AGILE ARCHITECTURE REFERENCE



Scope: Microservices based web application

Sector: Insurance

Tools: Eclipse IDE, SpringBoot, Oracle 11g, Jenkins toolchain

"A Walking Skeleton is a tiny implementation of the system that performs a small end-to-end function. It need not use the final architecture but it should link together the main architectural components. The architecture and functionality can then evolve in parallel." — Alistair Cockburn [Walking Skeleton]

The walking skeleton is a good strategy to introduce 'Just Enough Architecture' topics into the Agile Way-of-Working.

'Walking' means working, executable and testable, 'Skeleton' means minimized & simplified, end-to-end.

The goal of the Walking Skeleton is to be able to explain the whole product in terms of:

- Basic technologies and frameworks
- Architectural building blocks
- Core design patterns to be used

It's a minimal implementation that proves the intentional architecture through some end-to-end functionality. The goal is to get early feedback on whether the chosen design, architecture, and technology are suitable.

Walking Skeleton brings the following key benefits:

- Basic reference implementation
- Early feedback for the Minimal Viable Architecture (MVA)
- Working architecture that is easy to scale
- Set the initial direction that every developer must take when approaching a new implementation

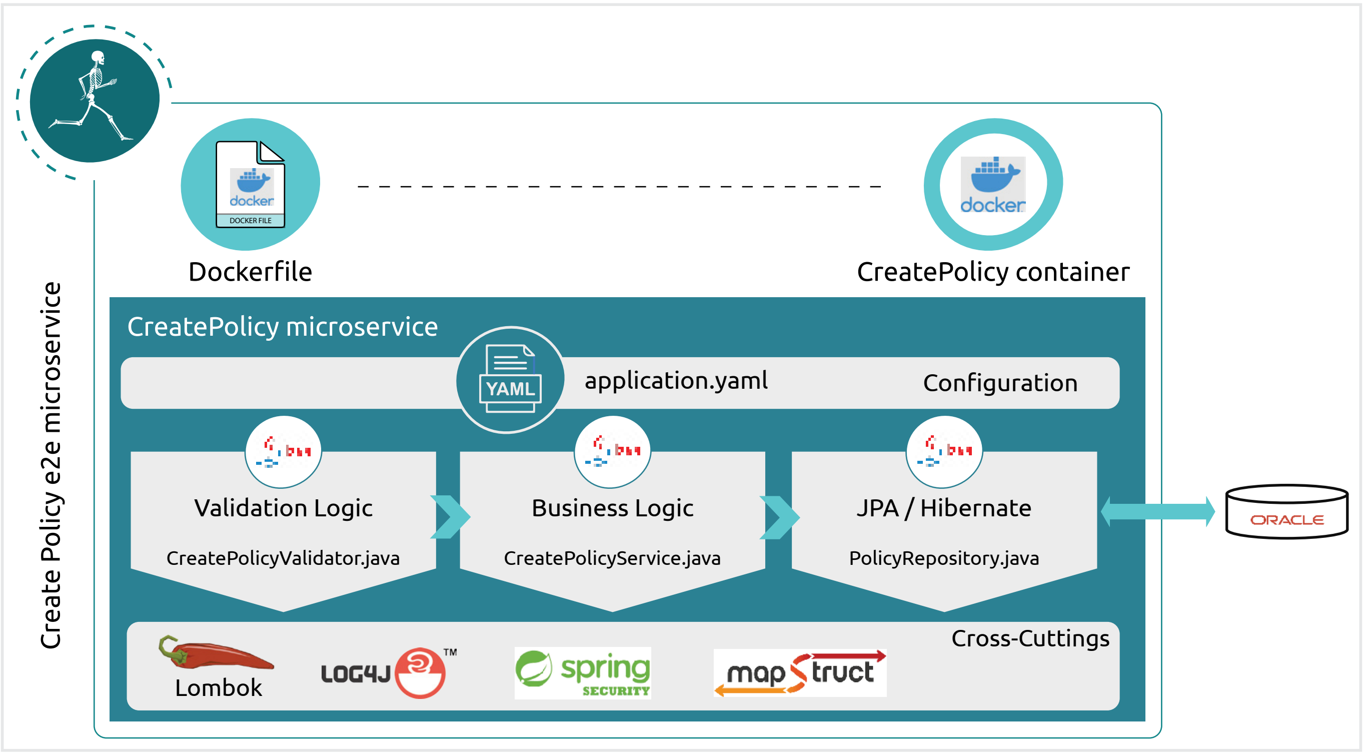
Walking Skeleton in action:

For an important insurance project, we had to implement a set of microservices to enable the communication between web and mobile applications and a legacy system.

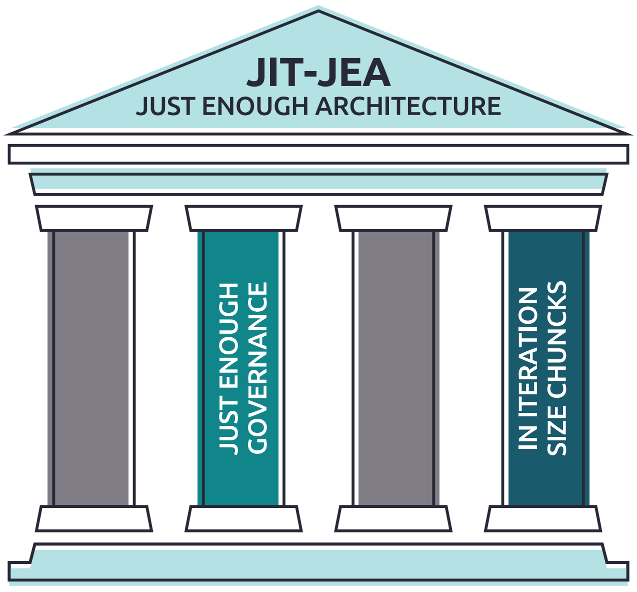
To create a microservices reference implementation for the developers and have a common guiding implementation among all microservices, we created a 'Walking Skeleton' based on the target architecture including Spring Boot, Spring MVC, Spring JPA. We implemented it as a medium-complexity microservice, not too complex to remain readable and easily understandable, and not too simple to remain meaningful.

The Walking Skeleton is the full-stack working implementation of the microservice, from the HTTP endpoint to the database and external legacy service invocation.

Most of the programming and cross-cutting aspect, including documentation, logging, security, naming convention, package and layer organization, feature flags, guardrails has been covered in the simplest but still meaningful way possible.



SAMPLE 6: ARCHITECTURAL RUNWAY



Scope: Digital Transformation

Sector: Public

The Architectural Runway ***consists of the existing code, components, and technical infrastructure needed to implement the near-term features without excessive redesign and delay*** [SAFe Architectural Runway].

This concept is in place when the architecture is capable of 'landing' the upcoming system or software releases. Iterations or releases are compared to airplanes coming in for landing, and the architecture is to ensure that the runway is suitable for it to land safely.

The Architectural Runway provides a way of integrating changes with the emerging architecture during the life of a long project.

These will be analysed and reviewed with the existing and future 'safe landing' at the heart of all the decisions reliant on constant communication and feedback between the teams, in the same way a pilot and control tower consistently update the local and changing conditions.

Architectural Runway brings the following key benefits:

- Provide consistency of the IT landscape
- Reduce rework
- Continuous evaluation against the changing external elements

Architectural Runway in action:

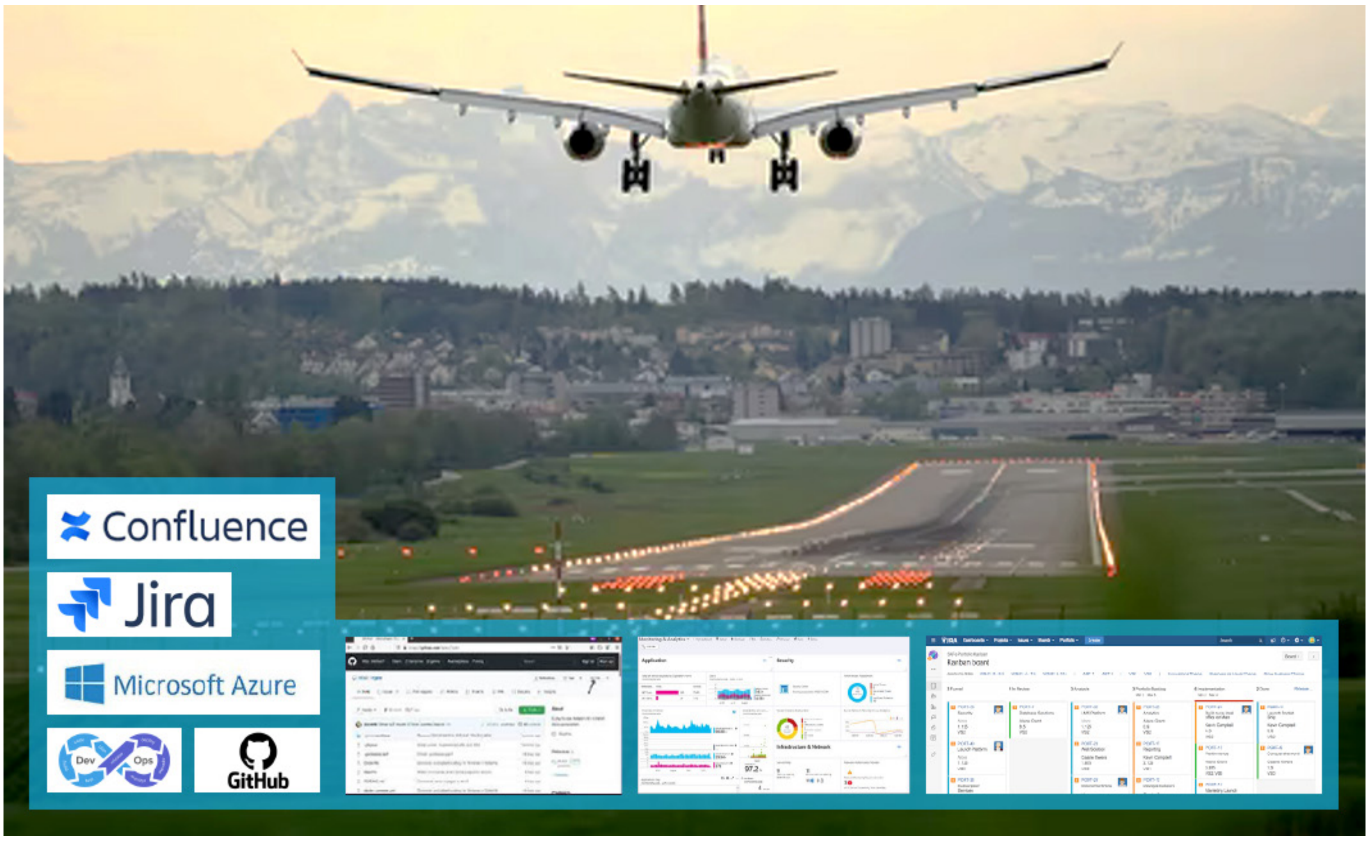
For a client on a digital transformation program, we have implemented an Architectural Runway as a real metaphor for the landing of an airplane. We needed an open communication channel with the control tower, a clear and long enough landing track (depending on the size and weight of the plane), the right airplane configuration (depending on the weather conditions), supporting systems to guide the airplane during the landing approach, well-defined procedures (e.g. a landing checklist) and continuous monitoring of all flight parameters (airspeed, altitude).

We started by putting in place the communication channel with the control tower—a dialogue zone between the transversal program architects in charge of intentional architecture and teams responsible for emerging design. Every week the backlog JIRA items were reviewed, the user stories related to architecture were identified and related to intentional architecture.

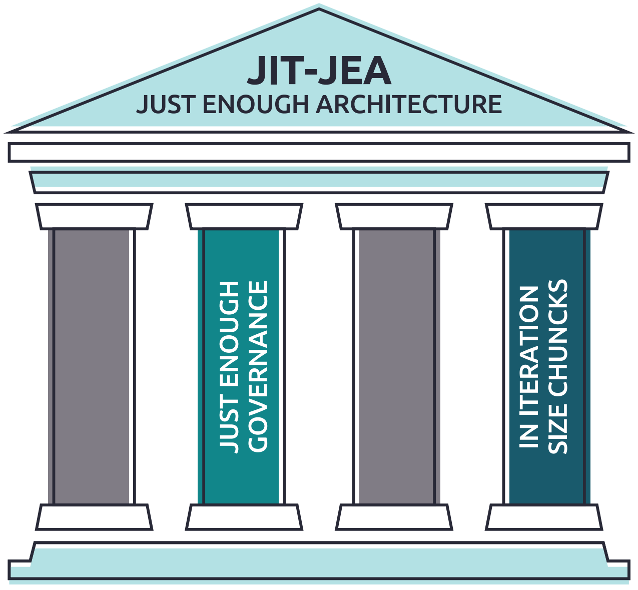
Debates in this dialogue zone allowed commonly finding the relevant design patterns, technologies, checklists, configuration with features toggles, launch procedures and architectural decisions—all those were stored in Confluence.

To be able to successfully deploy the application, we had to ensure that the Azure environment was ready on time and could cope with the non-functional requirements to let the application run smoothly and land safely. In addition, we had defined a CI/CD pipeline supporting automatic deployments in the right environments to guide the landing approach.

We also used KPIs to monitor the flight parameters of the evolving architecture, including Java-based microservices, REST APIs, Azure PaaS such as network usage using Dynatrace and API Consumption using API Gateway dashboards.



SAMPLE 7: DEVOPS GUARDRAILS



Scope: Capgemini Product

Sector: Banking

Tools: DevOps Toolchain Production Line, e3d Jira, Opensource tools

An architecture is defined with a set of principles, guidelines, technologies, good practices, and standards.

To be sure that architectural decisions remain respected during the project and maintenance lifecycles, we need to setup some mechanism to quickly check if these are followed in a proper way.

As with their real-world roadside equivalents, software and technologies guardrails are designed to keep people from straying onto dangerous or forbidden territory.

In real terms, guardrails represent a **lightweight governance structure.** [O-AA Guardrails]

DevOps Platforms provide a quick and effective way to establish IT Guardrails and kickstart the JIT-JEA way of working across all team members and stakeholders. Combined with Walking Skeletons and Reference Architectures, it can be used to steer architectural principles and assure consistency across the IT delivery.

Together with communication tools and fitness functions, including automatic unit tests, performance tests, and code coverage, these modern tools offer several different features on a single platform that can be leveraged as technology guardrails. Samples of such DevOps Toolchains are Azure DevOps, Atlassian Suite, GitLab and many others.

DevOps Guardrails bring the following key benefits:

- Automatic check of the architecture compliance
- Help decentralized decision-making
- Avoid dangerous decisions taken by single developers

DevOps Guardrails in action:

Within the banking sector, we used a DevOps Toolchain combined with Atlassian Jira Software and MS Teams to set up guardrails that ensure consistency across the delivery process.

We used the Production Line, the Capgemini Industrialization DevOps Toolchain based on the best-in-class open-source components such as Jenkins, Gitlab, Nexus, SonarQube and more [PRODUCTION LINE CAPGEMINI].

After defining the methodologies, the guidelines and good practices, we configured the tools to implement the architect decisions and detect any violation of the agreed behaviours at the earliest.

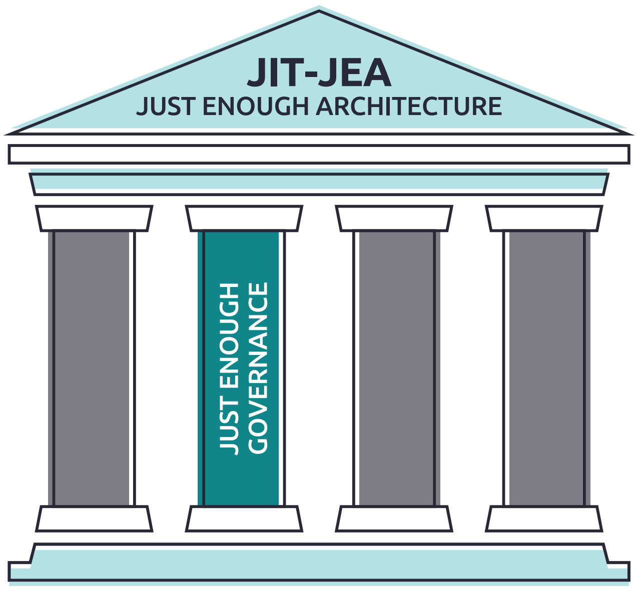
- We used ArchUnit to check source code against the agreed software architecture rules, including software layer consistency and naming conventions
- We used daily SonarQube, CAST and Checkmarks to check quality and security of the software code that ensured the absence of blocking or critical issues
- We used JUnit for unit testing and Selenium for Web pages testing integrated in the build process to ensure that all test cases passed before releasing the software
- JaCoCo Jenkins Plugin were used to check that the Test Code Coverage was above the acknowledged threshold of 50%
- We checked that the DevOps Toolchain is used on a day-to-day basis, at least once in a day
- We used Jira Software to setup Scrum boards for all the developer teams to have Scrum sprints synchronized and of the same length of two weeks

GREENSIGHT
sonarqube
Jenkins
Console Output
JUnit
ArchUnit
JACOCO
Java Code Coverage

wordLength

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
it.capgemini.productionline.example.service		77%		83%	2	6	3	22	1	3	0	1
it.capgemini.productionline.example.feature		80%		n/a	3	6	3	9	3	6	0	2
it.capgemini.productionline.example.entity		58%		n/a	3	7	4	10	3	7	0	1
it.capgemini.productionline.example.controller		25%		n/a	2	3	2	3	2	3	0	1
it.capgemini.productionline.example		72%		n/a	1	3	2	4	1	3	0	1
it.capgemini.productionline.example.exceptions		100%		n/a	0	1	0	2	0	1	0	1
Total	60 of 223	73%	1 of 6	83%	11	26	14	50	10	23	0	7

SAMPLE 8: LOW CODE FOR RAPID EVALUATION OF ARCHITECTURE ALTERNATIVES



Scope: Enterprise

Sector: Insurance

Tools: Low code (Mendix)

Set Based Concurrent Engineering [O-AA SBCE] [SAFe – Principle 3] is an approach to evaluate multiple product architecture alternatives. It builds upon principles to keep options open, while evaluating different alternatives and eliminating the weaker ones. While current technologies support this way of working, there often is not the time to create extensive prototypes and perform market research to evaluate the different alternatives. This is where rapid prototyping with low code comes in.

Low-code technology offers a cost-efficient way of creating working enterprise applications to test out real business products and architecture alternatives. And the best thing is the sunk or non-recoverable cost in case of a solution failure is very low. However, if the solution is a success, low code provides a solid enterprise level application that can easily be adapted and changed as required. Low-code technology lowers development cost while it speeds up the critical time-to-market that is crucial to stay competitive.

Note that low code is not mandatory to do rapid prototyping or evaluation of multiple architecture alternatives. However, experience shows it will speed up the process. Before applying low code, one should spend time on selecting the right platform that supports the use case(s) at hand.

Rapid Prototyping brings the following key benefits:

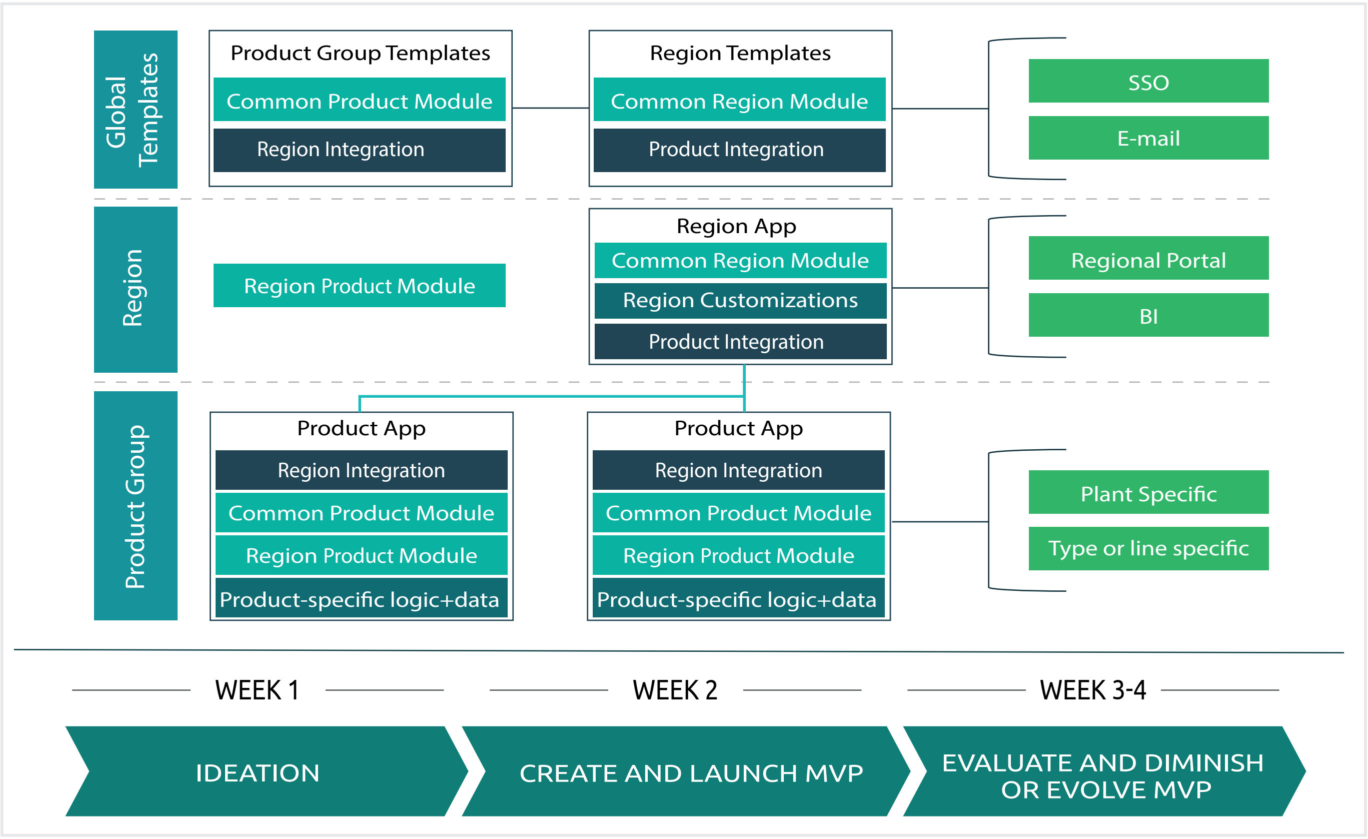
- Explore different architecture alternatives in days/weeks instead of months
- Capture requirements faster
- Low-code acts as an intrinsic guardrail, speeding up development and evaluation

Rapid Evaluation with Low Code in action:

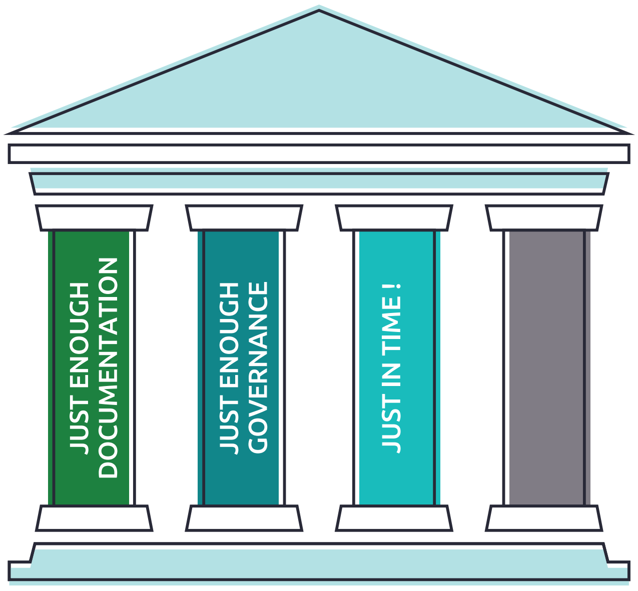
A global insurance company struggled bringing live new insurance products—the time from ideation to market research and launch of a product could easily take 9-18 months. In the current market scenario, that's a recipe for failure and, ultimately, bankruptcy.

We set up a composable microservice architecture based on the Mendix low-code platform, which allows experimentation of new insurance products within weeks instead of months. The initial architecture as shown below can be considered an Agile Architecture as it does not (yet) contain many details. Instead, it allows further evolution guided by prototyping and experimentation for many insurance products.

The advantage of this architecture, supported by low-code technology, is that it enables rapid prototyping and experimentation for new business ideas at low cost. It also makes it possible to launch new insurance products only two weeks after ideation. This supports the agile organization and staying ahead of the competition.



SAMPLE 9: DESIGN SYSTEMS



Scope: Front-end Apps

Sector: Energy Sector

Tools: Figma, Zeroheight and Miro

Companies are connecting with their customers in more ways than ever. While clients often use multiple channels, they always expect a seamless and consistent omnichannel experience.

Especially in larger organizations, with multiple agile teams working on different products, it becomes a challenge to deliver products with a consistent style and user experience. Teams often reinvent the wheel when new visual components are required. Sometimes there is reuse of previous material but, as there is no central repository, that material might already be outdated or even classified as 'bad examples'.

Design Systems define a **collection of design patterns, component libraries and good practices that ensure consistent digital products**. Very much like a set of instructions and a Lego kit for everyone.

From an architectural perspective, design systems clearly provide a layer of abstraction enabling a composable architecture within and across applications.

Based on the corporate style guides of the past, design systems offer shared libraries and documents that are easy to find and use. Such guidance is written down in code and kept under version control to make it less ambiguous and easier to maintain than simple documents.

Design Systems allow product development teams to focus. They can address strategic challenges around the product itself without reinventing the wheel every time a new visual component is needed.

Design systems bring the following key benefits:

- Faster time to market, thanks to structural reuse of a managed set of UX resources
- Reduced cost of software development (less rework, decrease maintenance work, less alignment discussions required across teams)
- Improved brand value and customer adoption, thanks to a consistent and recognizable user experience

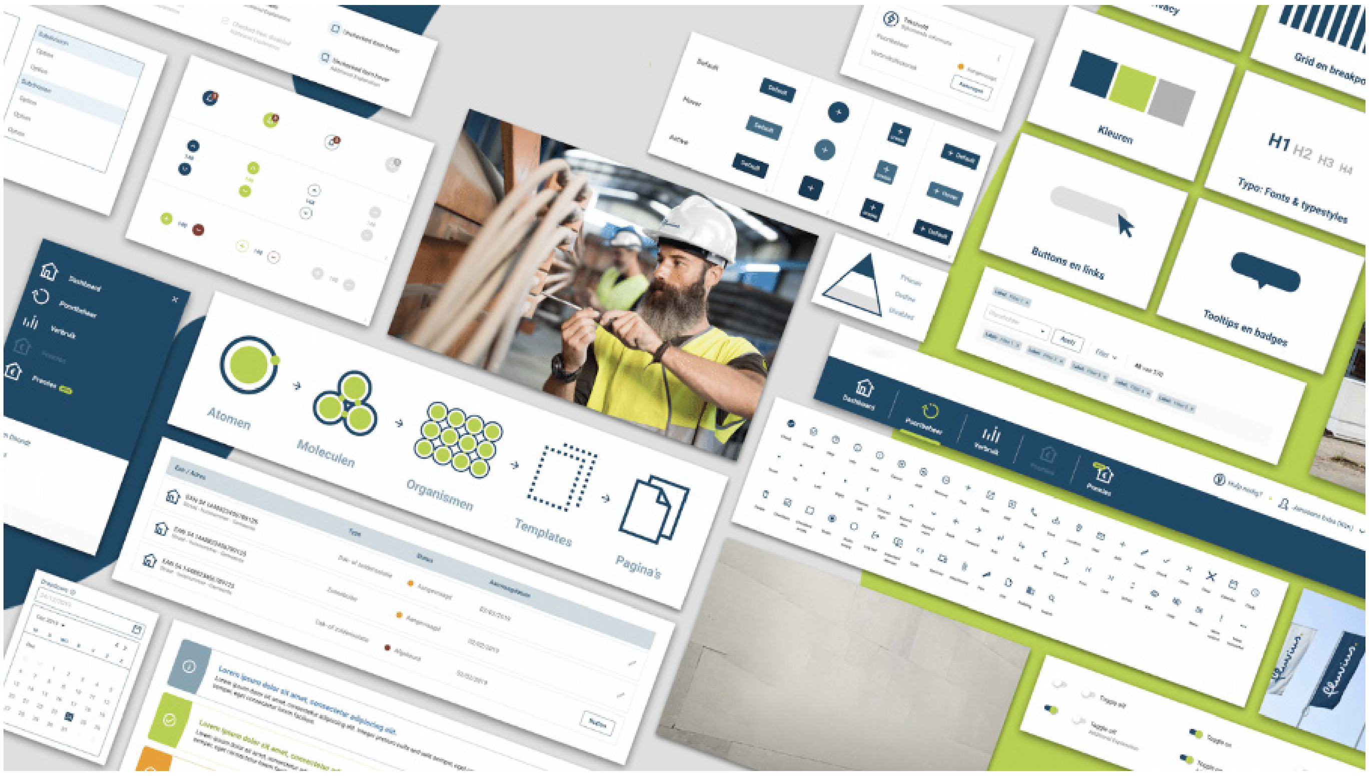
Design Systems in action:

From Minimum Viable Product (MVP) design system to a trusted and embedded partner

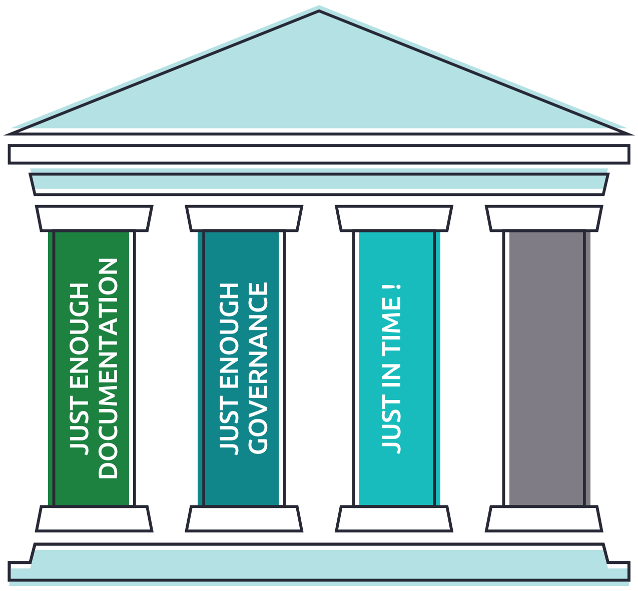
A Belgium grid operator for electricity and natural gas, needed to consolidate their design efforts, Capgemini proposed an MVP approach to create a new design system. This MVP firstly focused on auditing all the current components, consolidating, and defining new components in the library and redesigning the client portal & public website to be in line with the new foundation.

The work was so successful that the new design language has grown and been extended to all of company's digital touchpoints with the help of Capgemini's embedded design team that has been there since the implementation of the MVP. Capgemini also played a more strategic role in the design system, defining the companies' design principles and governance processes.

The design system brought speed and consistency for the new brand in the energy market.



SAMPLE 10: TECHNOLOGY RADAR FOR YOUR COMPANY OR DIVISION



Scope: Technology Market Watch

Sector: Water Sector

Tools: Open Source

An architect in an agile environment is expected to leverage relevant technologies in the right way, at the right moment and just in time. As technology evolves faster than ever, this poses several challenges:

- How to get inspired, bring the outside view and decide on which technology to assess, experiment with, or even adopt within a company or division?
- How to share experiences and communicate about decisions on technologies within the company?

A Technology Radar offers a platform to provide an answer to these questions:

- Producers of the radar: Creating a view and aligning with technologies that are relevant to the company requires teams to closely follow the industry trends and have good content discussions on the value, risks, and relevance of these technologies for the company.
- Consumers of the radar: A Technology Radar offers a centralized view for teams to indicate the evolution, status and potential technology that can be used as part of the IT solutioning.

In a Technology Radar, you typically have four rings:

- Adopt: The technology is mature and ready to be used by all teams in a company
- Trial: The technology is being used as part of a PoC or pilot
- Assess: The technology is being assessed for its relevance in the company
- Hold: The technology is not to be used in the company

The Radar itself can exist in any form, even a simple spreadsheet, and a good practice is to make it visual via a radial diagram. An open-source example is available here [Technology Radar OSS Sample].

We also recommend combining Technology Radar with Architectural Decision Records by logging the reasoning behind putting certain technologies in certain rings.

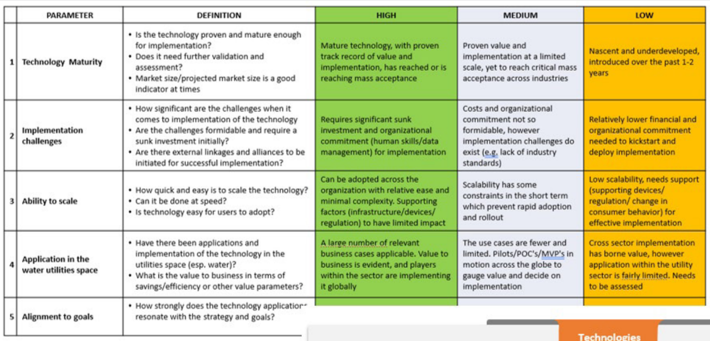
A Technology Radar brings the following key benefits:

- Allows improving decision making on the IT spend, anticipates and enables the adoption of new or evolving technologies
- Performs risk management related to technology disruption
- Can be used as a governance and communication tool to show what are the technologies being considered. It also indicates the readiness of new technologies to be part of the IT solutioning

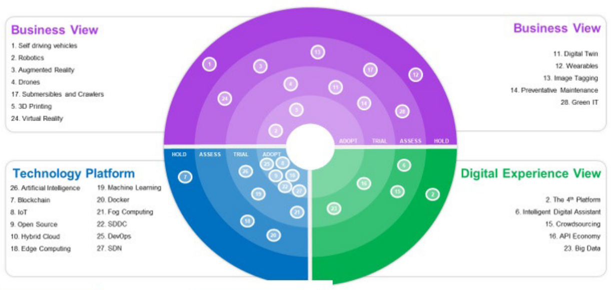
Technology Radar in action:

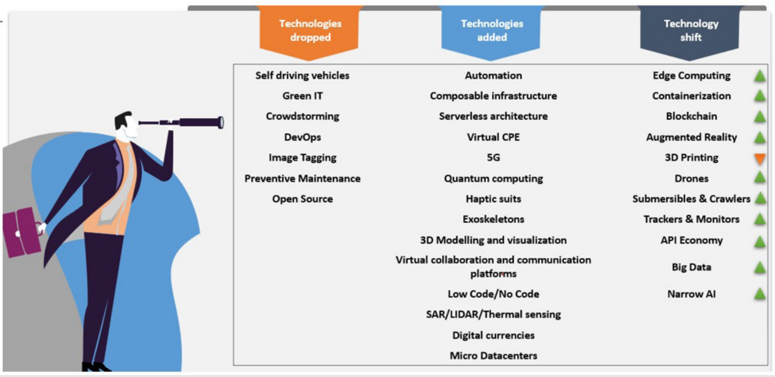
Since 2018, Capgemini has built a Technology Radar for a European water company. During this exercise, Capgemini conducts primary and secondary research on lots of technologies and options across different technology horizons. Once validated by Capgemini SMEs, the report is presented to key client stakeholders from innovation, sustainability, customer services, digital strategy, and field services teams.

The report allows the client to improve their decision making on the IT spend, increase their awareness of other operating units in the technology landscape and provide early identification of technologies, technological trends, and shocks. This Technology Radar is a living tool that, for this client, is refreshed twice a year.



	PARAMETER	DEFINITION	HIGH	MEDIUM	LOW
1	Technology Maturity	• Is the technology proven and mature enough for implementation? • Does it need further validation and assessment? • Market size/projected market size is a good indicator at times	Mature technology, with proven track record of value and implementation, has reached or is reaching mass acceptance	Proven value and implementation at a limited scale, yet to reach critical mass acceptance across industries	Nascent and underdeveloped, introduced over the past 1-2 years
2	Implementation challenges	• How significant are the challenges when it comes to implementation of the technology • Are the challenges formidable and require a sunk investment initially? • Are there external linkages and alliances to be initiated for successful implementation?	Requires significant sunk investment and organizational commitment (human skills/data management) for implementation	Costs and organizational commitment not so formidable, however implementation challenges do exist (e.g. lack of industry standards)	Relatively lower financial and organizational commitment needed to kickstart and deploy implementation
3	Ability to scale	• How quick and easy is to scale the technology? • Can it be done at speed? • Is technology easy for users to adopt?	Can be adopted across the organization with relative ease and minimal complexity. Supporting factors (infrastructure/devices/ regulation) to have limited impact	Scalability has some constraints in the short term which prevent rapid adoption and rollout	Low scalability, needs support (supporting devices/ regulation/ change in consumer behavior) for effective implementation
4	Application in the water utilities space	• Have there been applications and implementation of the technology in the utilities space (esp. water)? • What is the value to business in terms of savings/efficiency or other value parameters?	A large number of relevant business cases applicable. Value to business is evident, and players within the sector are implementing it globally	The use cases are fewer and limited. Pilots/POC's/MVP's in motion across the globe to gauge value and decide on implementation	Cross sector implementation has borne value, however application within the utility sector is fairly limited. Needs to be assessed
5	Alignment to goals	• How strongly does the technology application resonate with the strategy and goals?			





Technologies dropped	Technologies added	Technology shift	
Self driving vehicles	Automation	Edge Computing	▲
Green IT	Composable infrastructure	Containerization	▲
Crowdstorming	Serverless architecture	Blockchain	▲
DevOps	Virtual CPE	Augmented Reality	▲
Image Tagging	5G	3D Printing	▼
Preventive Maintenance	Quantum computing	Drones	▲
Open Source	Haptic suits	Submersibles & Crawlers	▲
	Exoskeletons	Trackers & Monitors	▲
	3D Modelling and visualization	API Economy	▲
	Virtual collaboration and communication platforms	Big Data	▲
	Low Code/No Code	Narrow AI	▲
	SAR/LIDAR/Thermal sensing		
	Digital currencies		
	Micro Datacenters		

CONCLUDING REMARKS ON JIT-JEA IN ACTION

In this second Agile Architecture Capgemini Point of View, we have detailed 10 pragmatic techniques and tools to practice JIT-JEA. They have been selected based on the value they brought to real engagements. We have described these techniques and tools and were able to share their real-life and successful implementation in different contexts by different team topologies.

This proves that practicing IT Architecture in an agile way and applying the JIT-JEA concepts is not only possible but brings outstanding value to teams and their broader organization. However, we recognize that Agile Architecture is in its early stages. Through this document, we were encouraged to find the trailblazers who have adopted this architecture for long enough to provide the insights presented here.

This is another stepping stone towards democratizing Agile Architecture and JIT-JEA. The journey has just begun! We intend to continue publishing, combining both the conceptual view provided by our first POV, and these concrete, real-life experiences. In the meantime, if you would like to learn more about JIT-JEA or the samples from this report, don't hesitate to reach out to us.

BIBLIOGRAPHY

[JIT-JEA part.1] Capgemini Agile Architecture Point of View: JIT-JEA Core Concepts
https://www.capgemini.com/wp-content/uploads/2022/01/AGILE-IT-ARCHITECTURE_PoV.pdf

[SAFe Enablers] https://www.scaledagileframework.com/enablers

[Documentation as Code] https://www.writethedocs.org/guide/docs-as-code/

[Markdown Language] https://markdown.github.io/

[Gollum] https://github.com/gollum/gollum

[Technology Radar OSS Sample] https://github.com/thoughtworks/build-your-own-radar

[LADR Example]
https://github.com/peter-evans/lightweight-architecture-decision-records/blob/master/0001-ladr-template.md

[Walking Skeleton] https://web.archive.org/web/20140329201356/http://alistair.cockburn.us/Walking+skeleton

[O-AA Guardrails] https://pubs.opengroup.org/architecture/o-aa-standard-single/#KLP-CAR-guardrails

[PRODUCTION LINE CAPGEMINI] https://www.youtube.com/watch?v=X_7qnEPsR4s

[O-AA SBCE]
https://pubs.opengroup.org/architecture/o-aa-standard-single/#_set_based_concurrent_engineering_sbce

[SAFe Principle 3] https://www.scaledagileframework.com/assume-variability-preserve-options/

[SAFe Architectural Runway] https://www.scaledagileframework.com/architectural-runway/

[Hexagonal architecture] https://alistair.cockburn.us/hexagonal-architecture/

[Clean Code] https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html

AUTHORS

The authors of this Agile Architecture paper are:

STEFANO ROSSINI

Italy BU Indus 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.

GERT HELSEN

Chief Architect in Capgemini

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 & Innovation. Led by passion, Pascal is a trainer and mentor within Capgemini Global Architects Community. Leader of the Architects Community of his practice, he is also part of the Core Team leading the 1500+ Architects of Capgemini France. Besides Architecture, Pascal's highest involvement relates to Sustainability.

ALESSANDRO PIRONI

Senior Software Architect and Agile Coach

Alessandro is an active member of the Italian Architect Community and is co-founder and co-leader of the Italy Agile Hub. Member of the Transform and Perform Office in Italy, he helps Italian engagements applying Agile methodologies and architectural principles to their day-by-day delivery.

JEAN PHILIPPE DEFRANCE

Vice President Edge Strategy, GTM & Operations, Capgemini Senior Architect

Over the last 15 years, Jean-Philippe combined his expertise in architecture, design thinking and agile to lead successfully large IT transformations. He is now developing the Group Edge Computing business.

SEBASTIAN SCHNELKER

Managing Architect and Agile Leader

Sebastian is an enthusiast of modern software development approaches (agile, lean, UX) and like to share these best-of-breed practices to provoke a change of our way of working and thinking. Within Capgemini Sebastian is very engaged in a diverse set of communities and conduct internal trainings on various design & architecture topics.

LISA ECKERSLEY

Integration Architect, Capgemini Chief Architect

As an Integration specialist Chief Architect in Capgemini, for 25 years Lisa has been integral to many large digital transformation programs across multiple sectors. She mentors and trains others and is active on steering groups for Women in Capgemini.

MARIEN KROUWEL

Low Code Expert, Agile Solution Architect and Lead Trainer

Marien Krouwel is an expert in low code/no code and cloud services, helping customers accelerate digital transformation and innovation by implementing low-code/no-code platforms. He is an experienced agile solution architect and lead trainer at Capgemini Academy.

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 over 360,000 team members in more than 50 countries. With its strong 55-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, fueled by the fast evolving and innovative world of cloud, data, AI, connectivity, software, digital engineering and platforms. The Group reported in 2022 global revenues of €22 billion.

Get the Future You Want | www.capgemini.com

MACS-Agile_14-02-2023_Surojit