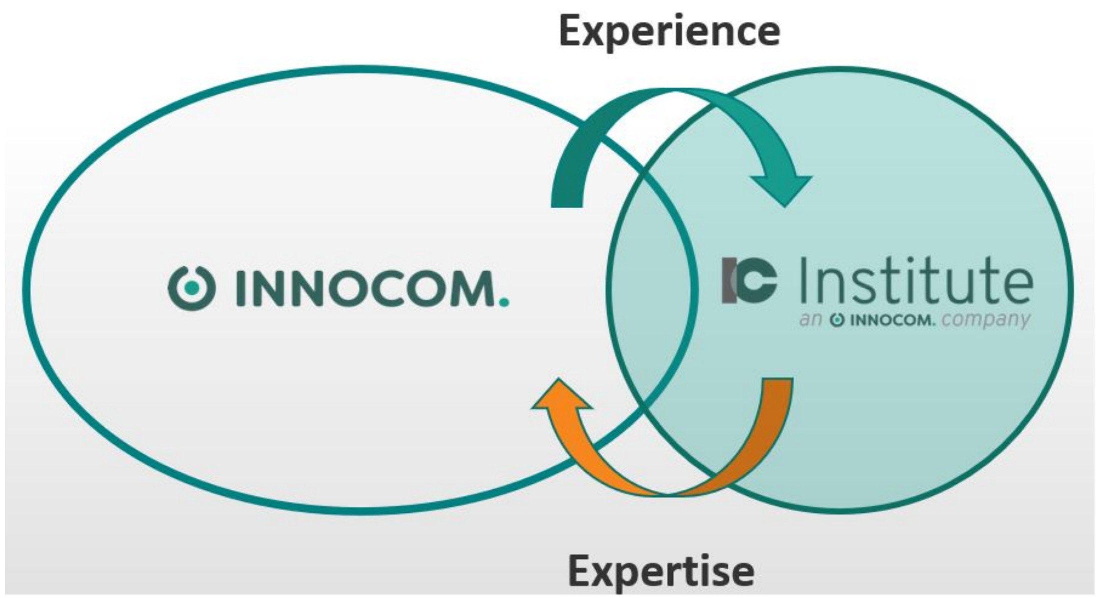
INNOCOM.

Modeling a Composable Architecture with EDGY

rudi.claes@inno.com

Bio



Rudi Claes
https://www.linkedin.com/in/rudi-claes-39a696b/
Rudi.Claes@inno.com

Background:

- 20+ years of experience in different Enterprise Architecture and Solution Architecture roles.
- Supporting large and medium sized organizations in their agile architecture journeys at all maturity levels.
- Guest lecturer at the IC institute on Solution Architecture and ArchiMate.

Talk outline

1 A short history of Composable Architecture

2 The link with Enterprise Design and EDGY

3 How this has delivered a solid proof-of-value in a real-world case

A short history of Composable Architecture

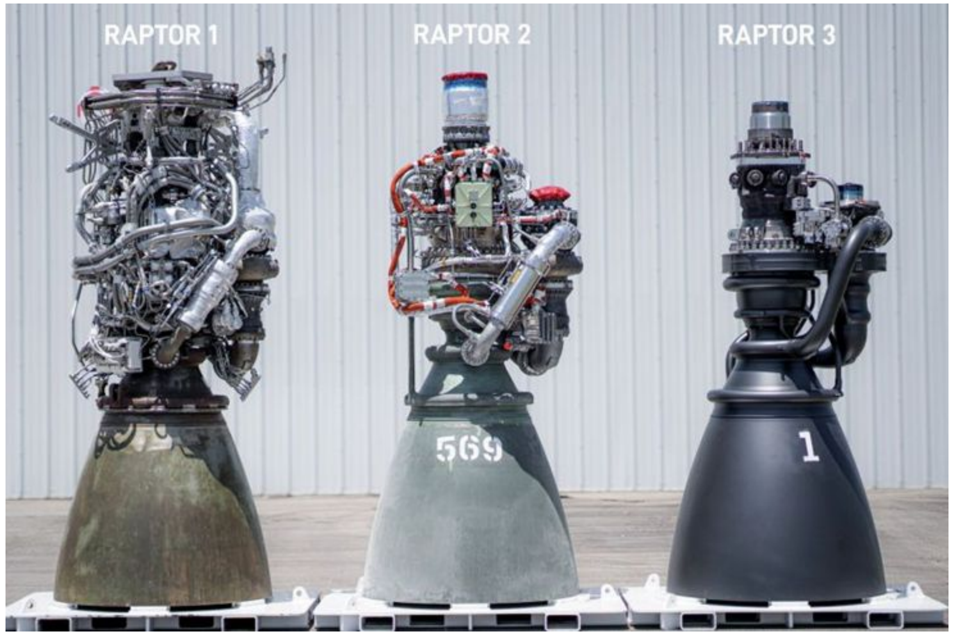
Modeling a Composable Architecture with EDGY

1. Composable Architecture is about what you build

The required agile characteristics of the system or architecture

- Managing complexity
- Allowing reusability
- Enabling evolvability

Easy



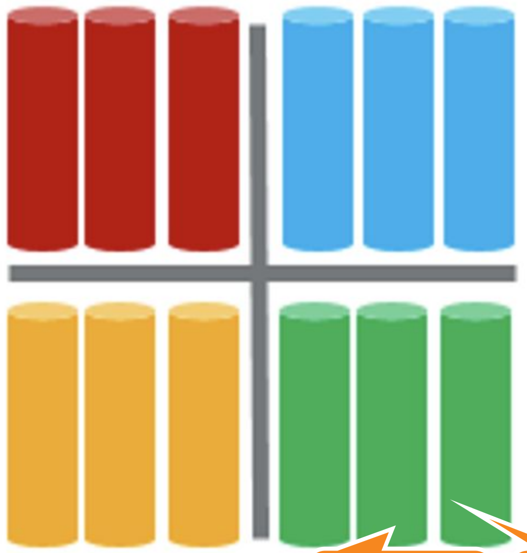
Difficult

All this while you have to keep the "engine" running in the process

Source: TCT Mag – SpaceX Raptor Engines

1. Composable Architecture is about what you build

- Managing complexity
- Allowing reusability
- Enabling evolvability



Shape ?

Size ?

Software is a dynamic system

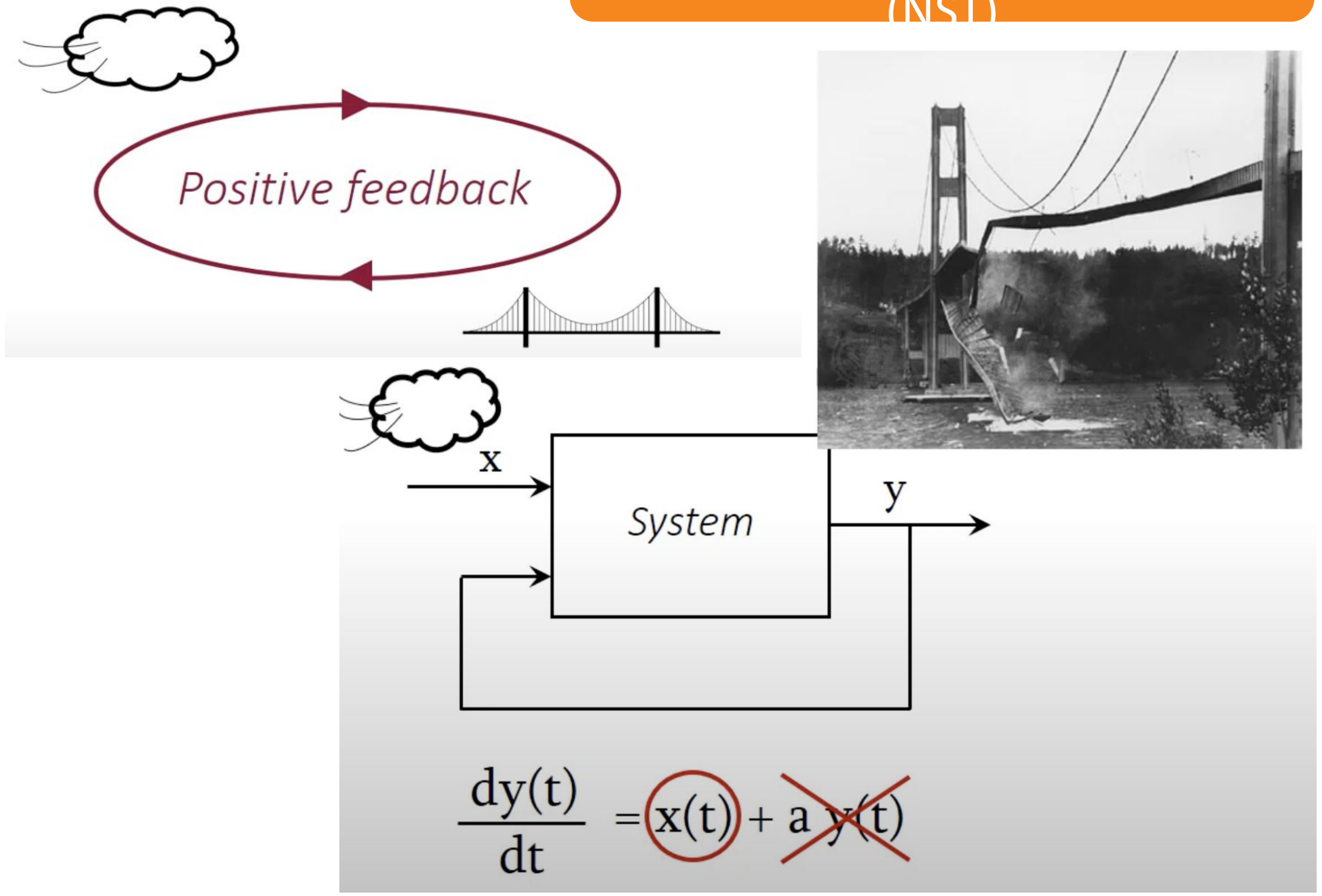
2. Composable Architecture should safeguard dynamic stability

Software is a dynamic system
Normalized Systems Theory (NST)

$a > 0 \longrightarrow a = 0$

$a > 0 \longrightarrow a = 0 \longrightarrow a > 0$

X = force of the wind
Y = deflection / energy of the bridge
a = 0 (no damping effect)



$$\frac{dy(t)}{dt} = \bigcirc{x(t)} + a\,\cancel{y(t)}$$

Collapse of the Tacoma Narrows Bridge (1940)

Source: NST Foundation Lecture 1 The Design Cycle as a Dynamic System

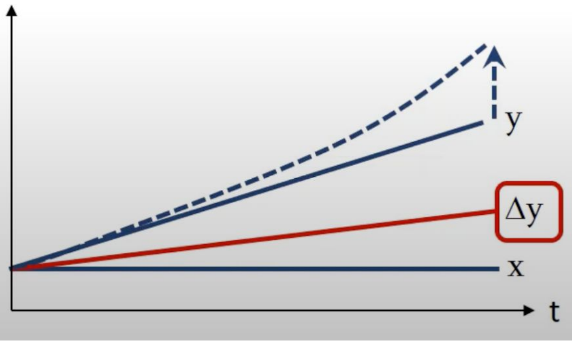
2. Composable Architecture should safeguard dynamic stability

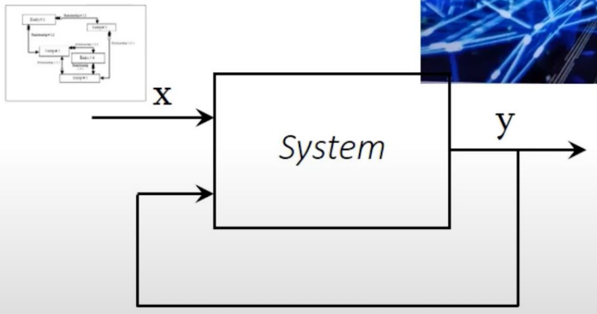
Software is a dynamic system
Normalized Systems Theory (NST)

X = requirements (new/changes)
Y = versions of software modules
a = change ripples (combinations of multiple changes)

a > 0
Principles of Normalized Systems Theory (NST)
Best practices & design patterns

- Dynamic instability
- Caused by positive feedback
  - between modular structure and change ripples
    → $\Delta y[k] \sim a y[k],\ a > 0$
  - with growing structure
    → $y[k+1] = y[k] + \Delta y[k] > y[k]$





$$\Delta y[k] = x[k] + a\,y[k]$$

Source: NST Foundation Lecture 1 The Design Cycle as a Dynamic System

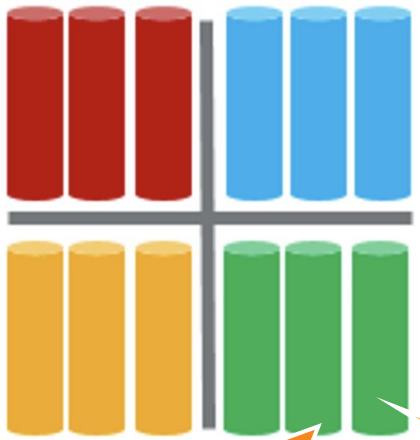
2. Composable Architecture should enable dynamic stability

Software is a dynamic system
Normalized Systems Theory (NST)

a > 0 → a = 0

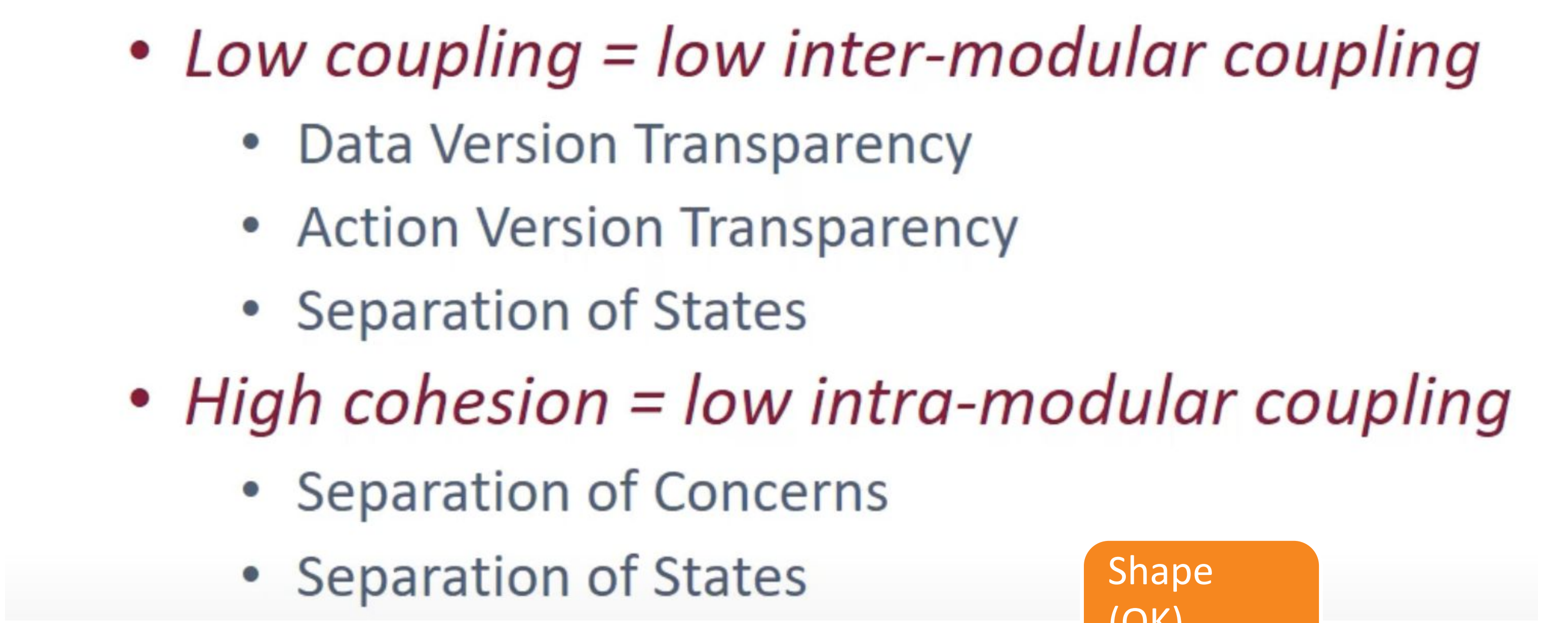
a > 0 → a = 0 → a > 0

a > 0
Principles of (NST)
Best practices & design patterns



Shape ?

Size ?

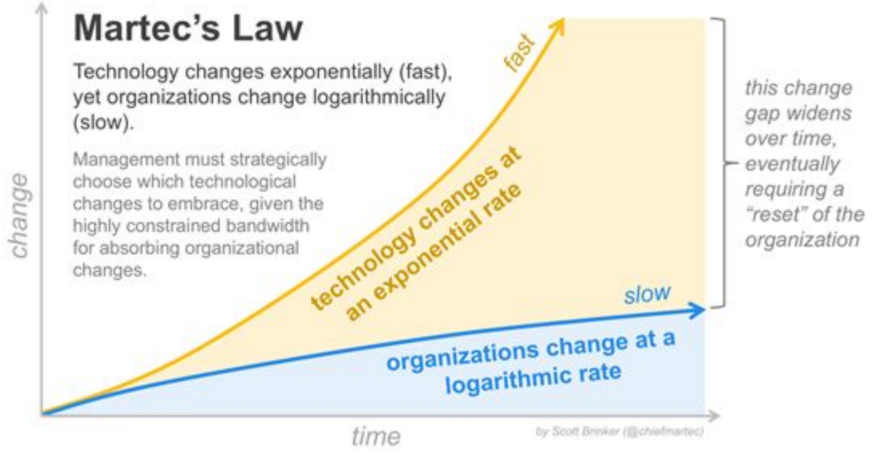


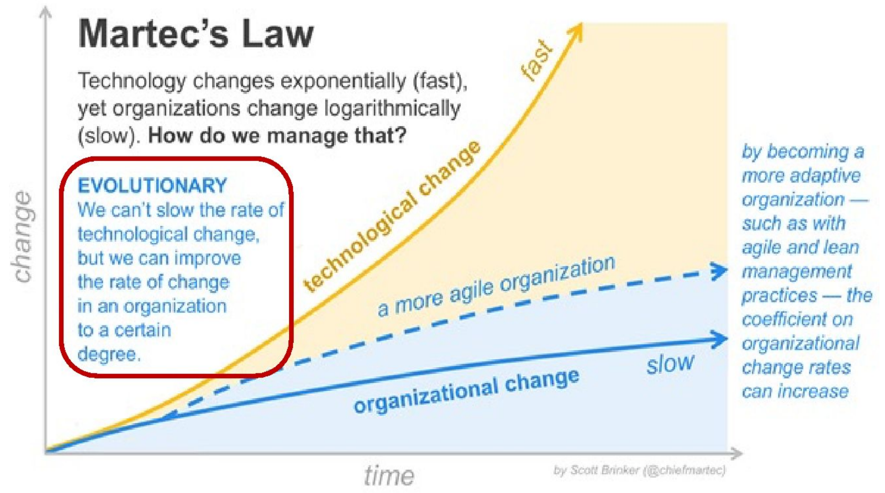
- *Low coupling = low inter-modular coupling*
  - Data Version Transparency
  - Action Version Transparency
  - Separation of States
- *High cohesion = low intra-modular coupling*
  - Separation of Concerns
  - Separation of States

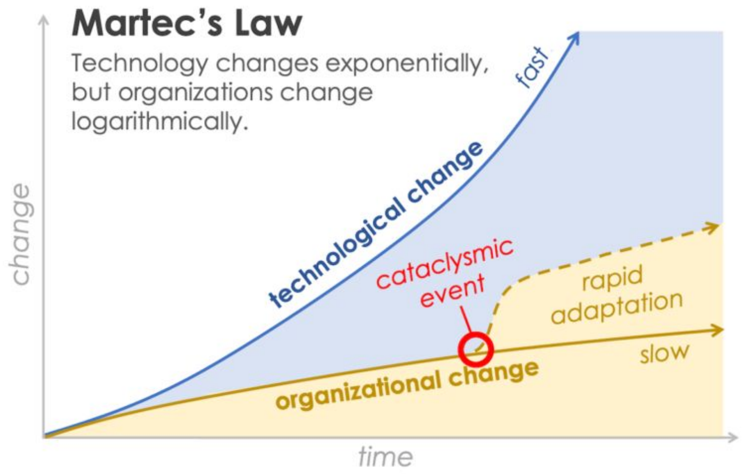
Shape (OK)
Size (?)

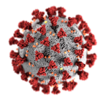
Source: NST Foundation Lecture 2 Design Theorems for Software Stability

3. Composable Architecture emerged in times of rapid adaptation

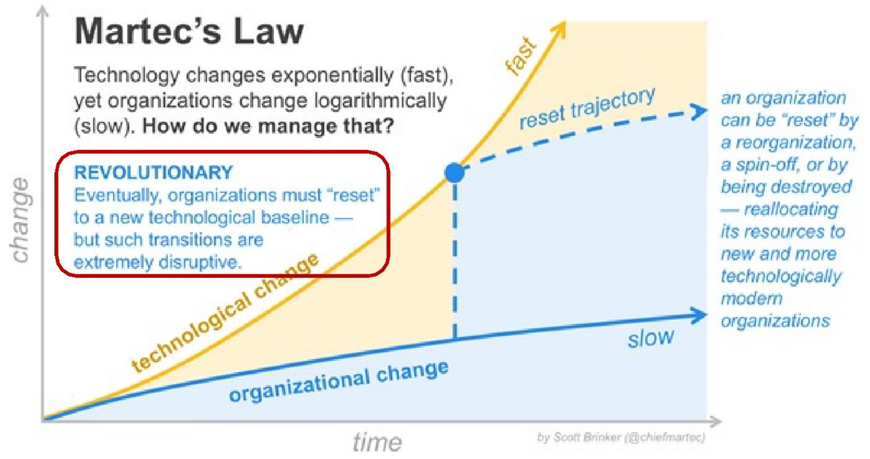






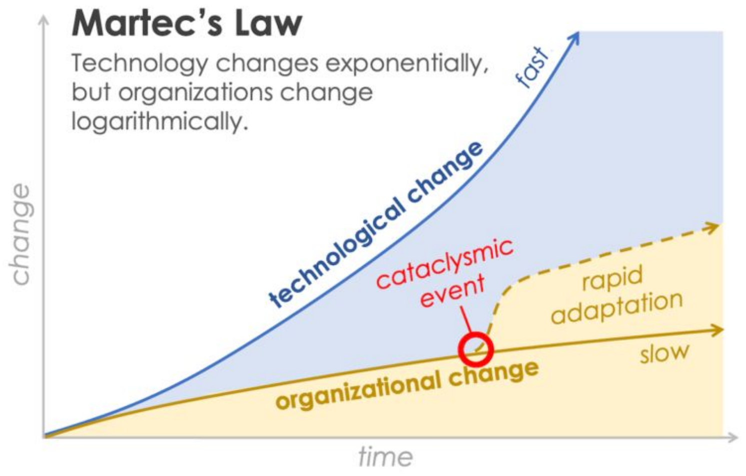


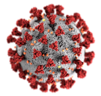
COVID-19



Source: Martec's Law

3. Composable Architecture emerged in times of rapid adaptation





COVID-19

Shape ?

Size ?

Huge change ripples

Very limited time to react
E-commerce tsunami

Source: https://chiefmartec.com

3. Composable Architecture as observed by Gartner during COVID

Packaged Business Capabilities (PBCs)

PBCs are encapsulated software components that represent a well-defined business capability, recognizable as such by a business user. Well-designed PBCs are:

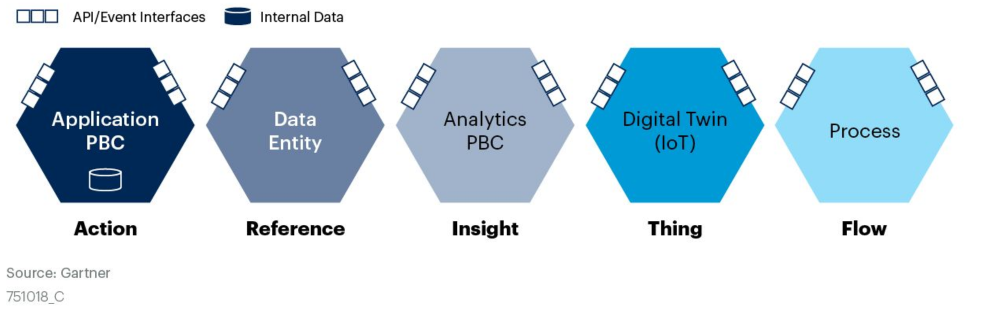
- **Modular:** Partitioned into a cohesive set of components.
- **Autonomous:** Self-sufficient and with minimal dependencies to ensure flexibility in composition.
- **Orchestrated:** Packaged for composition to assemble process flows or complex transactions through APIs, event interfaces or other technical means.
- **Discoverable:** Designed with semantic clarity and economy to be accessible to business and technical designers, developers and active applications.

Huge change ripples

x ⟶ X = a ⟶ A
Extra stabilization required!

Shape (OK)
Size (?)

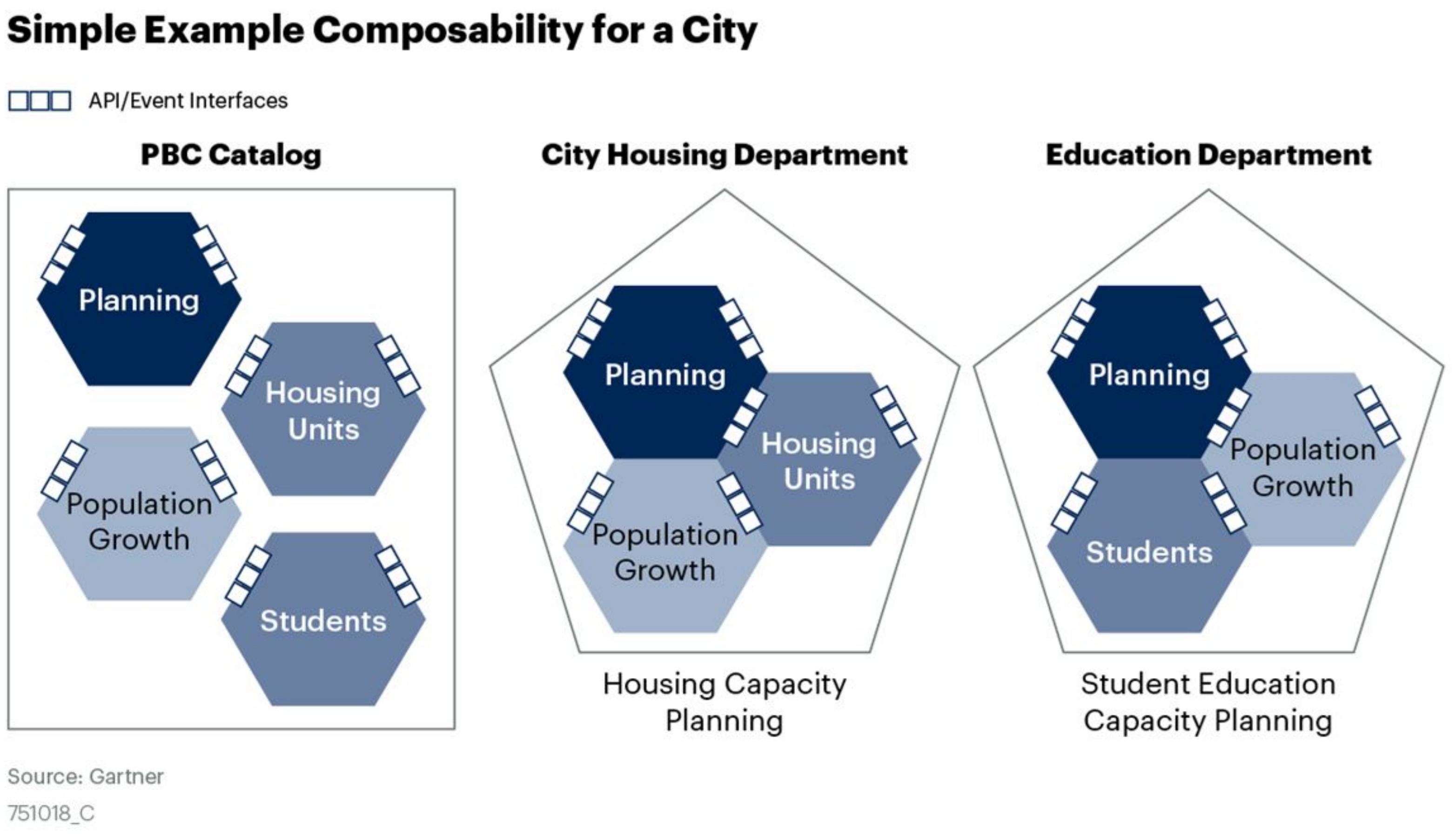
Example: PBC Types



Gartner

Source: Gartner

3. Composable Architecture as observed by Gartner during COVID



"The granularity of PBCs, as with all modular systems, is a common design challenge. Modular components that are too large may be easier to manage, but they are harder to change or use in new compositions. Components that are too small may be easier to assemble, but harder to isolate, identify, find or change."

Shape (OK)
Size (?)

Source: Gartner

4. Composable Architecture in Banking

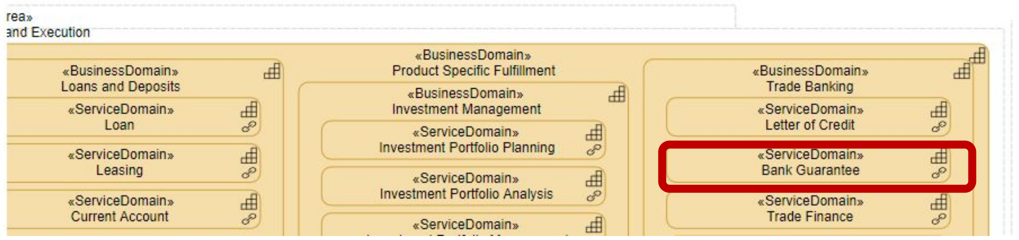
BIAN Service Landscape V10.0 Matrix View

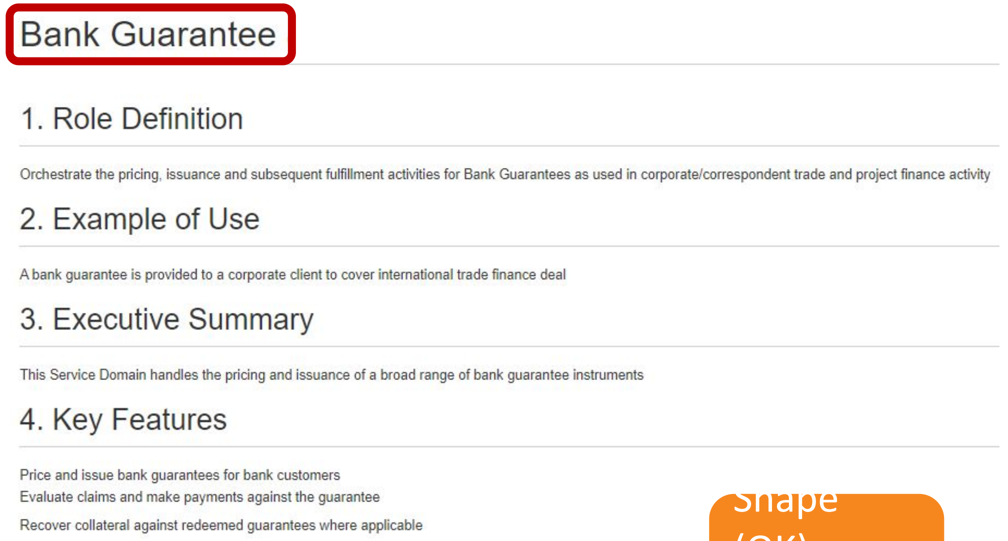
SERVICE DOMAIN

Service Domain	An elemental or atomic functional building block that can be service enabled as a discrete and unique business responsibility.	Service Domain

- The **Service Domain** is a core concept in the BIAN architecture – and standard.
- A BIAN Service Domain represents the smallest functional partition that can be service-enabled as a discrete and unique business responsibility.
- Service Domains are mutually-exclusive and collectively-exhaustive.
- A Service Domain offers its services (Service Operations) to other Service Domains. This allows Service Domains to fulfil their role by delegating the execution of functionality to other Service Domains.
- The interaction between the Service Domains can realize all the business activities that make up a bank.

Externalization


rea»
and Execution
«BusinessDomain» Product Specific Fulfillment
«BusinessDomain» Loans and Deposits
«ServiceDomain» Loan
«ServiceDomain» Leasing
«ServiceDomain» Current Account
«BusinessDomain» Investment Management
«ServiceDomain» Investment Portfolio Planning
«ServiceDomain» Investment Portfolio Analysis
«ServiceDomain»
«BusinessDomain» Trade Banking
«ServiceDomain» Letter of Credit
«ServiceDomain» Bank Guarantee
«ServiceDomain» Trade Finance


Bank Guarantee

1. Role Definition

Orchestrate the pricing, issuance and subsequent fulfillment activities for Bank Guarantees as used in corporate/correspondent trade and project finance activity

2. Example of Use

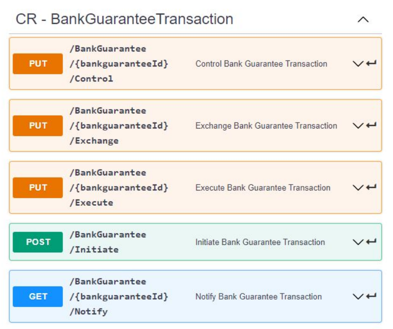
A bank guarantee is provided to a corporate client to cover international trade finance deal

3. Executive Summary

This Service Domain handles the pricing and issuance of a broad range of bank guarantee instruments

4. Key Features

Price and issue bank guarantees for bank customers
Evaluate claims and make payments against the guarantee
Recover collateral against redeemed guarantees where applicable


CR - BankGuaranteeTransaction

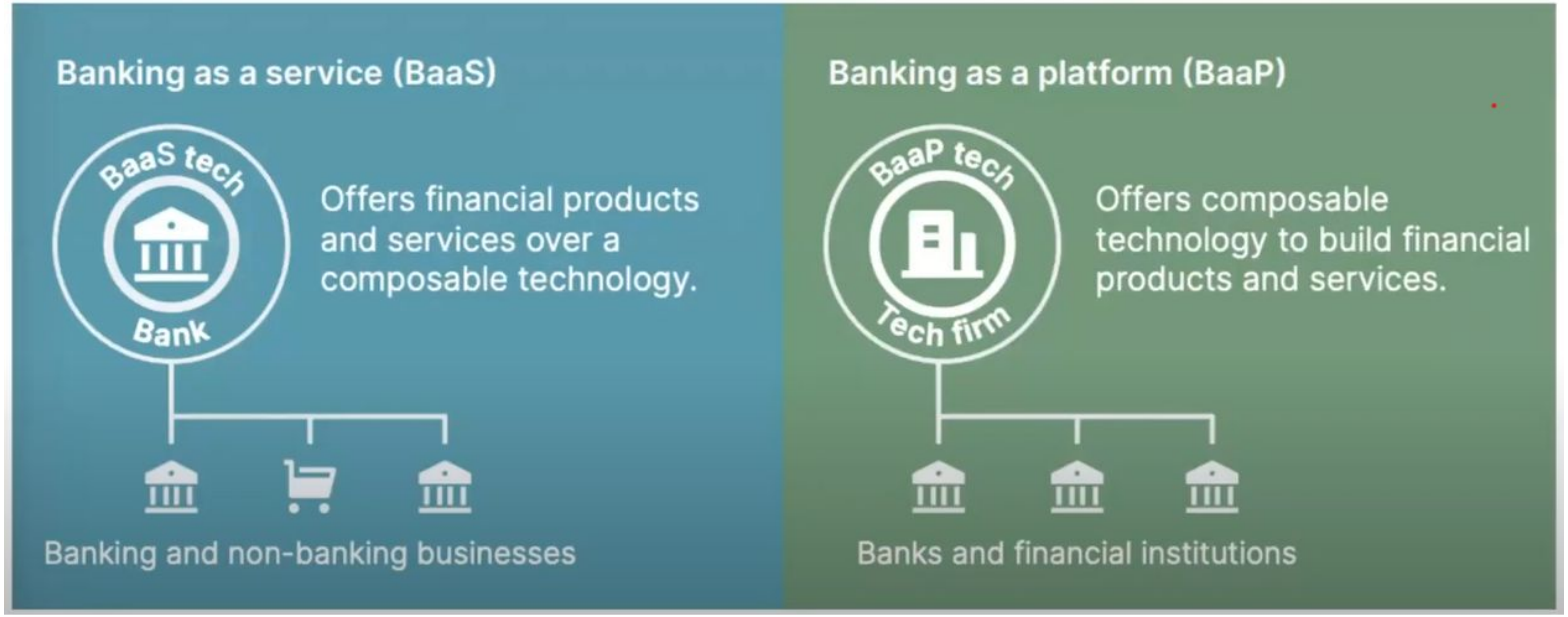
PUT	/BankGuarantee /{bankguaranteeId} /Control	Control Bank Guarantee Transaction
PUT	/BankGuarantee /{bankguaranteeId} /Exchange	Exchange Bank Guarantee Transaction
PUT	/BankGuarantee /{bankguaranteeId} /Execute	Execute Bank Guarantee Transaction
POST	/BankGuarantee /Initiate	Initiate Bank Guarantee Transaction
GET	/BankGuarantee /{bankguaranteeId} /Notify	Notify Bank Guarantee Transaction

BIAN
Banking Industry Architecture Network

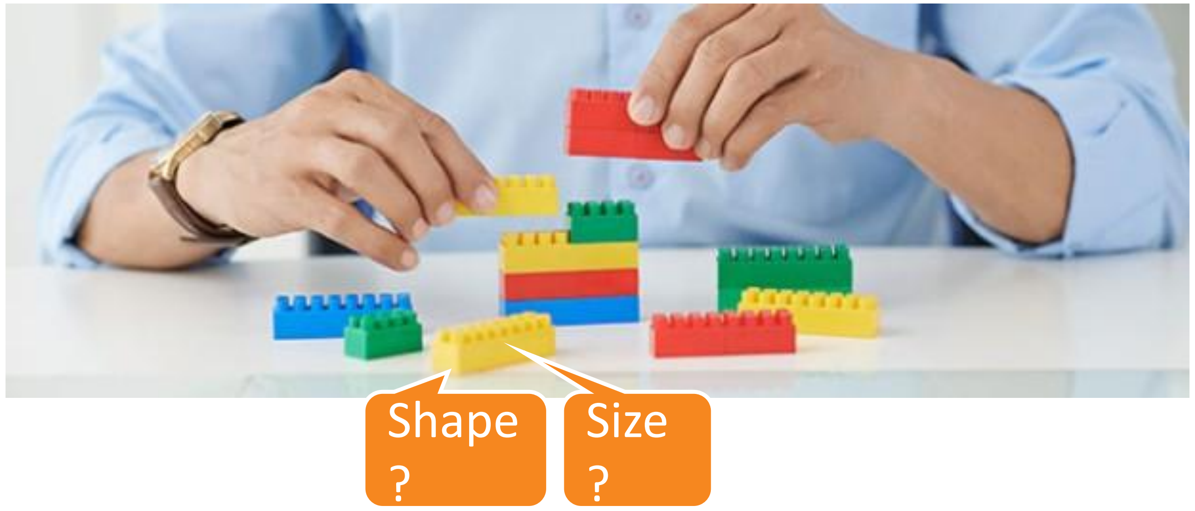
Shape
(OK)
Size
(OK)

Source: BIAN

4. Composable Architecture in Banking


Banking as a service (BaaS)
BaaS tech
Bank
Offers financial products and services over a composable technology.
Banking and non-banking businesses
Banking as a platform (BaaP)
BaaP tech
Tech firm
Offers composable technology to build financial products and services.
Banks and financial institutions

Source: Thoughtworks



The link with Enterprise Design and EDGY

Modeling a Composable Architecture with EDGY

Composable Architecture is the outcome of applying composability

Enterprise Architecture
How to define future state with composability in mind?

Solution Architecture
How to select fit for purpose solutions and/or solution building blocks?

Software Architecture
How to introduce, implement and evolve a composable architecture?

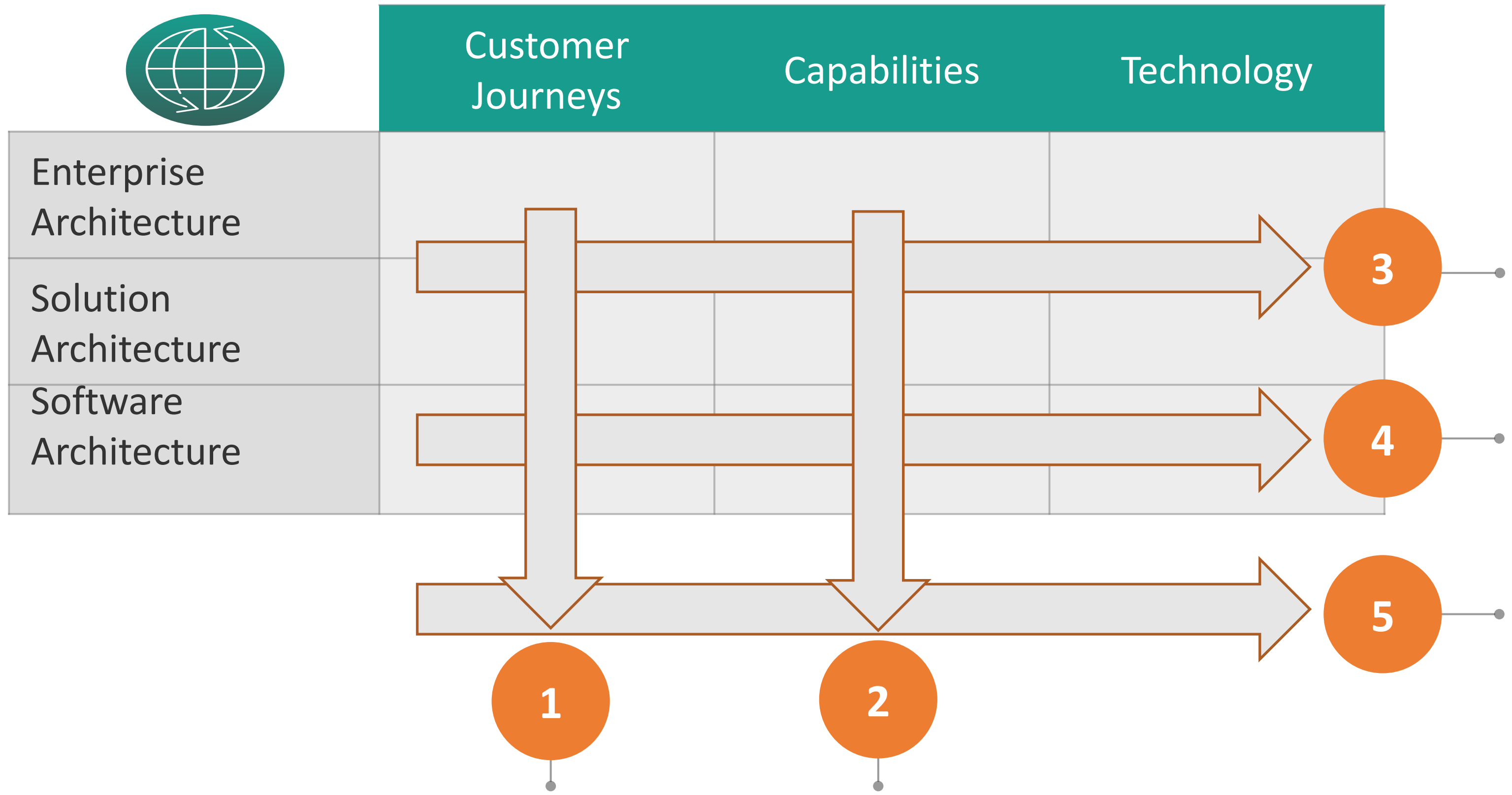
Customer Journeys
How to optimize user experience with composability in mind?

Capabilities

Packaged Business Capabilities (PBC)
How to evolve from current state to platforms and platform-based PBC's?

Technology

Composable Architecture is the outcome of applying composability



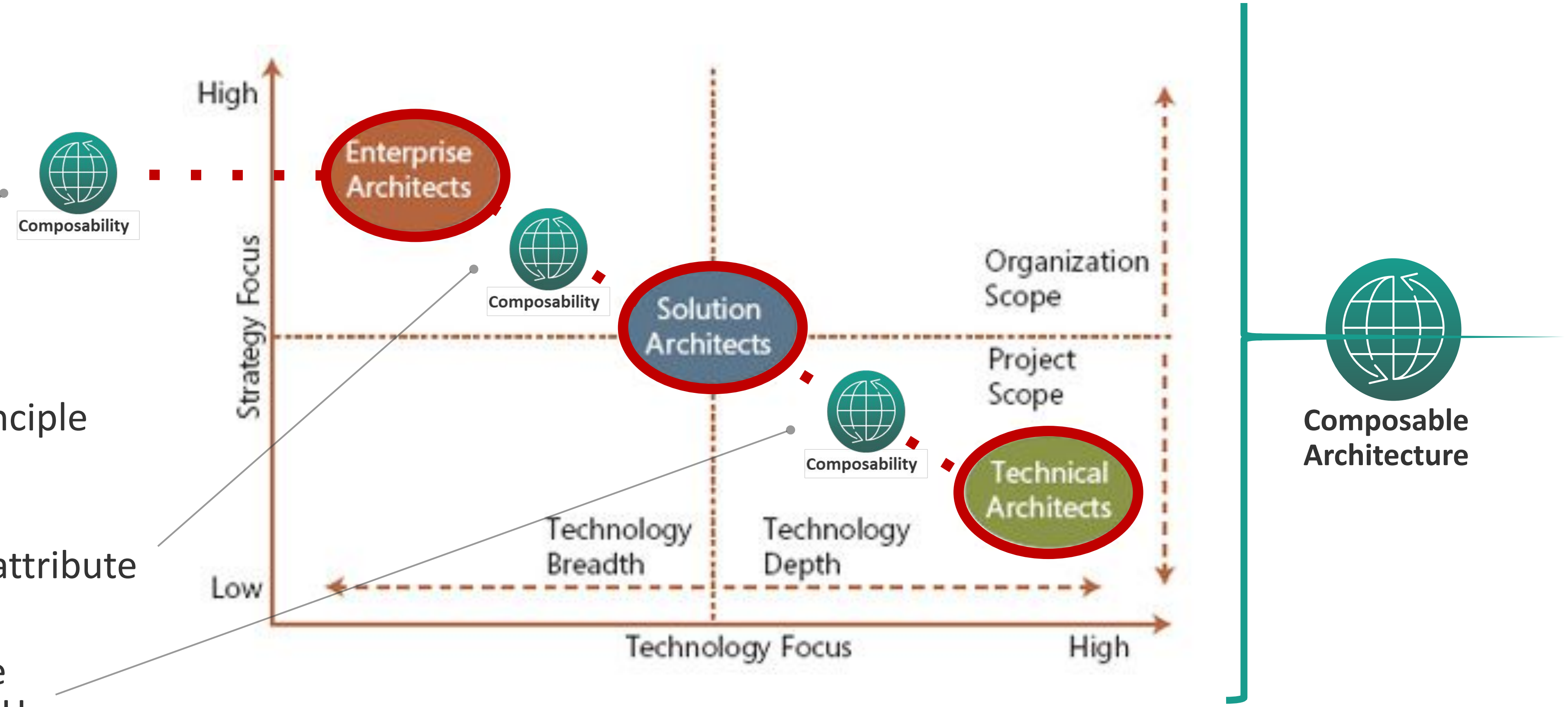
3. Define future state with composability as an EA principle
4. Solution architecting with composability as a quality attribute
5. Implement the composable architecture based on MACH
1. Define customer journeys via Enterprise Design
2. Shape PBC's based on BIAN Service Domains

Composable Architecture is the outcome of applying composability

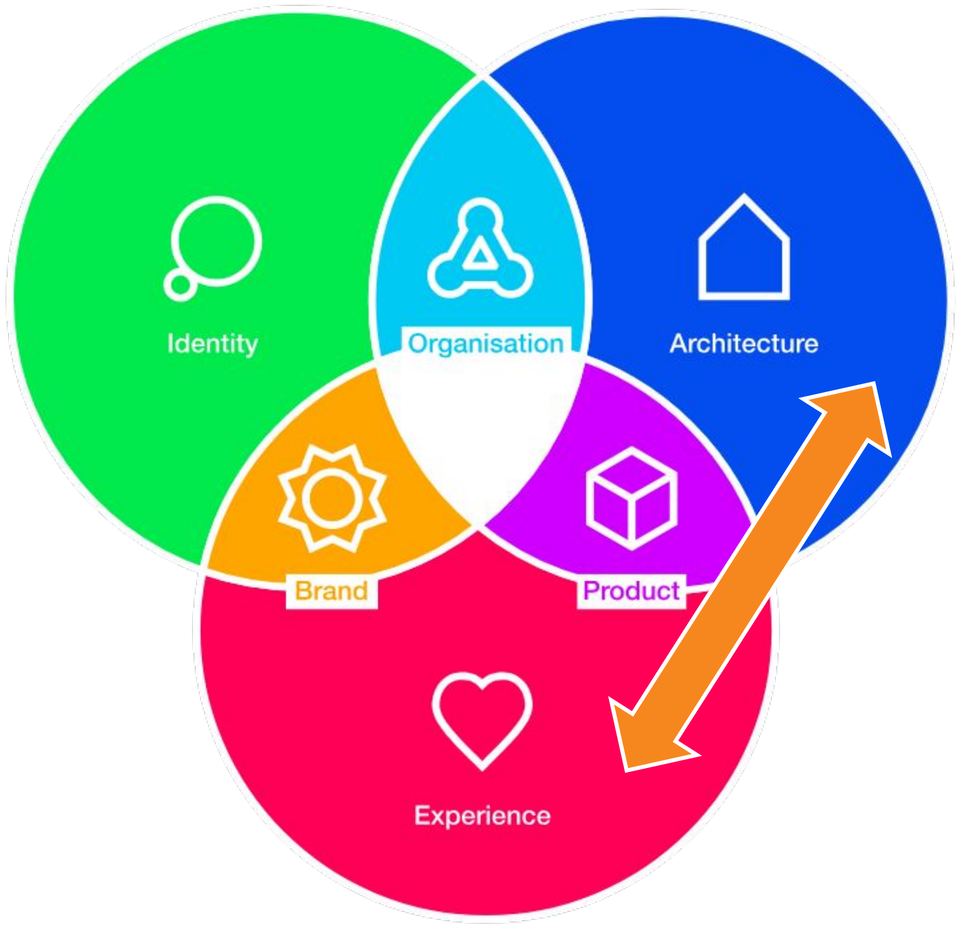
3 Define future state with composability as an EA principle

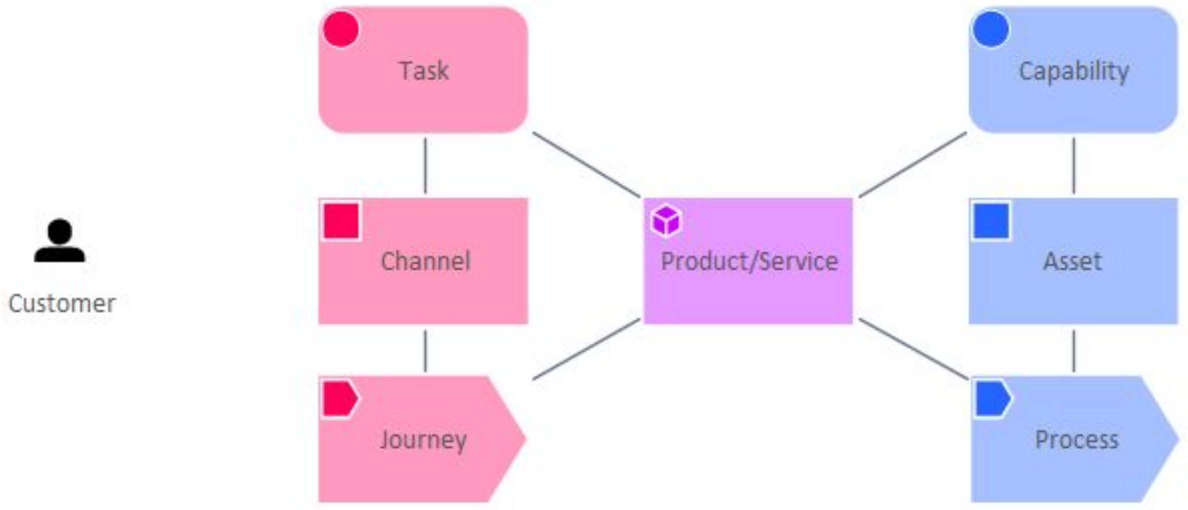
4 Solution architecting with composability as a quality attribute

5 Implement the composable architecture based on MACH



EDGY
Experience - Product/Service - Architecture

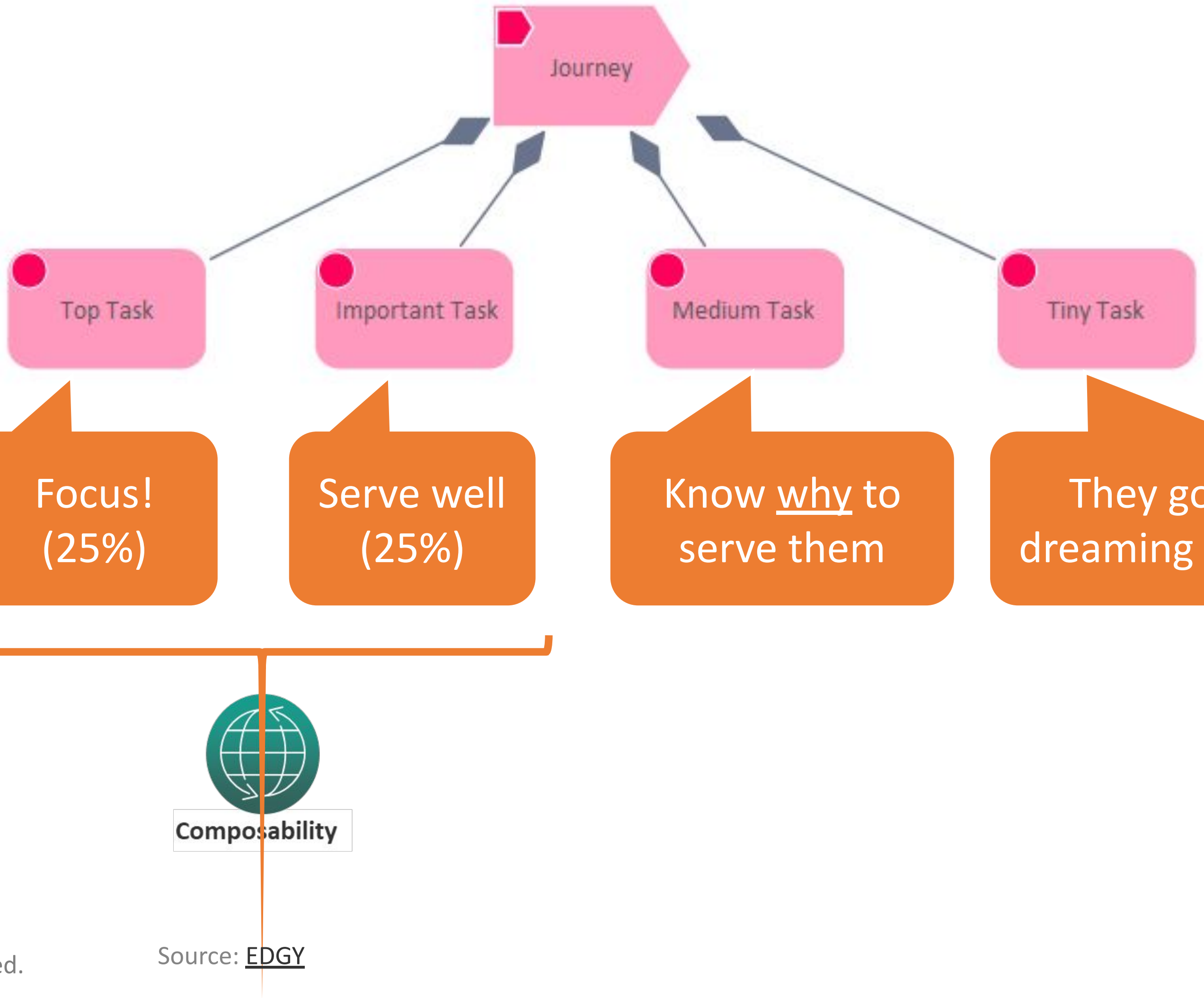




What does potentially have the most impact on composability?

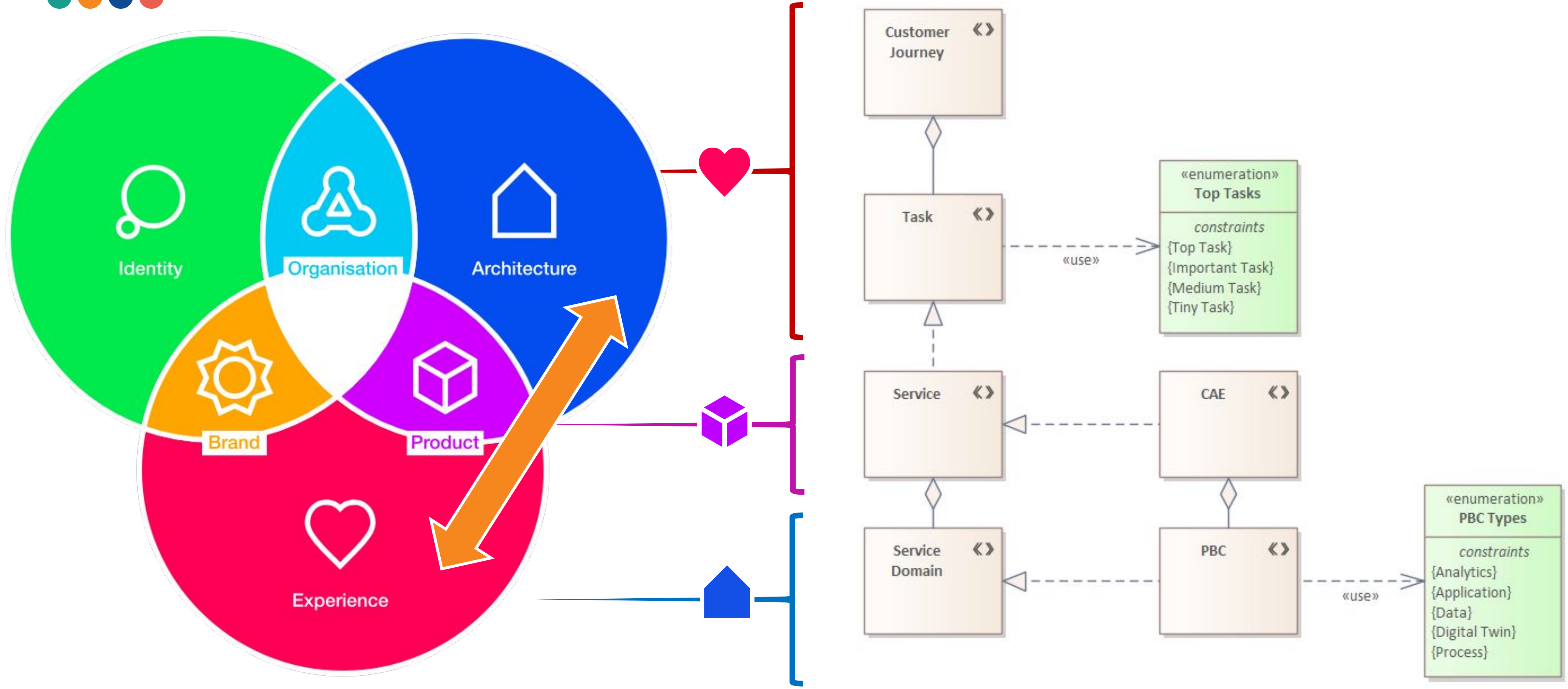
Source: EDGY

EDGY
Experience - Product/Service - Architecture



Source: EDGY

EDGY
Experience - Product/Service - Architecture



Source: EDGY

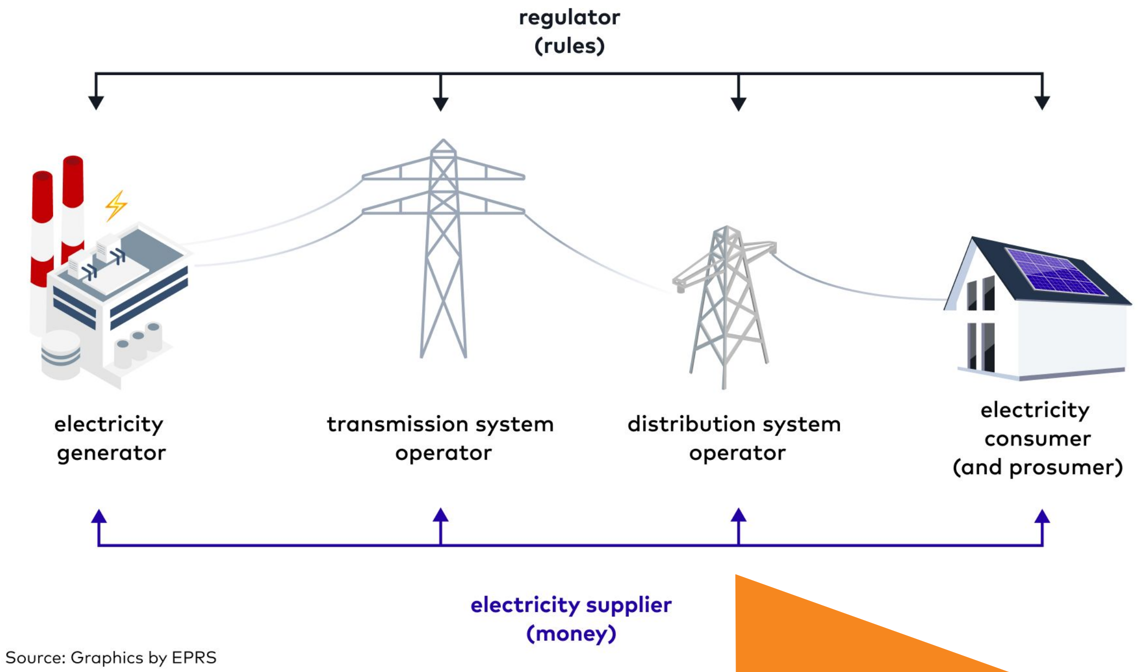
How this has delivered a solid proof-of-value in a real-world case

Modeling a Composable Architecture with EDGY

Case: Distribution System Operator (DSO) in Utility

Huge change ripples

x ⟶ X = a ⟶ A
Extra stabilization required!


regulator
(rules)
electricity generator
transmission system operator
distribution system operator
electricity consumer (and prosumer)
electricity supplier
(money)
Source: Graphics by EPRS

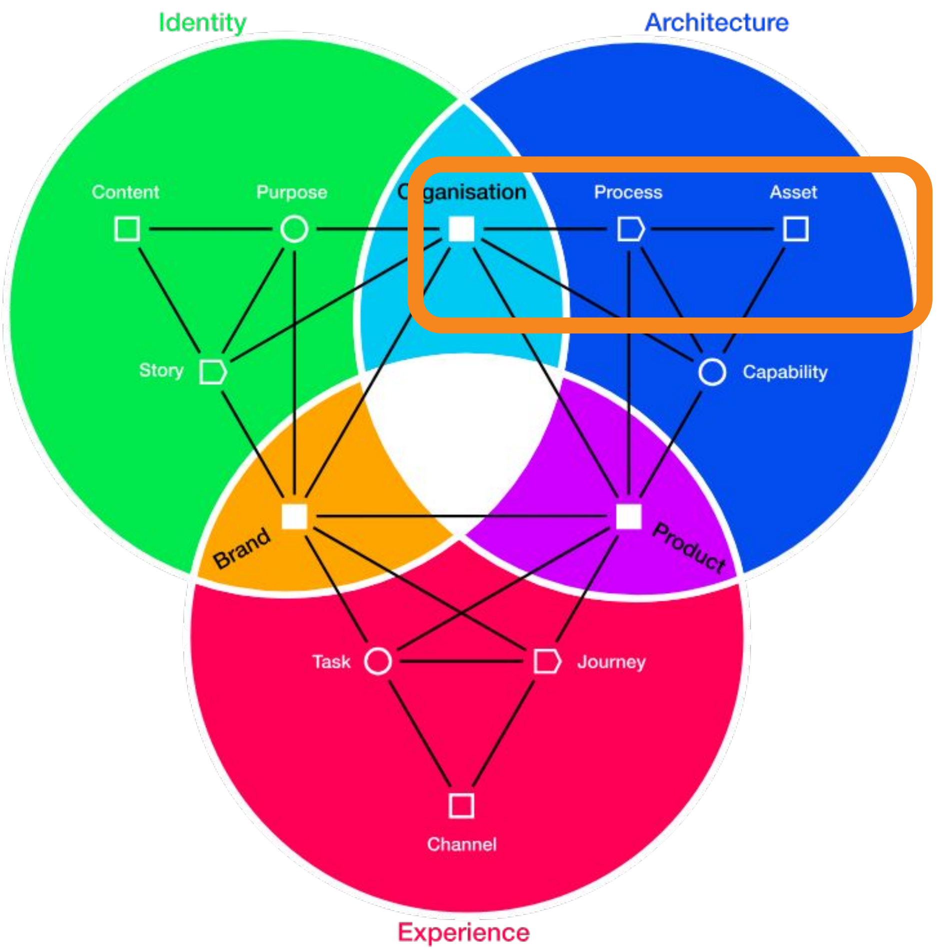
Questions:
1. How to get started with composable architecture?
2. How to identify platform-ready PBC's?
3. How to document & model PBC's ?

Source: EPRS

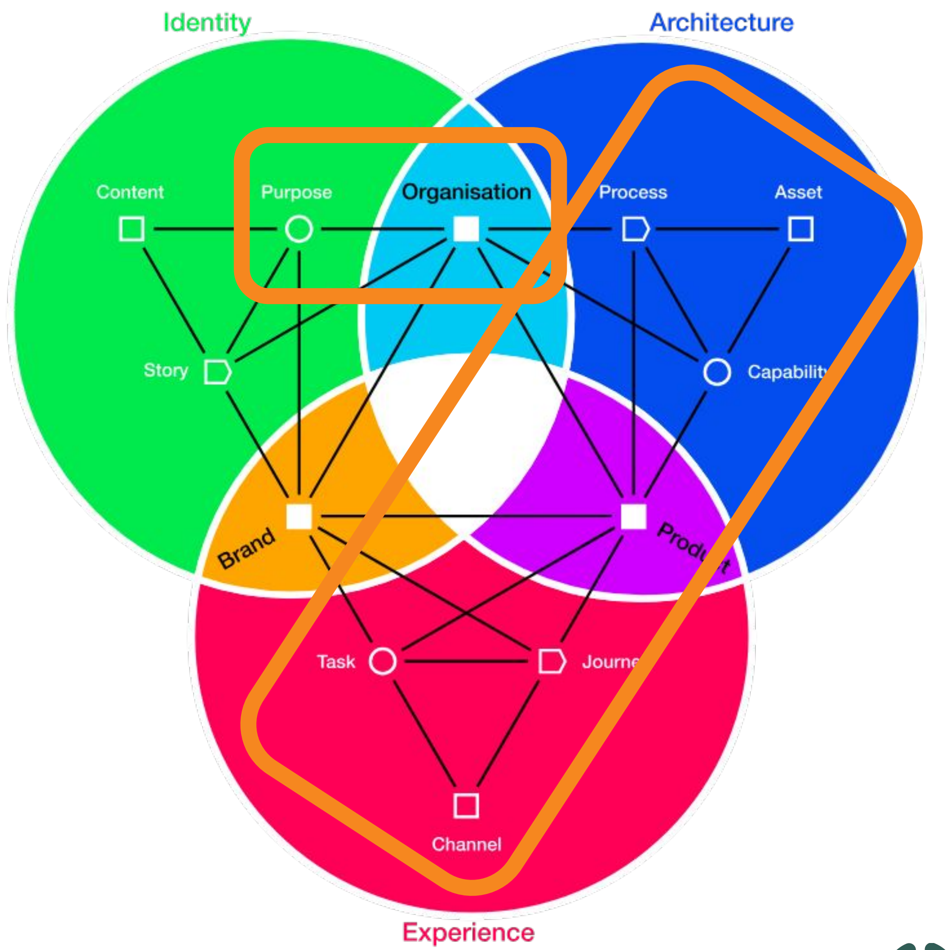
DSO Case Study

Questions:

1. How to get started with composable architecture?
   → Avoid silos and tight dependency between processes and applications



Reframing

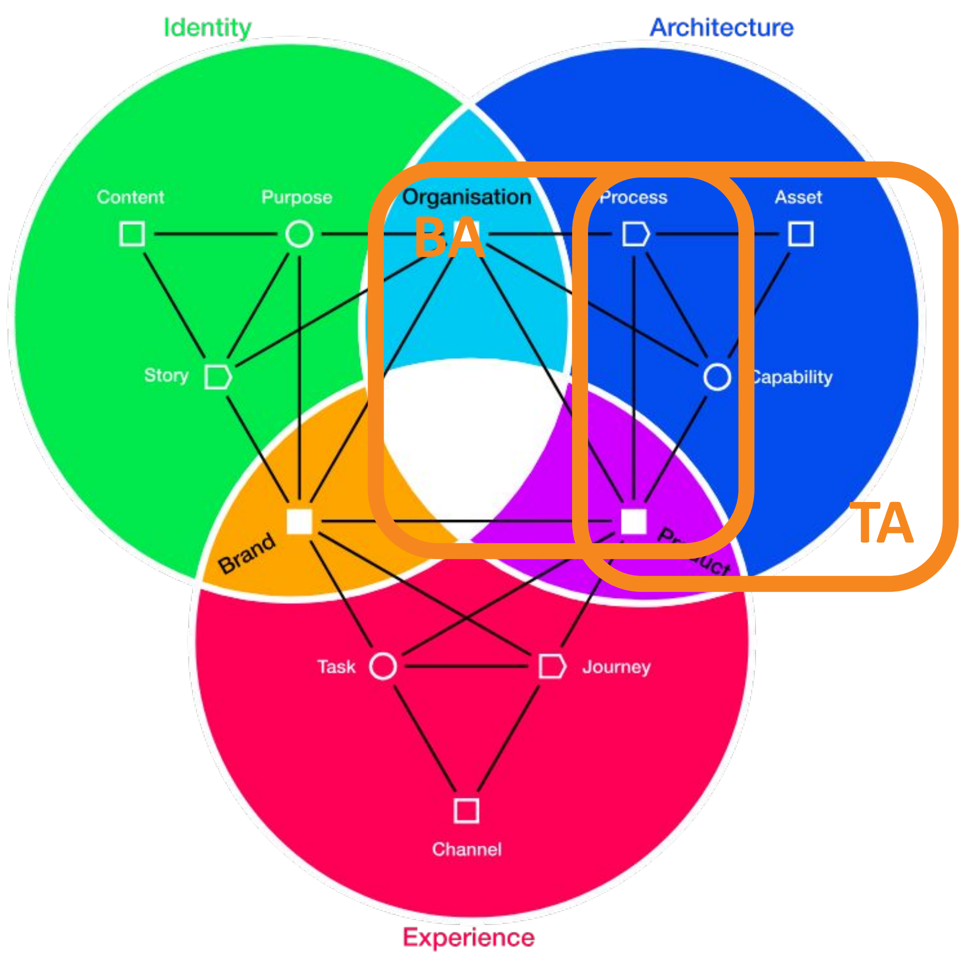


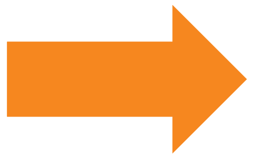
Source: EDGY

DSO Case Study

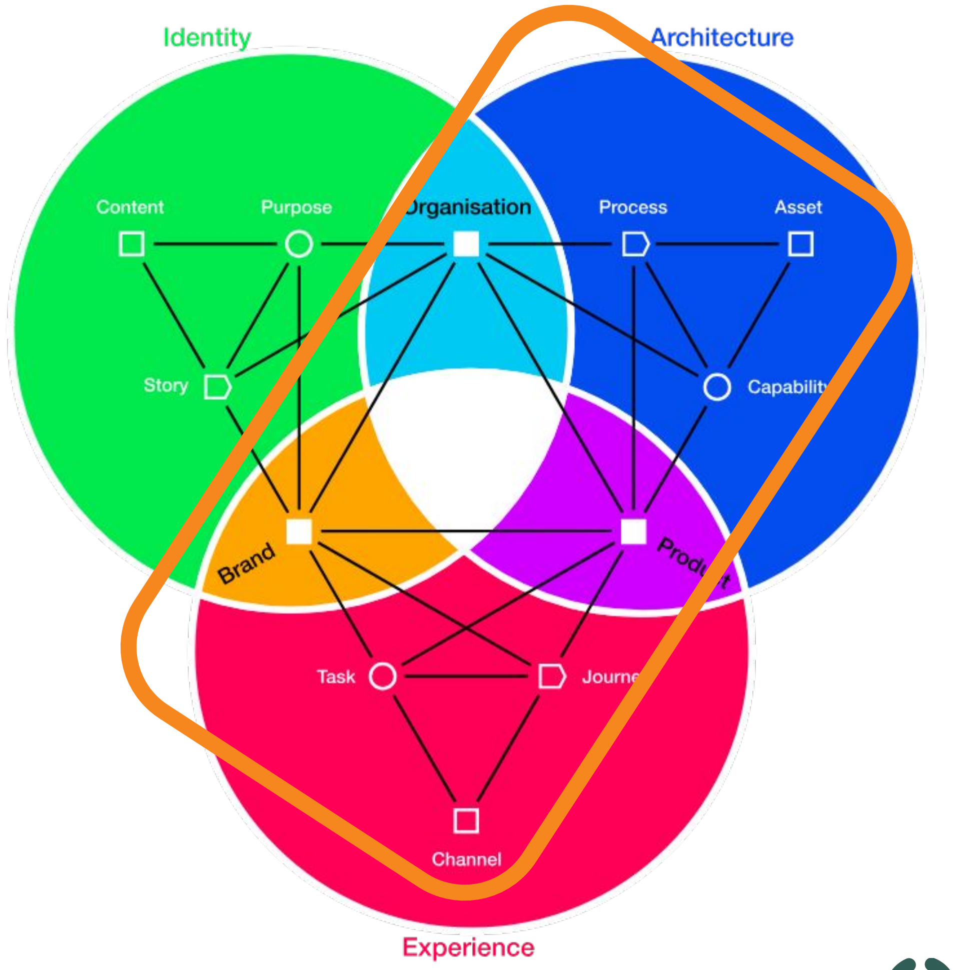
Questions:

2. How to identify platform-ready PBC's?
   → Bridge between business architecture and other architectural roles





Alignment
Training
Workshops

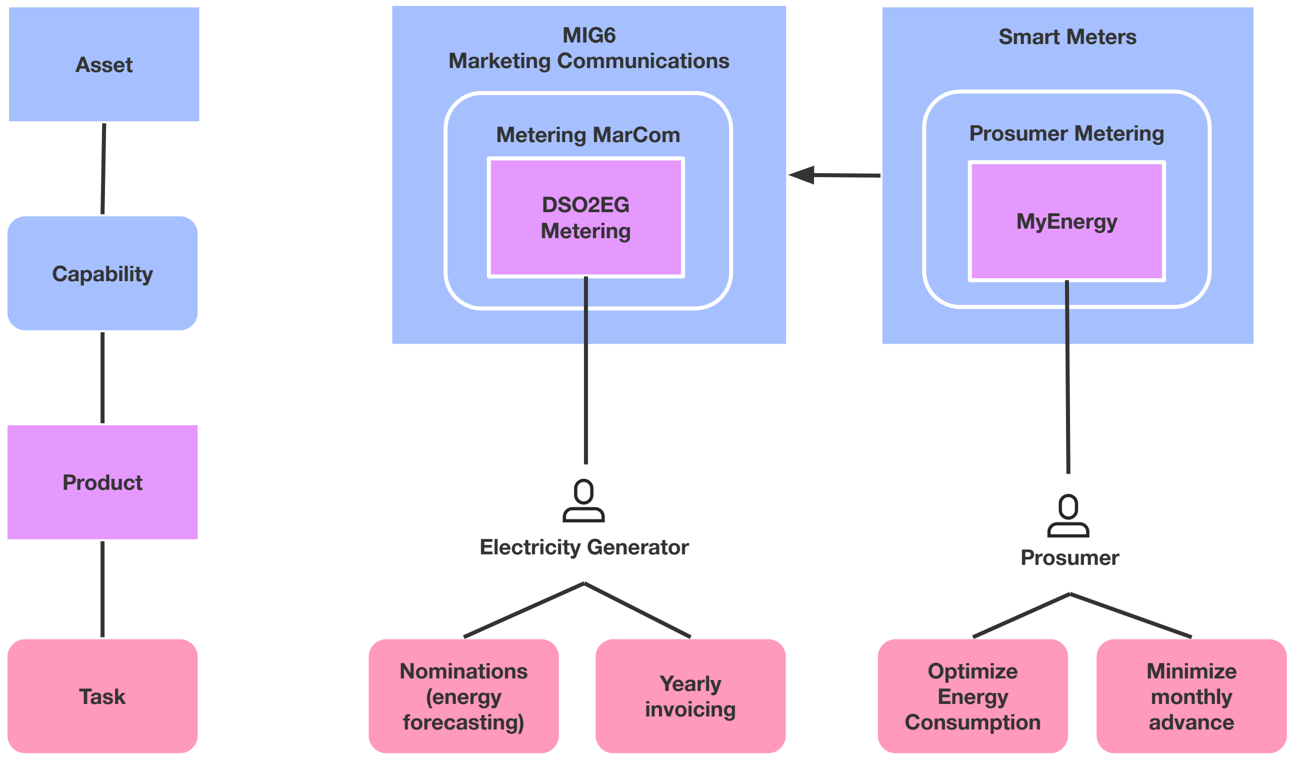


Source: EDGY

DSO Case Study

Questions:

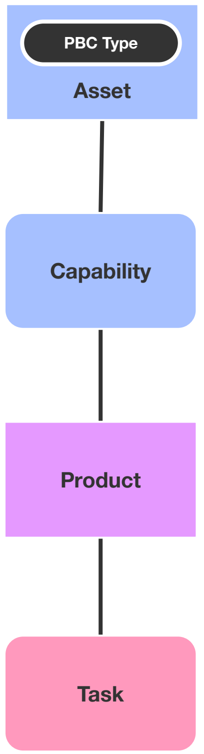
3. How to document & model PBC's?
   → Modeling language: ArchiMate subset
   → Modeling tool: Sparx Enterprise Architect

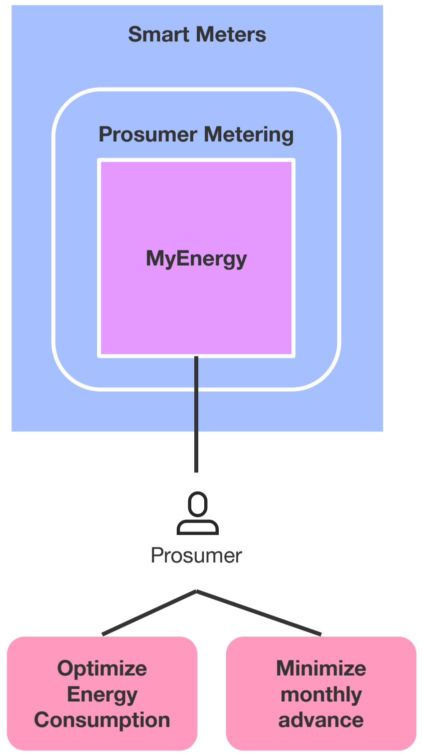


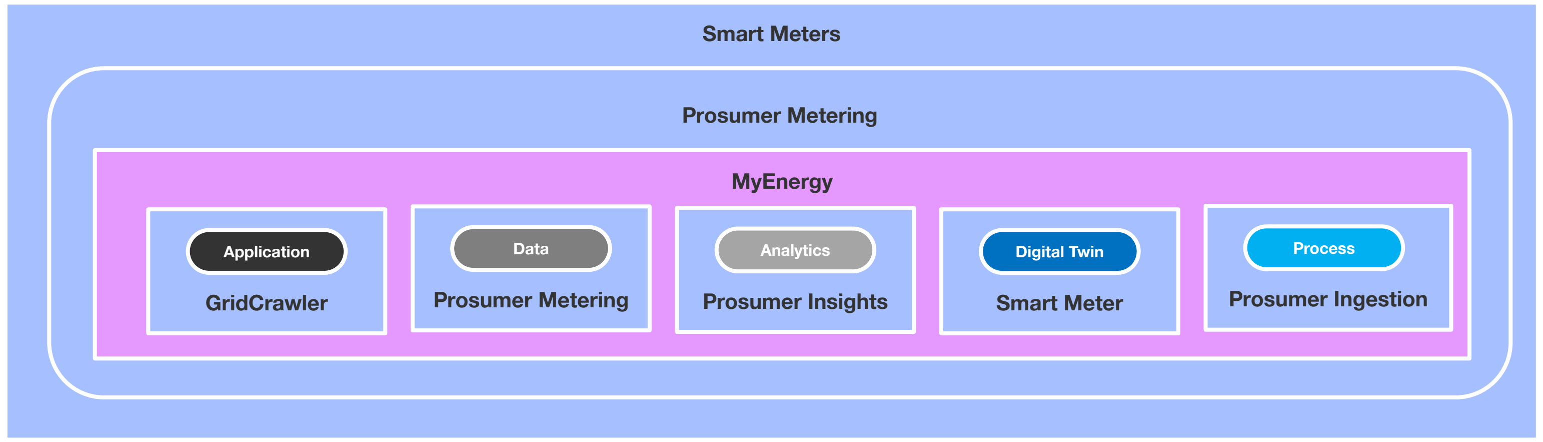
Source: EDGY

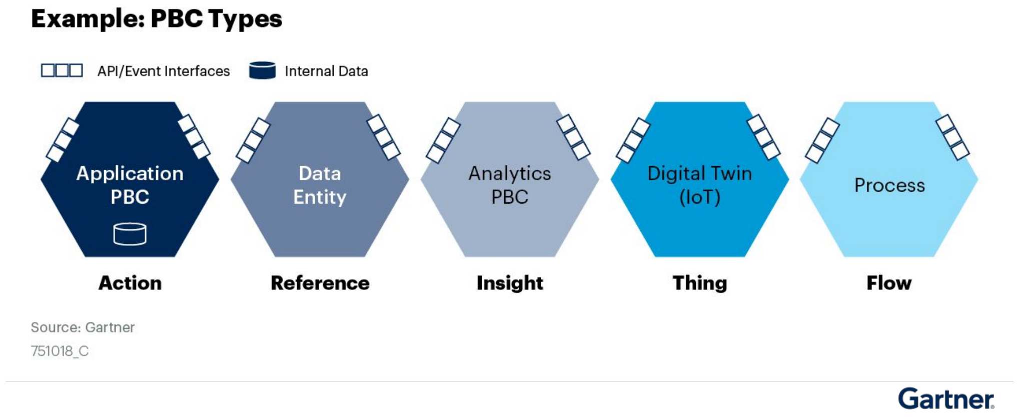
DSO Case Study

Questions:
3. How to document & model PBC's?
→ Modeling language: ArchiMate subset
→ Modeling tool: Sparx Enterprise Architect







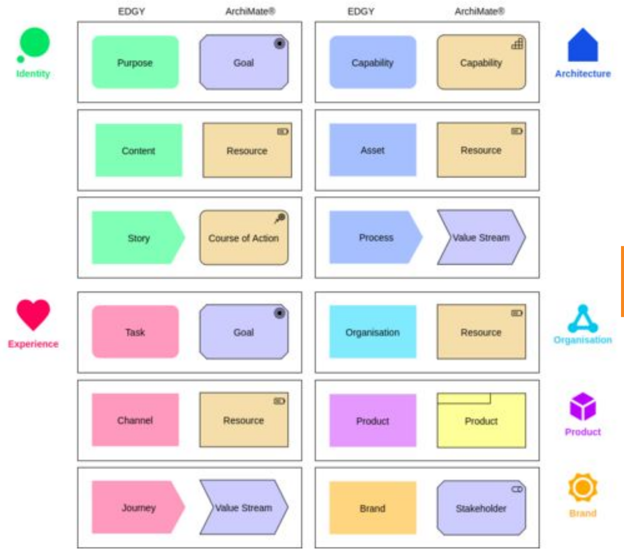


Source: Gartner

DSO Case Study

Questions:

3. How to document & model PBC's?
   → Modeling language: ArchiMate subset
   → Modeling tool: Sparx Enterprise Architect

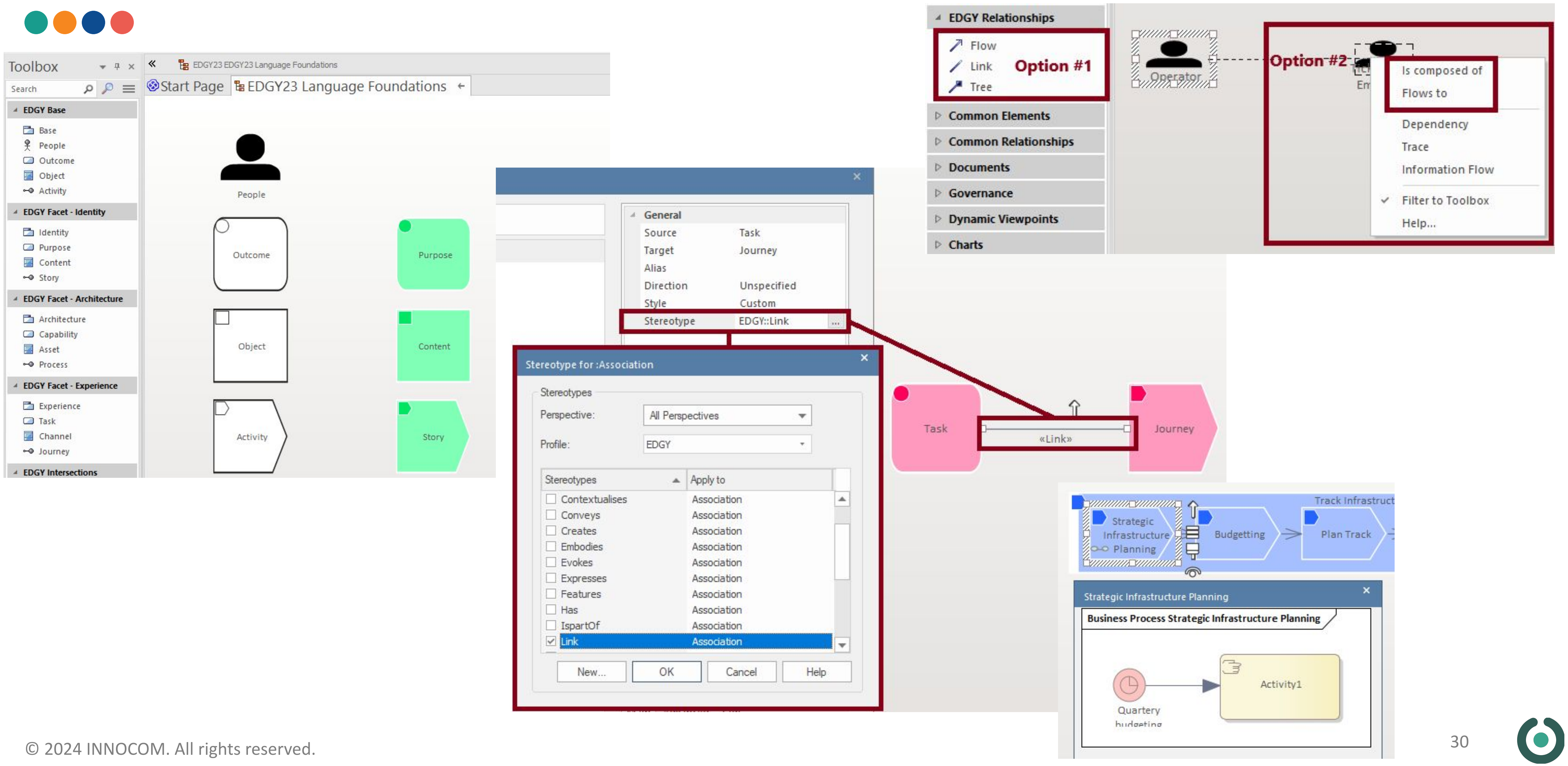


	EDGY	ArchiMate®	EDGY	ArchiMate®	
Identity	Purpose	Goal	Capability	Capability	Architecture
	Content	Resource	Asset	Resource	
	Story	Course of Action	Process	Value Stream	
Experience	Task	Goal	Organisation	Resource	Organisation
	Channel	Resource	Product	Product	Product
	Journey	Value Stream	Brand	Stakeholder	Brand

ENTERPRISE ARCHITECT
SPARX SYSTEMS

Source: EDGY

Free EDGY support for Sparx EA – SPARX4EDGY23



General	
Source	Task
Target	Journey
Alias	
Direction	Unspecified
Style	Custom
Stereotype	EDGY::Link

Stereotype for :Association

Stereotypes

Perspective: All Perspectives

Profile: EDGY

Stereotypes	Apply to
☐ Contextualises	Association
☐ Conveys	Association
☐ Creates	Association
☐ Embodies	Association
☐ Evokes	Association
☐ Expresses	Association
☐ Features	Association
☐ Has	Association
☐ IspartOf	Association
☑ Link	Association

New... OK Cancel Help

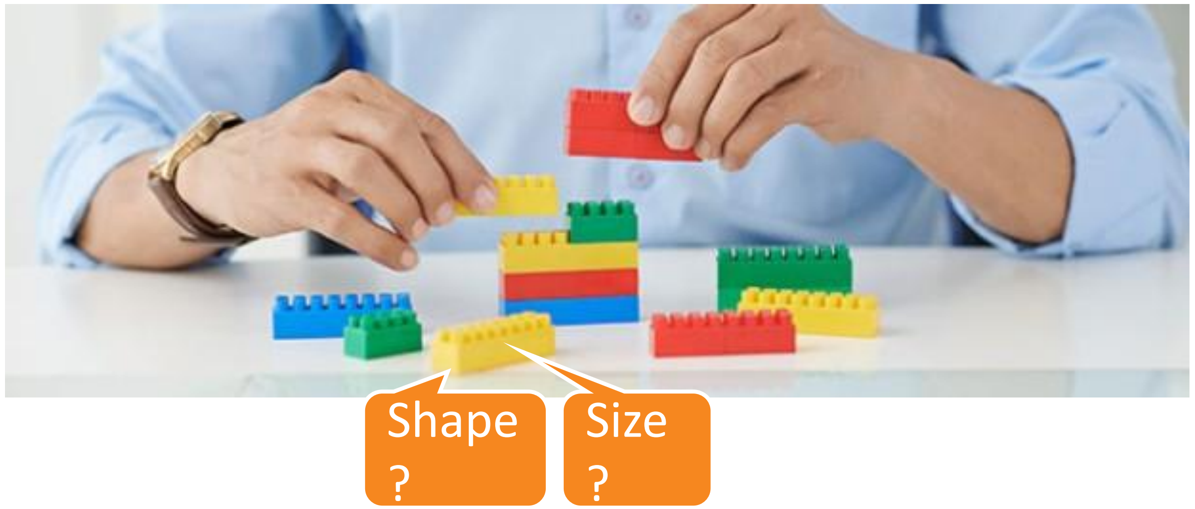
Talk outline

●●●●

1 A short history of Composable Architecture

2 The link with Enterprise Design and EDGY

3 How this has delivered a solid proof-of-value in a real-world case



rudi.claes@inno.com

References

1. TCT Mag – SpaceX Raptor Engines
2. NST Foundation Lecture 1 The Design Cycle as a Dynamic System
3. NST Foundation Lecture 2 Design Theorems for Software Stability
4. Martec's Law
5. Gartner
6. BIAN
7. Thoughtworks
8. EDGY
9. EPRS