Architecture Governance

Jacques Maeda

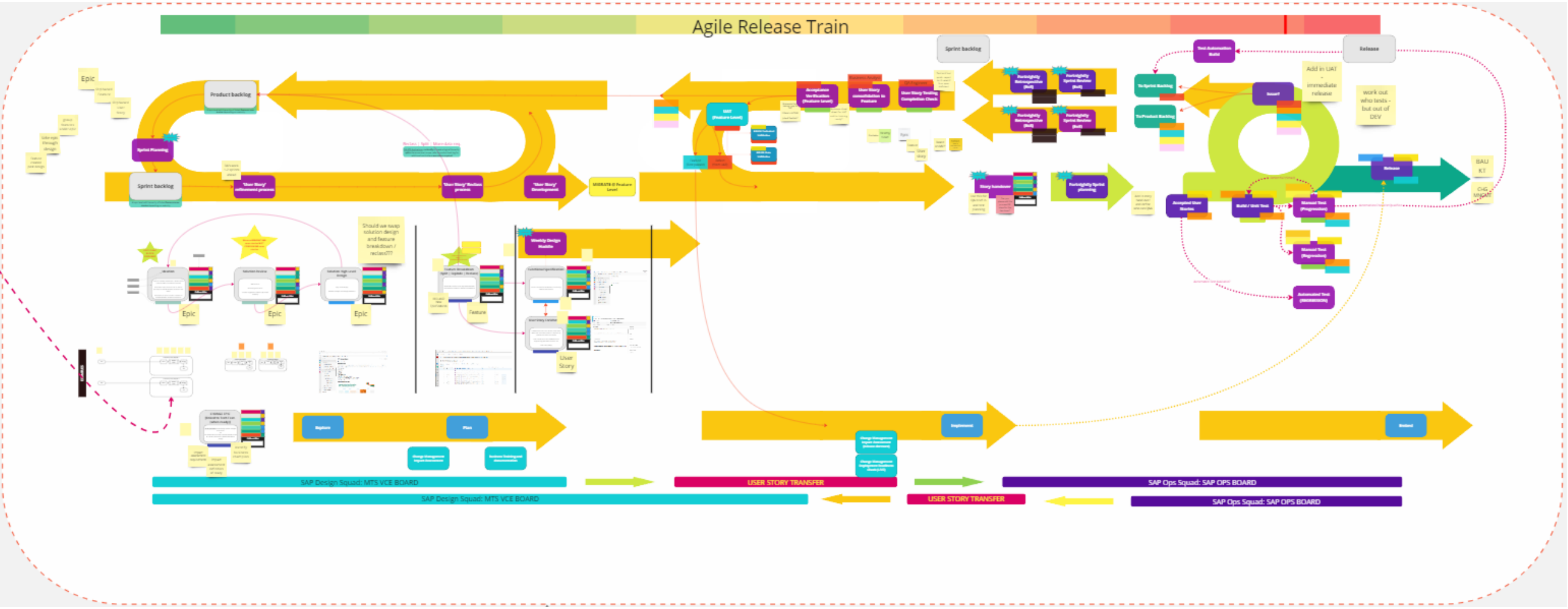
DATE
14 May 2025

Disclaimer

The content in this presentation is based on publicly available information, industry frameworks, vendor documentation, and common practices across retail and convenience store sectors.

All insights and references are for general knowledge sharing and do not reflect or disclose specific business practices, systems, or strategies of any clients.

C-Store's Ways of Working for MTS/VCE with SAP Ops



source: SpotifyScaling.pdf (crisp.se)

source: MTS_Structure & WoW, Visual Workspace for Innovation (miro.com)

Netflix’s Architecture Principles

It was a mammoth decision for the time and Netflix adopted some basic principles to guide them through this change:

Buy vs Build
- First, try to use or contribute to open-source technology wherever possible.
- Only build from scratch what you absolutely must.

Stateless Services
- Services should be built in a stateless manner except for the persistence or caching layers.
- No sticky sessions.
- Employ chaos testing to prove that an instance going down doesn’t impact the wider system.

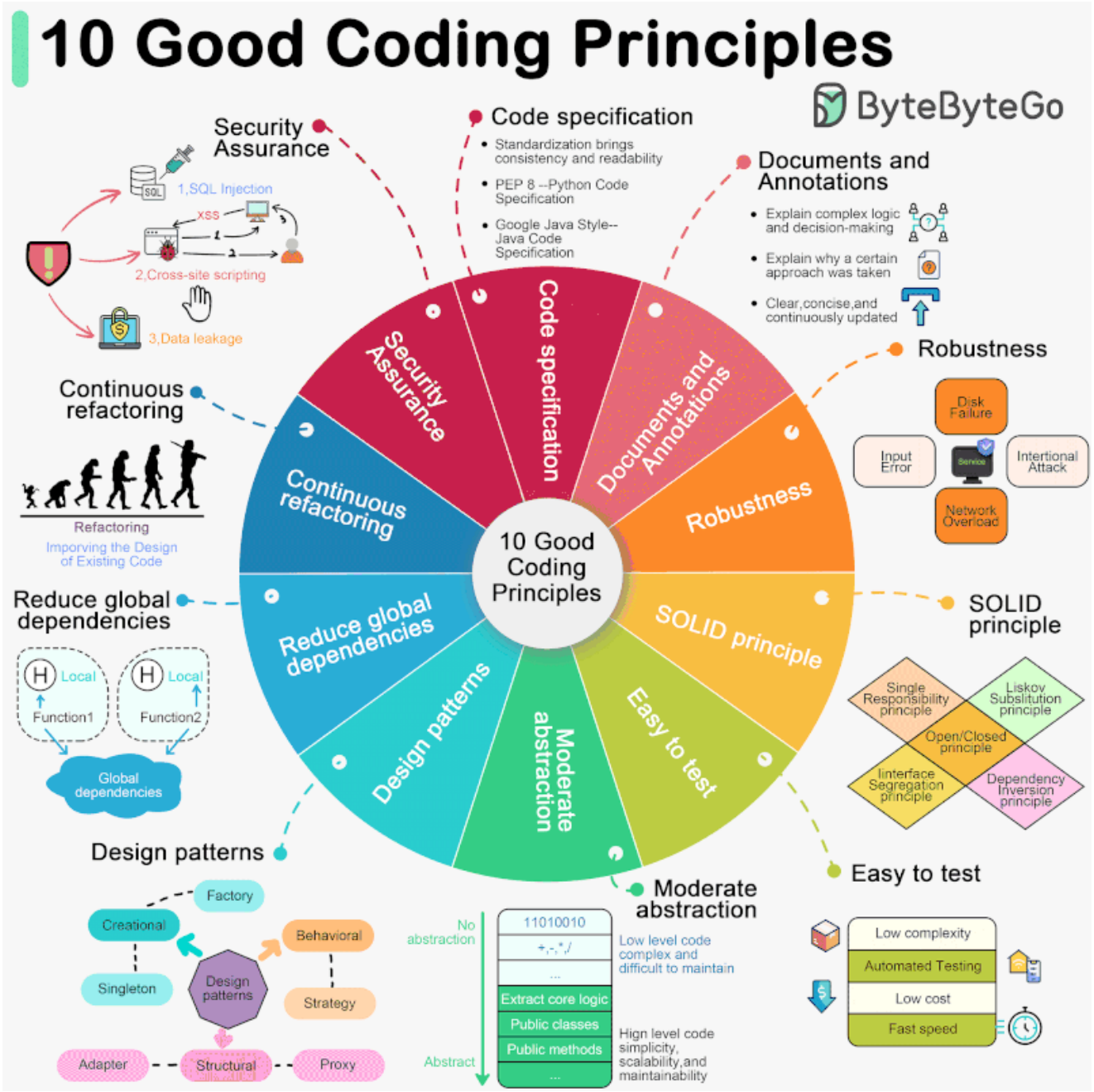
Scale-out vs scale up
- Horizontal scaling gives you a longer runway in terms of scalability.
- Prefer to go for horizontal scaling instead of vertical scaling.

Redundancy and Isolation
- Make more than one copy of anything. For example, replica databases and multiple service instances.
- Reduce the blast radius of any issue by isolating workloads.

Automate Destructive Testing
- Destructive testing of the systems should be an ongoing activity.
- Adoption of tools like Chaos Monkey to carry out such tests at scale.

These guiding principles acted as the North Star for every transformational project Netflix took up to build an architecture that could scale according to the demands.

source: A Brief History of Scaling Netflix - ByteByteGo Newsletter



The Challenges of Enterprise Architecture

Find the balance and role of enterprise architecture

Idea

Problem/Solution Fit

Continuous Exploration
(Strategic Excellence)

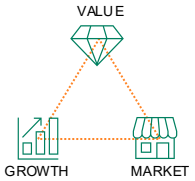
MVP

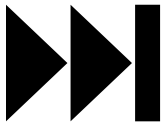
Product/Market Fit

Continuous Delivery
(Operational Excellence)

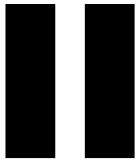
New Version

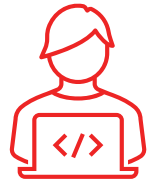
User





Business





IT

Creative Thinking

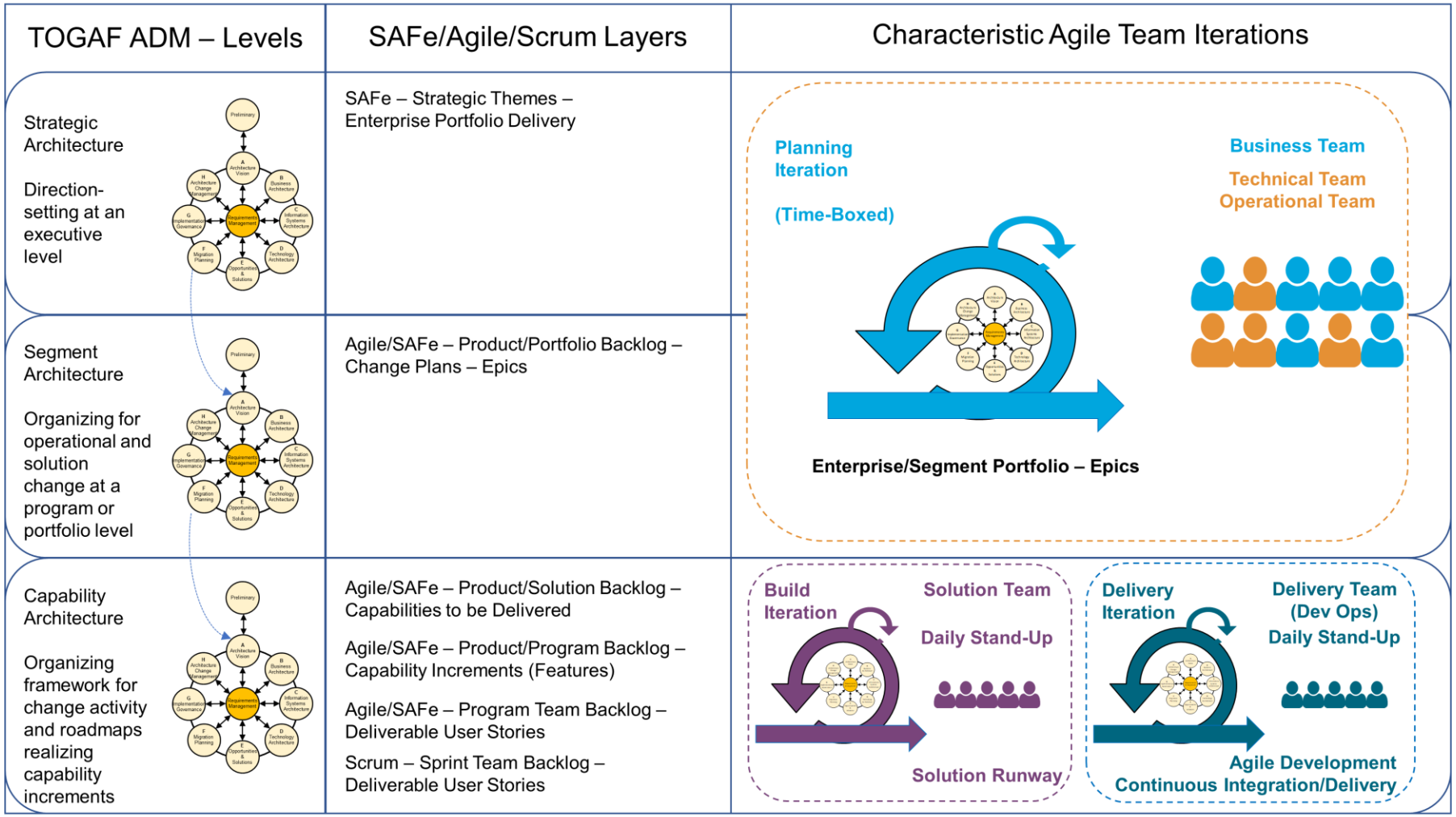
Customer-Centric Mindset
(Design Thinking)

Architectural Thinking

Enterprise Architecture
(Architecture Development Method)

Agile Architecture (Proposed)

Applying the TOGAF using Agile Sprints



source: Enabling Enterprise Agility

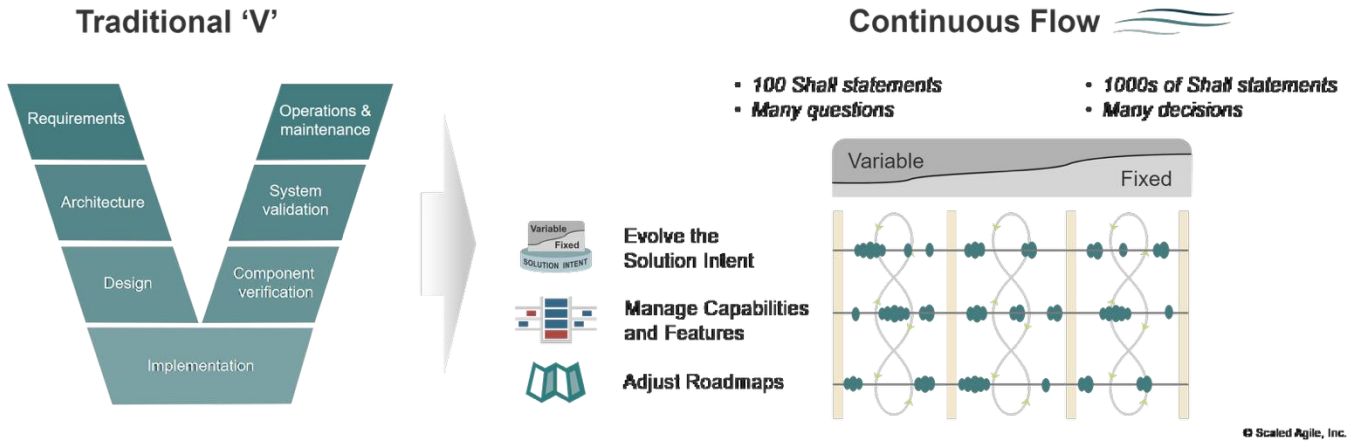
Agile architecture supports Agile development practices through collaboration, design simplicity, and balancing **intentional** and emergent design.

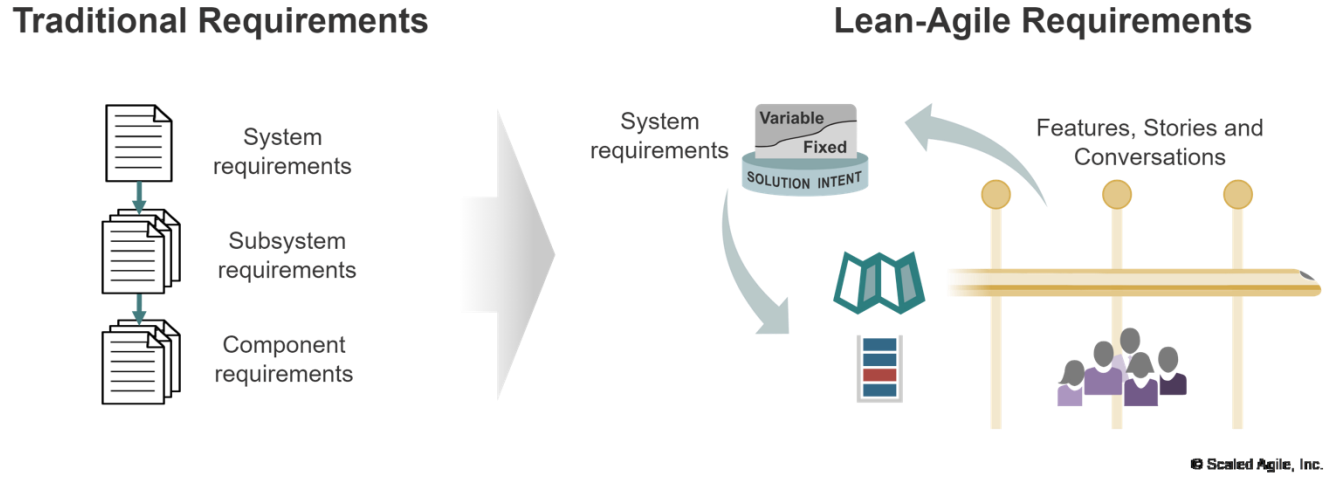
This approach embraces the **DevOps** mindset, allowing the architecture to evolve continuously while supporting current users' needs.

It avoids the overhead and delays associated with the start-stop-start nature and large-scale redesign inherent in **phase-gate processes and Big Design Up Front (BDUF)**.

Agile Architecture (Proposed)

Enterprise Solution Delivery

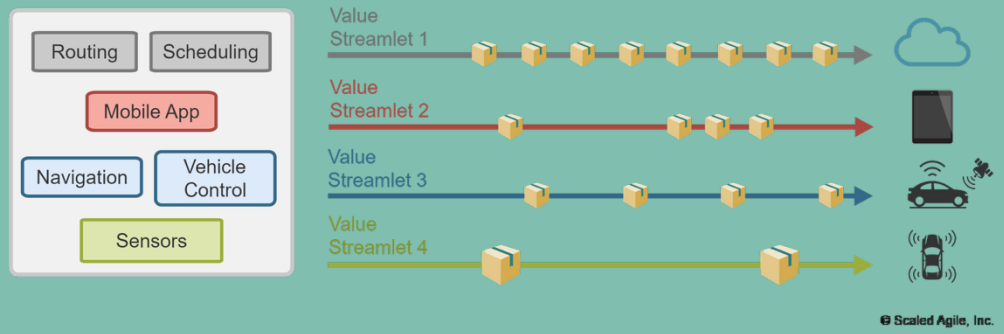




source: Enterprise Solution Delivery - Scaled Agile Framework

Architectural decisions are critical economic choices because they significantly impact the effort and cost required for future changes. Loosely coupled architectures are the strongest predictor of continuous delivery.

A good architectural design enables ARTs and teams to independently develop and release 'Value Streamlets' (solution components within a larger value stream).



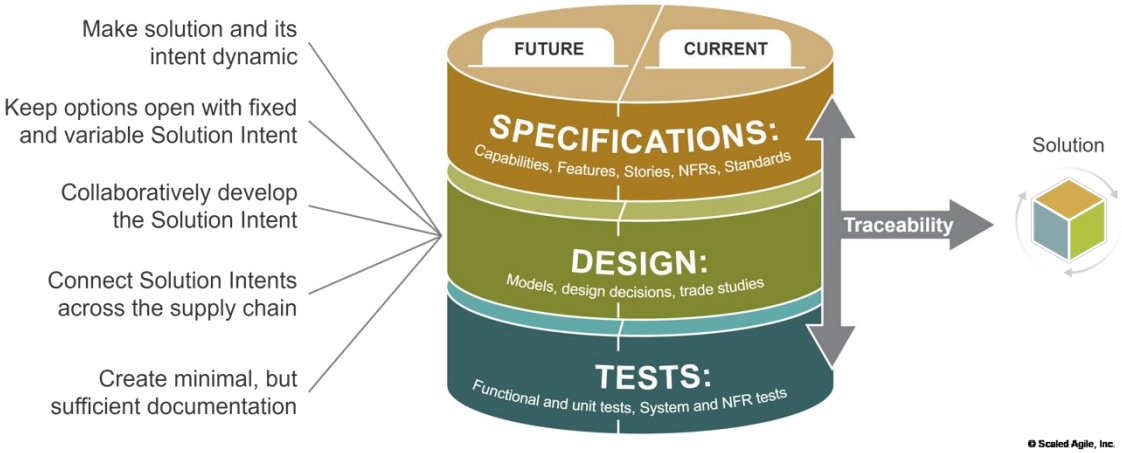
*Solution designs must balance the needs of operations and development. Design choices often optimize for operations to reduce unit, manufacturing, and delivery costs without fully understanding the total costs of delayed **value delivery**.*

Agile Architecture (Proposed)

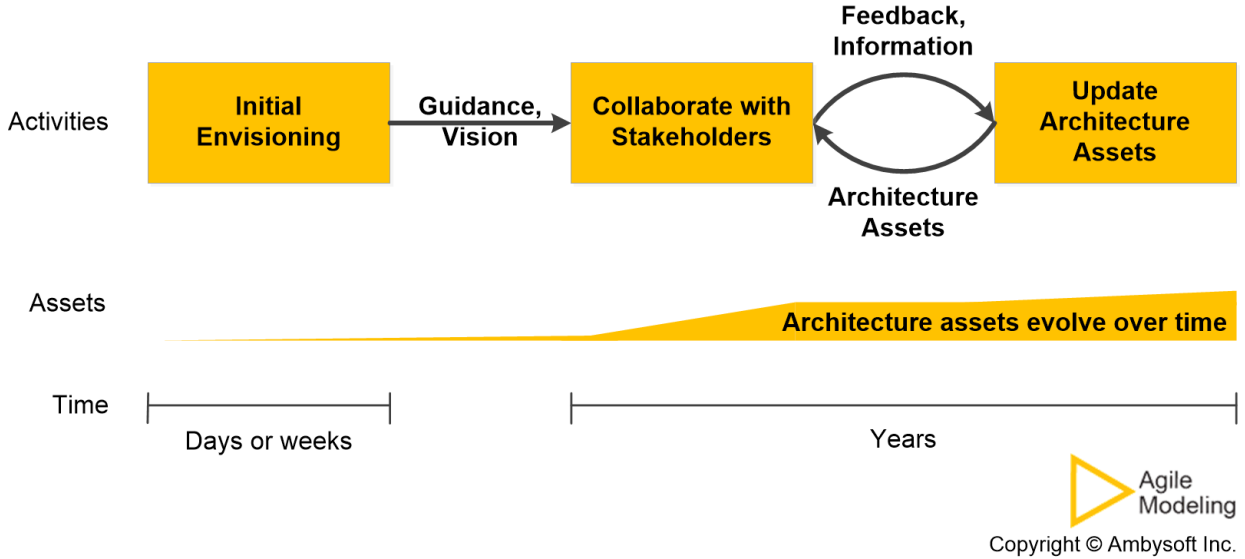
Solution Intent and Transition Architecture

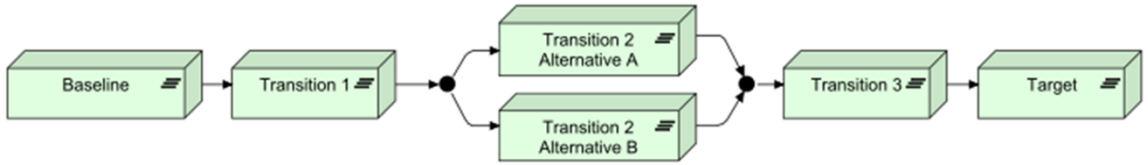
The Solution Intent (or Solution Architecture or High-Level Solution Design) should maintain the integrity of the current state and target state but allows creation of variants that include both fixed and variable specifications and designs.

Architect assets must be continuously updated and enriched.



source: [Advanced Topic - Agile Architecture in SAFe - Scaled Agile Framework]



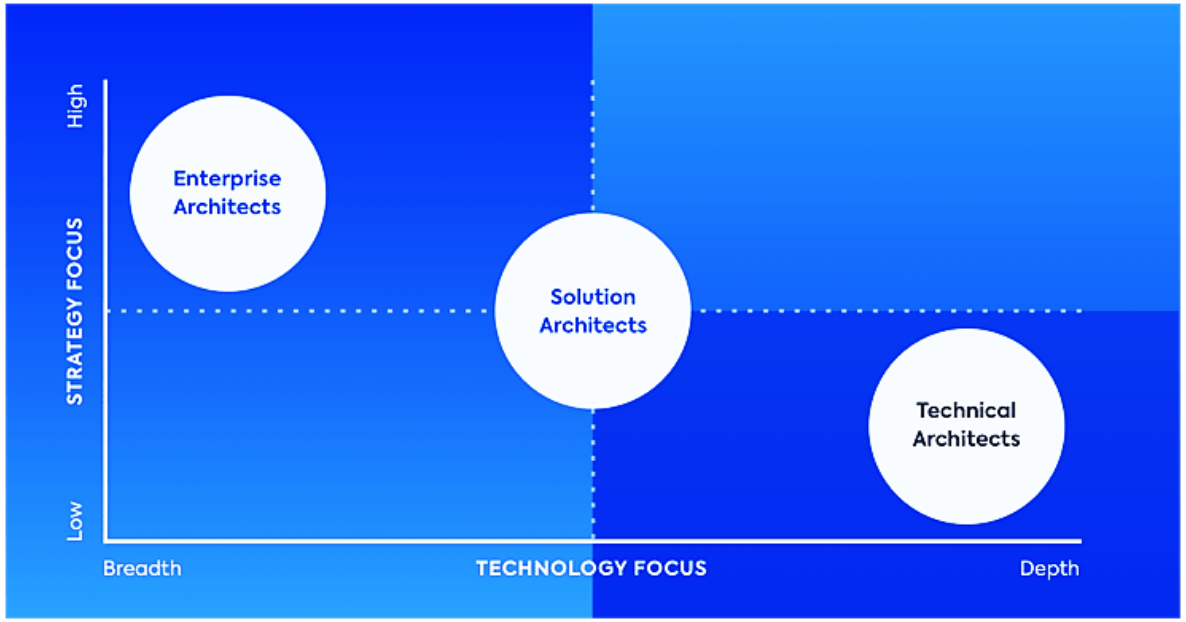


Clear transition architectures (capability increments usually implemented using sprints) might also be designed and documented with the Key Design Decision (KDD) to achieve the target state.

Comparison of Architecture Roles

Enterprise Architect Across Value Streams	Solution Architect Across Systems	System Architect Single System
		
• Aligns architecture with business strategy • Provides strategic technical direction across ARTs and teams • Collaborates with Lean Portfolio Management • Guides and supports Architectural Runway strategy • Promotes modern technical and DevOps practices • Synchronizes architecture functions across ARTs and teams	• Plans the Architectural Runway for a full Solution • Actively supports designing and steering of Continuous Delivery Pipeline • Establishes and supports definition of nonfunctional requirements (NFRs) • Partners with System Architects to elaborate Capabilities and Features • Fosters Built-in Quality for the entire Solution	• Plans the Architectural Runway • Actively supports design and steering of CI/CD pipeline • Establishes and supports the definition of NFRs • Partners with Solution and Enterprise Architects to elaborate Epics, Capabilities, and Business Capabilities • Fosters Built-in Quality for the ART's systems

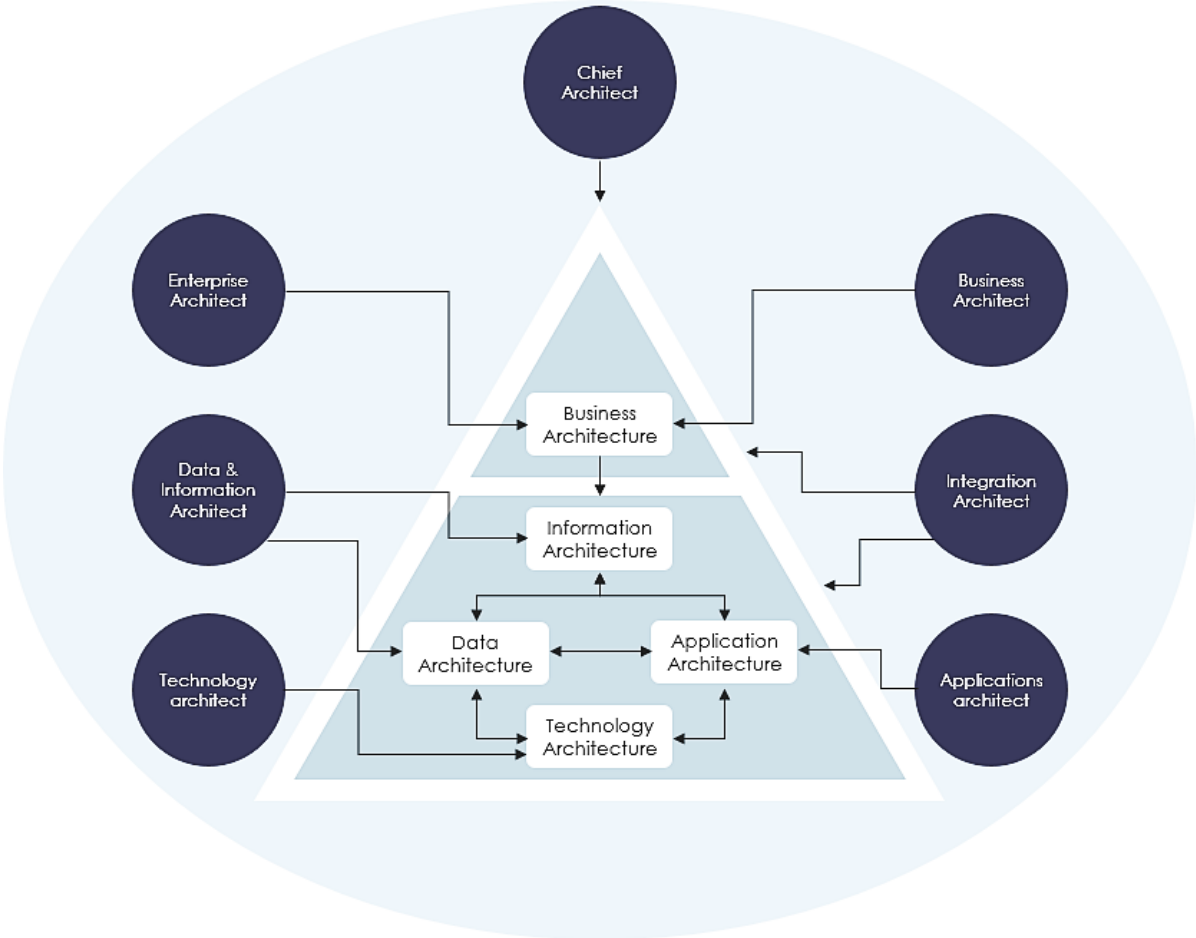
Comparison of Architecture Roles



source: Enterprise Architect vs Domain Architect vs Developer | LeanIX

- Business Architect (Process, Solution)
- Application Architect (**Platform**: SAP suite, Adobe suite, others)
- Information Architect (UX, Digital)
- Technical Architect (Infrastructure/Cloud, Store/POS)
- Data Architect (EDP)
- Security Architect (DevSecOps, Privacy, Compliance)
- Integration Architect (Salesforce MuleSoft, SAP, EDI VAN, others)

Domain Architects are specialists with in-depth knowledge within the particular domain of their expertise.



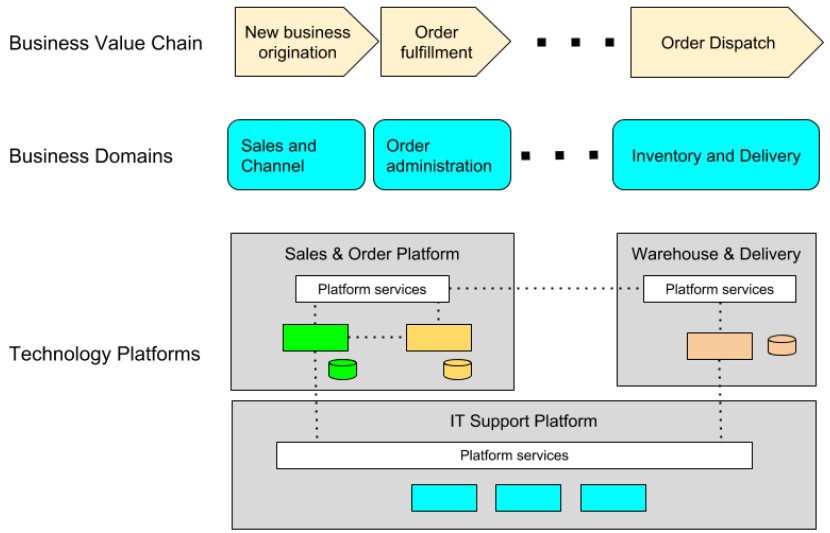
source: Enterprise Architects vs Solution Architects vs Domain Architects (visual-paradigm.com)

Comparison of Architecture Roles

C-Store's **services map** can be related to the business domains.

The technology or platforms that underpin those services are also identified in the service maps.

Review and fine tuning might be required. C-Store might look at Platform Engineering in the future, especially for Customer Digital.



source: Empowering Business with a Platform Architecture Approach (deloitte.com.au)

APQC’s Process Classification Framework

20799 8.5.3 Develop and manage service/solution architecture

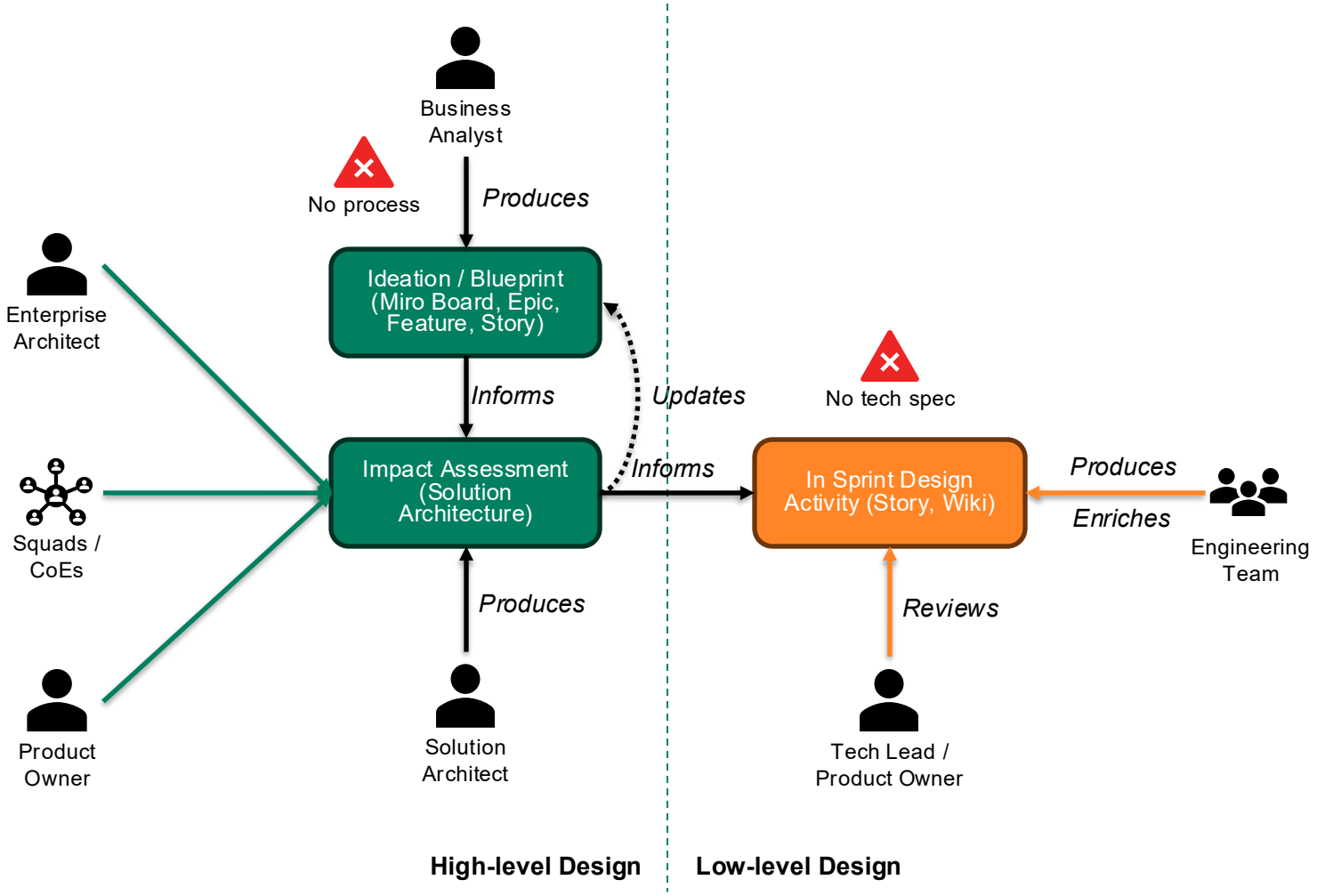
- 20800 8.5.3.1 Assess IT application and infrastructure architecture constraints
- 20801 8.5.3.2 Assess business constraints on IT service/solution
- 20802 8.5.3.3 Determine IT component integration requirements
- 20803 8.5.3.4 Identify opportunities for IT component reuse
- 20804 8.5.3.5 Promote adoption of existing service/solution architecture
- 20805 8.5.3.6 Develop and maintain service/solution architectures
- 20806 8.5.3.7 Assess IT service/solution architecture conformance
- 20807 8.5.3.8 Manage architectural exceptions

Examples of Technical Debt

- Quick-and-Dirty if-then-else
  - For the Canadian company, the decision to use *if-then-else* statements spread the change throughout the code, but it was a necessary quick-and-dirty solution from a business perspective to get a quick sale. Doing the right thing at that stage would have postponed the delivery of the system and likely lost them the deal. Now, would you continue down that path and add another layer of if-then-else for each language? Or would you rethink the strategy and decide to repay the original technical debt? Inserting the Japanese version of the quick fix, with its issues of character sets and vertical text, would be too much of a burden and a subsequent maintenance issue. You may argue that a good designer would have set up provisions for internationalization and localization right at the outset, but this is easy to say in hindsight. The demands and constraints at the beginning of development for this small venture were quite different, focused on the main features, and didn't foresee the need for a multilingual feature.
- Hitting the Wall
  - The development team neglected to consider architectural and technology selection issues or learn from the two existing systems at appropriate times during development. The team did not need to do all of that up front, but it needed to do it early enough not to burden the project downstream. Refactoring is valuable, but it has limits. The development team had to throw away large portions of the existing code weeks after its original production. Although the organization hoped to eliminate technical debt when it decided to implement a brand-new system after the merger, it failed to incorporate eliminating technical debt into the project management strategy for the new system.
- Crumbling Under the Load
  - There were hundreds of causes: code imperfections, tricks, and workarounds, compounded by no usable documentation and little automated testing. While the development team dreams of a complete rewrite, the economic situation does not allow delaying new releases or new products or abandoning support for older products. Some intermediate strategy must be implemented.
- Death by a Thousand Cuts
  - A pervasive lack of competence can result in many small, avoidable coding issues that are never caught. Lack of organizational competency--as in the case of this IT-service organization--easily activates a number of cascading effects. The unplanned and unmanaged hiring boom, the missed opportunity to enforce commonality across the products, and the limited testing all contributed to the accumulating technical debt.

source: Managing the Consequences of Technical Debt: 5 Stories from the Field (cmu.edu)
source: Helping Organizations Work Smarter, Faster, and with Greater Confidence | APQC
source: SAP Signavio Process Explorer

Roles & Responsibilities (Current)



Solutions Design (Business Analyst)

- Assumes responsibility for timely technology solution delivery and mapping dependencies between platforms.
- Complete view of technical stack to ensure best practices or avoid common pitfalls.
- Manages 3rd party development resources.

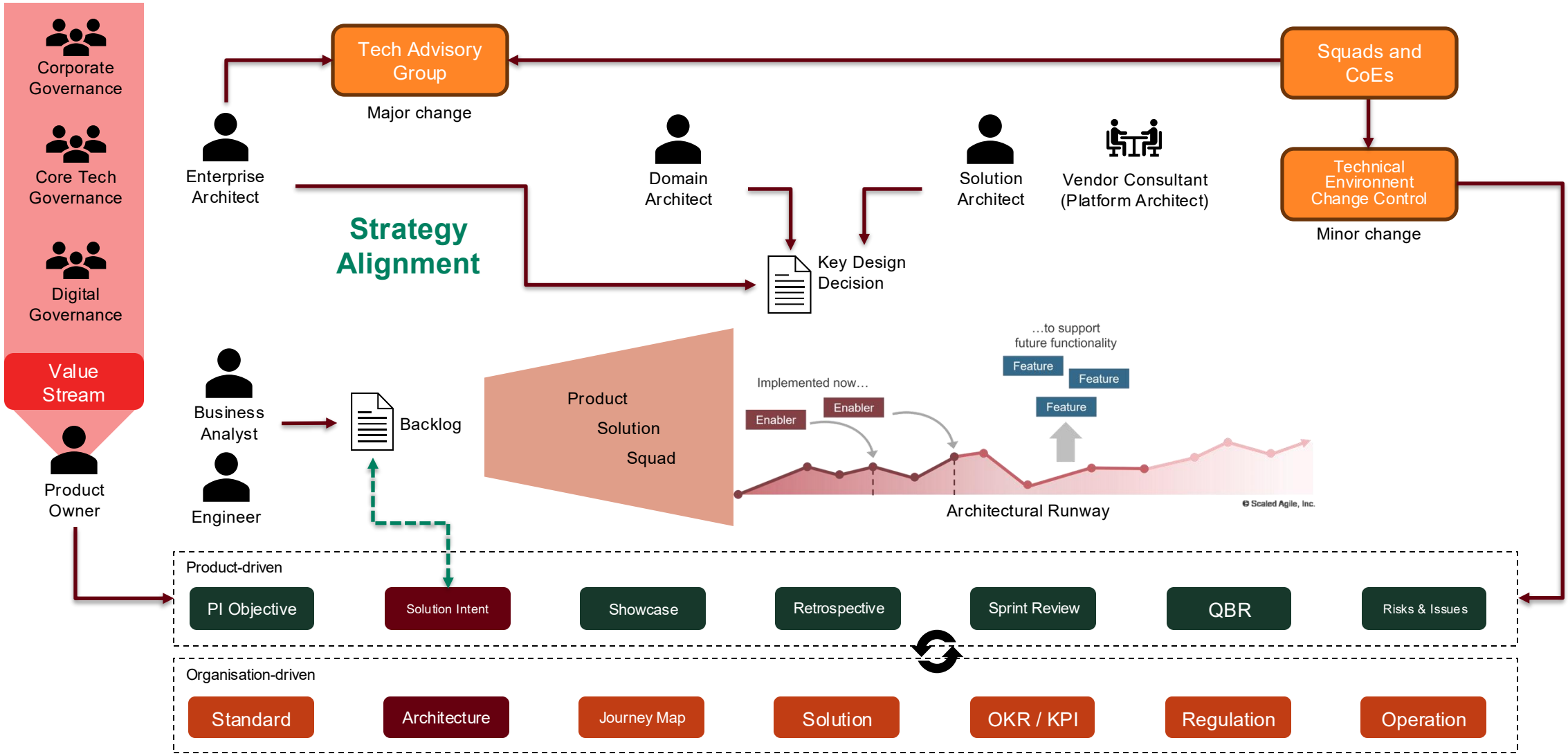
Architecture

- Enterprise and digital architecture design & principle management.
- Creation of non-functional requirements to ensure solution/security can scale/secure.
- Feature solution and security design and review.

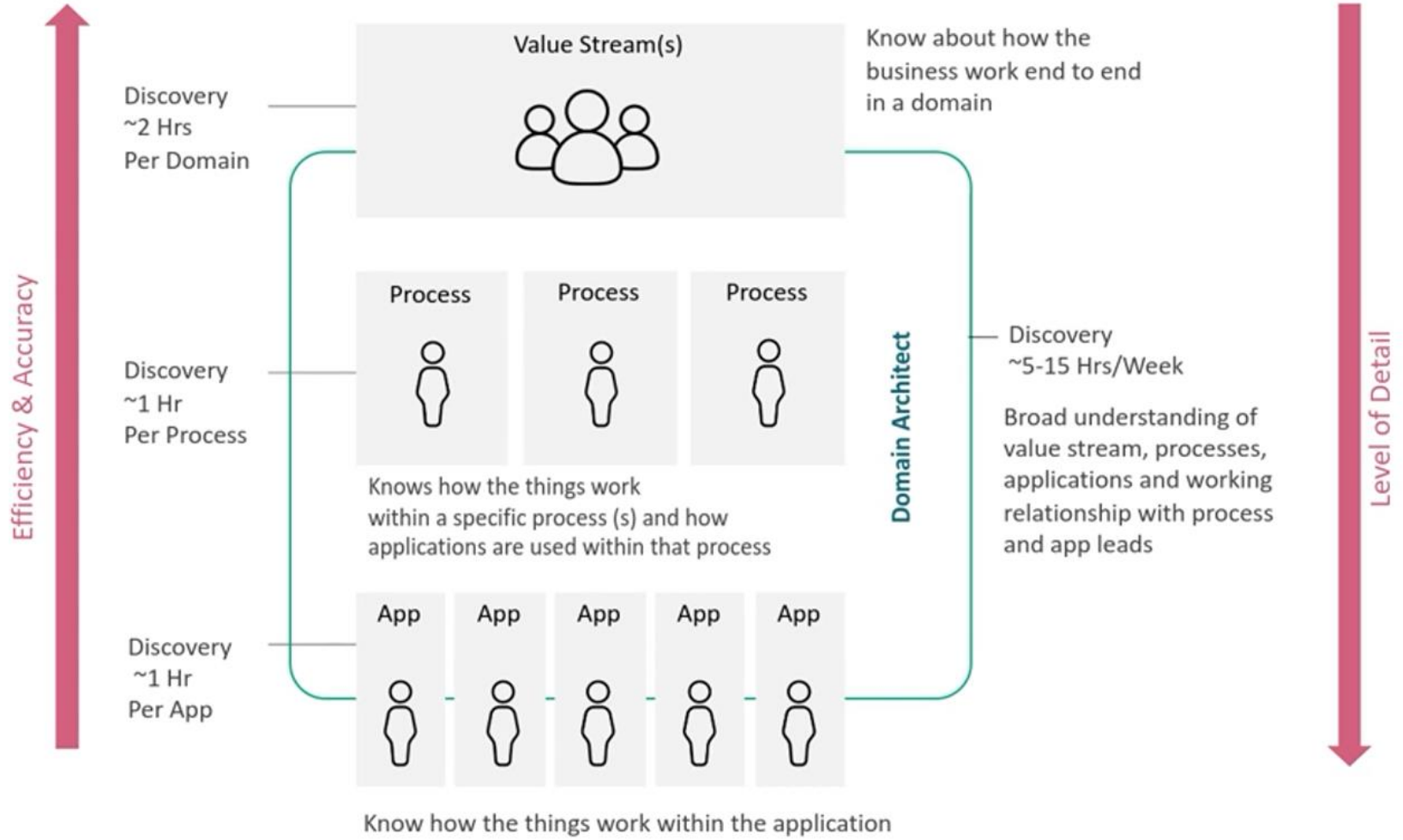
Engineering

- Understand and translate the business requirements into end-to-end functional applications.
- Application performance monitoring.
- Analyse and identify needs by investigating problem areas and implementing user feedback to the application that meets those needs

Roles & Responsibilities (Proposed)



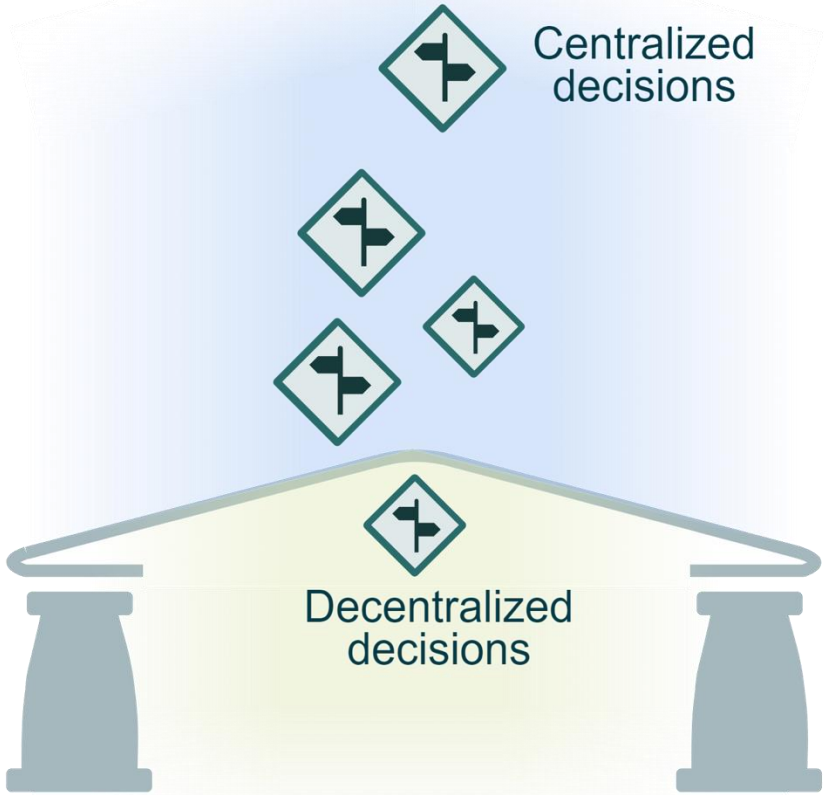
Who Has The Knowledge?

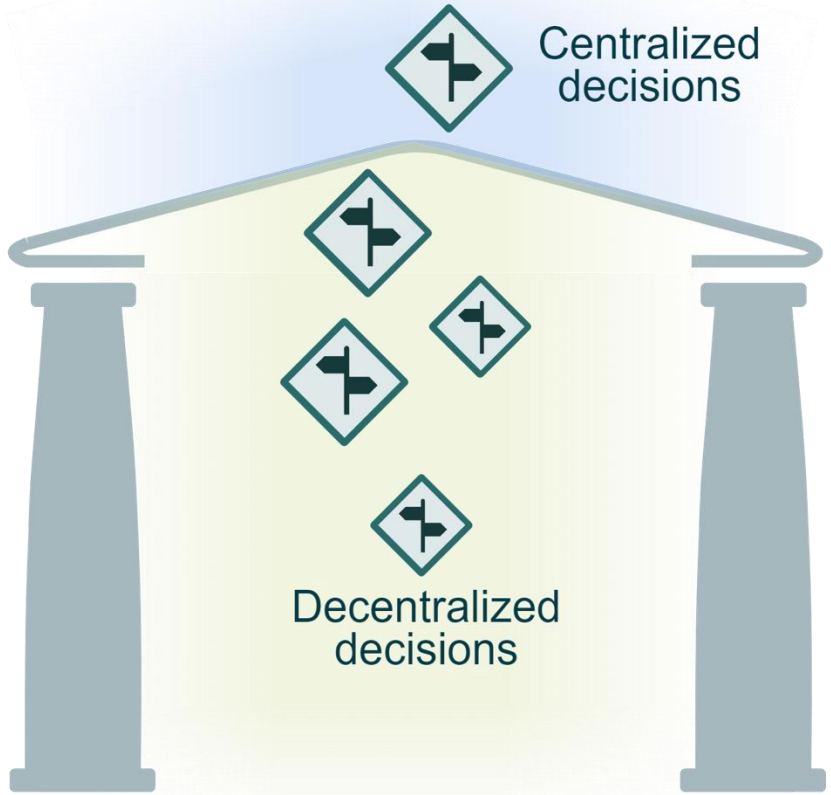


source: Unlock the Full Potential of Enterprise Architecture: A Three-Part Journey with SAP LeanIX and Vasa Digital Architects Explorer

Decentralised Decision-Making

A critically important aspect of decentralized decision-making is the communication of intent. Simply put, the decision maker must be clear to those impacted by the decision about the 'what' and the 'why' behind it.





Technical COMPETENCE

Organizational CLARITY

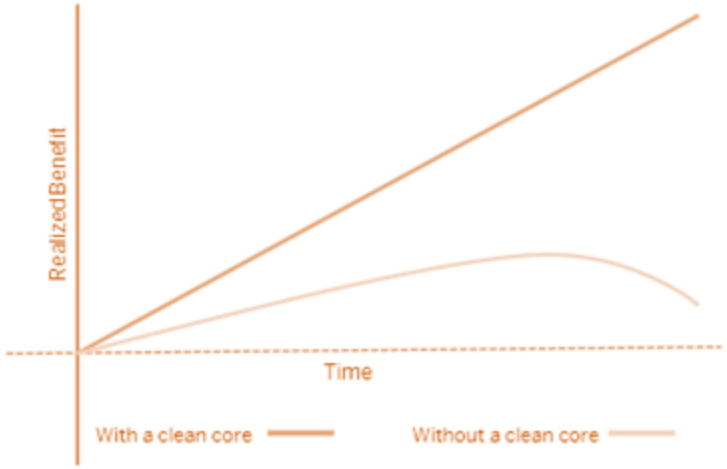
Technical COMPETENCE

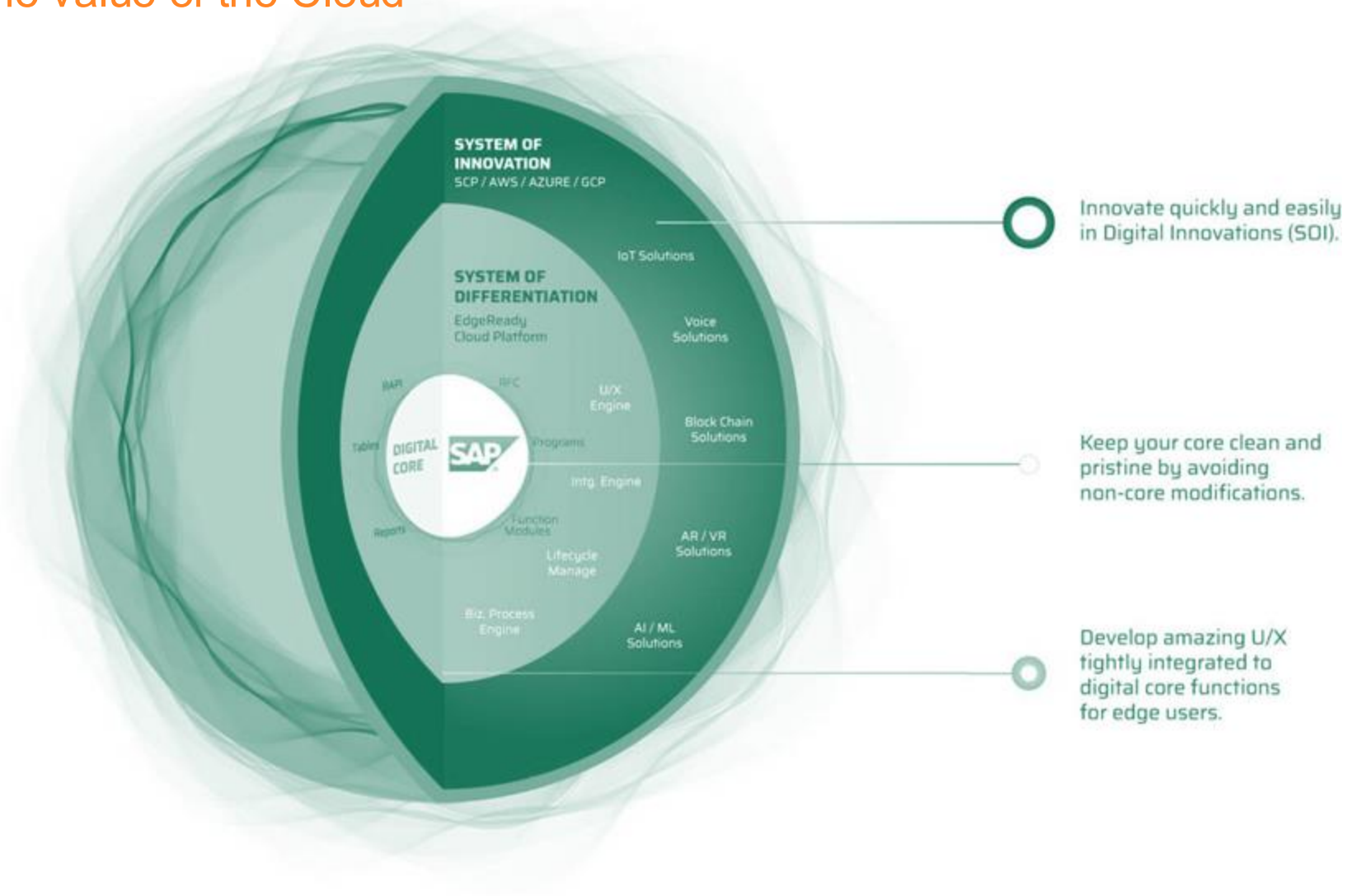
Organizational CLARITY

Clean Core Approach

Preparing for and maximising the value of the Cloud

- Clean code
- Reliable landscape
- Lean data
- Clear process
- Controlled modifications
- Effect & efficient operations

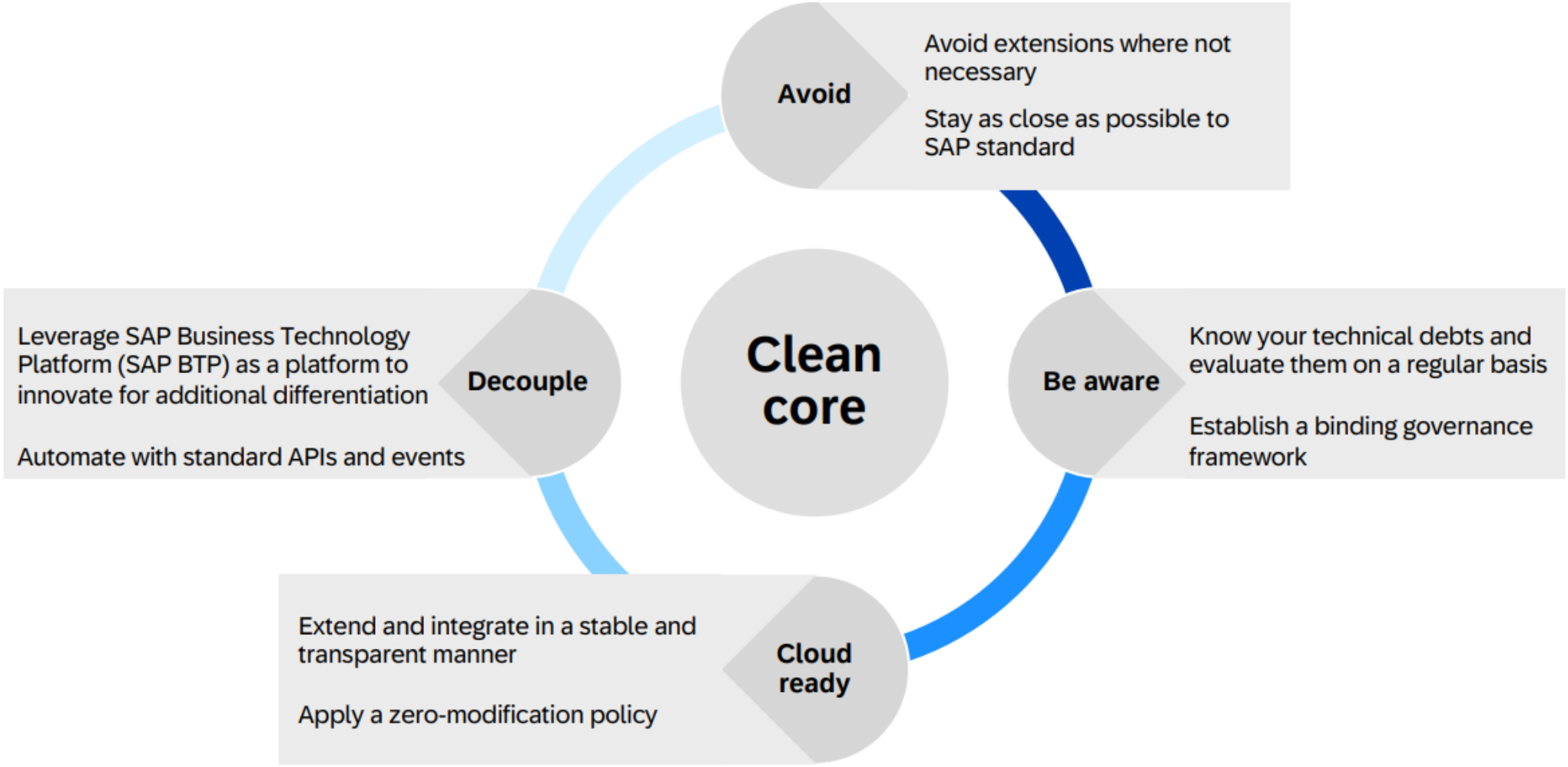




source: Point of View: Clean Core

Clean Core Approach

Top-level guiding principles



source: SAP Clean Core-1.pdf (asug.com)

What is a “clean” process?

- **State-of-the-art process governance**, such as including transparent organizational structures and requirement management
- **Application architecture** that leverages SAP’s standard solutions (SAP S/4HANA and beyond)
- Respective **SAP Best Practices packages** implemented where available and applicable
- Business **process design following defined principles** such as focusing on differentiating processes
- **Established process management**, documented processes and variants, and measurement of as-is process execution and process performance

- Governance model and organizational structures:
  - Proper requirement management includes a solution standardization board or a similar deciding structure, a suitable methodology (for example, SAP Application Extension Methodology – see slide “Clean extensions”), and proper documentation and tooling support such as the SAP Cloud ALM solution or the Focus Build solution for SAP Solution Manager.
  - A business process management structure is established and transparent across relevant organizations, including process owners and related contact persons.

- The solution architecture can be based on the SAP Enterprise Architecture Framework methodology. Supporting resources can be leveraged using the SAP Transformation Navigator tool, SAP Signavio Process Explorer solution, SAP Signavio Process Manager solution, SAP Signavio Process Collaboration Hub, and the partner solution LeanIX.

- SAP Best Practices can, in general, be compared to SAP Signavio Process Explorer, the SAP Signavio Process Navigator solution, SAP Signavio Process Manager, or SAP Signavio Process Collaboration Hub. Find more general guidance in the Administration Guide to Implementation of SAP S/4HANA with SAP Best Practices.

- For individual implementation projects, the SAP Activate innovation adoption process and the related road map viewer can be leveraged. Further preconfiguration is provided through the enterprise management layer for SAP S/4HANA. Industry-specific best practices can be requested through SAP standard content activation service.

- The solution design is focused on the following principles:
  - For nondifferentiating processes, SAP standard is strictly applied; for differentiating processes, SAP standard solutions are tailored to business needs and enhanced where needed.
  - A focus on mandatory and key process variants helps avoid unnecessary customizing and process complexity.
  - For required extensions the recommendations for “clean extensions” are followed.

- Business process management excellence covers the following three aspects:
  - Customer-specific process design is documented (SAP Signavio Process Manager and SAP Signavio Process Collaboration Hub).
  - As-is process execution can be measured (SAP Signavio Process Insights, SAP Signavio Process Intelligence, and using the plug and gain approach).
  - Business processes are efficiently executed based on measurable process performance indicators (SAP Signavio Process Insights and SAP Signavio Process Intelligence).

What is a “good” extension?

- **Avoid** extensions when possible
- Set up a **strong governance** to create decoupled extensions in a way that they would work in the cloud (three-tier model)
- Separate extensions by leveraging released APIs – **custom extensions** do not break an upgrade and upgrades do not break an extension*
- Leverage the full capabilities of extensibility **on the stack** as well as side by side **with SAP BTP**
- Create **technical debts** only as informed decision

- Establish a governance model – a clearly defined process with high demands to approve any extension.
  - Prefer standard over custom development by leveraging fit-to-standard best practices.
  - Avoid custom code where possible. Don’t extend for rarely needed use cases.
  - Prefer “clean” extension options over “unclean” ones.
  - Use SAP Application Extension Methodology and extensibility guidance to identify the best path in your landscape.

- If you need to extend, a clear separation is key.
  - Only access standard objects through released and stable APIs (either remote or locally; access for reading and changing access possible).
  - Choose only “clean” tools or environments and extensibility options (in SAP S/4HANA: key user, developer, or side-by-side extensions).

- Choose extension domain based on requirements only.
  - SAP BTP automatically decouples extension but is not the only “clean” approach.
  - Do not extend in the core simply because “we always do so.”

- Enable awareness.
  - In on-premise installations, you can actively decide to develop some extensions not clean core, as long as they are documented and informed decisions (use cases: copy routines; API not available, and more).
  - Mitigate missing APIs in private cloud or on premise by using wrappers as described in ABAP Cloud API Enablement Guidelines for SAP S/4HANA Cloud, private edition, and SAP S/4HANA.
  - Create requests for APIs using a customer influence tool (for public or private cloud editions).

*Ensuring upgrade stability can be a short-term workaround for transforming a whole application from traditionally developed code into cloud-compliant (Tier 1) extensions.

What is “clean” data?

Data quality (Configuration, master, and transactional data)*
- Accuracy
- Timeliness
- Completeness
- Validity
- Consistency
- Uniqueness

Data volume efficiency (Master and transactional data)
- Optimized memory and disk consumption
- No outdated, unused, or redundant information
- Data lifecycle management (creation, updates, end of life)

Data privacy compliance
- Storing and processing personal master data only with justifiable purposes

*Configuration data: General data that defines the organization‘s structure and is of static nature (such as company code, plants, purchase organisations, controlling area, or sales area); Master data: Consistent and uniform set of identifiers and extended attributes that describe the core entities of the enterprise, such as customers, vendors, products and general ledger accounts; Transactional data: Information directly derived as a result of transactions, this data always has a time dimension, a numerical value, and refers to one or more (master data) objects

For data quality:
- Analyze and define data quality measures for critical data objects. SAP provides data quality measures for several standard data objects.
  - If necessary, involve SAP or third-party vendors (such as CDQ) for getting further help and advice.

- Establish a “get clean” process.
  - Define a tool-based, reusable data cleansing process (such as through the SAP Master Data Governance application [consolidation], SAP Information Steward software, SAP Data Intelligence solution, quality services for SAP BTP, or other third-party tools) for deduplication, generation of best records, and more.

- Establish a continuous “keep clean” process.
  - Define a validation rule framework, approval process, and automated distribution framework to connected receiver systems for newly created, changed, or deleted data records (such as using SAP Master Data Governance or SAP Master Data Integration service).
  - Define continuous data monitoring.
- Adhere to SAP One Domain Model (universal language across SAP systems) and SAP Data and Analytics Advisory Methodology.

For data volume efficiency (from creation until end of life):
- Enable efficient continuous analysis and monitoring of the database by reducing outdated, unused, or redundant data.
- Define archiving or deletion of data-tiering processes to improve efficiency of the database (for example, using the SAP Information Lifecycle Management component).

For data privacy compliance:
- Analyze data usage to clarify business purposes of collecting and processing personal master data.
- Establish policies to govern personal master data lifecycle (using SAP Information Lifecycle Management).

What is “clean” integration?

- **Base integrations on standard APIs** (OData and SOAP)
- Aim for side-by-side extensibility with **API integration or even SAP Cloud SDK** by utilizing the tight coupling with **SAP Integration Suite**
- Realize loosely coupled integrations in an **event-driven design** based on standard **events***
- Avoid **traditional APIs** (RFC and IDoc) and their related **classical extension options**
- Ensure proper monitoring and error resolution capabilities using **SAP Application Interface Framework**

*Event: A data record expressing a significant change in state (for example, change of a business partner) and consisting of data representing the occurrence and context metadata. It is sent to an event provider such as the SAP Event Mesh capability, where consumers can subscribe to it. The business object data needs to be pulled by each consumer individually by using standard APIs.

- Establish a clearly defined integration strategy with SAP Integration Solution Advisory Methodology:
  - The central access point to discover integration artifacts like standard APIs, events, and integration flows is SAP Business Accelerator Hub.
  - If mediation is required, use SAP Integration Suite to benefit from the tight integration with other SAP BTP capabilities.

- Define a one-time “get-clean” process:
  - Create an integration repository (included in SAP Solution Manager) to identify existing integrations and the technology or protocol on which they are based.
  - Establish “get clean” service inside the company to evaluate how utilized traditional APIs (RFC and IDoc) as well as their classical extensions could be converted into standard interfaces (fit to standard).

- Establish a continuous “keep clean” process (governance model):
  - Define central governance functionality for evaluation of new interface requirements or any interface adjustment based on defined SAP Integration Solution Advisory Methodology characteristics. The Integration Assessment capability could be used as an accelerator.
  - Apply a “keep clean” process for the most important and critical integrations.

What are “clean” operations?

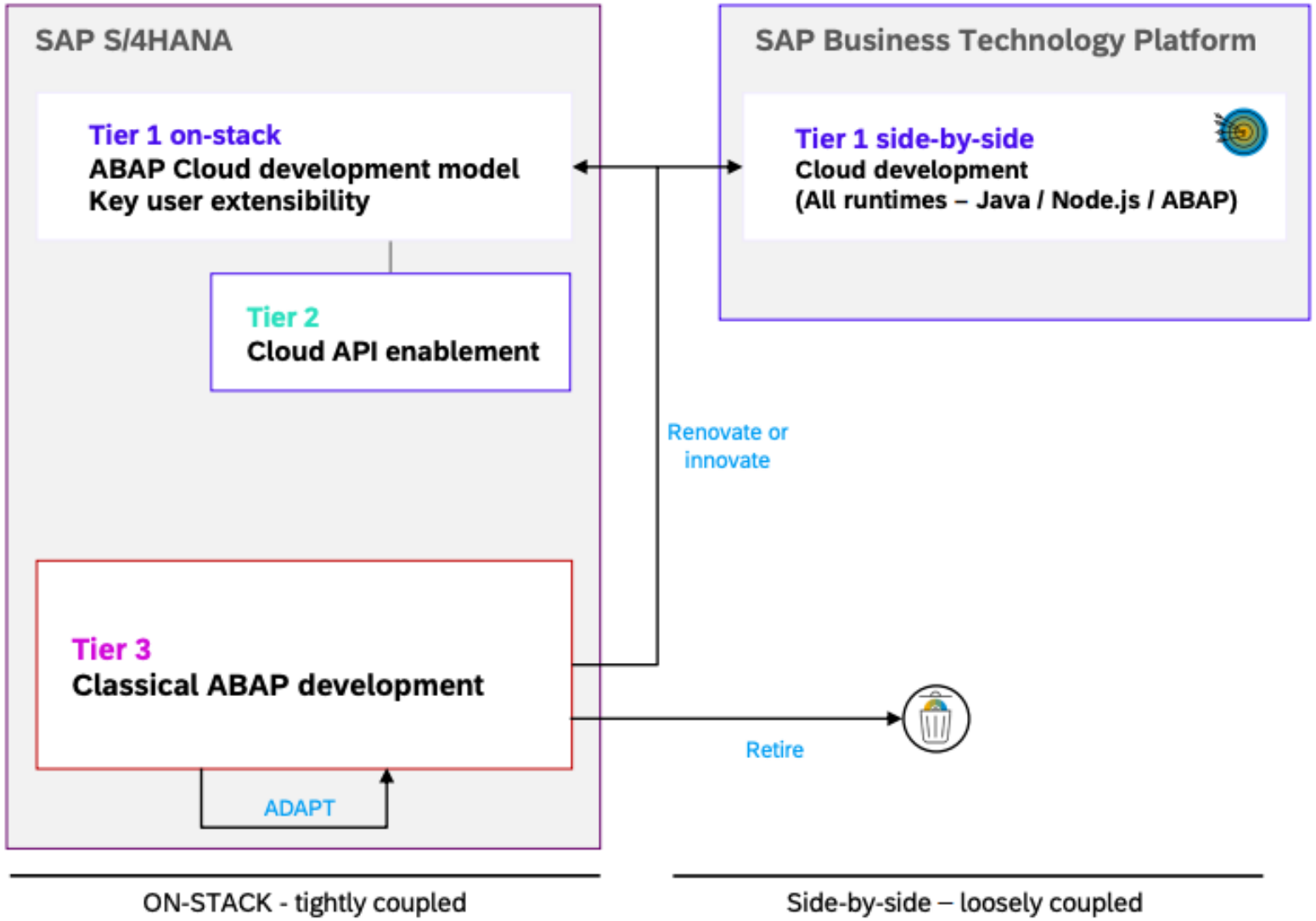
- The paradigm of keeping the core clean is integrated into the **end-to-end concept for operations**.
- **Release management** is an established foundation for a clean core; the latest release should always be targeted.
- **Housekeeping** activities that are in line with SAP Best Practices are pursued, and the distribution of roles and responsibilities that are agreed with SAP are followed.
- It is agreed that SAP performs maintenance for technology within the preapproved **contractual maintenance periods (CMPs)**.

- Establish keeping the core clean as an integral part of the end-to-end operations concept:
  - Consider “keep clean” as an IT service to add business value and establish IT as a service provider to own the end-to-end view and end-to-end processes.
  - Integrate monitoring and alerting of a “keep clean” process into the overall concept for operations to have an integrated view on KPIs for affected areas that define a clean core (integration, extensibility, processes, data).
  - Establish procedures for event management and escalations that are in line with the established governance models for integration, extensibility, processes, and data. Consider using SAP’s operations platforms to achieve this – SAP Cloud ALM or SAP Solution Manager.

- Release management is based on two core principles:
  - CMPs create preapproved monthly maintenance windows that are agreed by all involved parties and adequately documented.
  - Only in exceptional rare cases, it is requested to skip maintenance windows ("opt out").

- Regular housekeeping is established and supports keeping the core clean, for example:
  - Background job management (including approval, documentation, monitoring, and improvement) is implemented to contribute to an efficient utilization of infrastructure.
  - The usage of file interfaces for importing or exporting data from or into the system is avoided to strengthen security, maintainability, and consistency.
  - End-user authorizations are reviewed and adapted on a regular basis. Unneeded authorizations are unassigned from users and discontinued.

Extensibility Dimension: Three-Tier Extensibility Model



Tier 1 – Cloud extensibility model

Development of cloud-ready and upgrade-stable applications and extensions

Same development model as used in SAP S/4HANA Cloud, public edition

Default for new extensions and custom apps

Tier 2 – Cloud API enablement

Extends and enables tier 1 for private cloud and on-premise

Mitigates missing public SAP APIs or extension points:

- Develop custom wrapper objects for not released SAP objects to be used in tier 1
- Retire wrapper once a released SAP API is available

Tier 3 – Classic ABAP extensions

Legacy/existing custom ABAP code or new on-stack extensions code that cannot follow the rules of tier 1 and 2

Guidance to avoid and reduce the content in tier 3

source: Explaining Extensibility Model Best Practices

Clean Core Approach

3R STRATEGY

Transitioning customizations from one version of enterprise software to the next is one of the most challenging technological hurdles and a significant threat to return on investment during upgrades.

Therefore, the establishment and maintenance of a modification-free Digital Core is crucial. This allows extensions and innovations of your standard business processes, which is an essential capability in your journey towards adopting SAP S/4HANA and transitioning to the cloud.

As you transition from a legacy ERP system to SAP S/4HANA, it's vital to have a strategic plan for your custom code. The strategy for your custom code will guide you on how to handle your existing custom code during the transition. Generally, the three options are - retiring, refactoring, or rebuilding the code.

A starting point for a custom code strategy is SAP Intelligent Custom Code Management, it is a packaged service that helps you analyze existing custom code and define your 3R strategy for it.

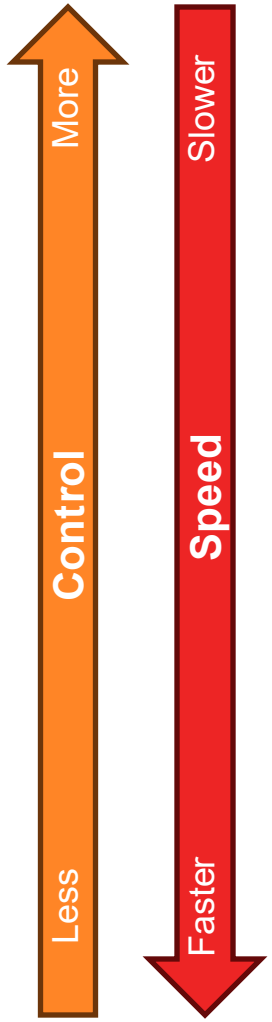
RETIRE
Retire involves the discontinuation of current custom code. This might be due to the code no longer being in use or because a standard functionality that fulfils the required capabilities is now available.

REFACTOR
Refactoring involves modifying custom code to ensure that it is compatible with the new SAP S/4 HANA system. An example is moving existing custom code to SAP BTP and adjusting it to the ABAP Cloud Programming model.

REBUILD
Rebuilding involves the redesigning of critical innovations and extensions as well as relocating them onto a future proof platform with SAP BTP. This ensures high productivity, enabling you to drive extensions and innovations.

source: SAP Clean Core - SAP Clean Core (brandcast.io)

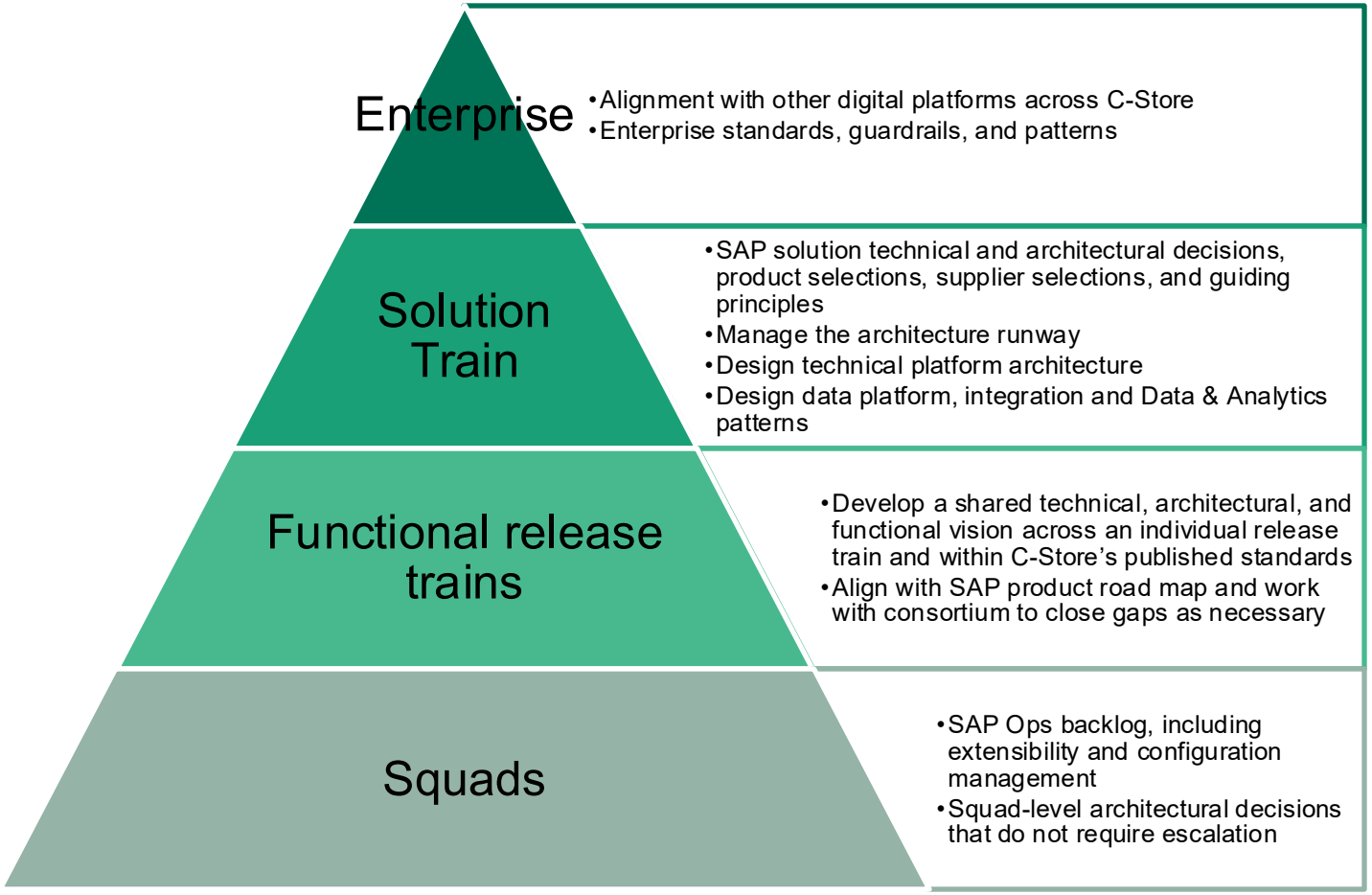
Architecture Governance



Key roles

- Cross-platform enterprise architect
- SAP architects
- Digital architects
- Integration architects
- Data architects
- Functional system architects
- Engineering
- Design architects
- Solution design

Key responsibilities and decision authority



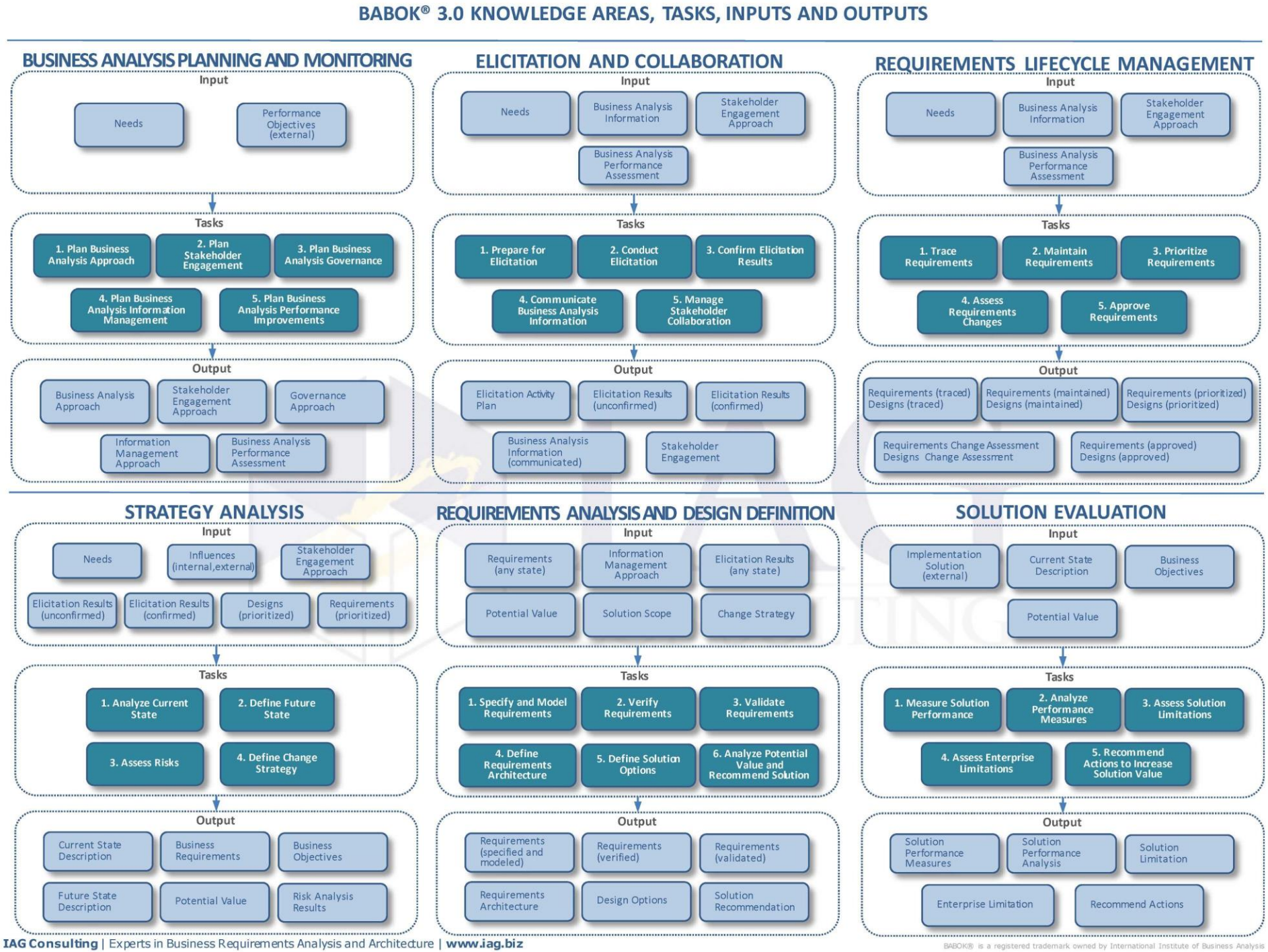
Pattern Classification

Classification	Technical Options	Upgrade Implications
GREEN	• Standard configuration as intended to be used by SAP • Side-by-side extensions or SAP's custom business object app using SAP Business Technology Platform (SAP BTP) • SAP Fiori design system for enhanced user experience	• Expected to have no impact to upgrades, although some level of regression testing will be required • Allowed in single-tenant and multi-tenant editions
AMBER	• SAP's custom fields and logic app to define additional custom fields • Creation of custom configuration tables • Using predefined Customizing Includes, Append Structures, or Extension Includes to define additional custom fields	• Requires regression testing after upgrade • Upgrades could cause these types of extensions to no longer work as designed or not work at all • Increases regression testing requirements during change windows, future deployments, application of support packs, and upgrades • Total cost of ownership needs to be evaluated across the entire set of yellow extensions, not individually • Allowed in single-tenant edition but not able to migrate to multi-tenant edition
RED	• Classic extensions using ABAP workbench • Traditional user exits, use of enhancement spots, and other methods to add callouts within core SAP code • Copying / cloning standard SAP code to apply modifications • Configuring or utilizing standard fields in a manner that was not intended by SAP • Modifications to core SAP code	• Requires redeployment or redevelopment after upgrade • Typically involves changing or adding callouts within core SAP code • May not be allowed in single-tenant and not allowed in multi-tenant editions

Balancing Speed and Control

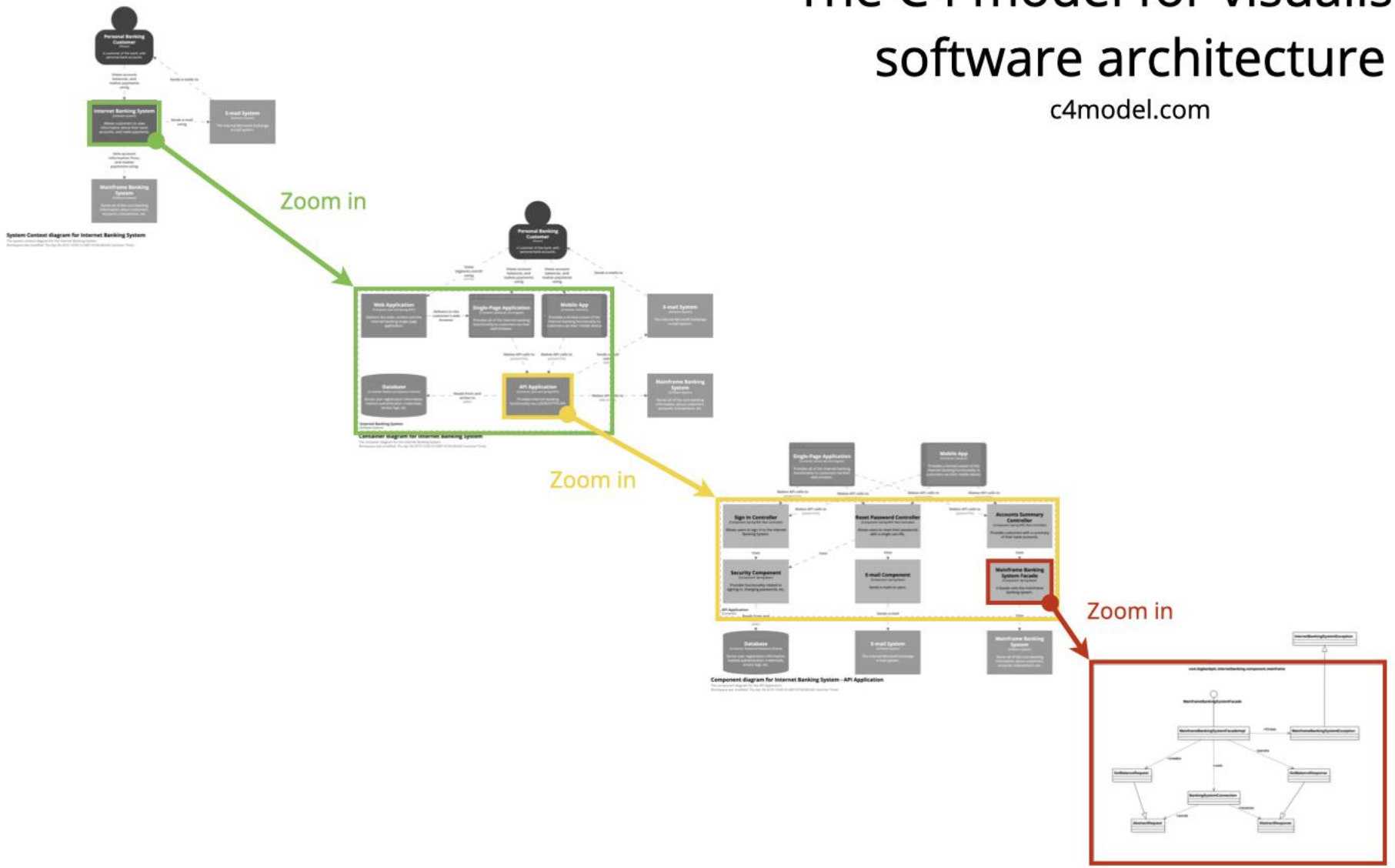
Classification	State	Action
GREEN	When the light is green, you may proceed through the intersection provided it is safe to do so. Watch out for vehicles disobeying the traffic control signals because many serious crashes are caused by vehicles driving through intersections against a red light.	The functional system architects are empowered to approve green extensions without presenting them to the architecture runway. They review the overall design quality to ensure they use green patterns and have a high-quality design.
AMBER	A yellow traffic signal light means “CAUTION.” The light is about to turn red. When you see a yellow traffic signal light, stop (if you can do so safely). If you cannot stop safely, cautiously cross the intersection.	Yellow extensions incur a level of technical debt. The functional system architects and the architecture runway should understand the technical debt and trade-offs before proceeding with caution.
RED	A stop sign means that you must stop your vehicle before the intersection. After checking for oncoming traffic, you may proceed if it is safe to do so.	Red extensions incur a higher technical debt and could prevent C-Store from achieving our intent. The system architects should stop, understand the problem, and explore other approaches. A red extension could be the right solution, but we want to be deliberate in decision-making.

BABOK 3.0


BABOK® 3.0 KNOWLEDGE AREAS, TASKS, INPUTS AND OUTPUTS
BUSINESS ANALYSIS PLANNING AND MONITORING
Input
Needs
Performance Objectives (external)
Tasks
1. Plan Business Analysis Approach
2. Plan Stakeholder Engagement
3. Plan Business Analysis Governance
4. Plan Business Analysis Information Management
5. Plan Business Analysis Performance Improvements
Output
Business Analysis Approach
Stakeholder Engagement Approach
Governance Approach
Information Management Approach
Business Analysis Performance Assessment
ELICITATION AND COLLABORATION
Input
Needs
Business Analysis Information
Stakeholder Engagement Approach
Business Analysis Performance Assessment
Tasks
1. Prepare for Elicitation
2. Conduct Elicitation
3. Confirm Elicitation Results
4. Communicate Business Analysis Information
5. Manage Stakeholder Collaboration
Output
Elicitation Activity Plan
Elicitation Results (unconfirmed)
Elicitation Results (confirmed)
Business Analysis Information (communicated)
Stakeholder Engagement
REQUIREMENTS LIFECYCLE MANAGEMENT
Input
Needs
Business Analysis Information
Stakeholder Engagement Approach
Business Analysis Performance Assessment
Tasks
1. Trace Requirements
2. Maintain Requirements
3. Prioritize Requirements
4. Assess Requirements Changes
5. Approve Requirements
Output
Requirements (traced) Designs (traced)
Requirements (maintained) Designs (maintained)
Requirements (prioritized) Designs (prioritized)
Requirements Change Assessment Designs Change Assessment
Requirements (approved) Designs (approved)
STRATEGY ANALYSIS
Input
Needs
Influences (internal,external)
Stakeholder Engagement Approach
Elicitation Results (unconfirmed)
Elicitation Results (confirmed)
Designs (prioritized)
Requirements (prioritized)
Tasks
1. Analyze Current State
2. Define Future State
3. Assess Risks
4. Define Change Strategy
Output
Current State Description
Business Requirements
Business Objectives
Future State Description
Potential Value
Risk Analysis Results
REQUIREMENTS ANALYSIS AND DESIGN DEFINITION
Input
Requirements (any state)
Information Management Approach
Elicitation Results (any state)
Potential Value
Solution Scope
Change Strategy
Tasks
1. Specify and Model Requirements
2. Verify Requirements
3. Validate Requirements
4. Define Requirements Architecture
5. Define Solution Options
6. Analyze Potential Value and Recommend Solution
Output
Requirements (specified and modeled)
Requirements (verified)
Requirements (validated)
Requirements Architecture
Design Options
Solution Recommendation
SOLUTION EVALUATION
Input
Implementation Solution (external)
Current State Description
Business Objectives
Potential Value
Tasks
1. Measure Solution Performance
2. Analyze Performance Measures
3. Assess Solution Limitations
4. Assess Enterprise Limitations
5. Recommend Actions to Increase Solution Value
Output
Solution Performance Measures
Solution Performance Analysis
Solution Limitation
Enterprise Limitation
Recommend Actions
IAG Consulting | Experts in Business Requirements Analysis and Architecture | www.iag.biz
BABOK® is a registered trademark owned by International Institute of Business Analysis

The C4 model for visualising software architecture

c4model.com



Level 1
Context

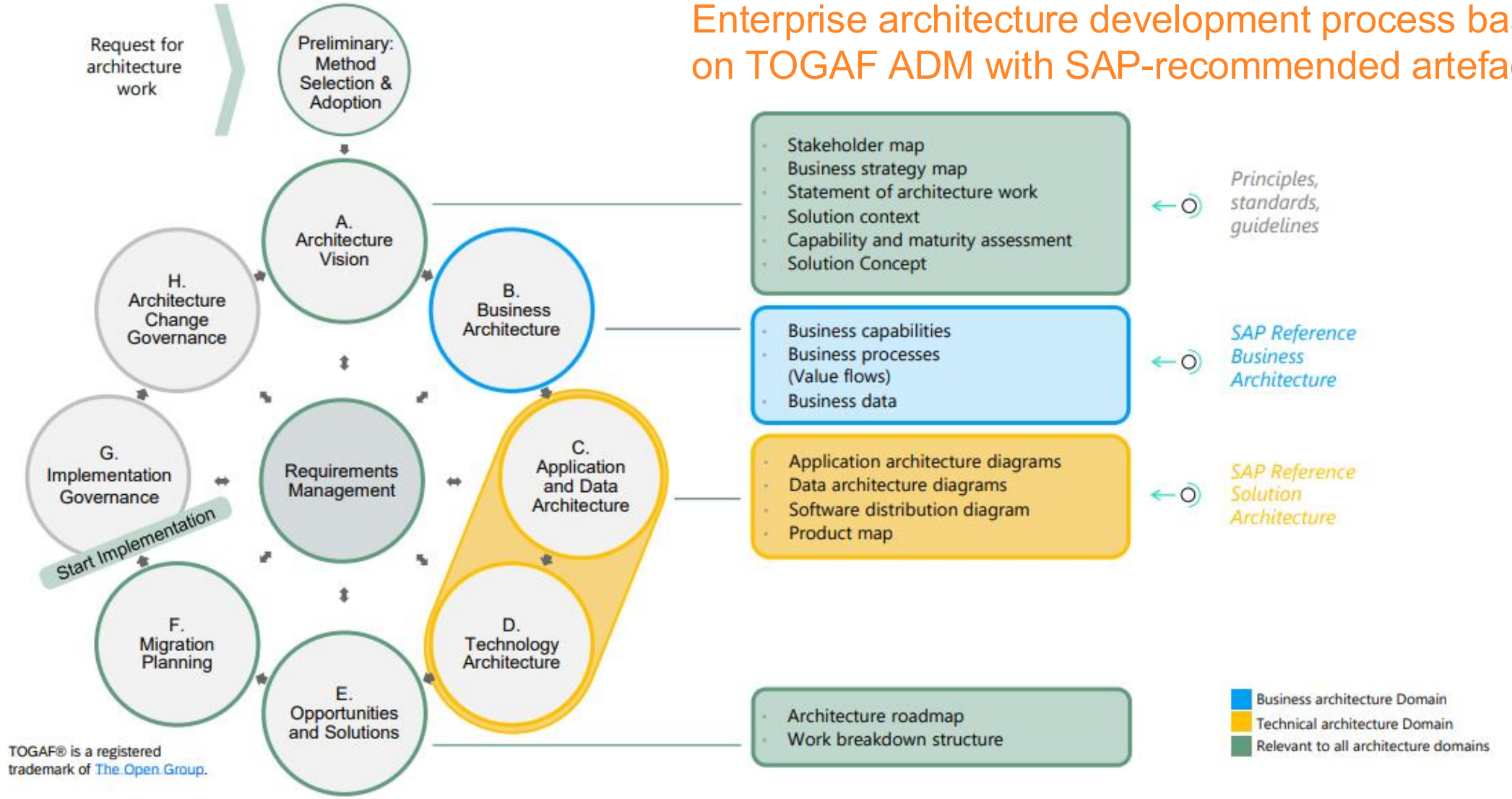
Level 2
Containers

Level 3
Components

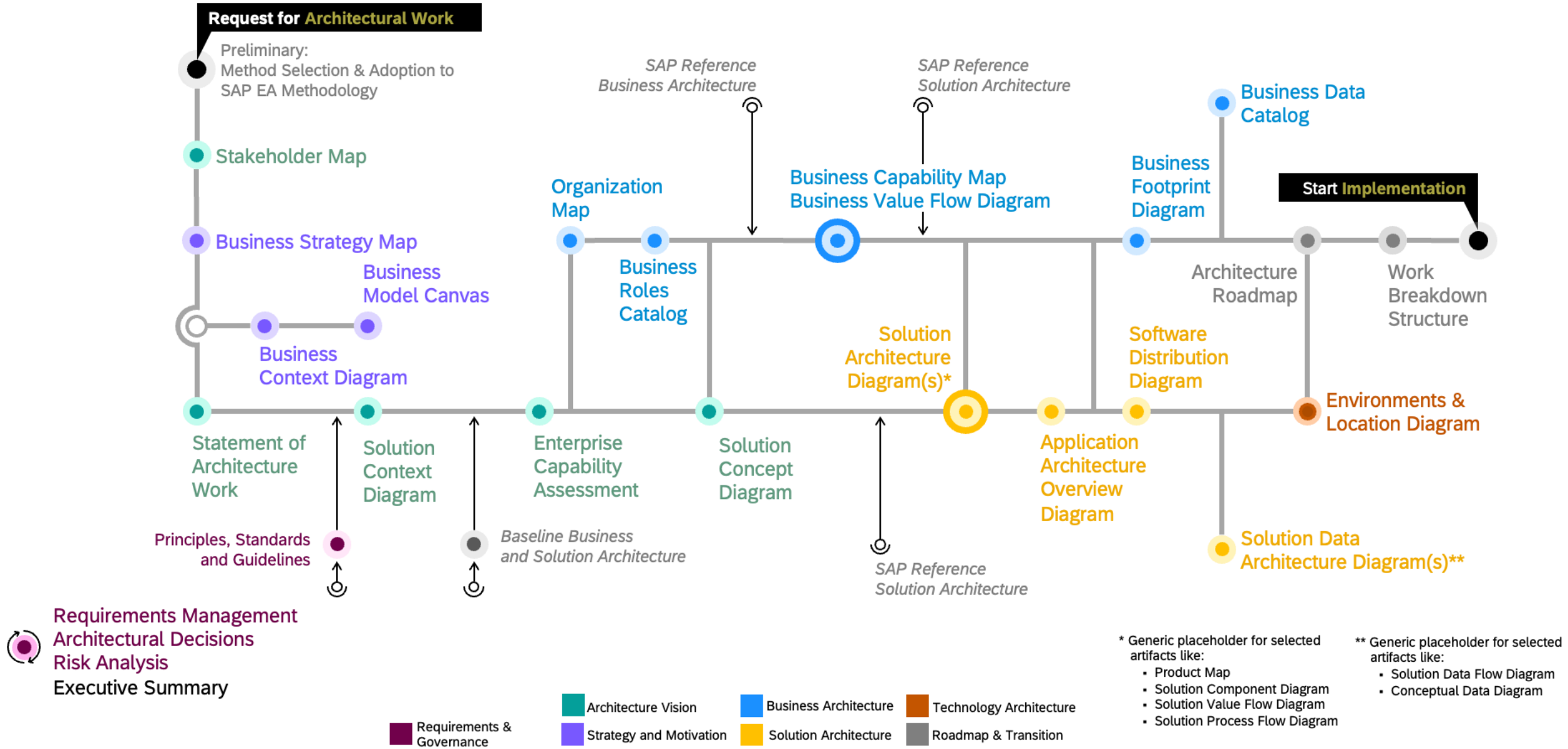
Level 4
Code

SAP Enterprise Architecture Framework

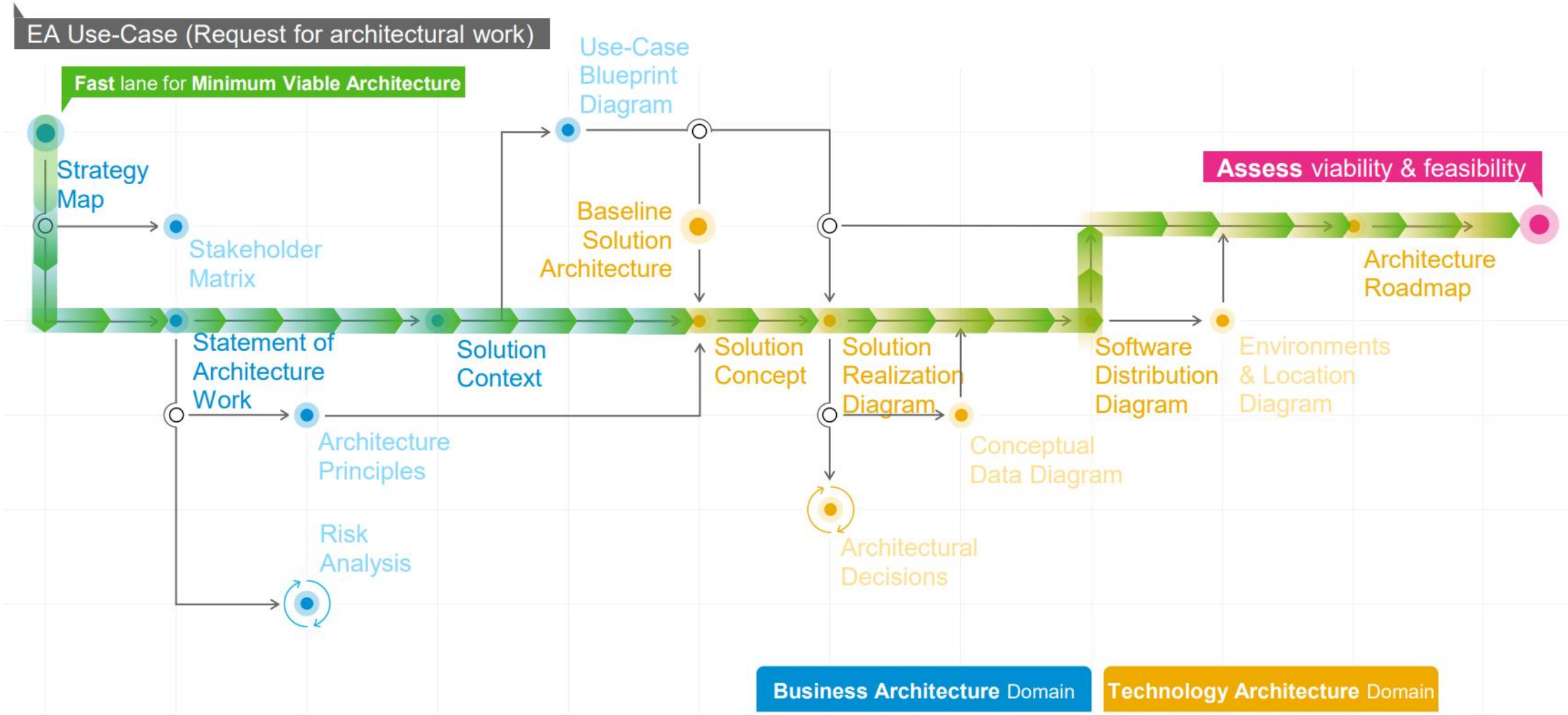
Enterprise architecture development process based on TOGAF ADM with SAP-recommended artefacts



SAP Enterprise Architecture Framework (2024)



SAP Lean EA Toolkit (Minimum Viable Architecture) (2023)



Strategy Map

Vision

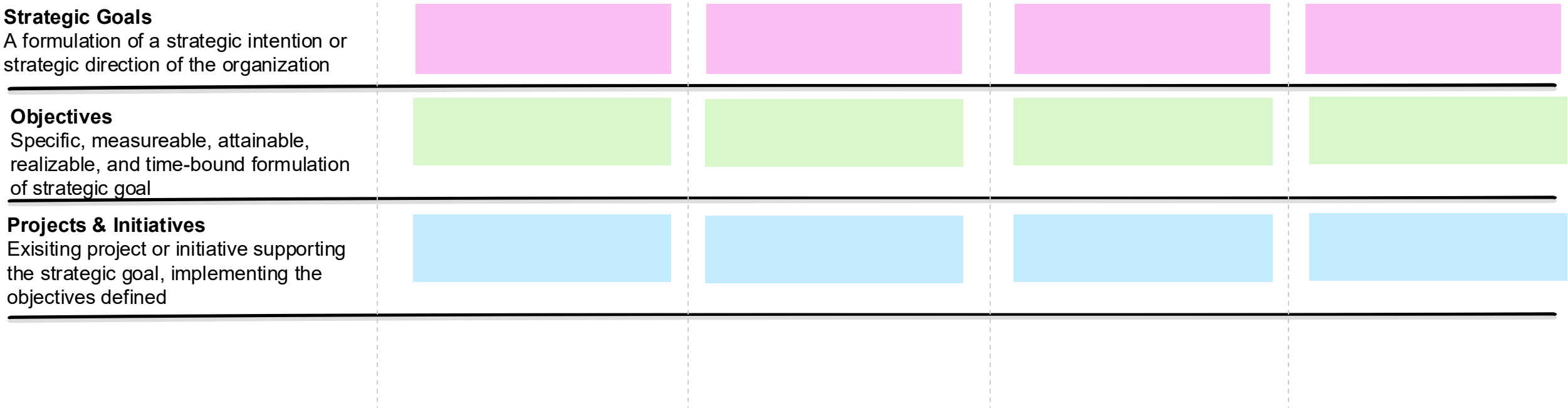
Add company's vision statement

Driver(s)

An external or internal condition(s) that motivate(s) the organization to define its goal s, such as customer and market behavior, competitive forces, legislation, etc.

Corporate Goal(s)

Add a formulation of intention(s) or direction(s) of the company or organization.



Strategy Map

Vision

Create world-changing technology that enriches people's lives.

Driver

Unleash the potential of artifical intelligence by reliable, sclable,trusted data processing inspired by quantum computing.

Corporate Goal

Global market leader in technology optimized for machine learning algorithms.

Strategic Goals	Fast and effective approval of business initiatives and research & development projects to support massive growth as well as innovation and adaptability	Engineer solutions for our customers' success with reliable, cloud-to-edge computing inspired by quantum technology	Provide new digital services supporting AI as a service	Sell digital services to customers via a marketplace. Offer API Hub to consume services
Objectives	▪ Reduce average time of approval from 25+ days to 2 days ▪ Automatic processing of at least 25% of requests with confidence	▪ Improve training speed of massive data sets by 75% until 2022 ▪ Reduce power consumption by 50% until 2022	Offer AI based services via APIs out of own data centers	A marketplace will be created to so sell services to stakeholders
Projects & Initiatives	▪ Global Growth Initiative 2025	▪ Quantum Flagship Initiative	▪ Digital Agenda 2023 ▪ Data Center Strategy	▪ Digital Agenda 2023 ▪ Project STELLAR

Statement of Architecture Work

source: TOGAF Standard, Version 9.2

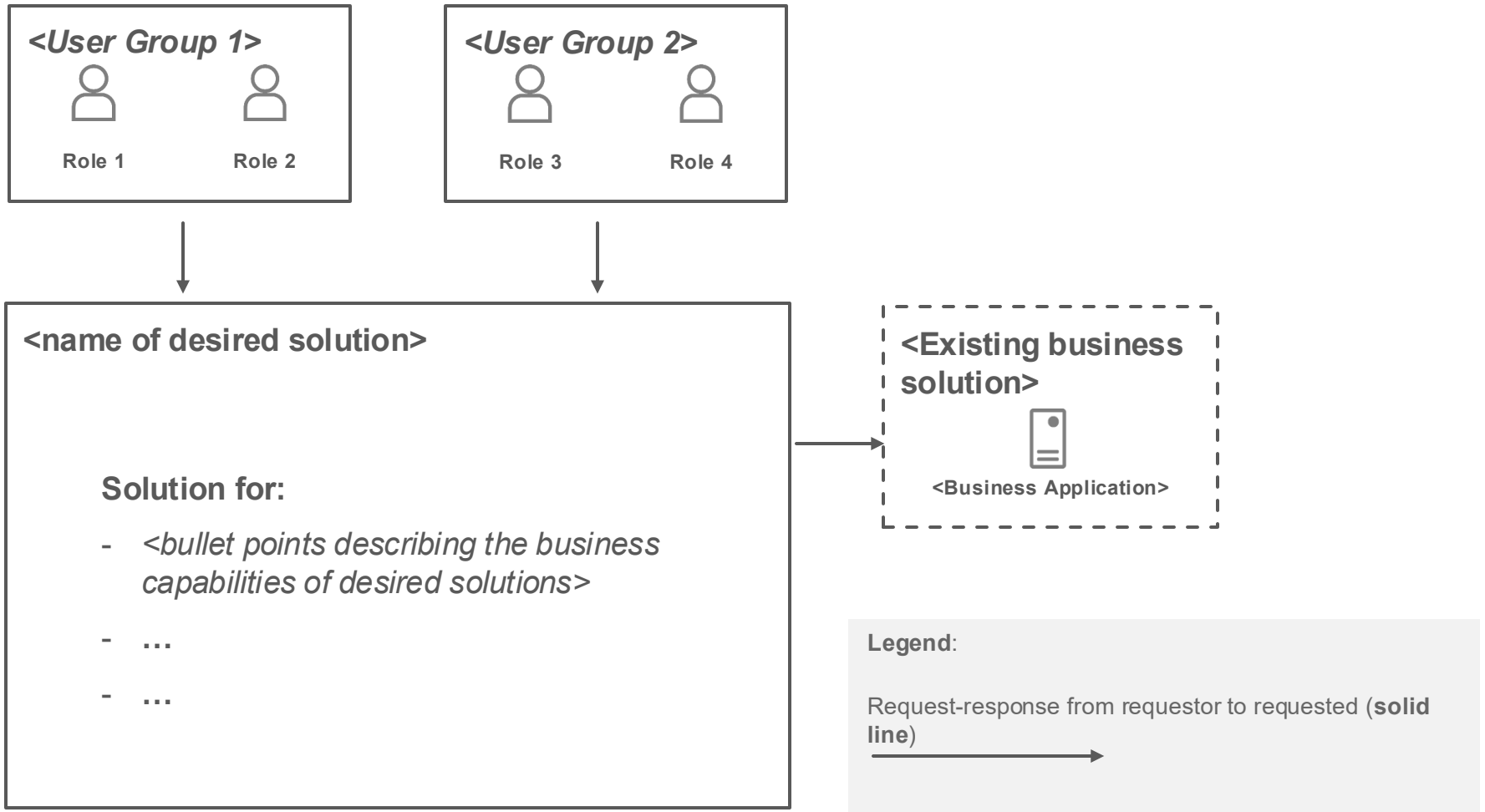
1. Title	*<Title of the project>*
2. Architecture project request and background	*<Short description of the reason for the project and its background>*
3. Architecture project description and scope	*<Brief description of the project and its scope>*
4. Overview of architecture vision (high-level-architecture)	*<Provide a high-level picture of the target architecture, including core functional components and users / roles>*
5. Roles, responsibilities, and work products	*<List all the roles and their responsibilities in the project>*
6. Acceptance criteria and procedures	*<Describe the acceptance criteria and acceptance procedures of the project>*
7. Architecture project plan & schedule	*<Provide the main milestones and their schedule for delivering the project>*

Statement of Architecture Work

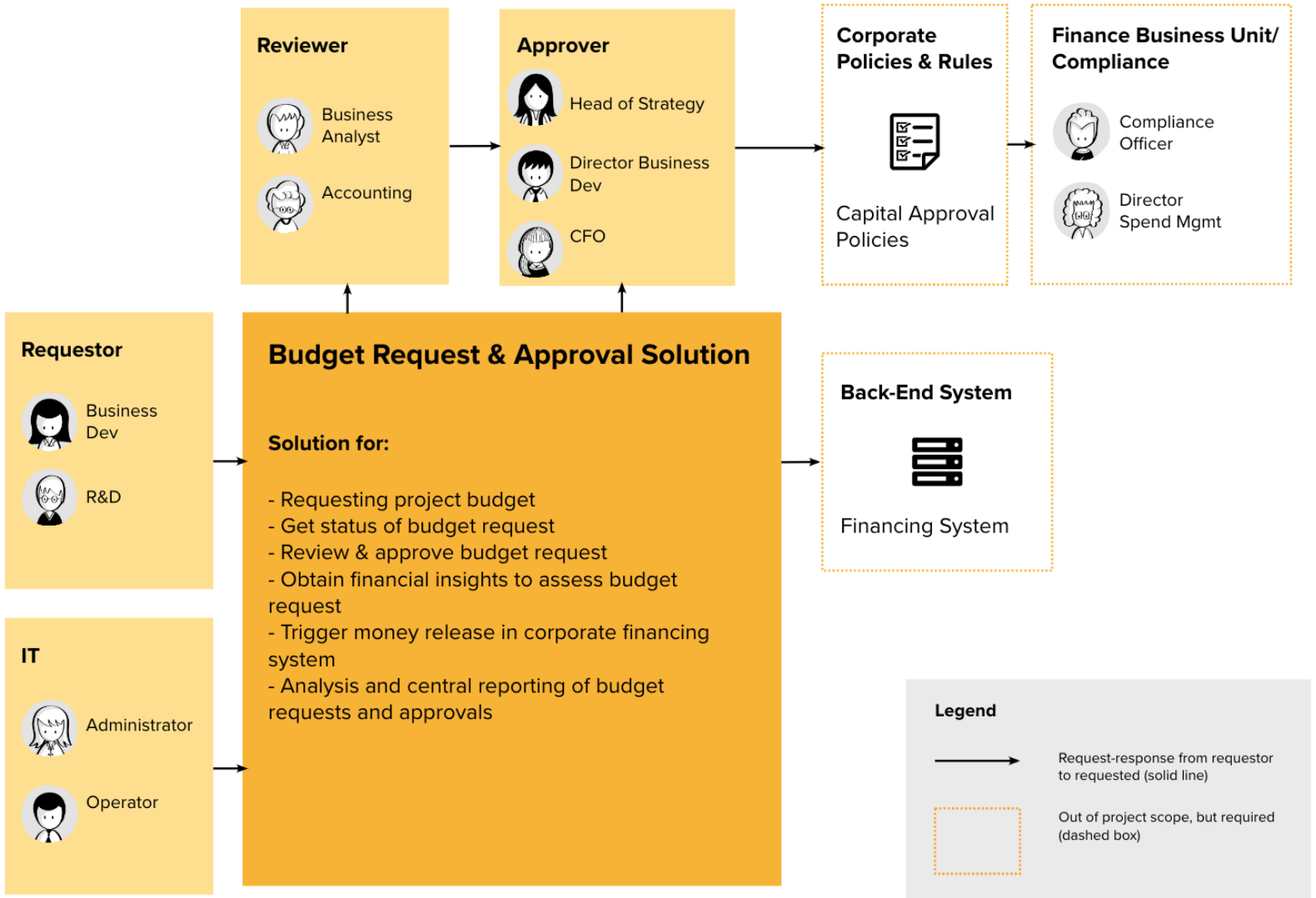
source: TOGAF Standard, Version 9.2

1. Title	Budget Request & Approval Solution
2. Architecture project request and background	To support the Global Growth Initiative 2025, design, develop, and operate a solution to support the financing of more than 2000 employees working on business dev. and R&D projects worldwide.
3. Architecture project description and scope	Plan for the development of a finance request & approval workflow aplication integrating with the existing finance system.
4. Overview of architecture vision (high-level-architecture)	
5. Roles, responsibilities, and work products	▪ **Business Architecture**: Director Business Development (Rocket Chips), Financial Analyst (Rocket Chips), Lead Architect (you) ▪ **Technical Architecture**: Lead Architect (you), Technical Architect (Vendor), Lead Developer (Rocket Chips), Head of IT (Rocket Chips) ▪ **Work products**: All artifacts from Lean EA toolkit without Architecture Principles, Risk Analysis and Use-Case Blueprint Diagram. Delivery format and content via PowerPoint and Word.
6. Acceptance criteria and procedures	▪ Approval for business architecture: Director Business Dev. (Rocket Chips), CFO (Rocket Chips) ▪ Approval for technical architecture: Head of IT (Rocket Chips)
7. Architecture project plan & schedule	▪ Architecture Roadmap work product, proposal for license and implementation effort

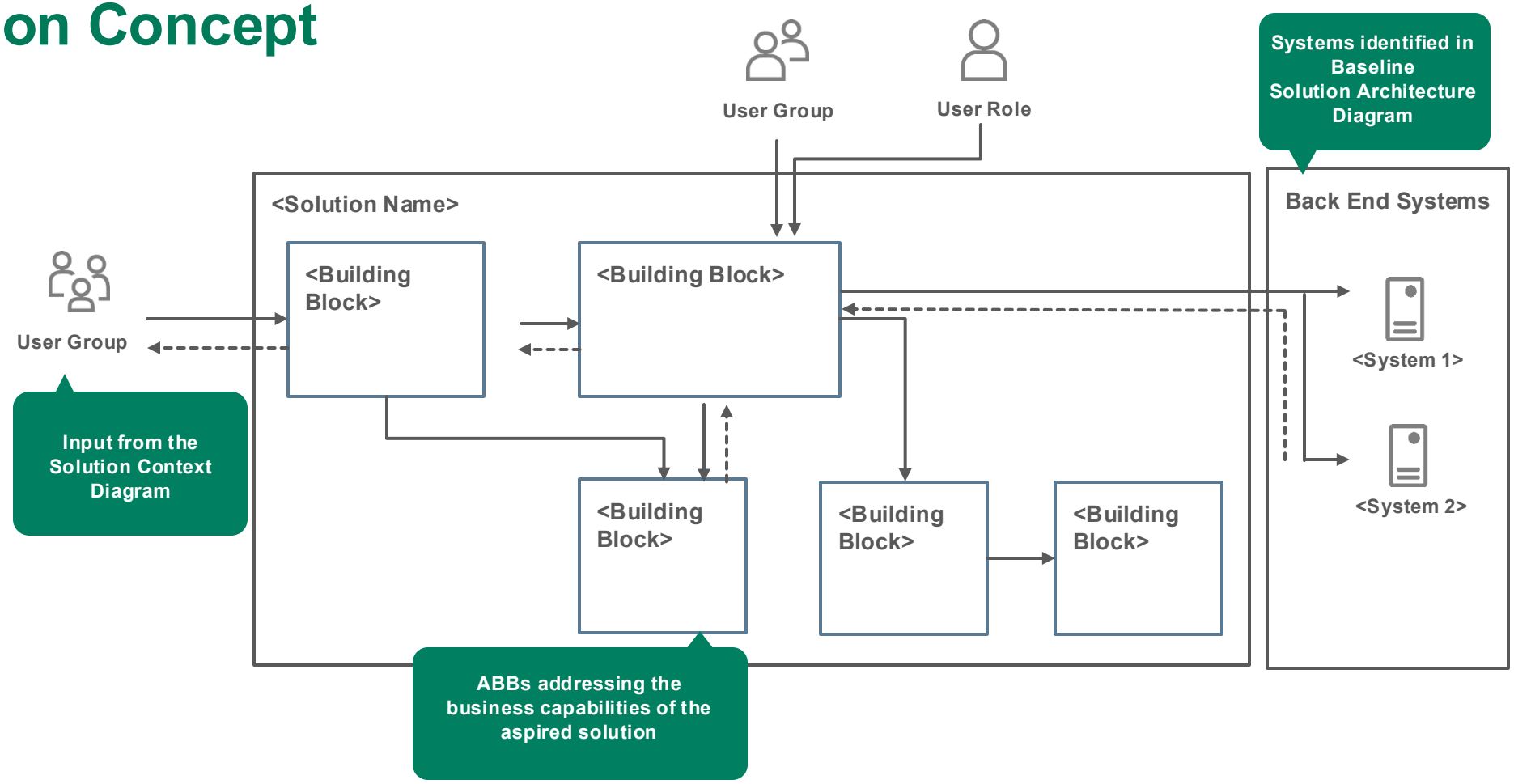
Solution Context



Solution Context



Solution Concept

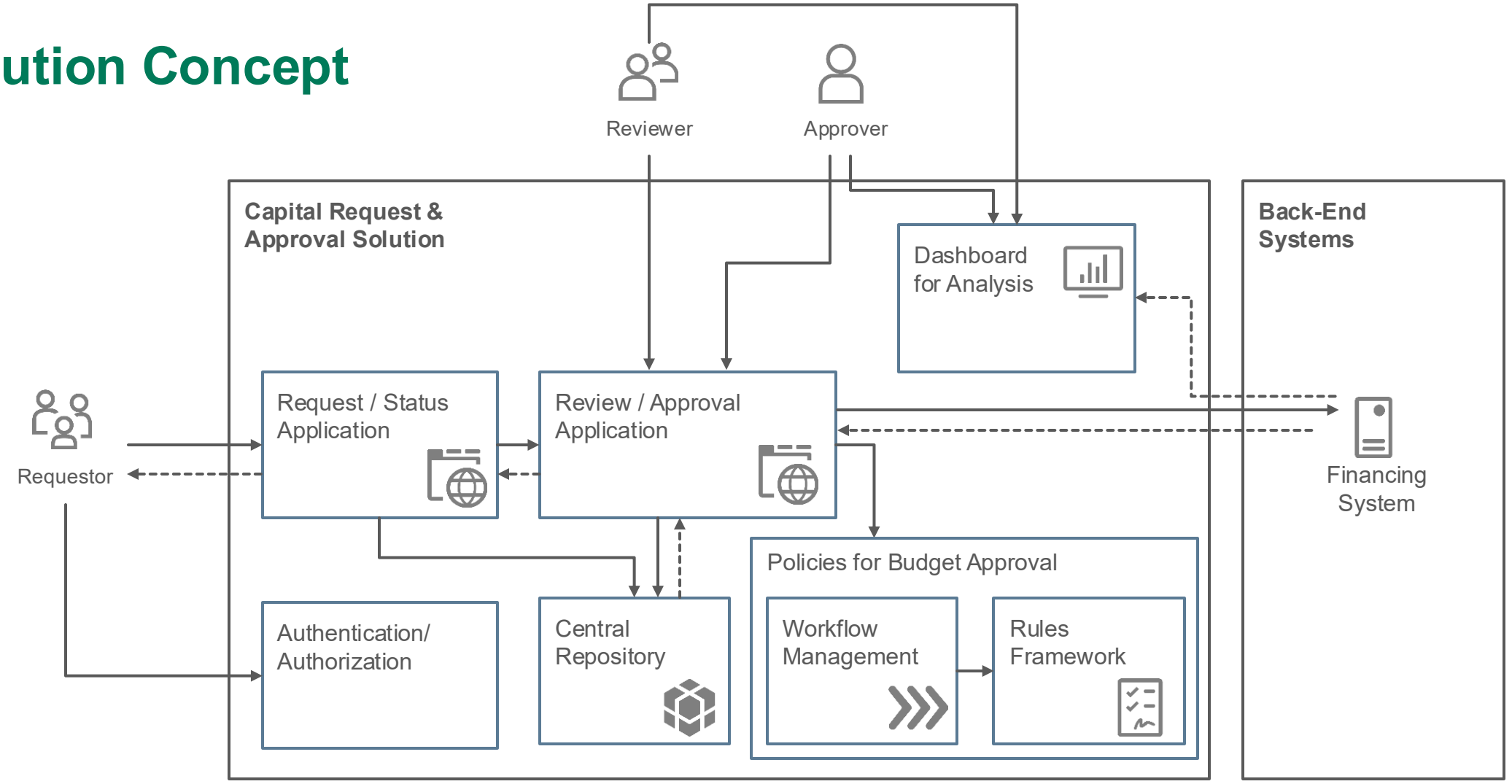


Legend:

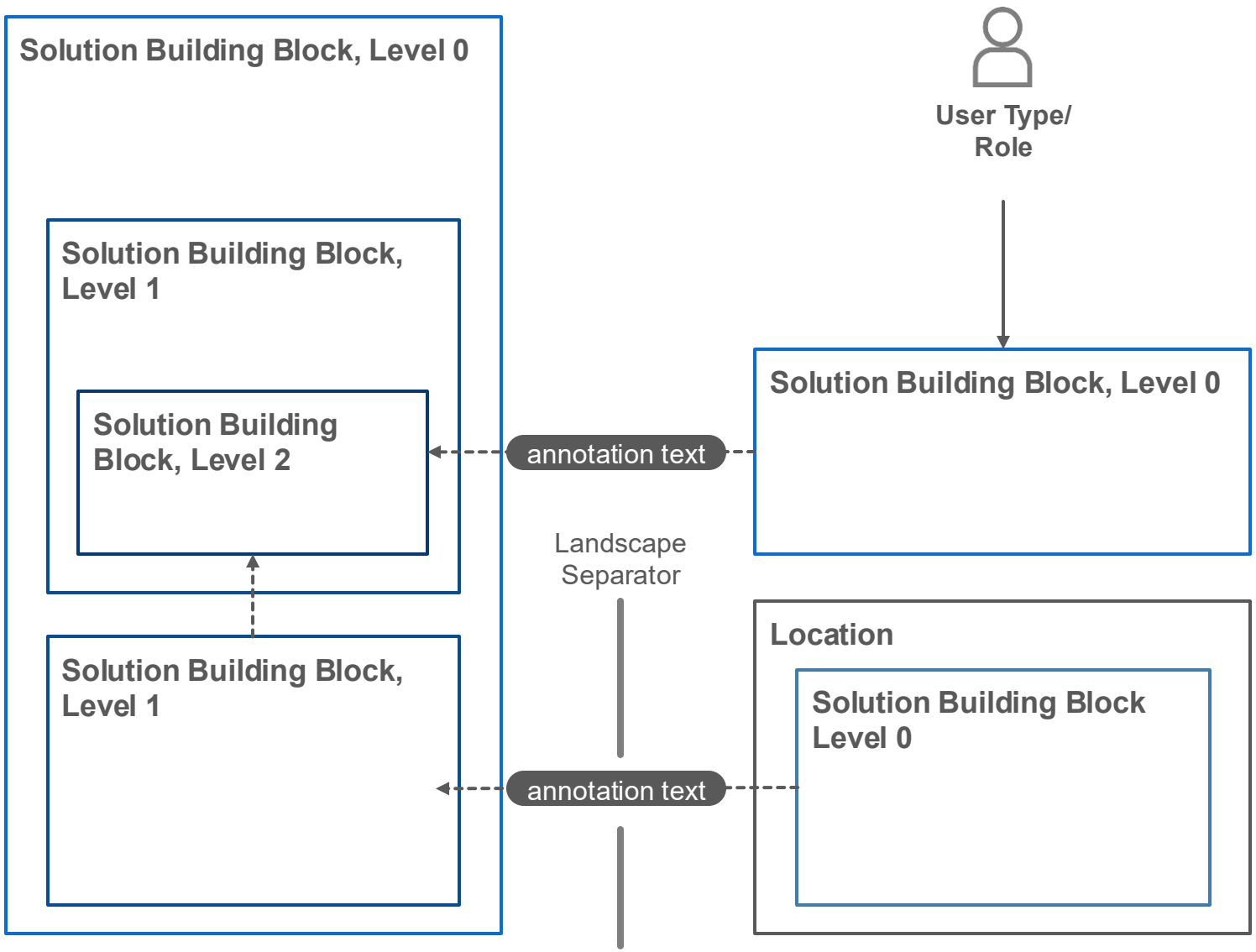
Request response: ——————→

Information flow from data source to target: ←- - - - - - - - - -

Solution Concept



Solution Realisation

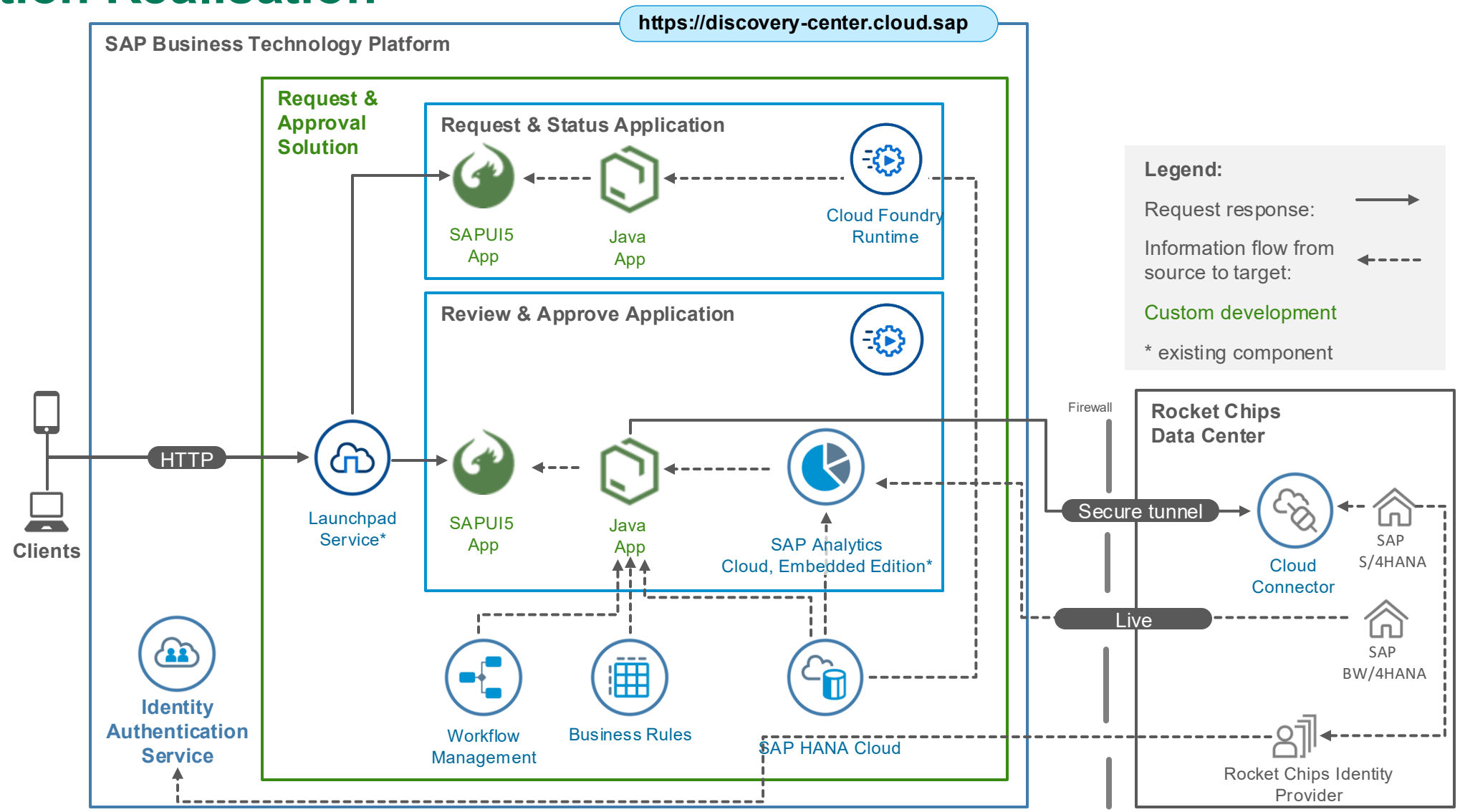


Legend:

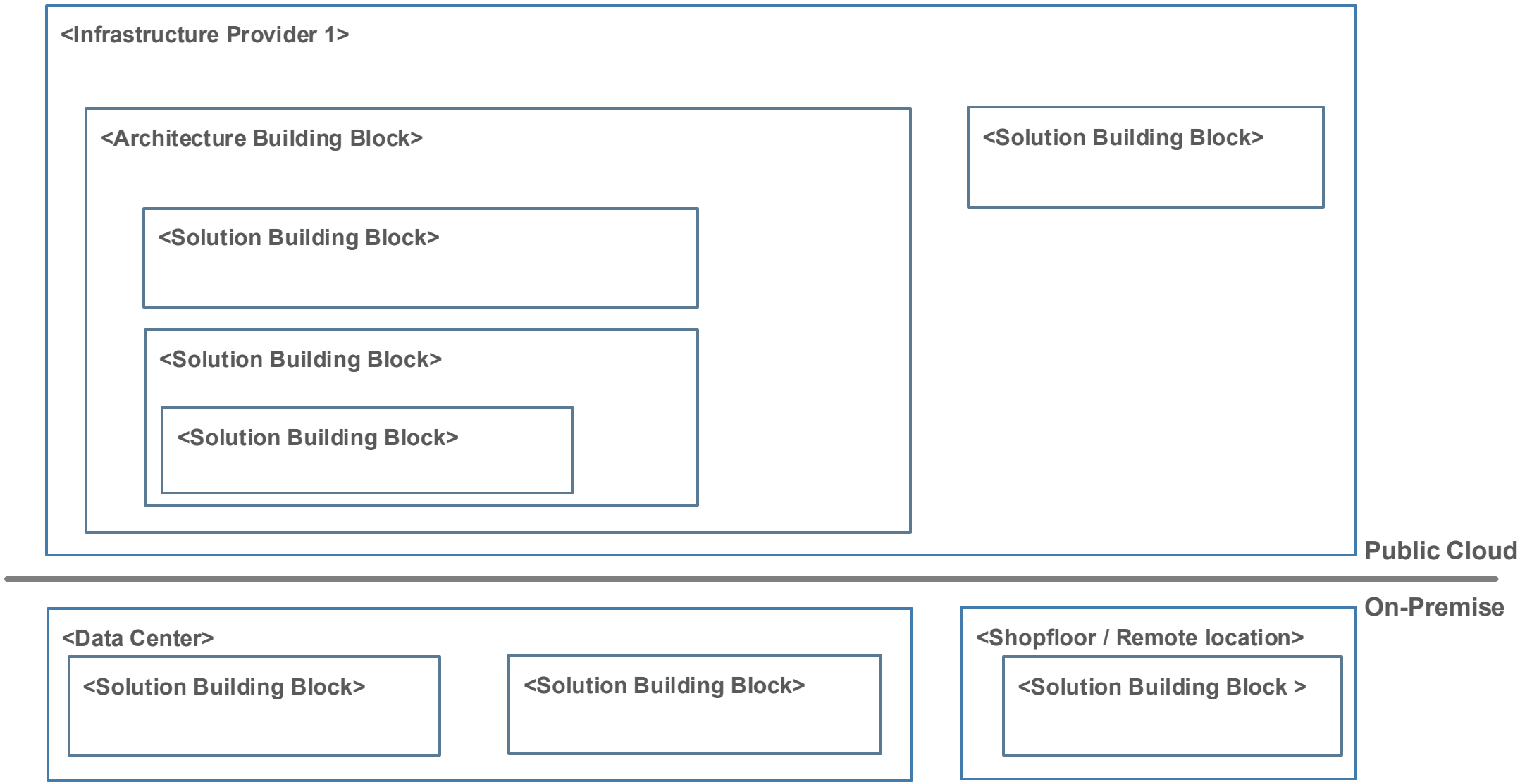
Request response: ⟶

Information flow from source to target: ⟵

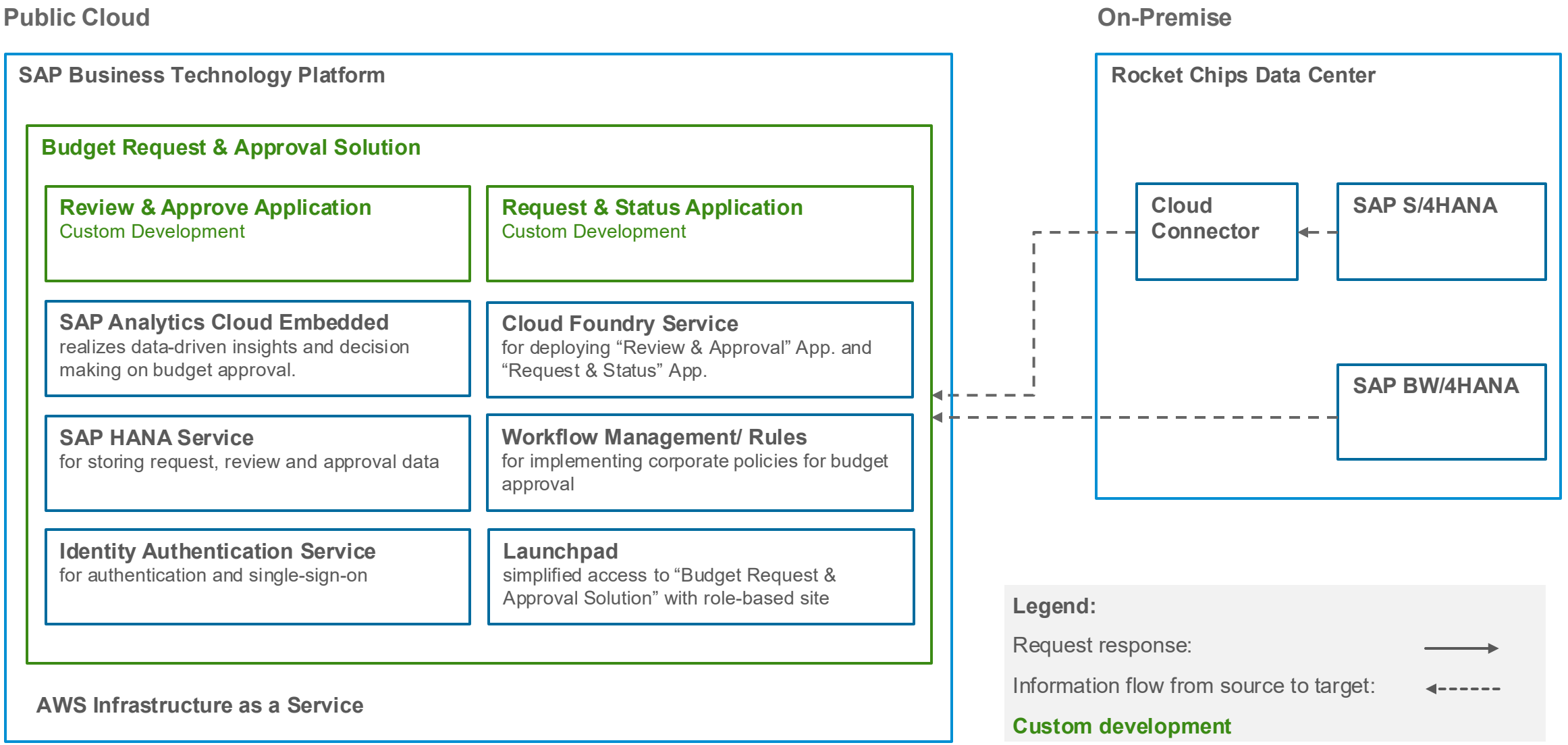
Solution Realisation



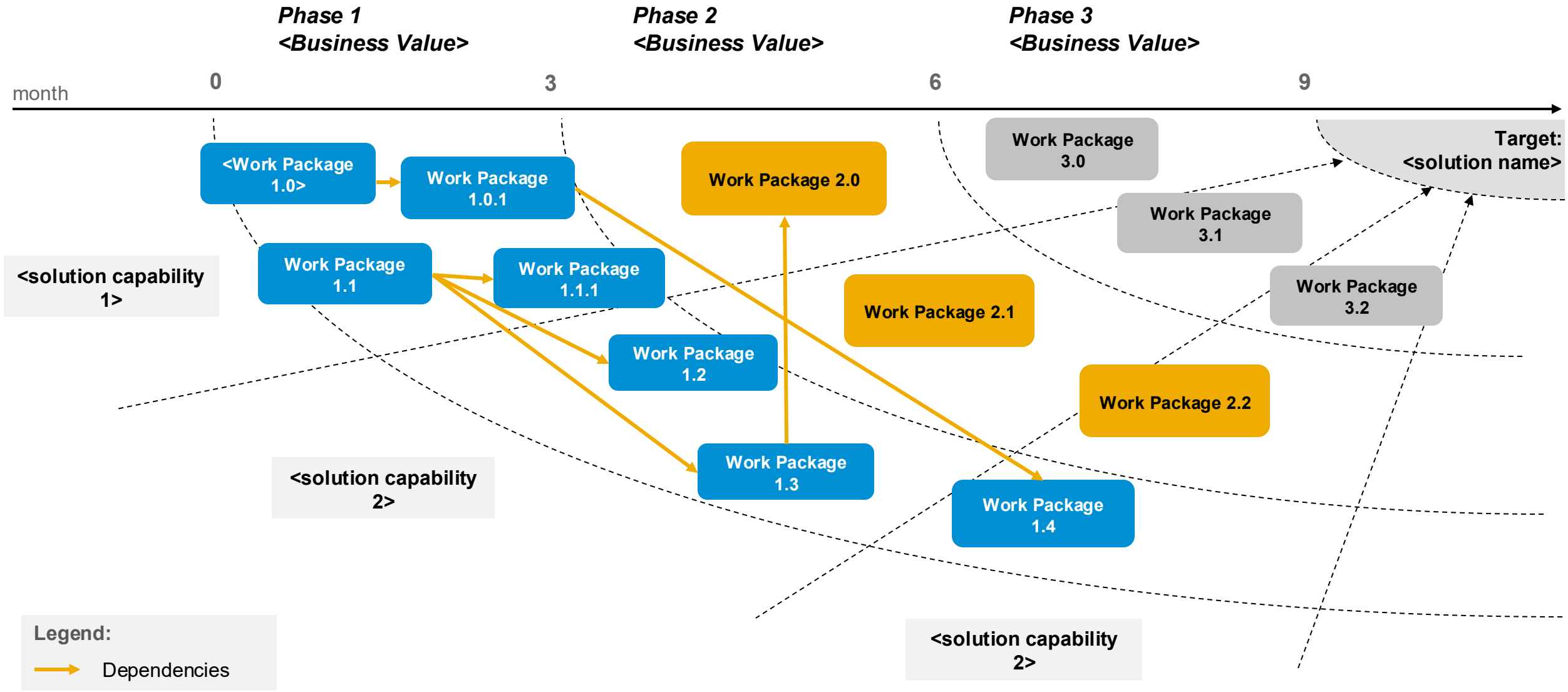
Solution Distribution



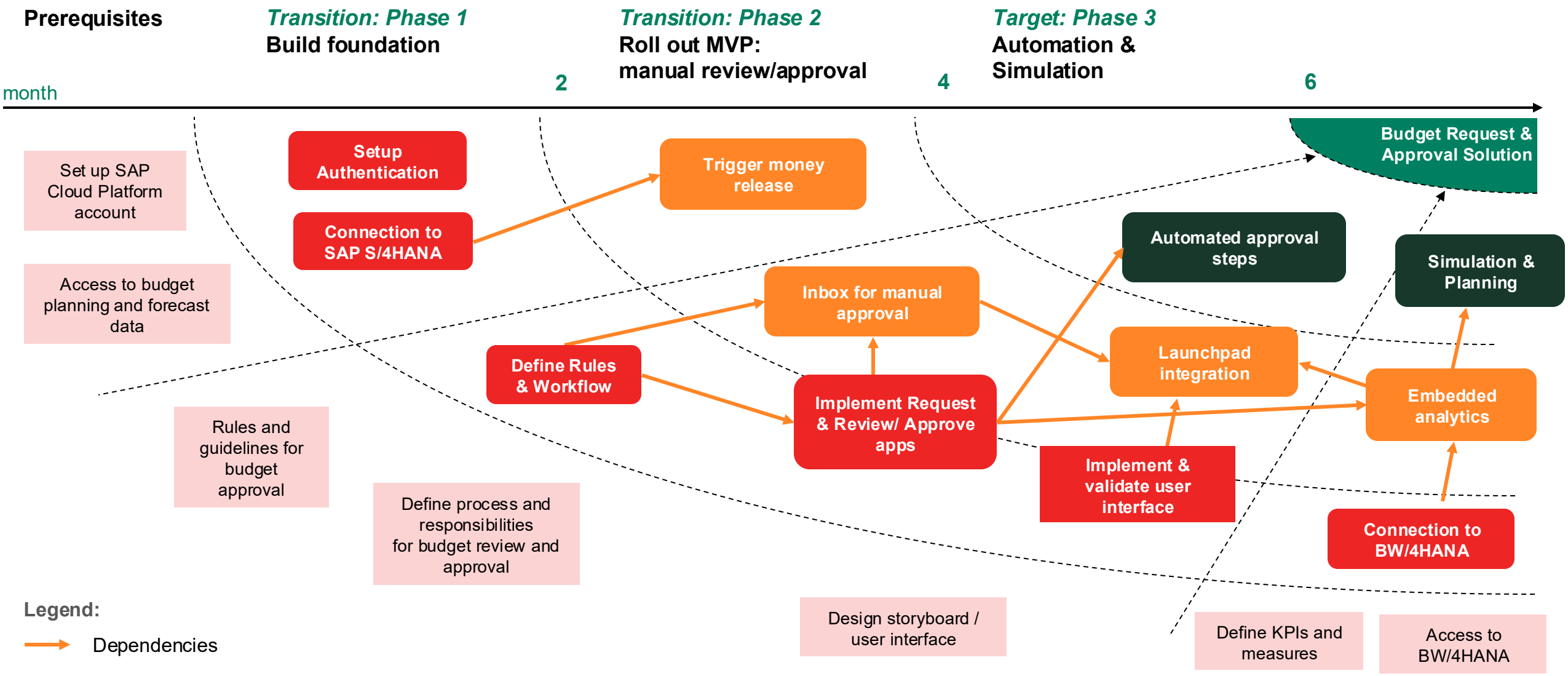
Solution Distribution



Architecture Roadmap



Architecture Roadmap



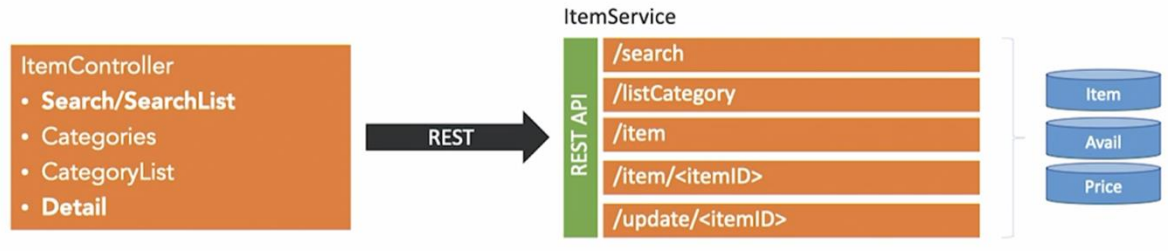
Architecture Approach

	Advantages	Disadvantages	Best For
Monolithic	● Low cost to build and maintain due to tightly coupled structure ● Requires relatively less technical skills to manage ● Easily launched with prebuilt features and out-of-the-box functionality	● Challenging to customize for variety of touchpoints ● Lacks flexibility ● Slow and cumbersome to build new features, unless offered by service provider ● Unable to mix and match services to build impactful experiences and capitalize on market opportunities ● Adopting new capabilities requires upgrades to the entire platform	Small and midsized businesses (SMBs) that are just starting out with ecommerce and don't need custom applications. This approach is also best for businesses that don't have access to the technical skills or budget to build and maintain microservices.
Microservices	● Can combine best-of-breed services from various vendors to maintain operational excellence ● Highly customizable with freedom to build unique user experiences ● Each service runs independently, allowing for changes without impacting other services ● Each service can be scaled independently depending on business needs	Requires a high level of technical skills to build and operate both the infrastructure and organizational processes ● Costly to maintain multiple, disparate microservices ● Can accumulate technical debt without a mature IT organization or agency ● Often lack unified business user tooling, especially if services are delivered by different vendors	Large or mature commerce companies looking for a hyper-customized ecommerce platform. These businesses have the budget, organizational maturity, and resources to build and operate across a variety of microservices.
Composable	● Gives development teams immense flexibility to make changes and build new features without impacting the rest of the platform ● Allows businesses to freely implement the features that will best serve their needs ● Reduces operational costs by only paying for needed features	● Can be costly as more microservices or PBCs are added ● Highly technical resources needed to build and maintain the platform ● Too many microservices distributed across various vendors can significantly extend implementation times and cost	Mature commerce organizations looking to become industry leaders. These businesses can innovate quickly and build fresh, creative experiences, driven by either in-house IT organizations or a trusted agency. They have both the budget and technical expertise and resources to build and maintain a customized, flexible platform.
Headless	● Opportunities to expand commerce functionality to new channels and touchpoints ● Opportunities to personalize and customize customer experiences and UI ● Ability to add best-of-breed features and capitalize on new market opportunities ● Ability to integrate third-party services ● Lower technology barrier than building a microservices-based or composable commerce platform	● Requires talented engineers familiar with a storefront language like React to build APIs ● Slower time to market due to more coordination between design and technical implementation ● Higher technical debt if IT organization isn't mature ● Stage and preview limited or not available ● Often lacks tooling that gives marketers and merchandisers control over site content	Any business looking to explore how to decompose their traditional monolithic platform. A headless platform is a small step in the direction toward composable commerce by decoupling the storefront interface layer from the back-end commerce engine. It allows businesses to adjust the customer experience by changing the storefront UI without needing to rebuild the back-end engine and processes.

Architecture Approach

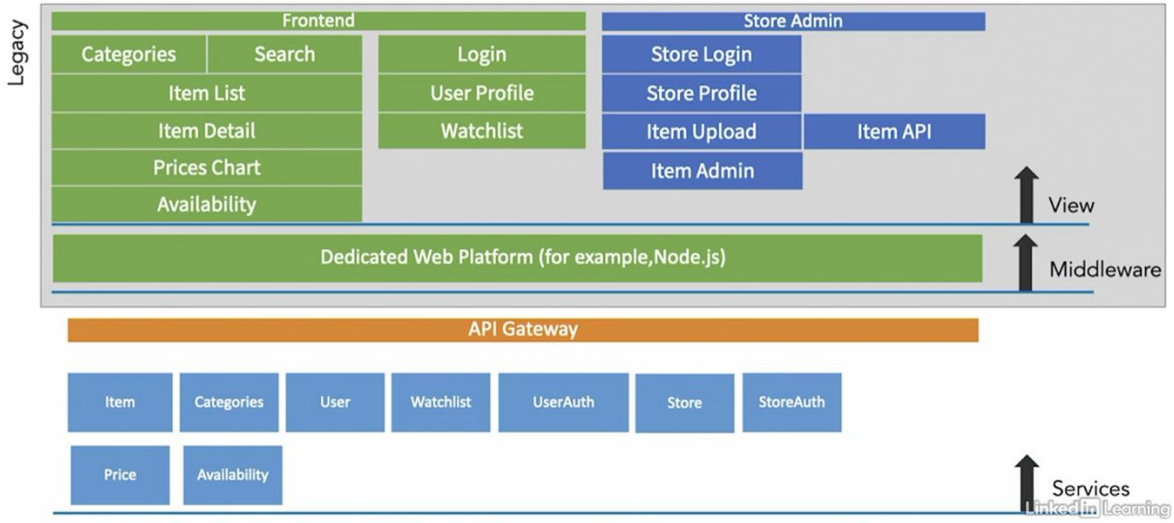
	Advantages for business leaders	Disadvantages for IT leaders	Best For
Hybrid	● Remain competitive by reacting quickly to customer, business, and market needs ● Maintain access to the testing and experimentation benefits of microservices without needing to re-platform ● Retain preintegrated business user-friendly tools to decrease reliance on developers and empower marketing and merchandising teams ● Maintain a shared dataset that enables consistency of experience across touchpoints and consolidated data management and reporting ● React quickly to changing customer preferences while introducing and experimenting with new digital capabilities to differentiate the business ● Maintain usability of the platform across internal teams at varying levels of digital maturity ● Operate efficiently across a common UX and UI using a single login and support organization	● Update services more quickly and easily than with a monolithic approach ● Lower burden on IT than a full microservices-based approach due to prebuilt integrations ● Harness a single organization's support to resolve issues across service updates and releases ● Gain flexibility to make the right technology decisions that serve specific business needs with the ability to seamlessly integrate with the platform through APIs	Most businesses will find the greatest value in building an ecommerce platform with a hybrid approach. With this kind of approach, a set of services is purchased and bundled from a single organization. Not only does the hybrid approach lower technical requirements and total cost of ownership (TCO), but it also reduces the need for complex integrations across services from different vendors. To keep risks low, businesses need to choose a vendor whose core services are flexible enough for API-led integrations and customization. That way, businesses can harness opportunities to quickly make changes in a headless system and build innovative services within the competitive ecommerce landscape.

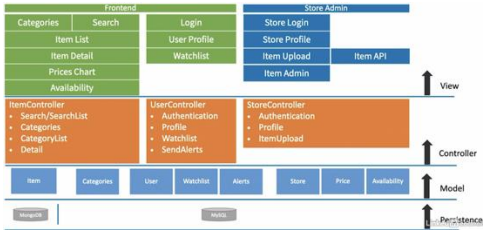
Breaking a Monolith into Microservices



Best Practices

- Design your services around capabilities and not around single functions.
- Make it a mantra that each service should be developed and deployed individually.
- Make evolutionary, atomic iterations and use the proxy pattern to delegate functions to respective service calls.





Refactor or rewrite

- Splitting up monoliths is about capabilities and not code
- Much of the code is specific to the monolith and won't be usable
- You miss the opportunity to improve code
- You miss the opportunity to use the right tool for the job
- It's rarely faster than starting from scratch
- Exception: complex orthogonal calculations and logic

Choosing the first migration target

- Start with an orthogonal, edge service.
- Small and not too business critical.
- Limited, simple functionality that can be developed fast.

Deployment plan (cutover): feature toggle

- Implement Proxy Methods on the monolith and hide them behind a feature toggle called "ServiceAuthentication."
- Add another toggle for migration "ServiceAuthenticationMigration" for the migration process that creates a friendly message for users attempting to use the features being migrated.
- Implement automated migration scripts and tests.
- Release the new version of the monolith.

Operating model

- Organisation culture, CI/CD, cross-team collaboration with inner source
- Automated API documentation
- Monitoring: telemetry and observability to detect degradations
- Distributed tracing

Thank you!

Jacques Maeda