Dr. Hans-Peter Hoidn
Distinguished IT Architect (Open Group certified)

Enterprise IT Architectures

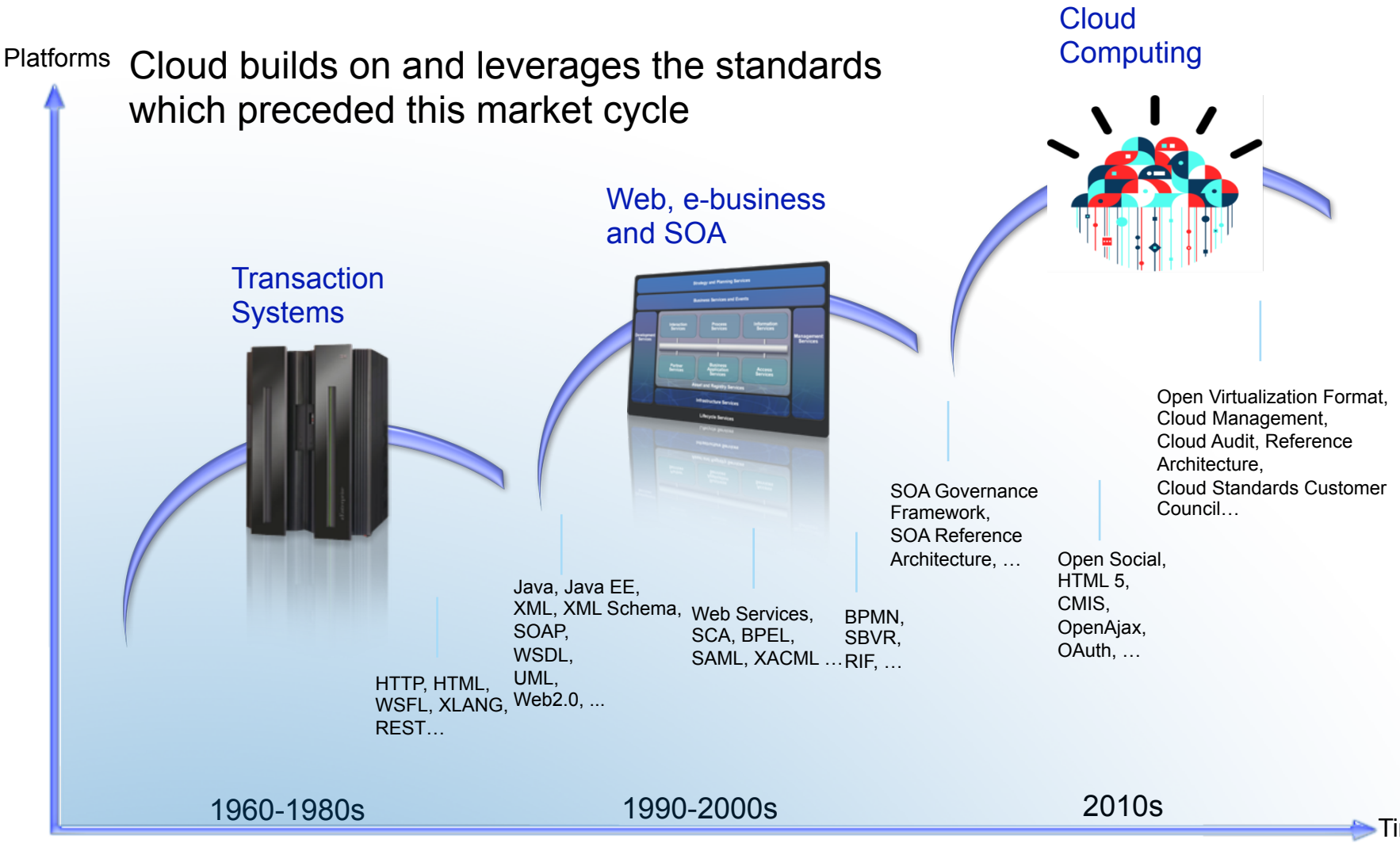
Technology Standards, SOA (Service Oriented Architecture), Technology Architecture, Information Systems Architecture

Agenda: Evolution of Technology, Standards, and Architecture; Technology Architecture, Information Systems Architecture

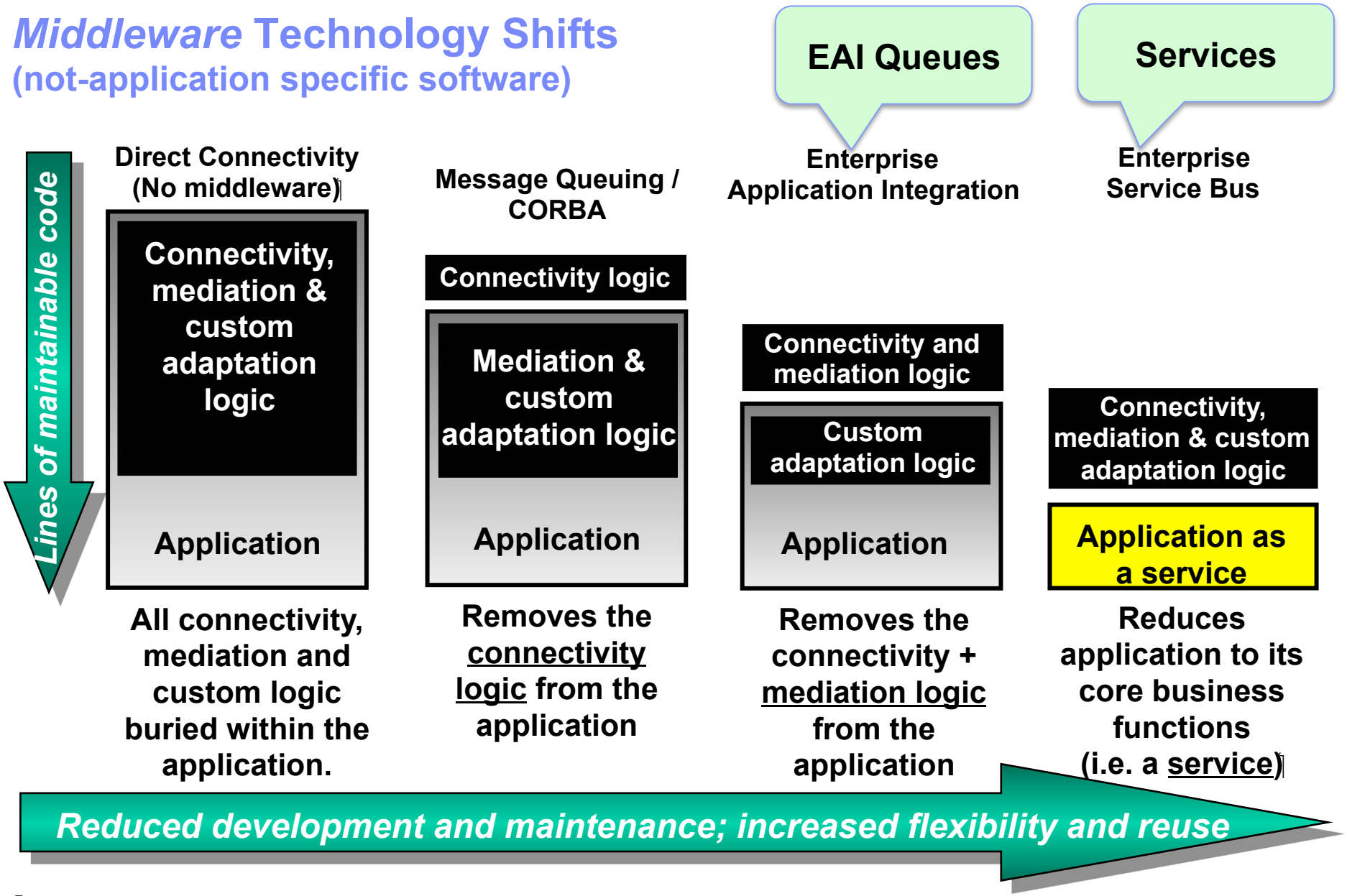
- Technology evolution and Standards
  - Technology Standards, like XML, SOAP, HTTP
  - "Quasi"-Standards, like REST
  - The Open Group and OMG provide Architecture Standards and more
  - *Without Standards IT would not be as it is today!*
- Technology Architecture
  - Various styles of architecture
  - Reference architectures (and patterns)
- Information Systems Architecture
  - Architecture style SOA (Service Oriented Architecture)

Technology Standards

Constantly shifts in Technology
(Distribution of Computing)



Middleware Technology Shifts
(not-application specific software)



Source: IBM

Technology Standards and Quasi-Standards

- Technology shifts drive the evolution of standards
  - XML, SOAP, REST, HTTP, JSON
  - Representing technology shifts

- Impact
  - Keeping IT components independent from technology
  - Architecture style adapts to technology and standards
  - Services and SOA (Service Oriented Architecture)
    (capital letter for Services in the sense of Service-Orientation)

- Reference Architectures and Patterns
  - SOA RA (Reference Architecture)
  - Patterns

SOAP

- SOAP is a protocol specification for exchanging structured information in the implementation of web services

- Its XML-based protocol consists of three parts:
  - an envelope, which defines the message structure
  - a set of encoding rules
  - a convention for representing procedure calls and responses

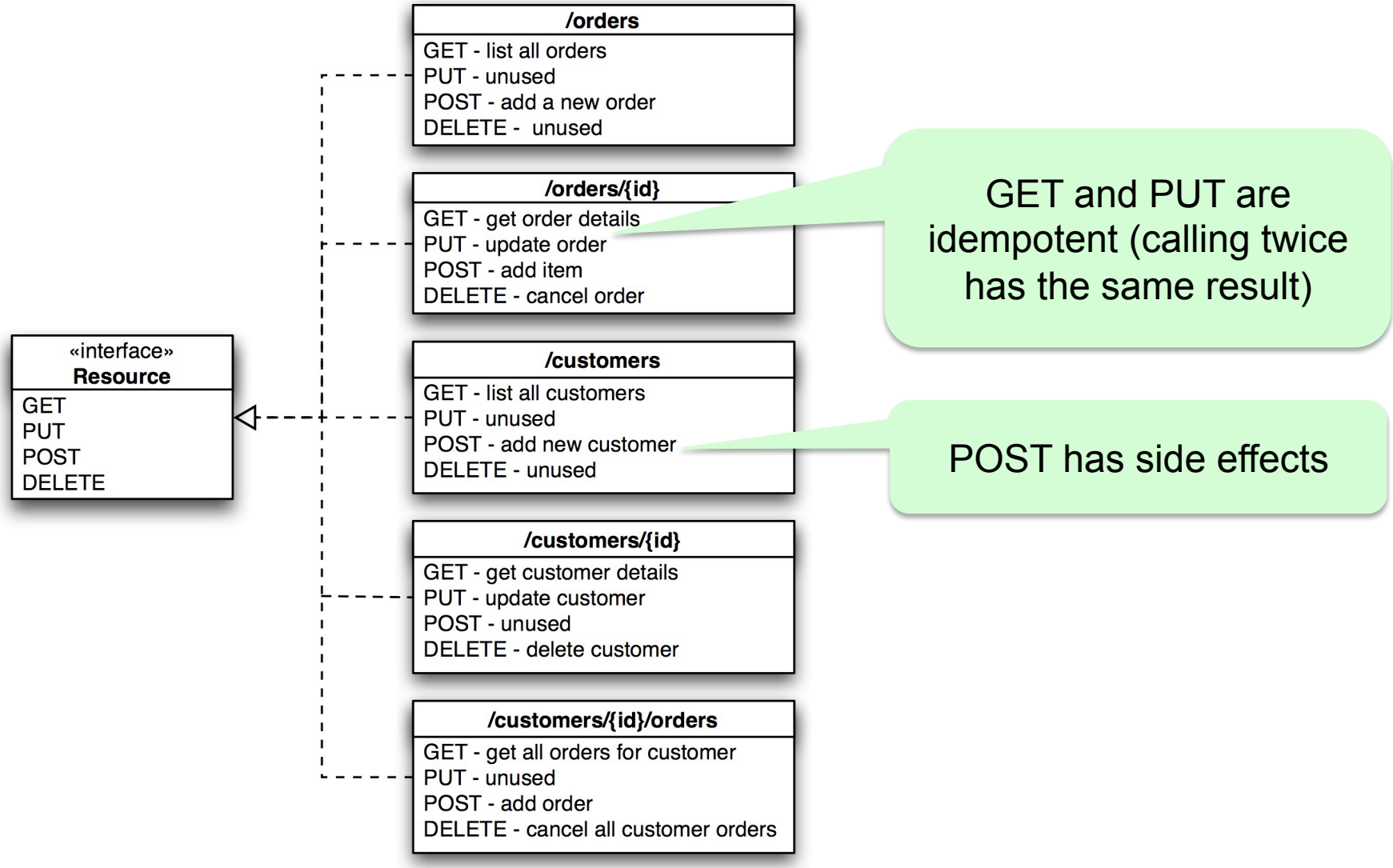
- SOAP has three major characteristics:
  - extensibility (security and WS-routing are among the extensions under development)
  - neutrality (SOAP can operate over any transport protocol such as HTTP, SMTP, TCP, UDP, or JMS)
  - independence (SOAP allows for any programming model)

REST (Representational State Transfer)

- REST is providing a generic interface, such that with REST an application is treated as a web application
- HTTP is the transport protocol. URIs access resources, a resource becomes a web resource by simply making the information associated with it accessible to the web
- The basic functions on the resources (entities as well as other "things" like a process) are GET, POST, PUT and DELETE
  - GET, PUT, DELETE are idempotent (no harm when called twice)
  - POST has side effects (may also on other resources)
- The advantage is that the lean data format JSON can be used

Source: Roy Fielding defined REST in the year 2000 in his thesis

Example RESTful Services



Services: API Management
(Reference: "API for Dummies" by Claus T. Jensen)

- "API Management is the process of publishing, promoting and overseeing application programming interfaces (APIs) in a secure, scalable environment" – including:
  - Software Lifecycle support
  - Access proxies including security procedures and policies
  - Version control.

- Almost every vendor provides API Management environment and tools providing:
  - Management for APIs that are (in principle) services
  - May support multiple service styles (Web as well as REST)

- *Sometimes it seems that "API" replaces "Service" and "API Management" replaces SOA – and the rest is unchanged.*

Services: Microservices
(Reference: http://martinfowler.com/articles/microservices.html)

- **Microservices**
  - **Defined by Martin Fowler & James Lewis**
  - **Monolithic application versus Microservice Architecture with every functional element in one service**

- **Microservice architectural style**
  - **“is an approach to developing a single application as a suite of small services”**
  - **“communicating with lightweight mechanisms, often an HTTP resource API”.**

- ***Comments*:**
  - **Granularity is key (Microservices focus on small Services)**
  - **Note that a single application is in focus, not integration**

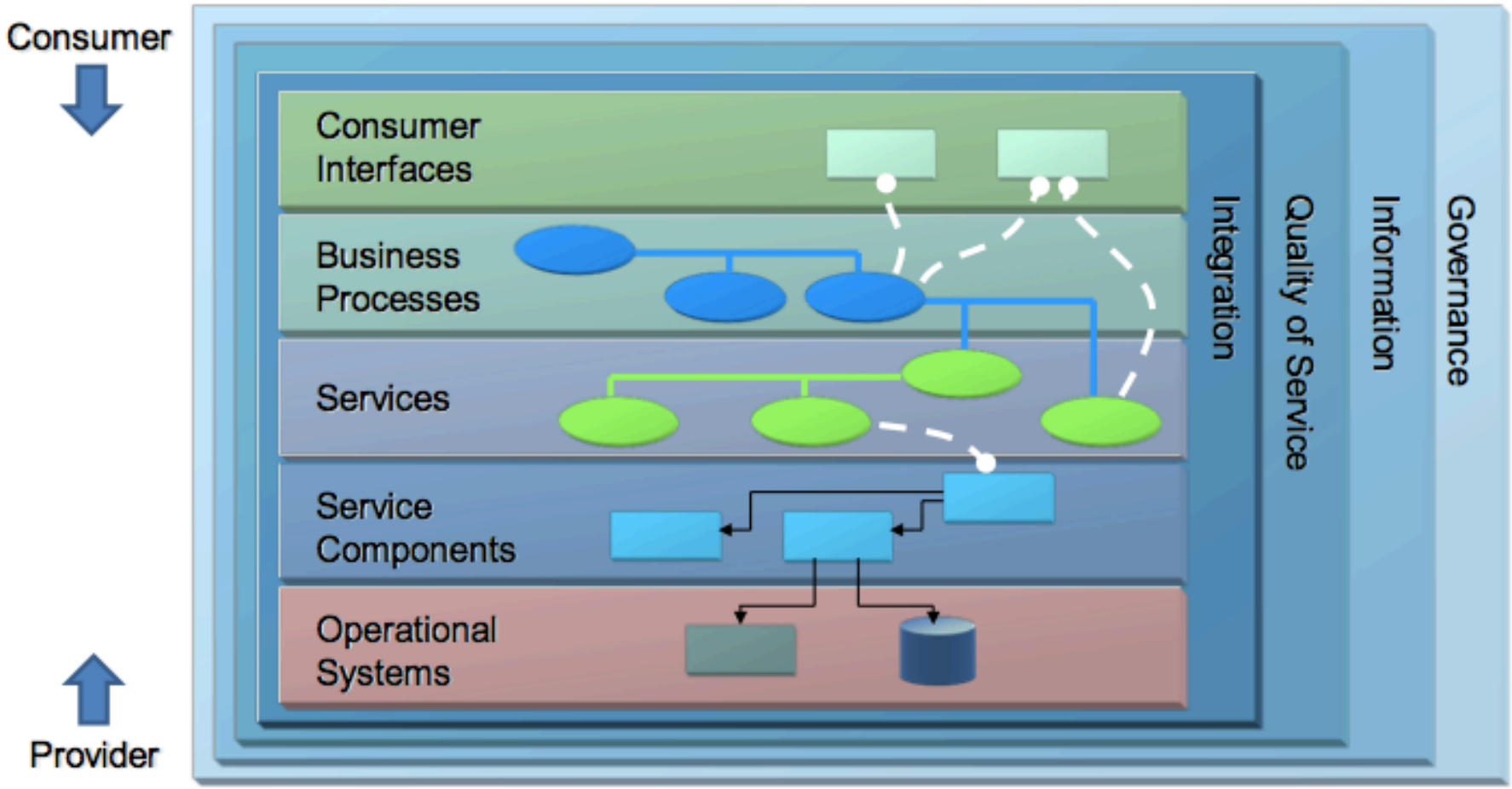
TOGAF Reference Architectures - SOA and Microservices

- **SOA Reference Architecture**
  - **Technical Standard (Document C119, November 2011)**
  - **Specifying a specific layered view**

- **Microservices Reference Architecture**
  - **White Paper (Document W169, July 2016)**
  - **Microservices is a subset of SOA (e.g. no composition of Services)**

- ***Comments*:**
  - **There is a large debate SOA versus Microservices (because SOA is often associated with – heavy and costly – integration)**

SOA RA (The Open Group SOA Reference Architecture)
(Reference: The Open Group Technical Standard, Document C119)



Source: Open Group

SOA is dead, long live Services
(Anne Thomas Manes)

Services: Definition "Service"

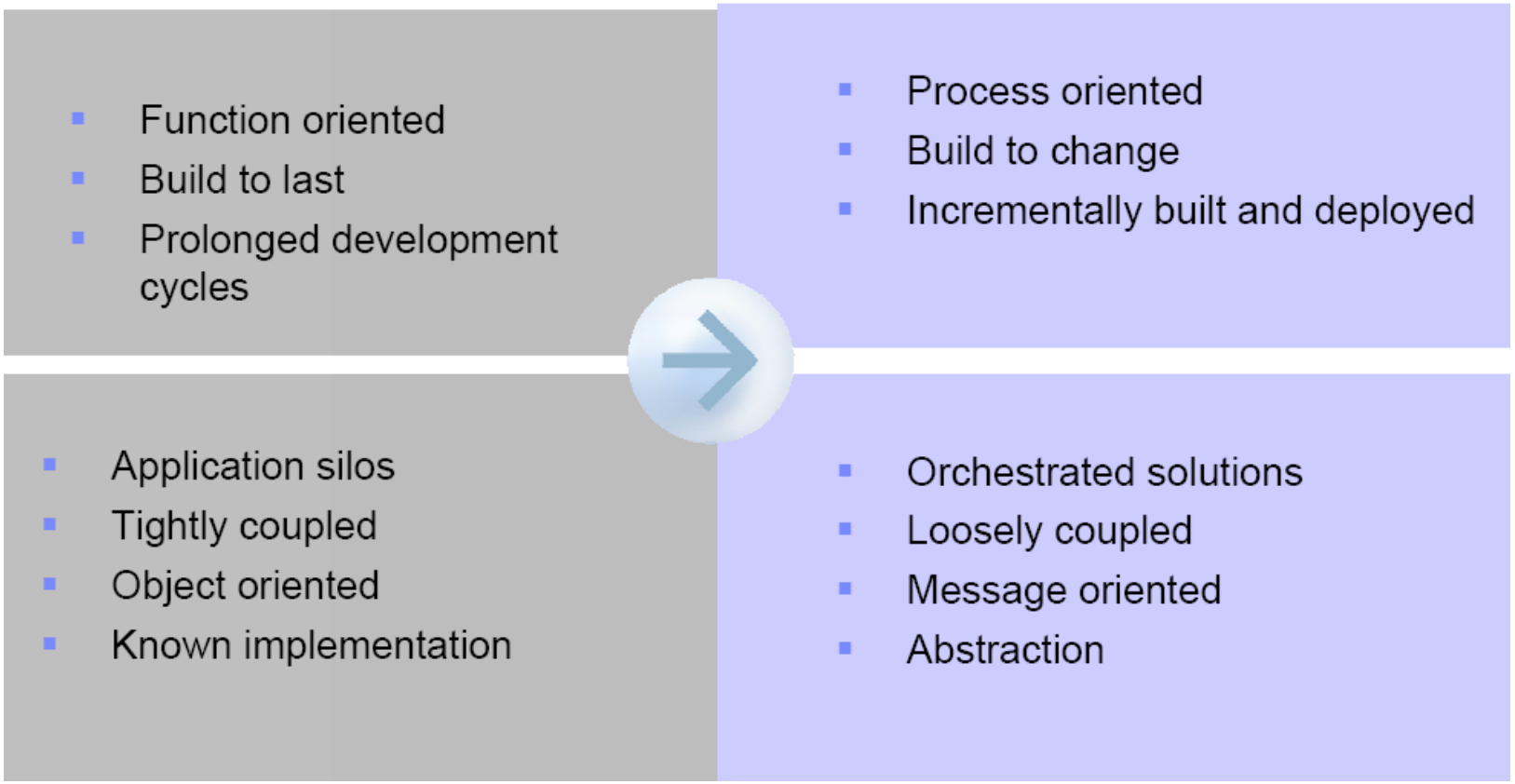
- **Business Perspective: A Service**
  - **is a well- defined, encapsulated, reusable, business-aligned capability.**
  - **is fully defined by a service description, a published document or artifact that outlines the overall objective of the service and its inputs, purpose, outputs, scope, responsibility, governance, sustainability (provision period, maintenance, and repair), and qualities of service provisioning**

- **IT Perspective: A Service**
  - **is a discoverable, invokable software resource that has a service description and interface and is configurable using policies.**
  - **implementation is realized through a service provider that delivers quality of service (QoS) requirements for the service consumer**

Major Observations of Common Shift

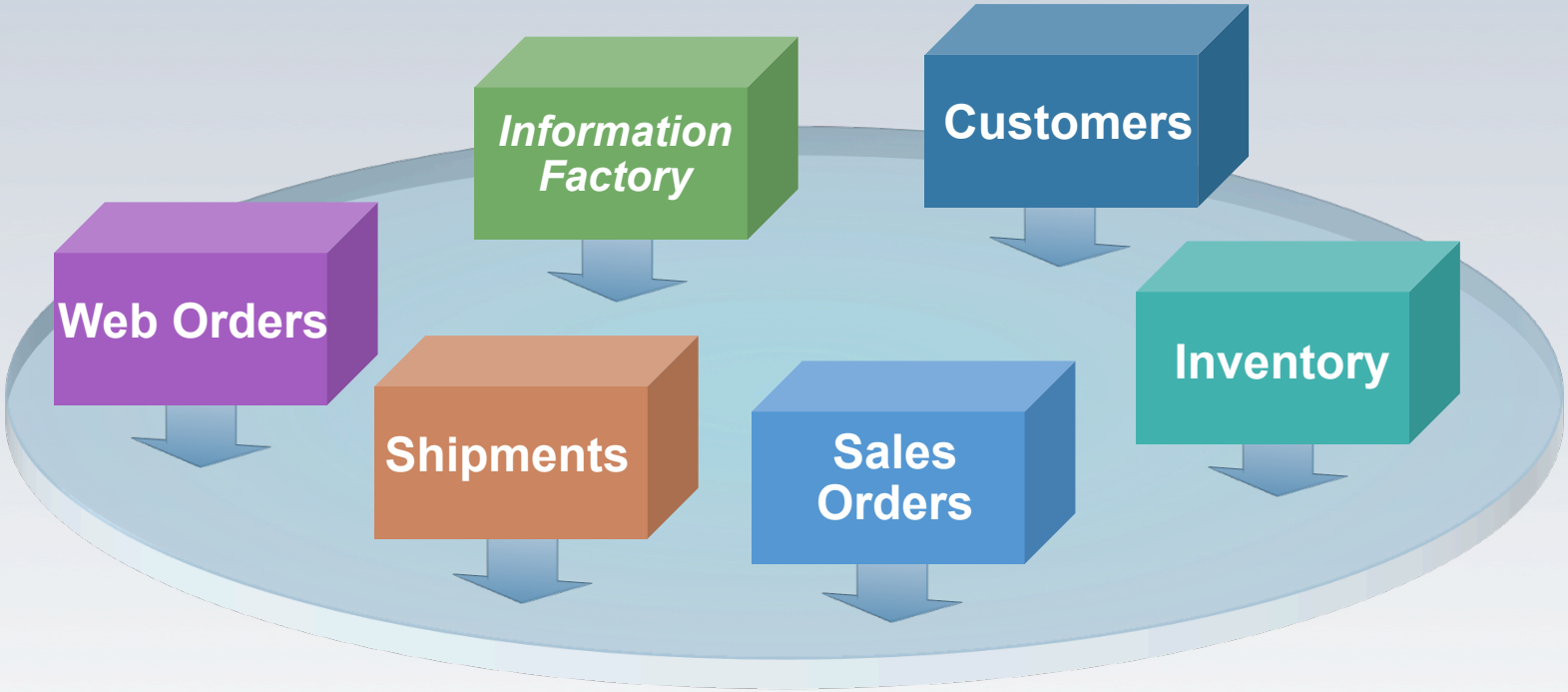
Note Conway's Law: "Any piece of software reflects the organizational structure that produced it"c

Shift to a Service-Oriented Architecture



Services & Service Oriented Architecture (SOA)

Moves IT Logic Out of Services



Services defined as units of business logic separated from…

- Flow of control and routing
- Data transformation and protocol transformation

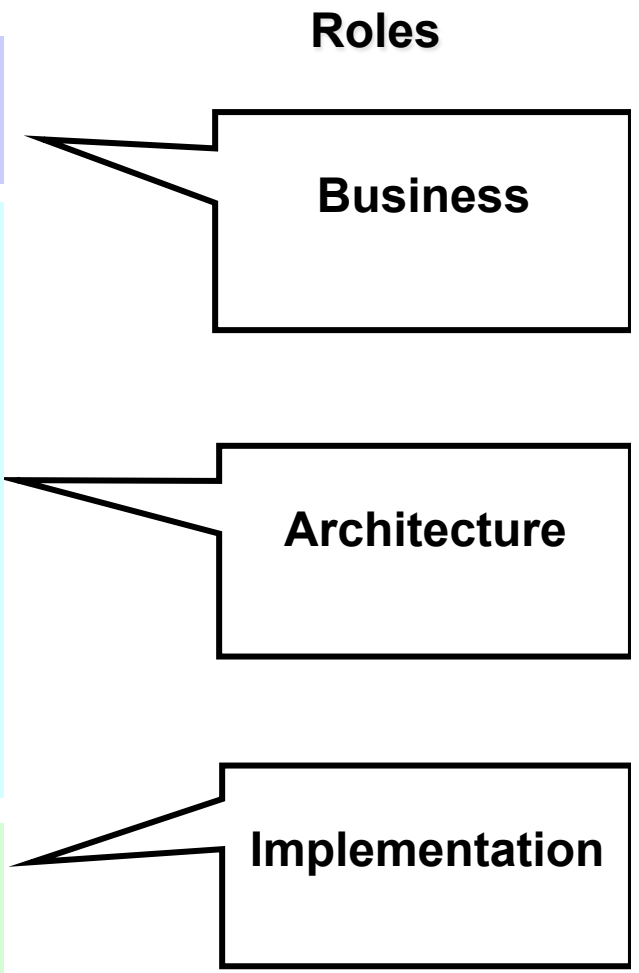
Services & SOA is different things to different people

A set of services that a business wants to expose to customers and clients

an architectural style which requires a service provider, requestor and a service description.

a set of architectural principles and patterns which address characteristics such as *modularity, encapsulation, loose coupling, separation of concerns, reuse, composable and single implementation*.

A programming model complete with standards, tools, methods and technologies such as web services.



SOA and Microservices: SOA is the general idea, where microservices are a very specific way of achieving it

All of the tenets of SOA also apply to microservices

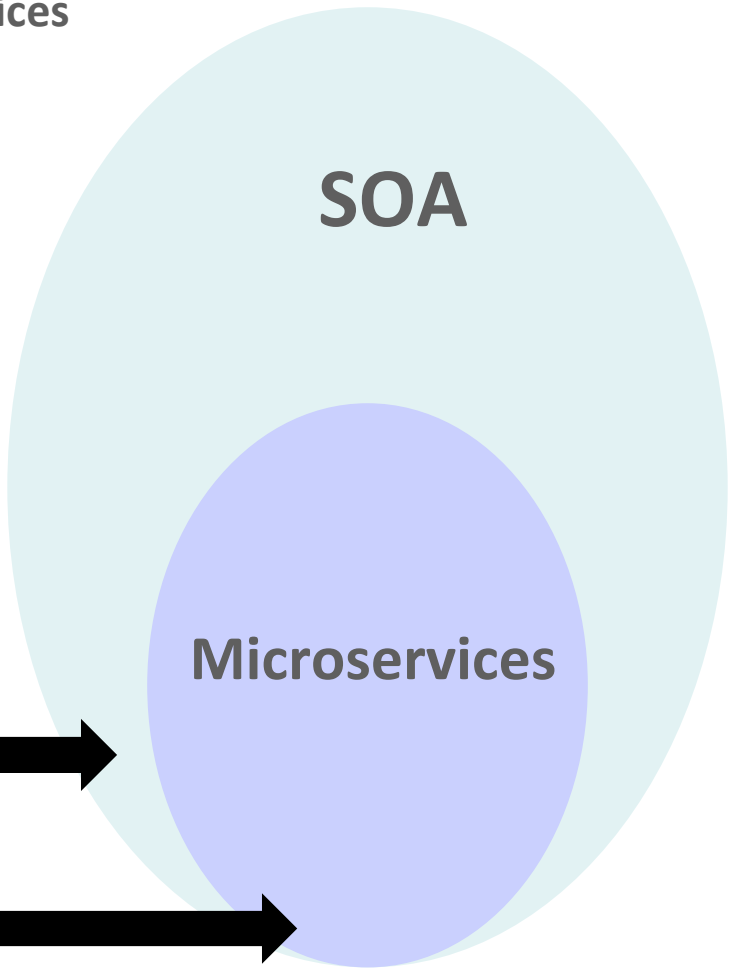
1. Keeping consumption of services separate from the provisioning of services
2. Separating infra management from the delivery of application capability
3. Separating teams and decoupling services

Implementation Differences

- Favors centralized orchestration
- Needlessly complicated by SOAP
- "Dumb endpoints, smart pipes"

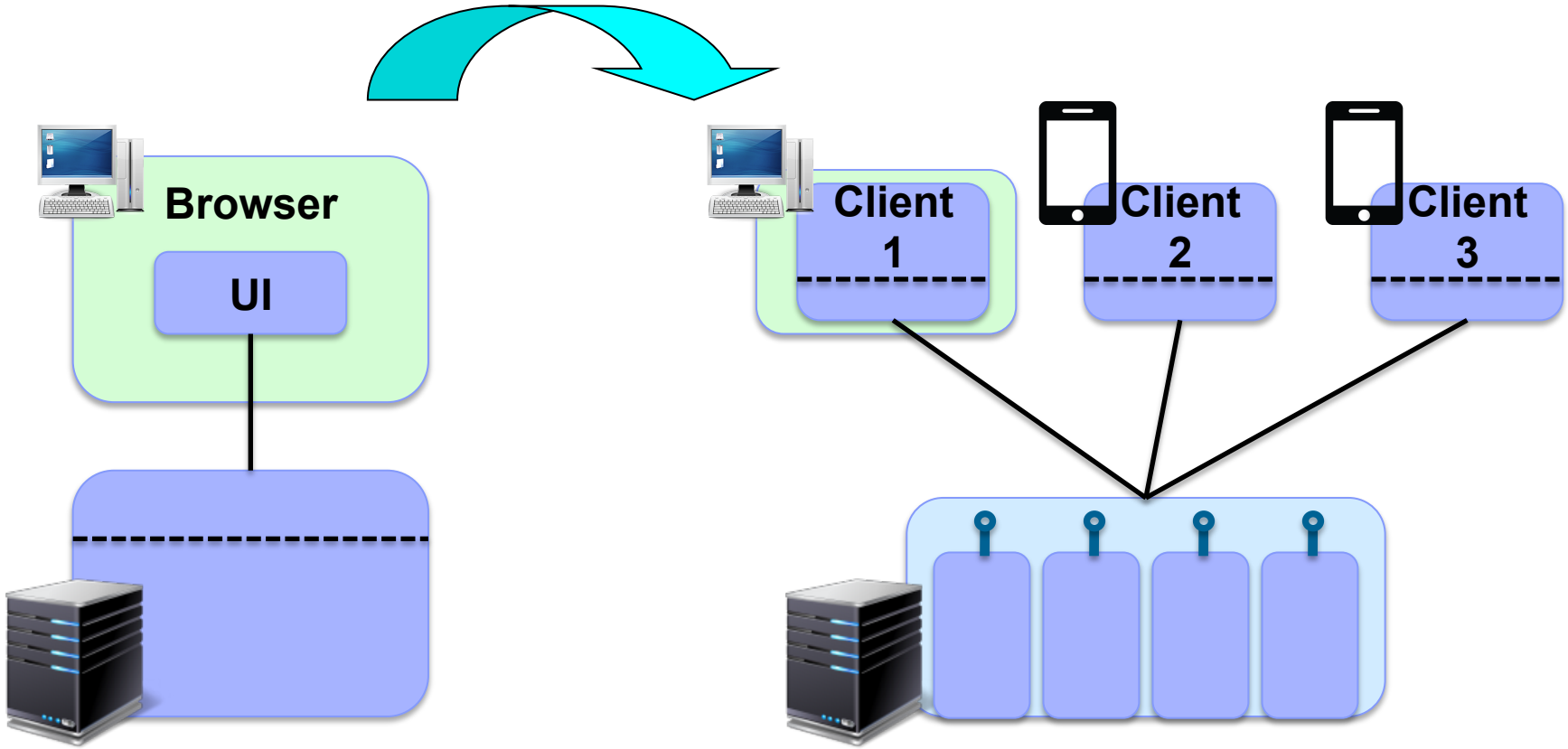
- Favors distributed choreography
- REST + HTTP/S = simple
- "Smart endpoints, dumb pipes"

"Two Speed Application Strategy"

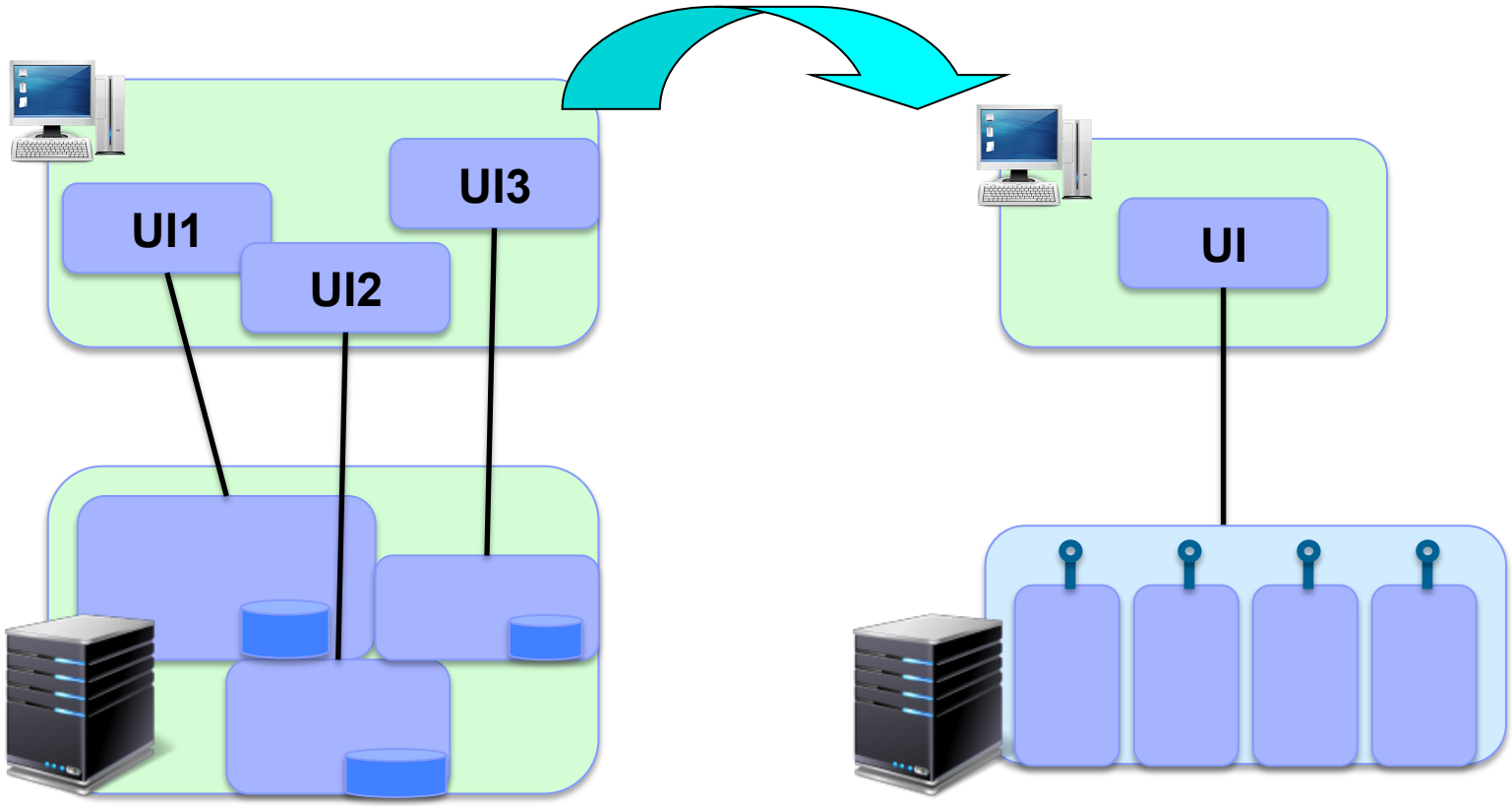


Source: ORACLE

Services-Example:
New Frontend with Apps accessing Services in the Backend



Services-Example:
Migration of Frontend and Backend

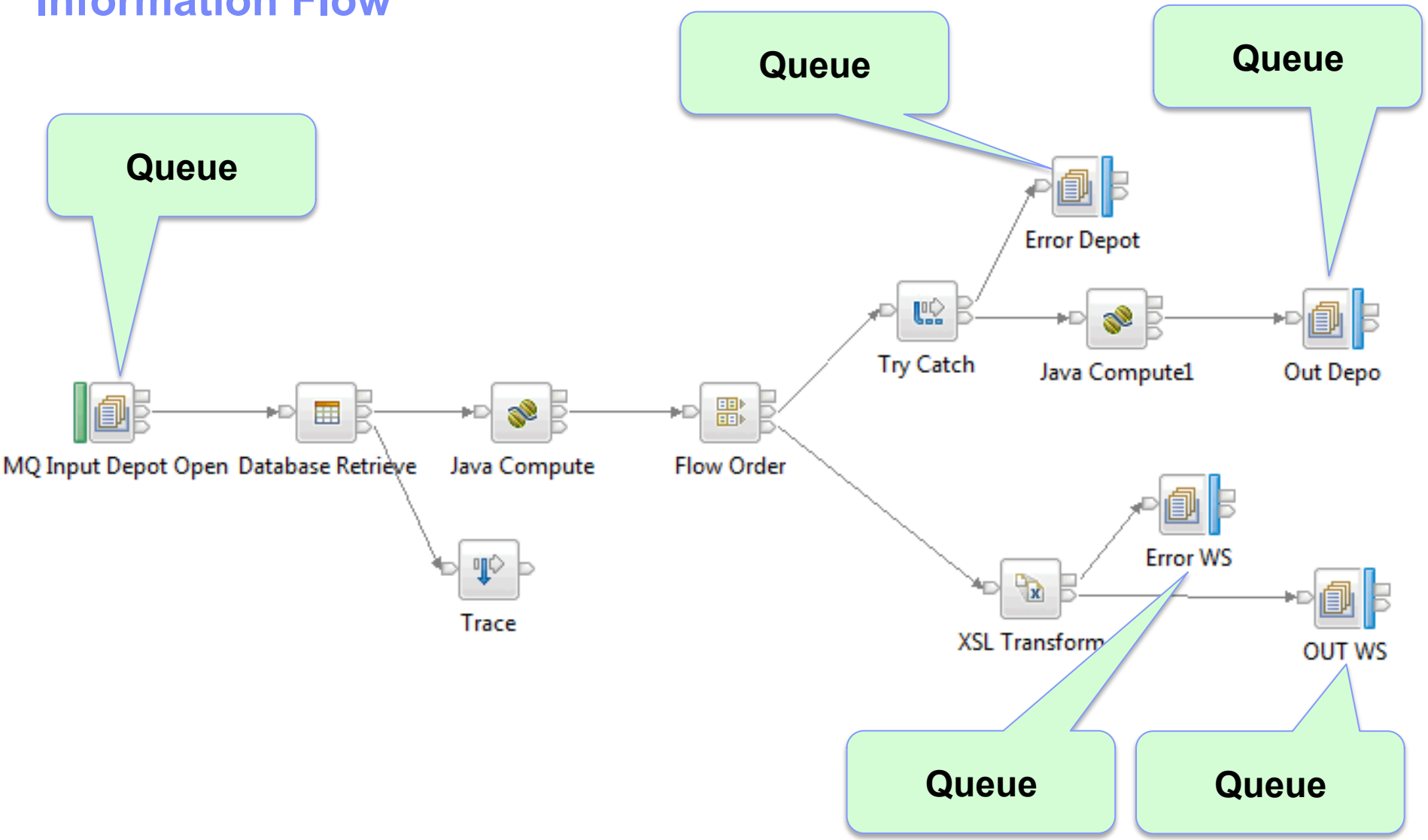


Integration: EIP (Enterprise Integration Pattern)
(Reference: Gregor Hohpe & Bobby Woolf, Addison-Wesley)

- **Firstly EIP (Enterprise Integration Pattern) – defined 2004 by Hohpe und Woolf differentiating 65 integration patterns**
  - **Messaging Systems (incl. Message Translation)**
  - **Messaging Channels (incl. Publish & Subscribe, Guarantied Delivery, Message Bus)**
  - **Message Routing (incl. content-based, Aggregation, Load-Balancing)**
  - **Message Transformation (incl. Enrich, Filter)**
  - **Message Endpoints (incl. Gateway, Consumer und Subscriber)**

- ***Patterns describe the integration properties and after identifying a pattern the proper technology may be chosen***

Integration – Message Queuing: Illustration of (Message) Broker Information Flow



ESB (Enterprise Service Bus) – Definition and Purpose

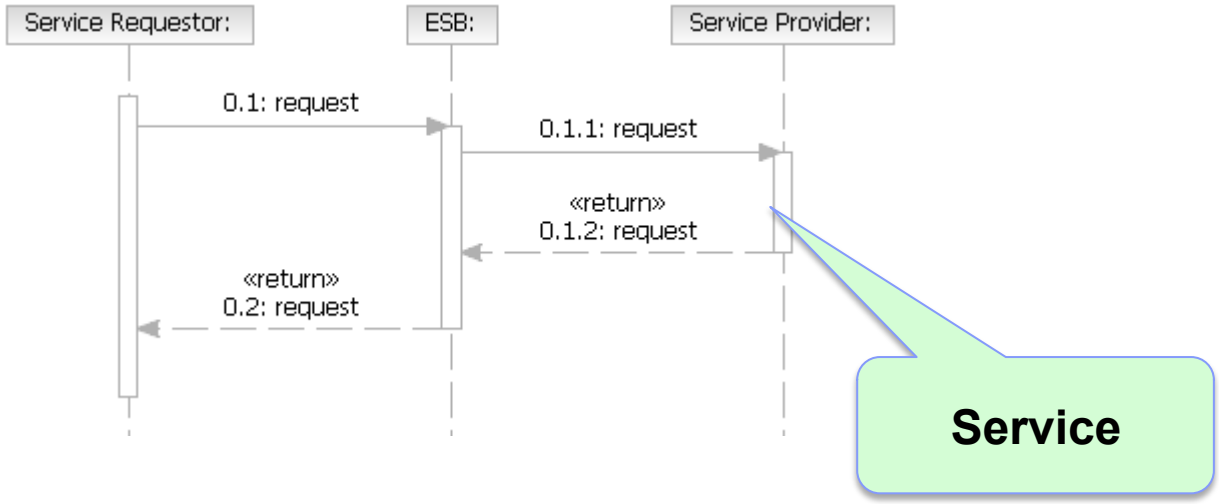
- An Enterprise Service Bus (ESB) is an architectural pattern defining a flexible connectivity infrastructure for integrating applications and services.

- The architecture pattern is a guiding principle to enable the integration and federation of multiple service bus instantiations.

- An ESB performs:
  - Routing messages between services
  - Converting transport protocols between requestor and service – managing multiple protocols
  - Transforming message content between requestor and service
  - Handling business events from disparate sources

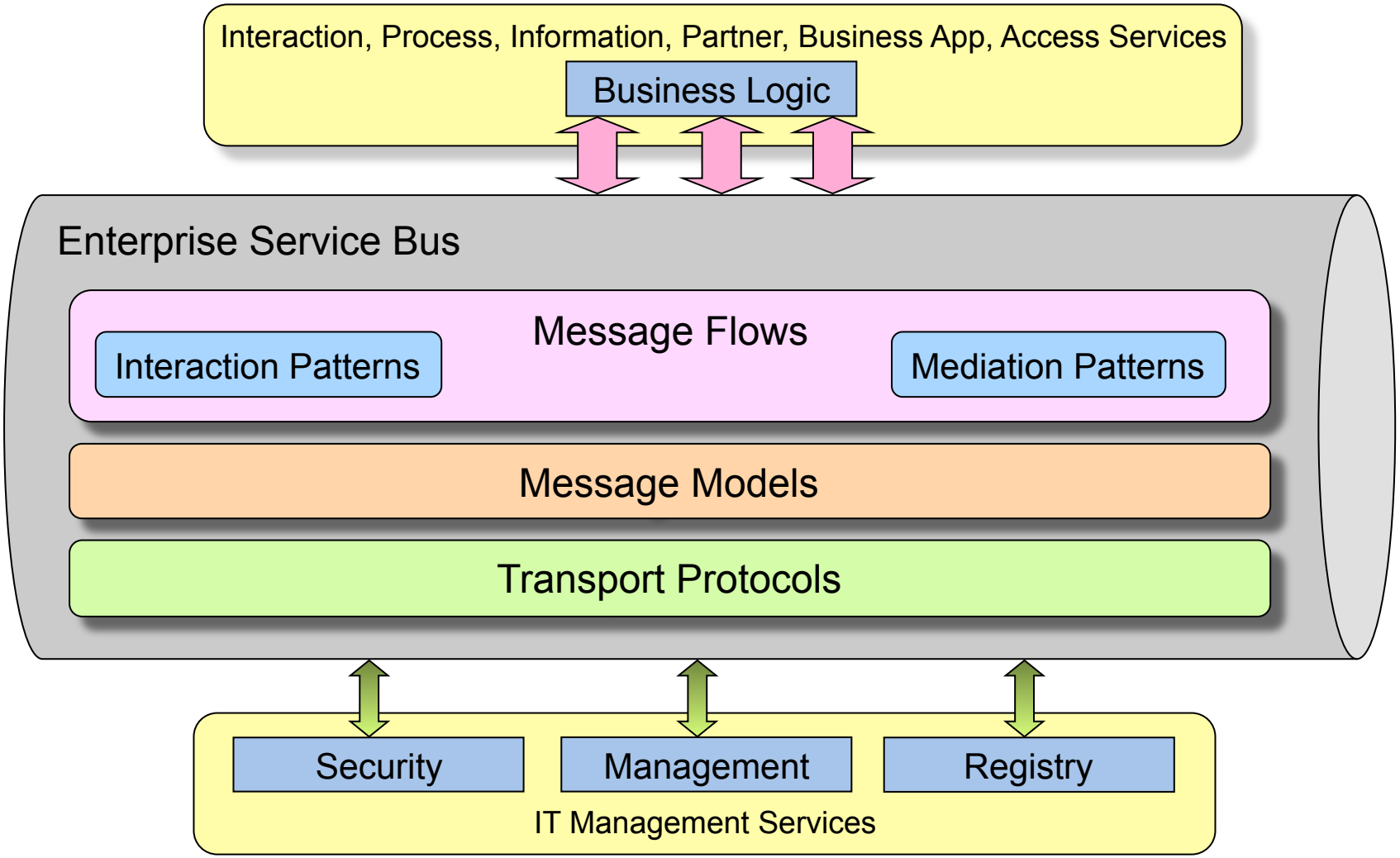
ESB (Enterprise Service Bus) – Service Virtualization
Decoupling of Service Requester and Provider

- **ESB acts as an intermediary (proxy) between requestor and provider**

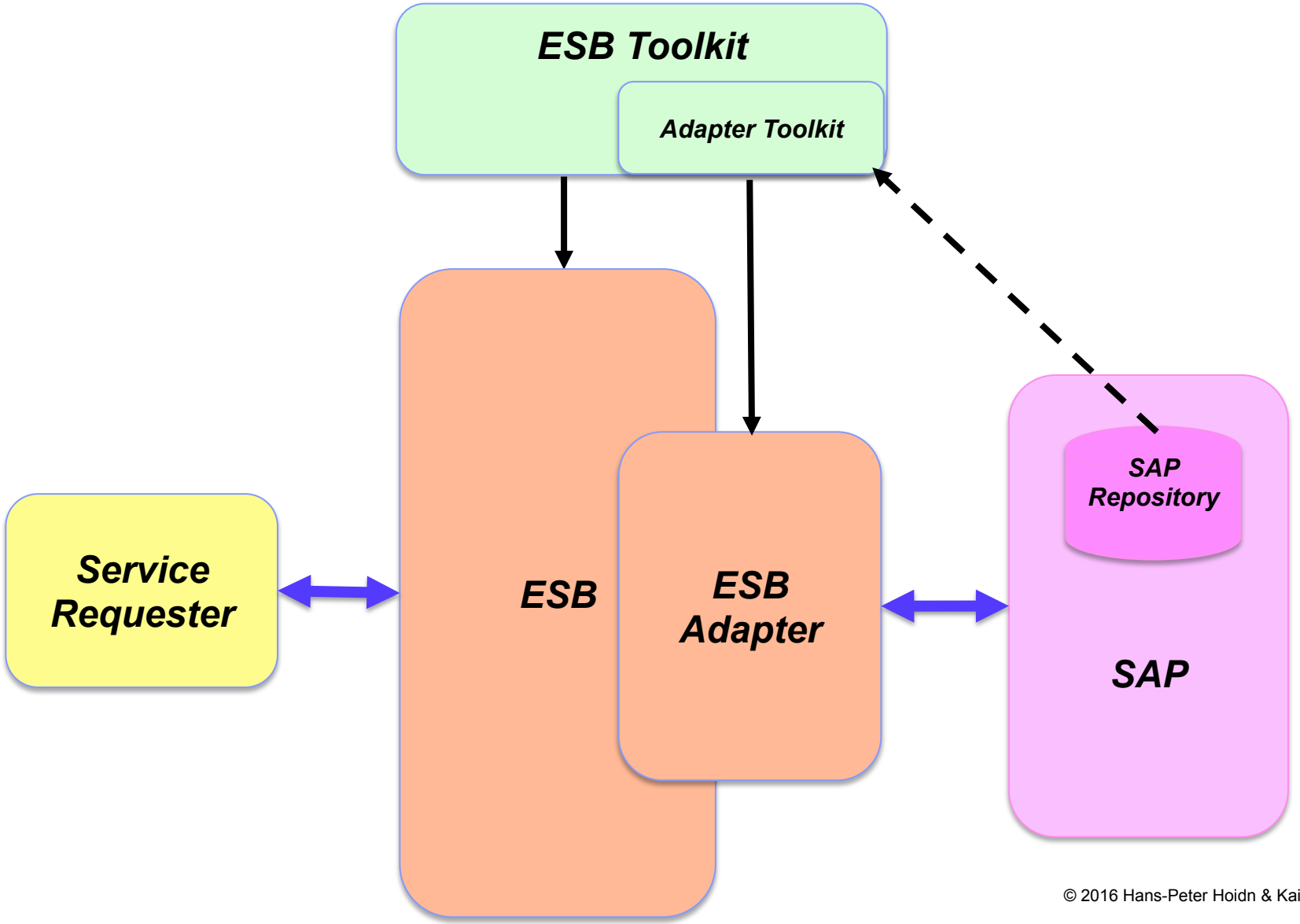


- **ESB provides *service virtualization* of**
  - ***Location and identity***
  - ***Interaction protocol***
  - ***Interface***
- **Interactions are *decoupled*, supporting *separation of concerns***

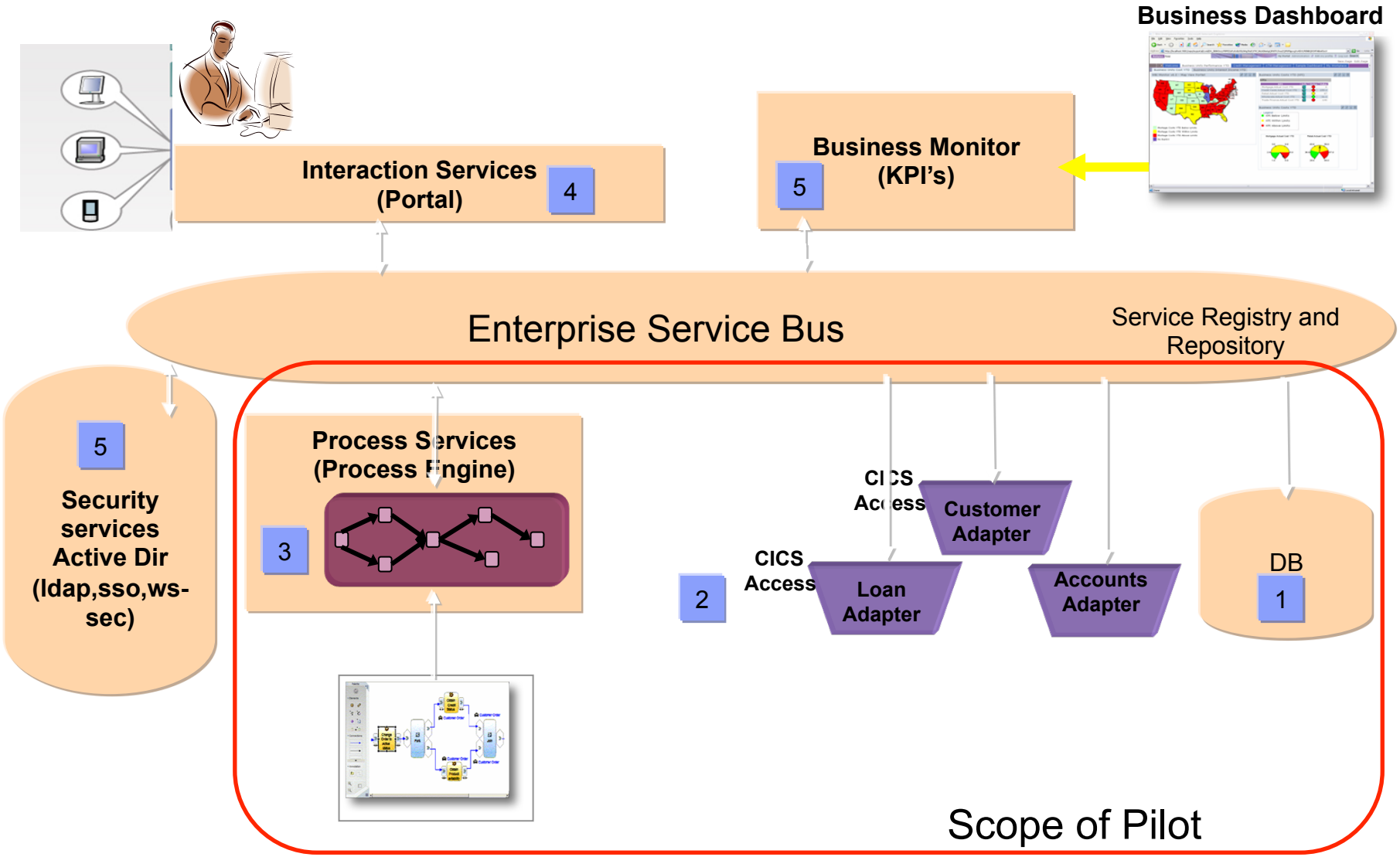
ESB (Enterprise Service Bus) – Expanded View



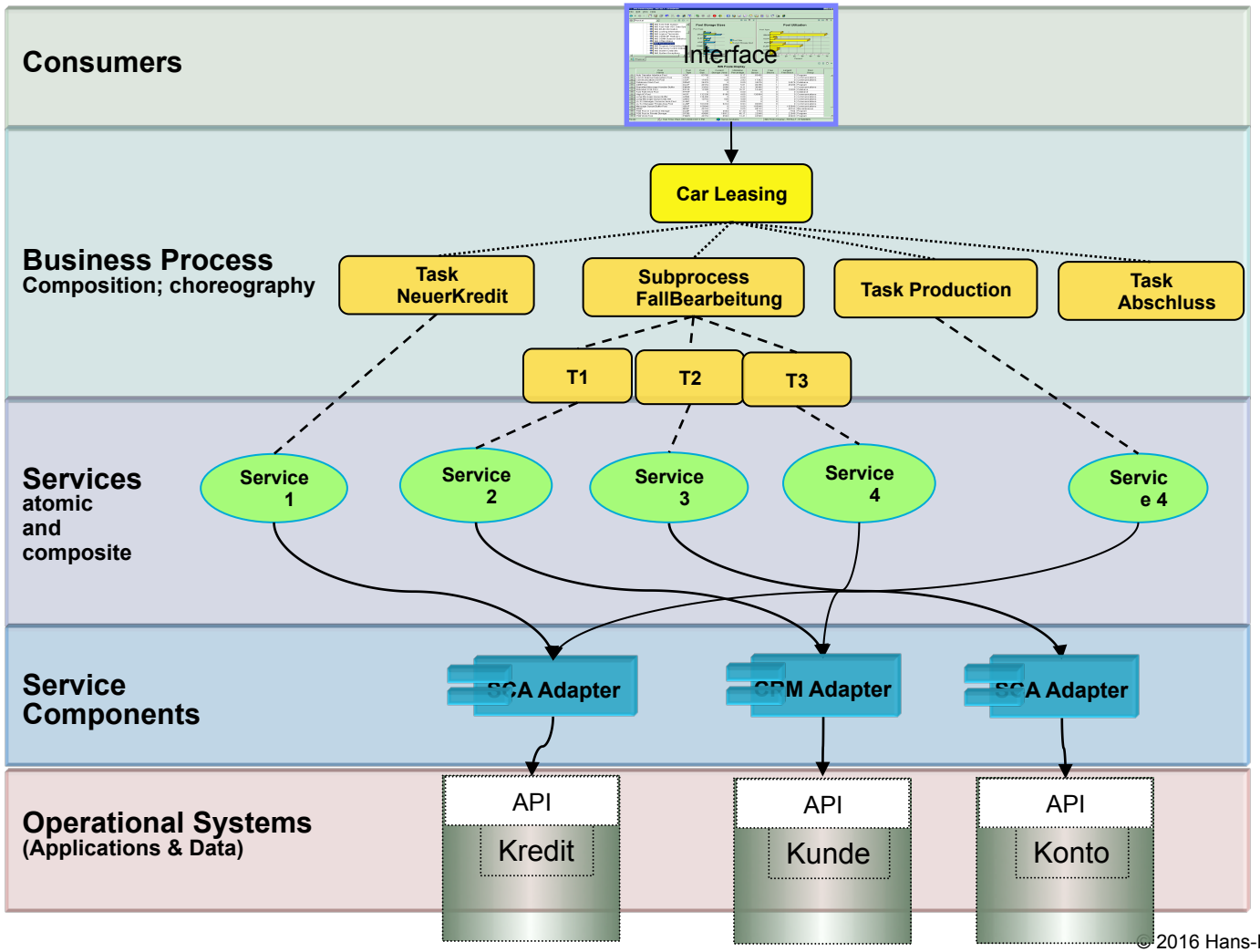
ESB with Adapter: Access to Backend Application



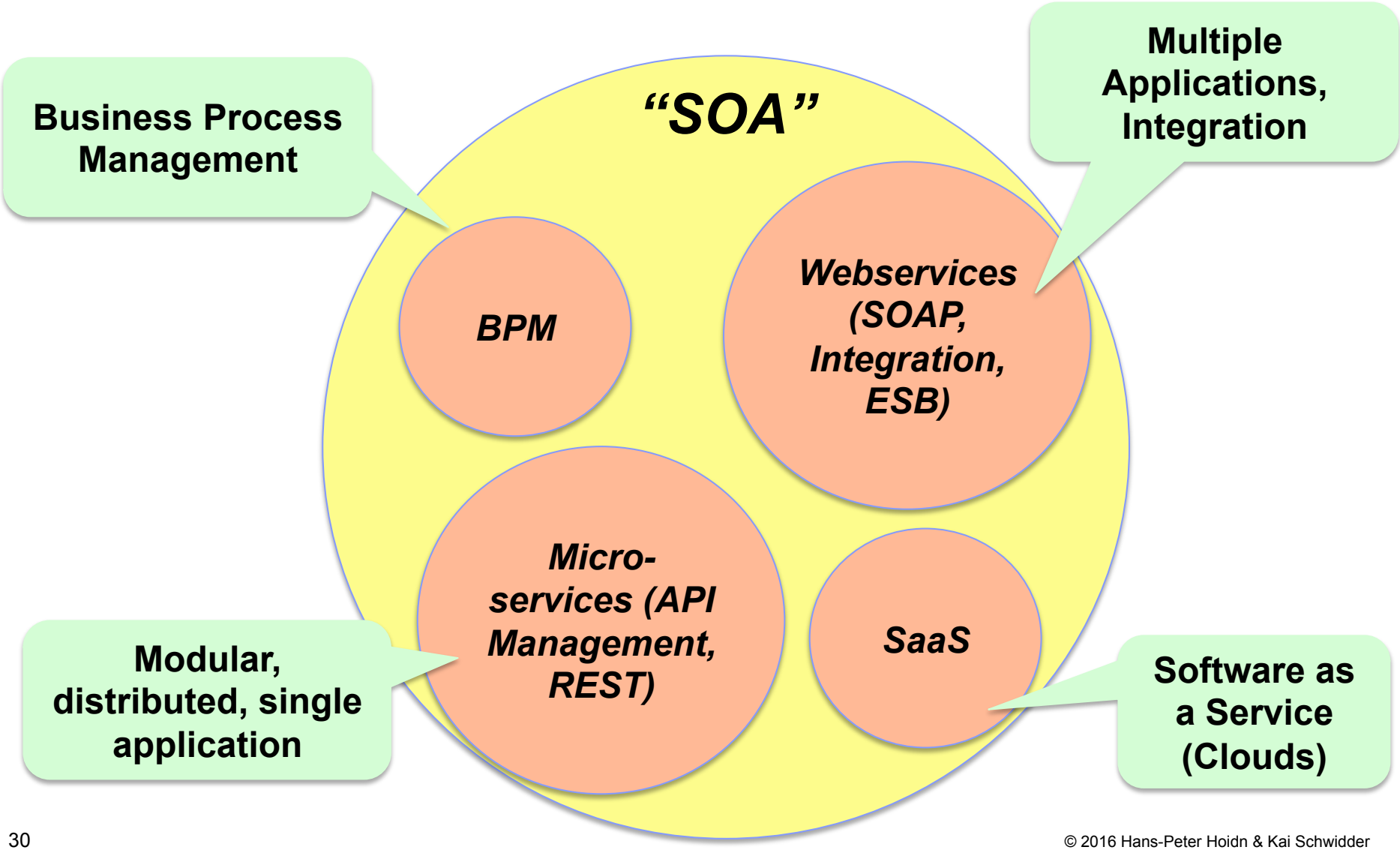
Example Car Leasing: Technical View – Car Leasing



Example Car Leasing – SOA Layers



Conclusion: Types of Applications requires types of "SOA"
(there is NO one-size fits all approach)



Technology Architecture

Technology Architecture – Purpose

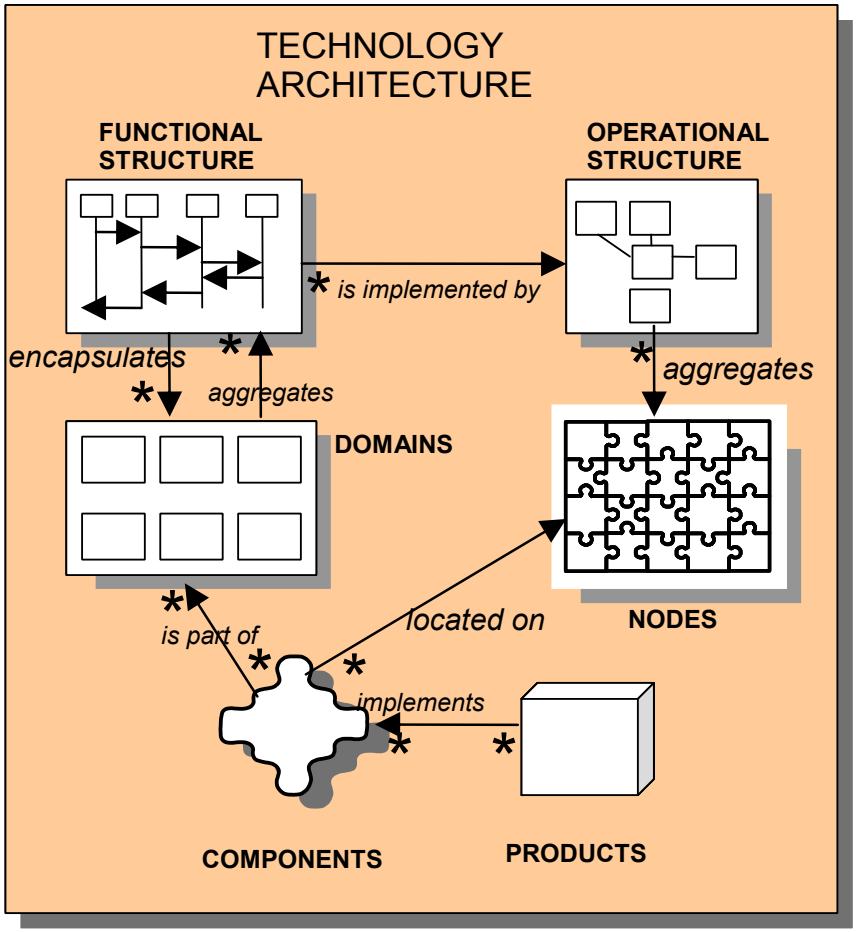
- provides the information for the physical implementation
  - Mapping from logical to physical
  - Dealing with the distribution of components to nodes
- Is heavily influenced by technology evolution
  - Defines a Technology "Platform" (adapts to technology)
  - Provides the basis for the implementation of the solutions
- Patterns and Reference Architectures play a major role:
  - Reference Architectures provide patterns
  - E.g. SOA Reference Architecture
  - E.g. Cloud Reference Architecture

First Step for "*Technology Architecture*"
(some key words highlighted)

- **Review and validate the set of technology *principles*. These will normally form part of an overarching set of architecture principles**
- **Select relevant Technology Architecture resources (*reference models*, *patterns*, etc.) – adapting to architectural style**
- **Select relevant Technology Architecture viewpoints that will enable the architect to demonstrate how the stakeholder concerns are being addressed in the Technology Architecture**
- **Identify appropriate tools and techniques to be used for capture, modeling, and analysis, in association with the selected viewpoints**

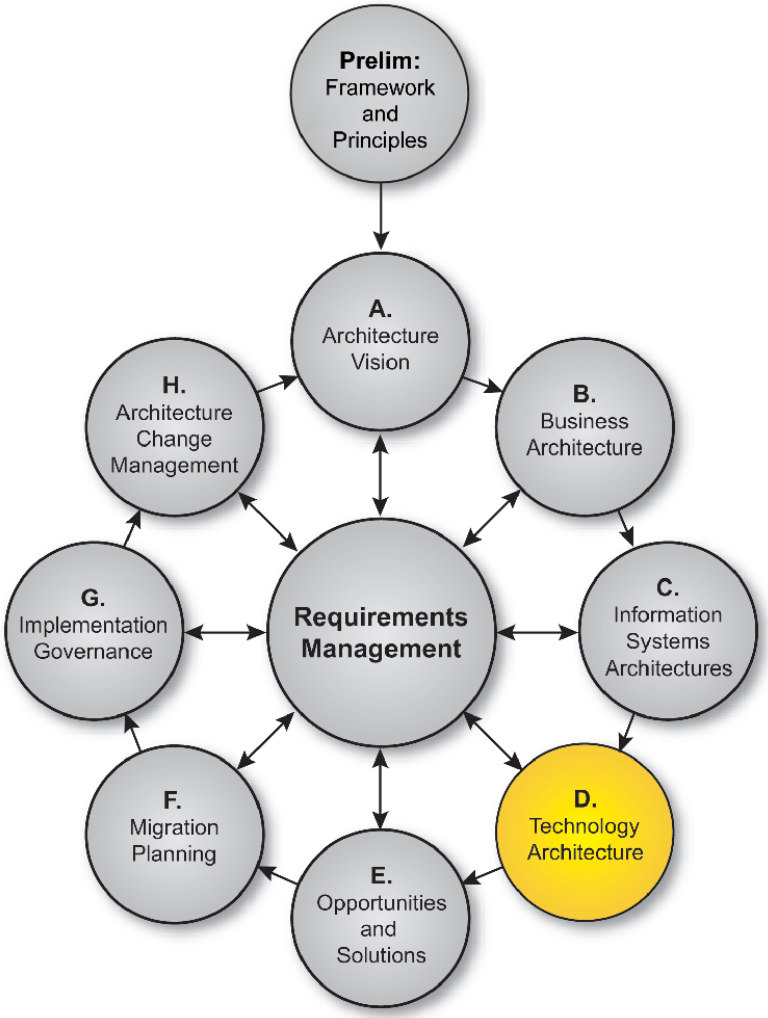
Technology Architecture – Overview

- **Defines the essential guidance (rules) for selecting, building and maintaining a technical infrastructure composed of a set of hardware and software environments and platforms with defined interfaces, standards & services**
- **Defines the nodes, components, connections, collaborations and locations required**
- **Is based largely on proven architectural patterns, reference models or templates for the domains**
- **Provides design guidance, where appropriate, for developers**
- **Provides a framework for setting technology and product standards**



Source: IBM Architectural Thinking

Technology Architecture – Content according to TOGAF



- **The fundamental organization of an IT system, embodied in**
  - **its hardware, software and communications technology**
  - **their relationships to each other and the environment,**
  - **and the principles governing its design and evolution**

Technology Architecture – Guidance
described in accordance with four key aspects

Components

Technology "Building blocks" – often "standards"

Nodes

Logical Servers

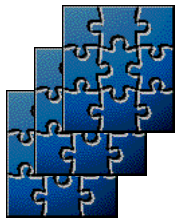
Logical Clients

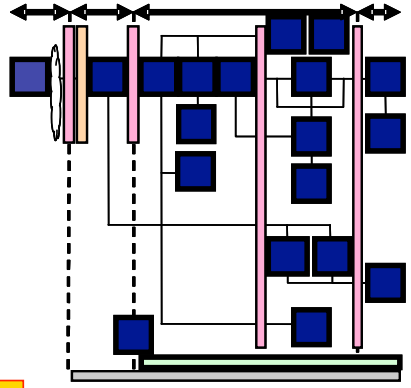
Operational Model

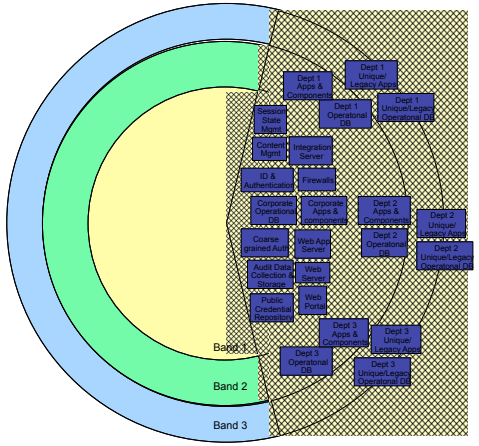
Assemblies of nodes required for common solutions

Placement:

Implementation guidance







Standards Criteria

E.g. RDBMS

ANSI SQL 92

E.g. Operational Database

E.g. Data Warehouse

E.g. B2P Pattern

E.g. B2P Placement

Source: IBM Architectural Thinking

Technology Architecture – Objectives (TOGAF 12.1)

- **Develop the Target Technology Architecture that enables the logical and physical application and data components and the Architecture Vision, addressing the Request for Architecture Work and stakeholder concerns**

- **Identify candidate Architecture Roadmap components based upon gaps between the Baseline and Target Technology Architectures**

Technology Architecture – Steps (TOGAF 12.4)

- Select reference models, viewpoints, and tools
- Develop Baseline Technology Architecture Description
- Develop Target Technology Architecture Description
- Perform gap analysis
- Define candidate roadmap components
- Resolve impacts across the Architecture Landscape
- Conduct for mal stakeholder review
- Finalize the Technology Architecture
- Create Architecture Definition Document

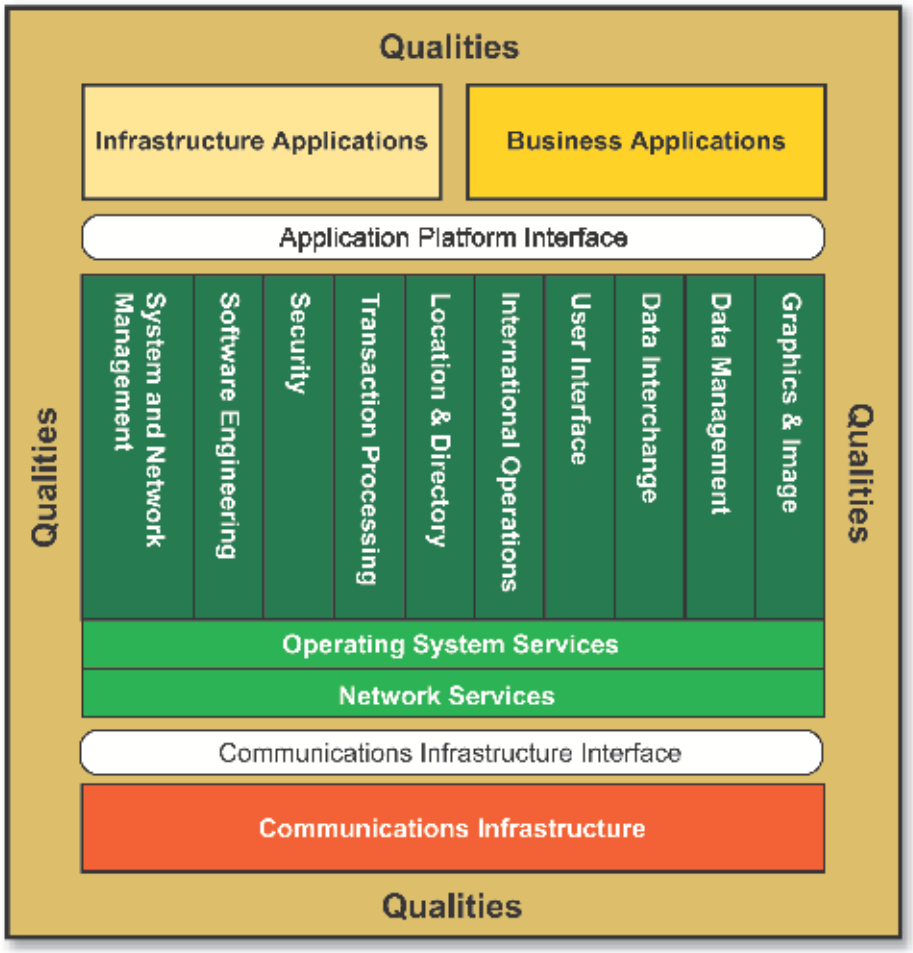
Technology Architecture – Topics (TOGAF Pocket Guide)

- Baseline Technology Architecture, if appropriate Target Technology Architecture, including:
  - Technology components and their relationships to information systems
  - Technology platforms and their decomposition, showing the combinations of
- Technology required to realize a particular technology "stack"
  - Environments and locations – a grouping of the required technology into computing environments (e.g., development, production)
  - Expected processing load and distribution of load across technology components — Physical(network)communications
  - Hardware and network specifications
  - Views corresponding to the selected viewpoints addressing key stakeholder concerns

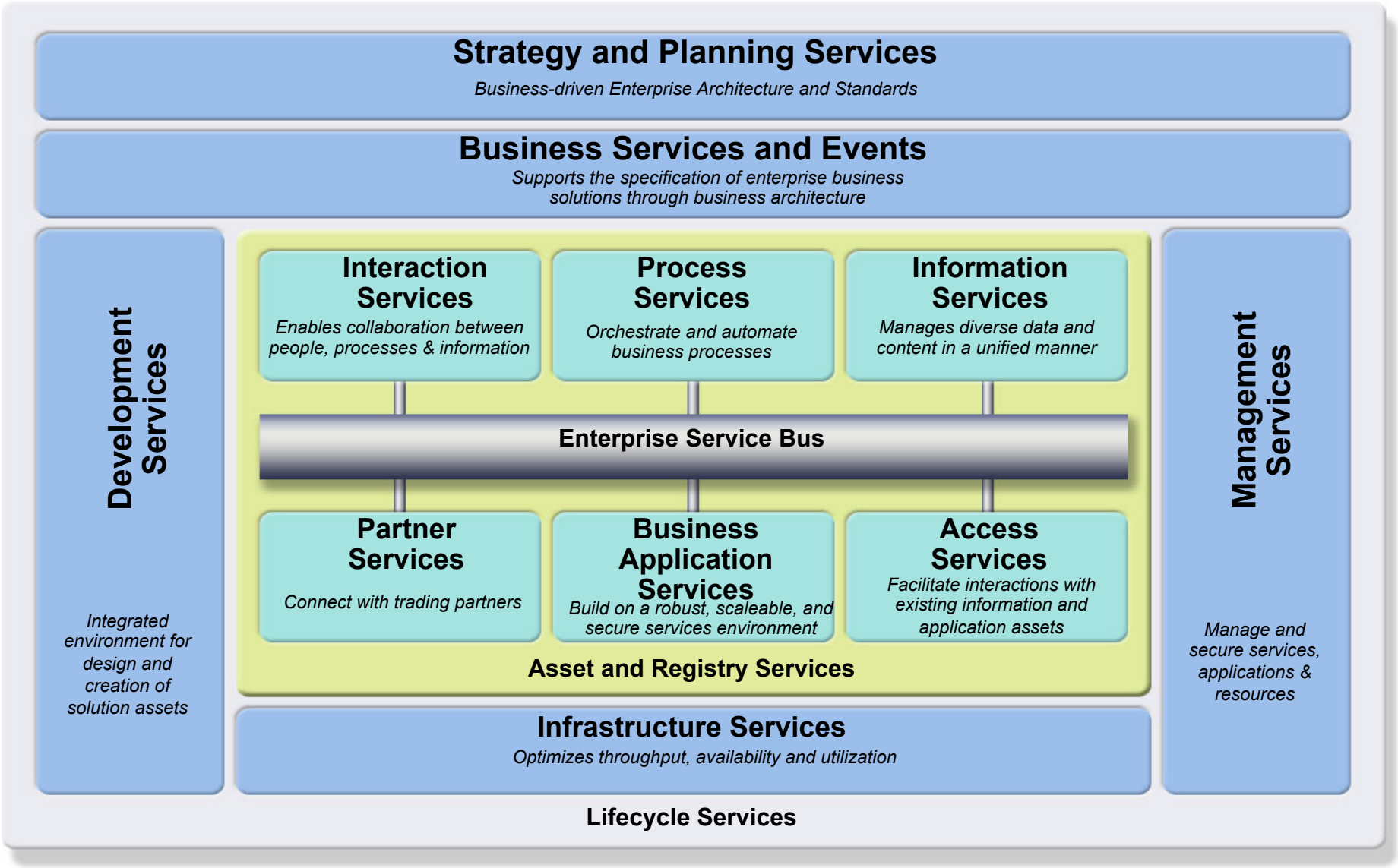
Reference Architectures (once more) – Overview

- **TOGAF**
  - **Is providing two reference architectures, one of those is the TRM (TOGAF Technical Reference Model)**
  - **TRM specifies various technical services, the details showing service categories, see TOGAF Chapter 43, pp. 491 – 521**
  - ***If not a huge project, this taxonomy is too much !***
- **SOA Reference Architecture (The Open Group Technical Standard)**
  - **Provides appropriate basis for an Integration Platform, firstly logically but can be augmented by products**
  - ***Much better suitable for "normal" usage and consulting***
- **Cloud Reference Architecture (and some more)**

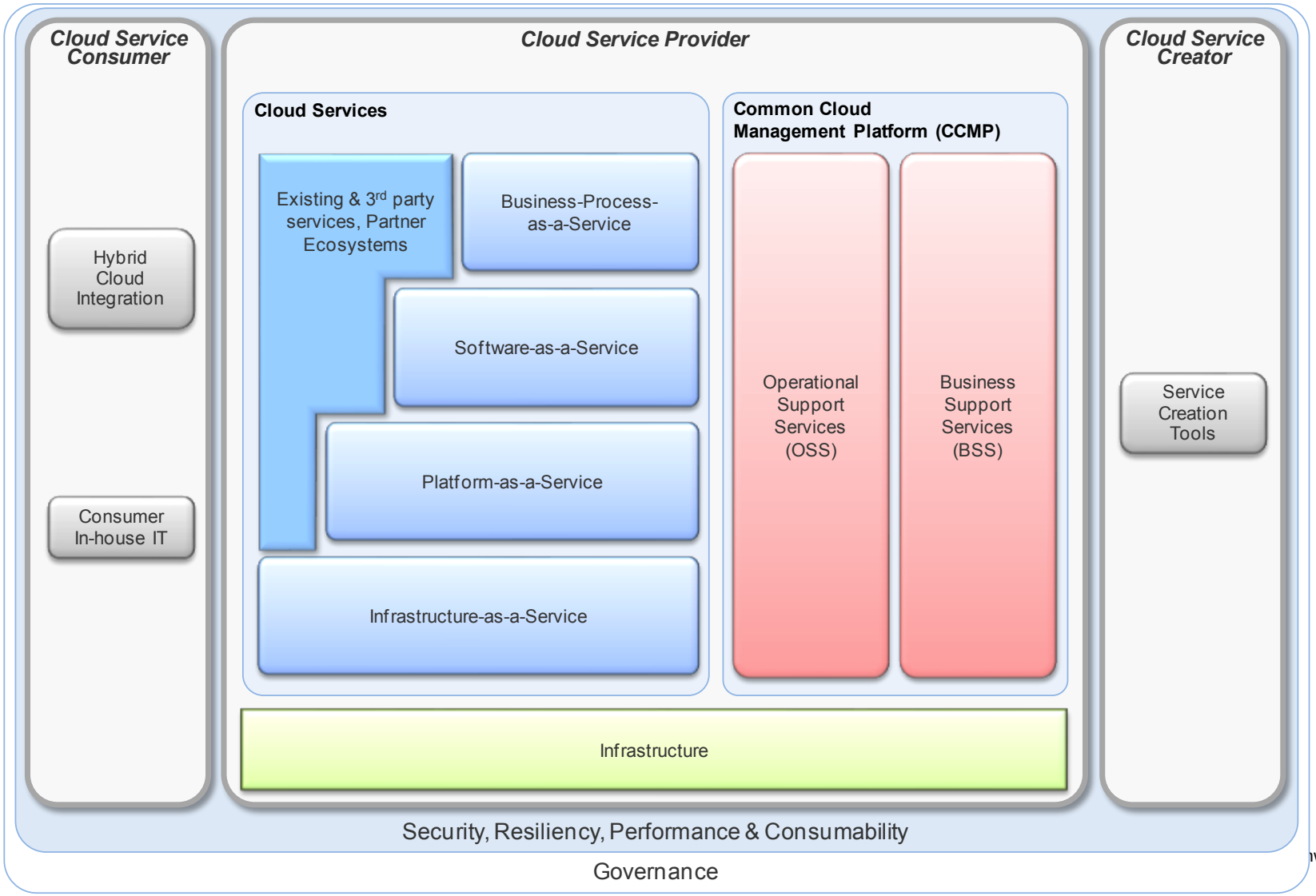
Reference Architectures (1)
TRM (TOGAF Technical Reference Model) Detailed View



Reference Architectures (2): The Open Group SOA RA contains SOA Foundation Reference Model (originally from IBM)

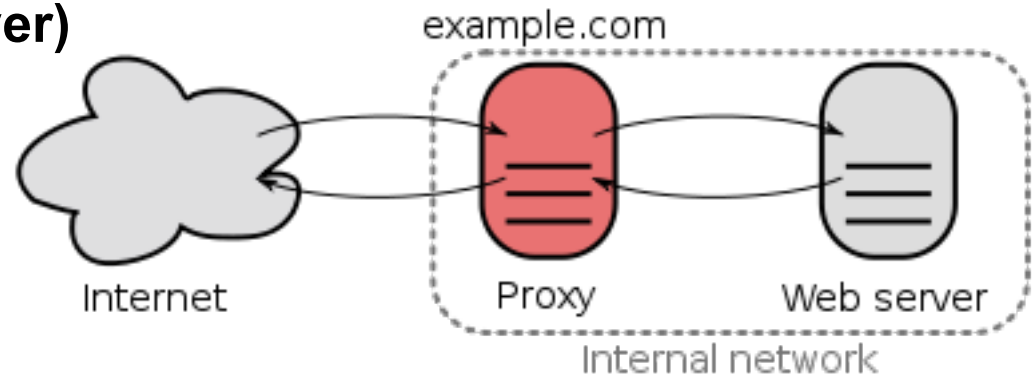


Reference Architectures (3)
Cloud Computing Reference Architecture (CCRA) Version 4.0



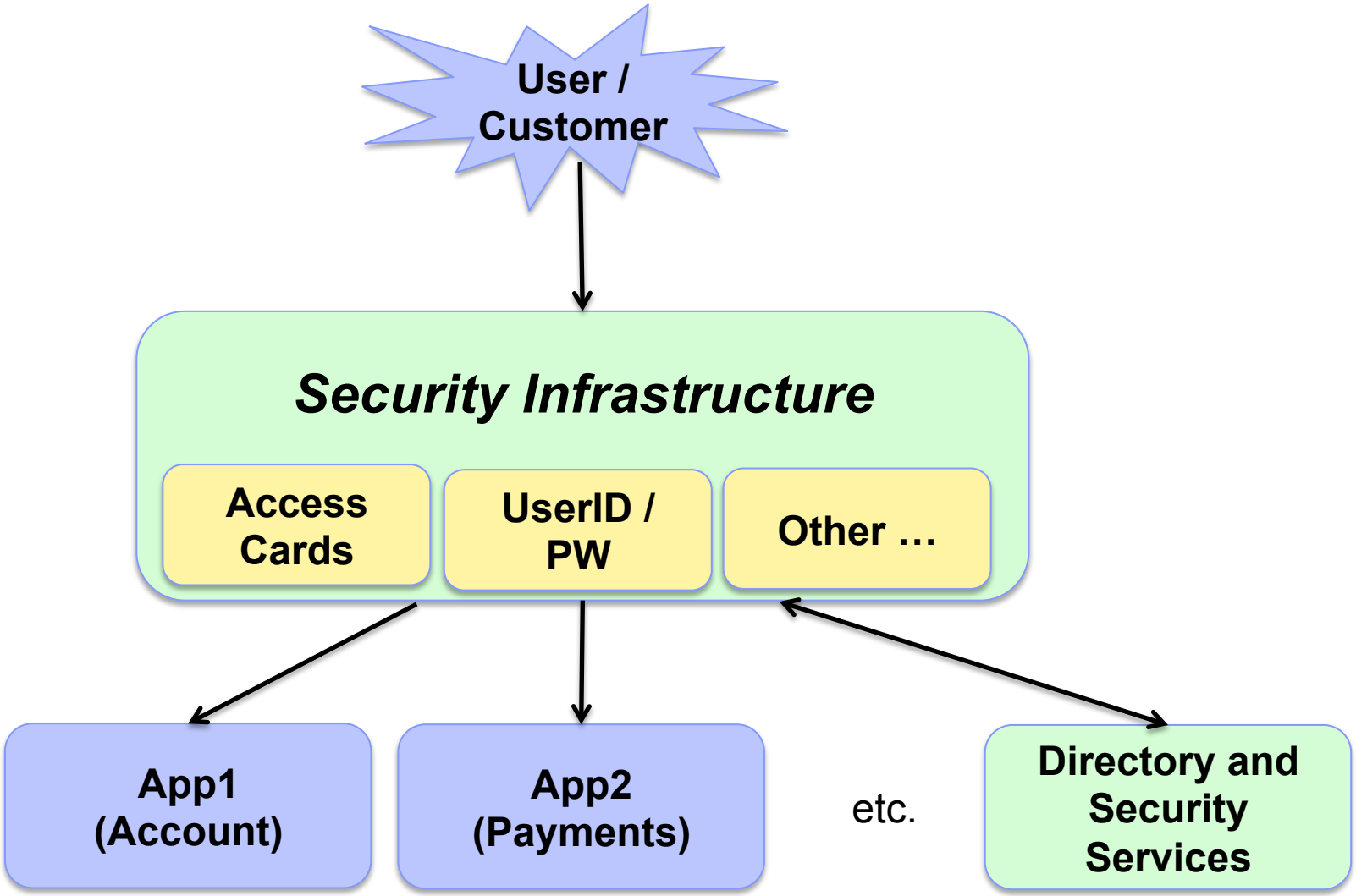
Reference Architectures (4)
Security Key Patterns: Proxy & Firewall

- (Secure) Reverse Proxy ***protects*** and provides:
  - Session Handling (supporting SSO – Single Sign-on)
  - Authentication (using Security Services and Identity Management)
  - SSL (Secure Sockets Layer)



- Firewall (WAF – Web Application Firewall) provides:
  - Validation of Input and Protocol
  - Content Inspection
  - Filtering of Requests und Responses
  - Encryption

Reference Architectures (5)
General Structure of Security Infrastructure



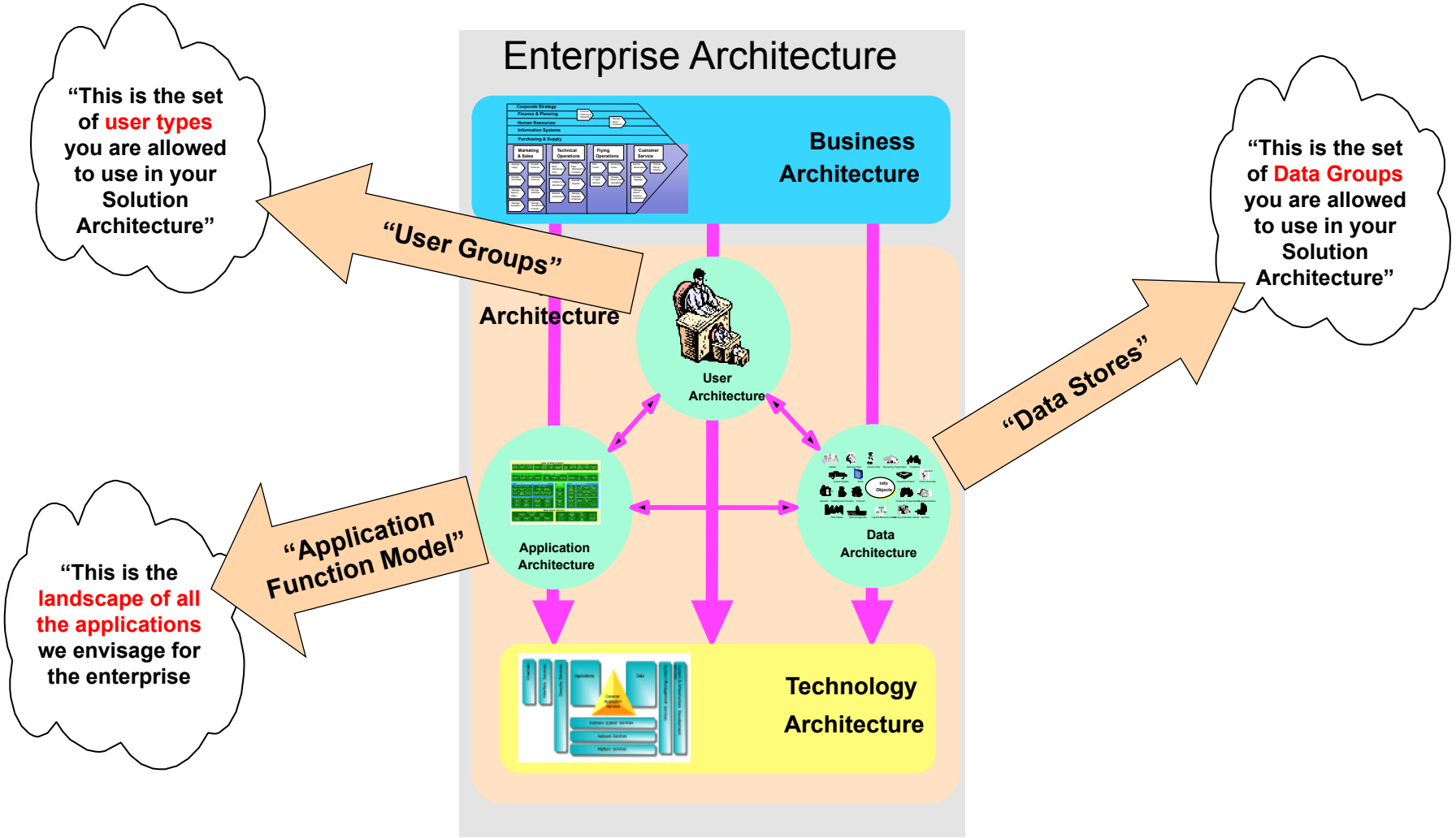
Information Systems (IS) Architecture

Information Systems (IS) Architecture – Overview

- The Information Systems Architecture provides the information about the buildings blocks to support the business
  - Objectives general addressing “application systems” and “data”
  - Capture “Baseline” and “Target”, different views

- Definition: An IS architecture describes aspects of business that are to be automated – known as the “business dependent IT architecture”. The IS Architecture manages
  - Functional Architecture
  - User Groups
  - Application Groups
  - Data Stores
  - Application Components
  - IS Services

Information Systems (IS) Architecture – Illustration



In more detail: There are two steps to defining IS Architecture: Step1: Select what functionality and information to automate…

- Understand currently automated business activities;
- Assess users groups, their roles and information requirements;
- Understand strategic decisions already made and being implemented;
  - Package selection decisions
  - Sourcing decisions
  - Programs, projects in flight
- Evaluate current portfolio against Business Architecture;
- Understand essential non functional requirements.
- Use Reference Architectures and Patterns as accelerators;
- Assess and prioritize gaps and challenges;

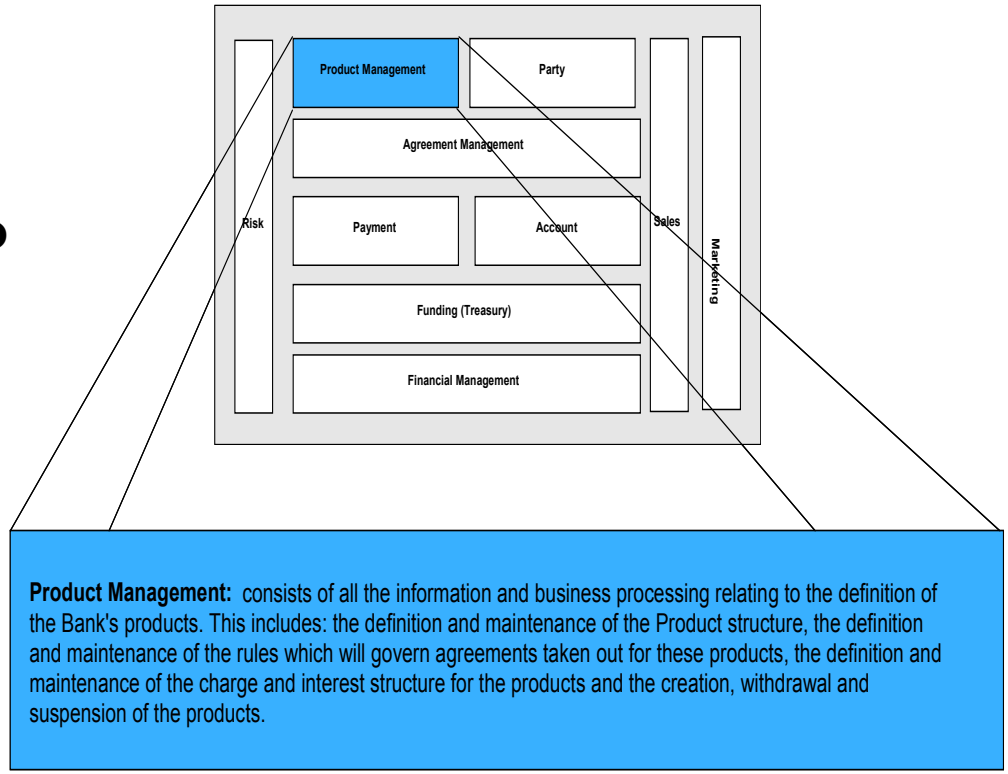
In more detail: Step 2: Decide how to package and structure this automation to deliver measurable business value.

- **"Upstream EA" (Enterprise Level)**
  - Enterprise-wide picture of all automated functionality and information ("on-a-page")
  - Coarse grained components (e.g Account Management)
  - Minimal redundancy of functionality - potential for high levels of replaceability
  - Constraints for more granular (solution) level components
  - Loose coupling between components
  - Minimize technology dependencies
- **"Downstream EA" (Solution level)**
  - Set of building blocks (ABBs) at a level Solution Architects can use.
  - Finer grained components (e.g. create account, assess credit score, check background, etc)
  - No redundancy of functionality - potential for high levels of reusability
  - Loose coupling between components
  - High cohesion within components

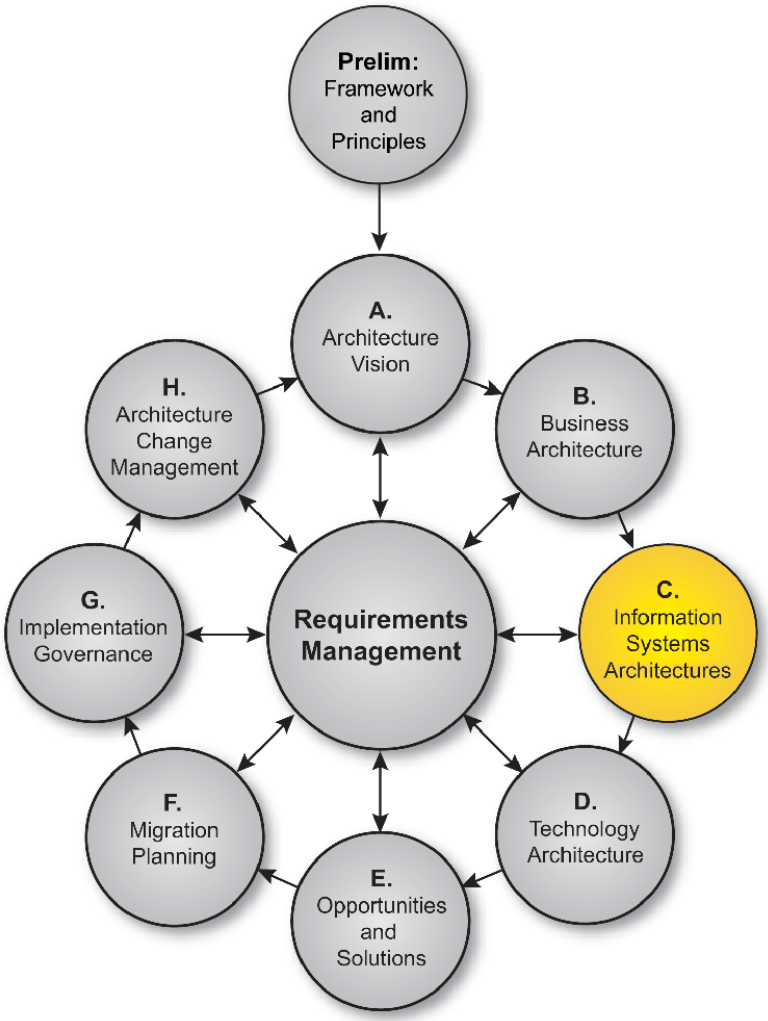
Function analysis describes how business information, business activities and user roles will be grouped & partitioned into functionality (e.g. application groups or services)

Functionality is optimal when:

- **Business functions are closely related;**
- **Scope and boundaries of each functions are clearly defined and do not overlap;**
- **Interface and interactions between functions are well-defined;**
- **Functions are able to access and share common data, rather than 'owning' data;**
- **Functions represent "reasonable" pieces of work that can be tackled by projects.**



IS Architecture Content according to TOGAF



- The fundamental organization of an IT system, embodied in
  - relationships to each other and the environment, and the principles governing its design and evolution
- Shows how the IT systems meets the business goals of the enterprise

Information Systems Architecture – Objectives (TOGAF 11.1)

- Develop the Target Application Architecture that enables the Business Architecture and the Architecture Vision, while addressing the Request for Architecture Work and stakeholder concerns

- Identify candidate Architecture Roadmap components based upon gaps between the Baseline and Target Application Architectures

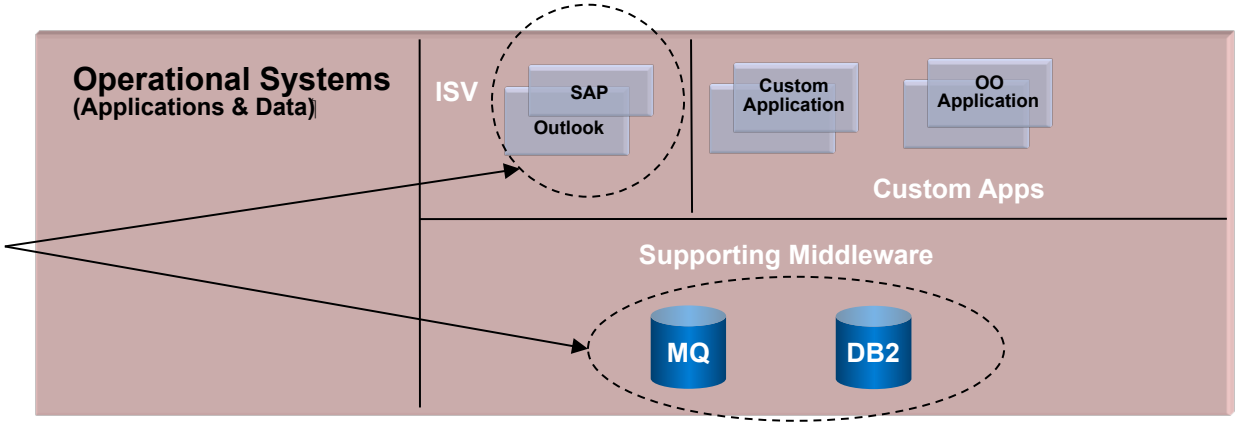
Information Systems Architecture – Steps (TOGAF 11.4)

- Select reference models, viewpoints, and tools
  - Determine Overall Modeling Process
  - Identify Required Catalogs of Application Building Blocks
  - Identify Required Matrices
  - Identify Required Diagrams
- Develop Baseline Application Architecture Description
- Develop Target Application Architecture Description
- Perform gap analysis (see Section 11.4.4)
- Define candidate roadmap components
- Resolve impacts across the Architecture Landscape
- Conduct formal stakeholder review
- Finalize the Application Architecture
- Create Architecture Definition Document

APPENDIX: SOA Layers
(see SOA RA – Open Group C119)

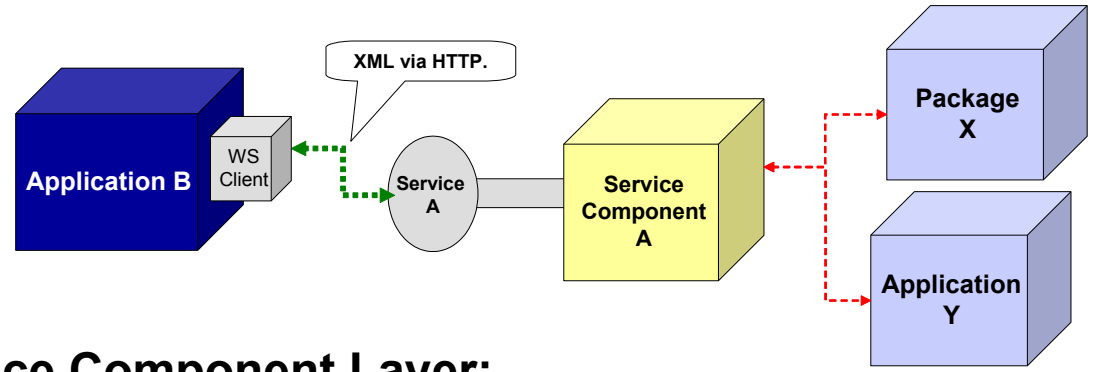
Layer 1: Operational Systems (Leverage Existing Investment)

Examples for illustration: specifics are not in the scope of the reference model.



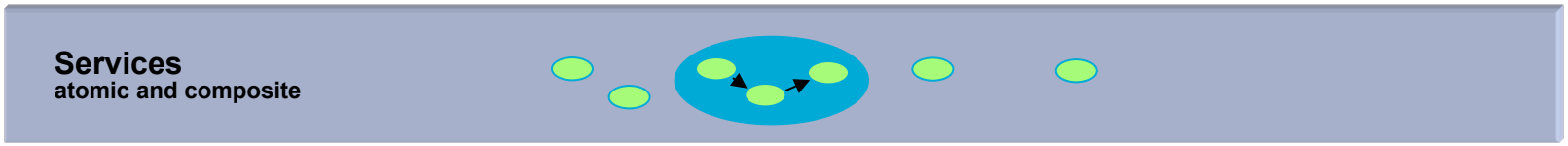
- **Recognizes the value of existing IT investment**
  - Use of existing "legacy" applications (e.g. COBOL application) and / or packages (e.g. SAP)
- **Some SOA Related Activities:**
  - Asset Inventory
  - Refactor existing applications to unlock business value

Layer 2: Service Components



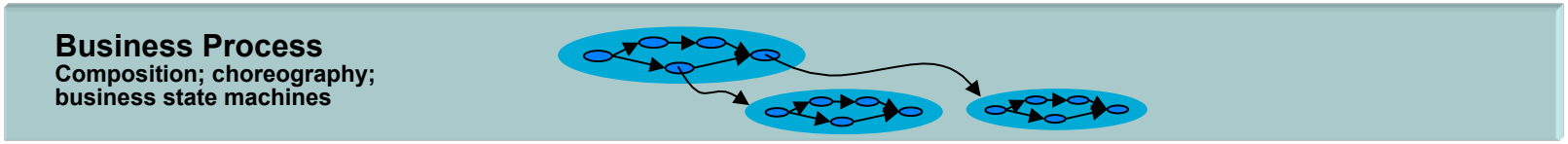
- **The Service Component Layer:**
  - **Enables IT flexibility by strengthening the decoupling in the system. Decoupling is achieved by hiding volatile implementation details from consumers.**
  - **Often employs container based technologies like EJBs**
- **Each Service Component:**
  - **Provides an enforcement point for service realization**
  - **Offers a facade behind which IT is free to do what they want/need to do**

Layer 3: Services (Decouple Business and IT)



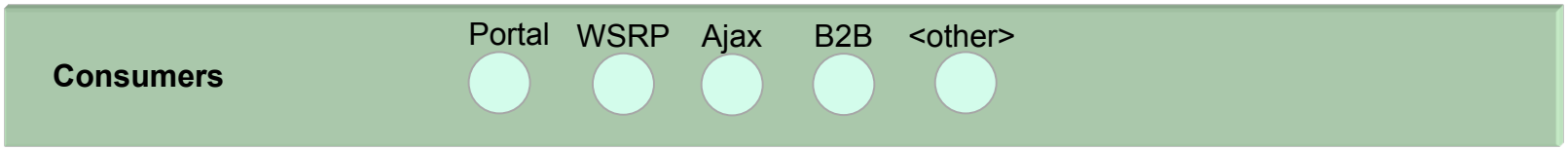
- **The Services Layer forms the basis for the decoupling of Business and IT.**
  - **Captures the functional contract (incl. QoS – Quality of Service) for each standalone *business* function or each task in a business process**
- **The assumption is that (within an SOA) IT responsibility is to realize/ manage service implementations that faithfully conform to the set of services in the service model.**
- **This layer contains all the *exposed services* in the SOA**
- **Each service is a contract between the consumer(s) and the provider(s)**

Layer 4: Business Processes (Business process alignment of IT)



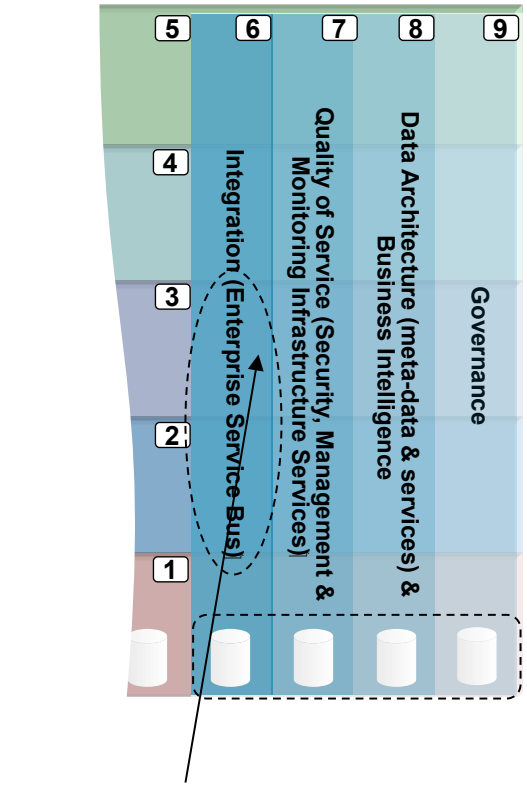
- This layer contains operational IT artifacts that implement business processes as a choreography of services
- The set of services that are composed is restricted to those services that are defined in Layer 3
- The choice of technology depends on a set of realization decisions that must be made when establishing a physical Reference Model for a given SOA

Layer 5: The Consumer Layer
(Channel independent access to business processes)



- **This layer exists to recognize that the technology chosen to expose Business Processes/Services must permit access from a wide set of interaction *channels*.**
- **It is important to populate this layer with the set of *channels* types that are required in a solution.**

Cross-cutting concerns/capabilities



- Several concerns are not restricted to a single layer in the Reference Model, these concerns are captured in ‘Layers’ 6-9
- These are not really layers but treating them as such gives us the ability focus discussions/ decisions, for example “What is found where Governance intersects Services? i.e. what are the Governance concerns specific to Services?”
- Clearly there is interaction among these ‘layers’ also. For example, it is likely that most data architectures will be subject to governance