SERVICE ORIENTED ARCHITECTURE

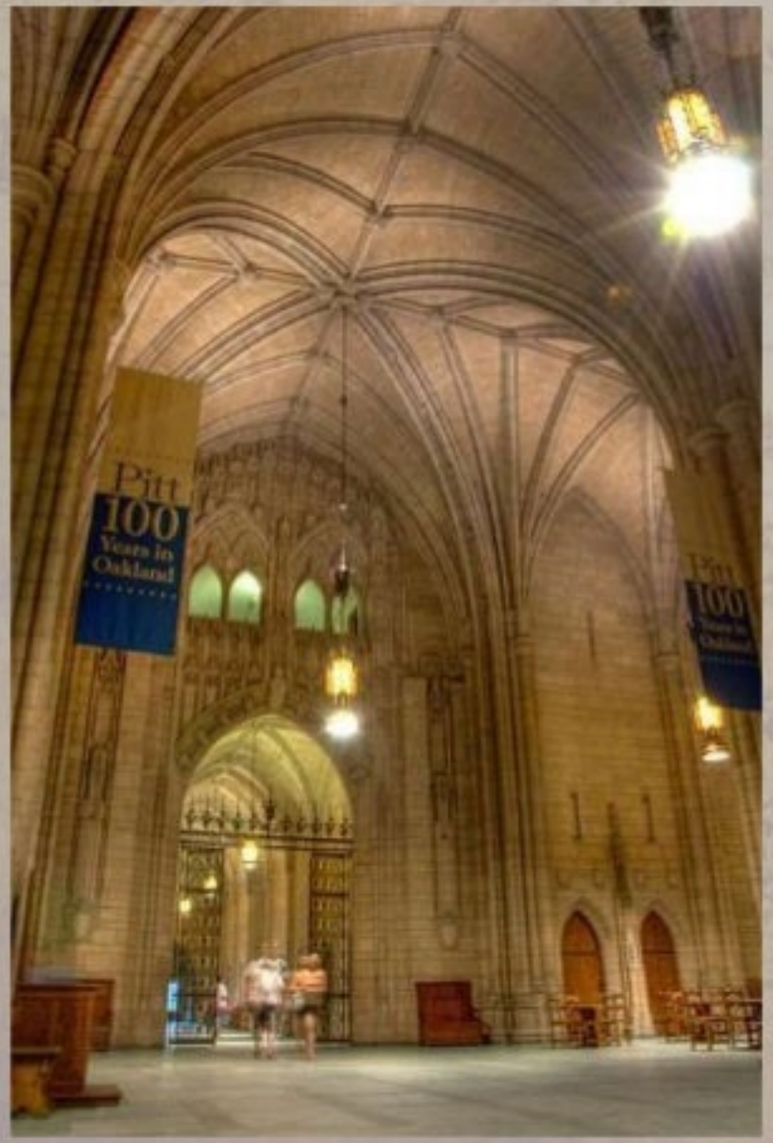
-SOA-

SOA DEFINITION

SOA DEFINED

- An architecture based on **reusable**, well-defined **services** implemented by **IT components**.
- Components are **loosely coupled.**
- Provides **platform, technology** and **language independence**

WHAT IS AN ARCHITECTURE?



Architecture implies a consistent and coherent design approach.

Essential principles include:

- **Consistency**: The same challenges should be addressed in a uniform way.
- **Reliability**: The structures created must be fit to meet the demands for which they are designed.
- **Extensibility**: A design must provide a framework that expanded in ways both foreseen and unforeseen.
- **Scalability**: The implementation must be capable of scaled to accommodate increasing load by adding hardware to the solution.

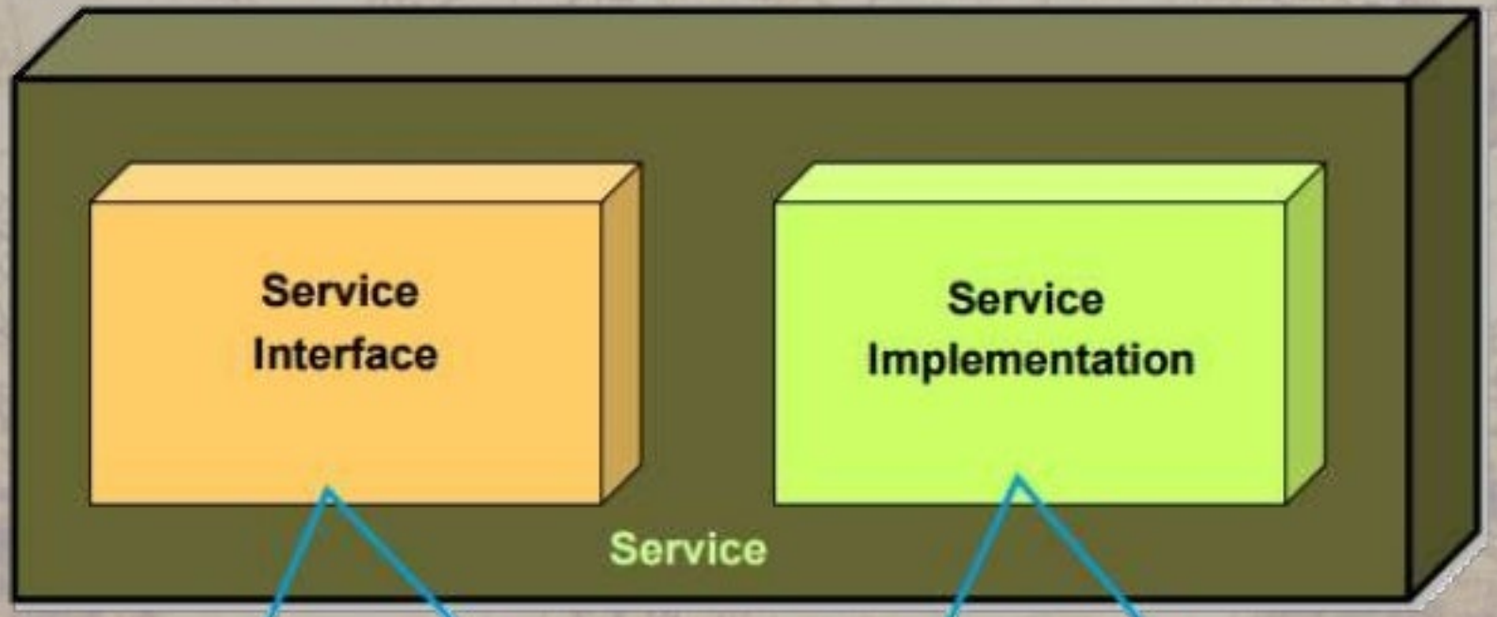
WHAT IS A SERVICE?

- A ***service*** is a package of closely related standardized functions, which are called repeatedly in a similar fashion, and should therefore be implemented by a dedicated facility, which can be specialized to perform them.

- A service can be partitioned and have multiple ***service functions***.

- The smallest subunits within service functions are called ***service primitives***.

ANATOMY OF A SERVICE



Access layer between the service consumer and service provider. It contains,

- Service Identity
- Service Input & Output data information
- Service Purpose & Function Metadata

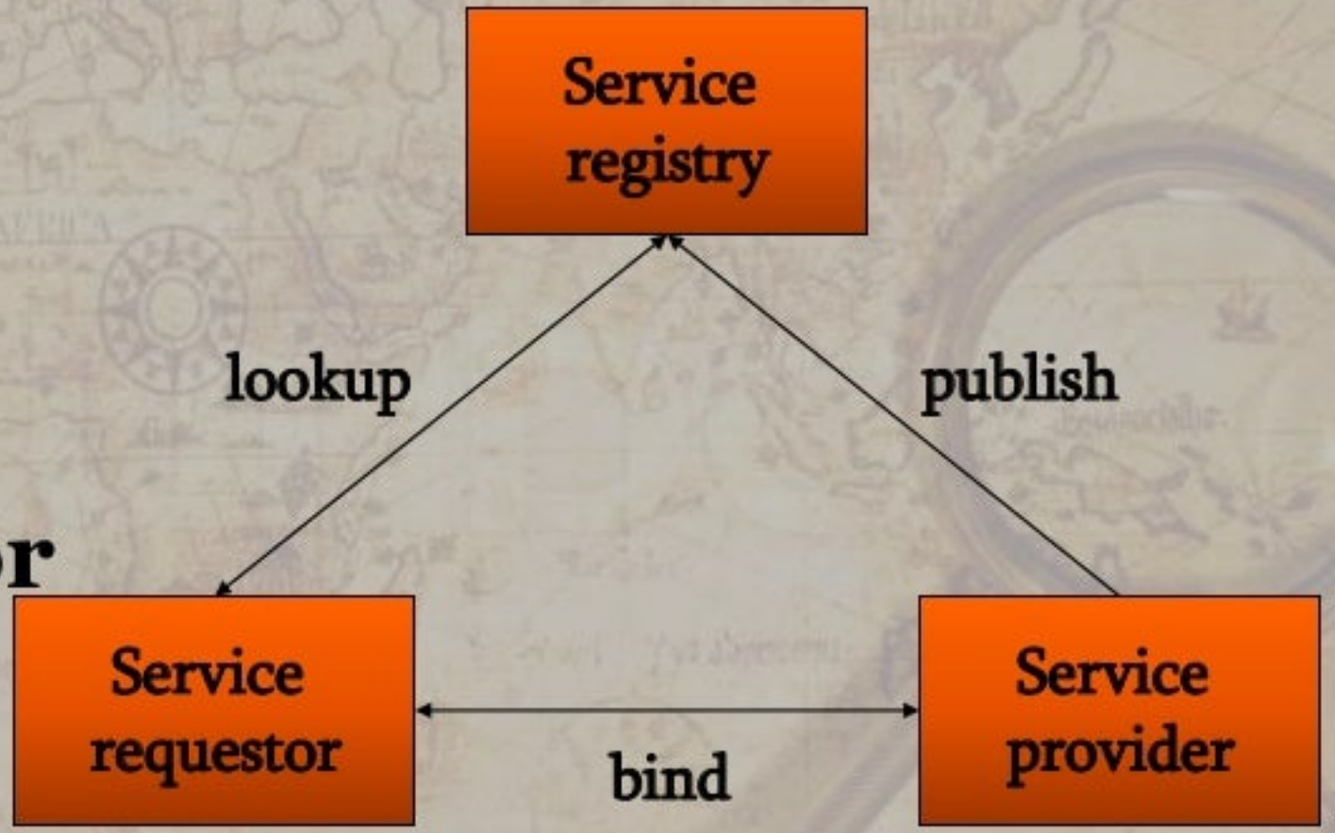
- Contains the core functional or business logic of the service.
- The implementation should be totally transparent to the service consumer, with no knowledge necessary about the implementation specifics.

SERVICE DISCOVERY

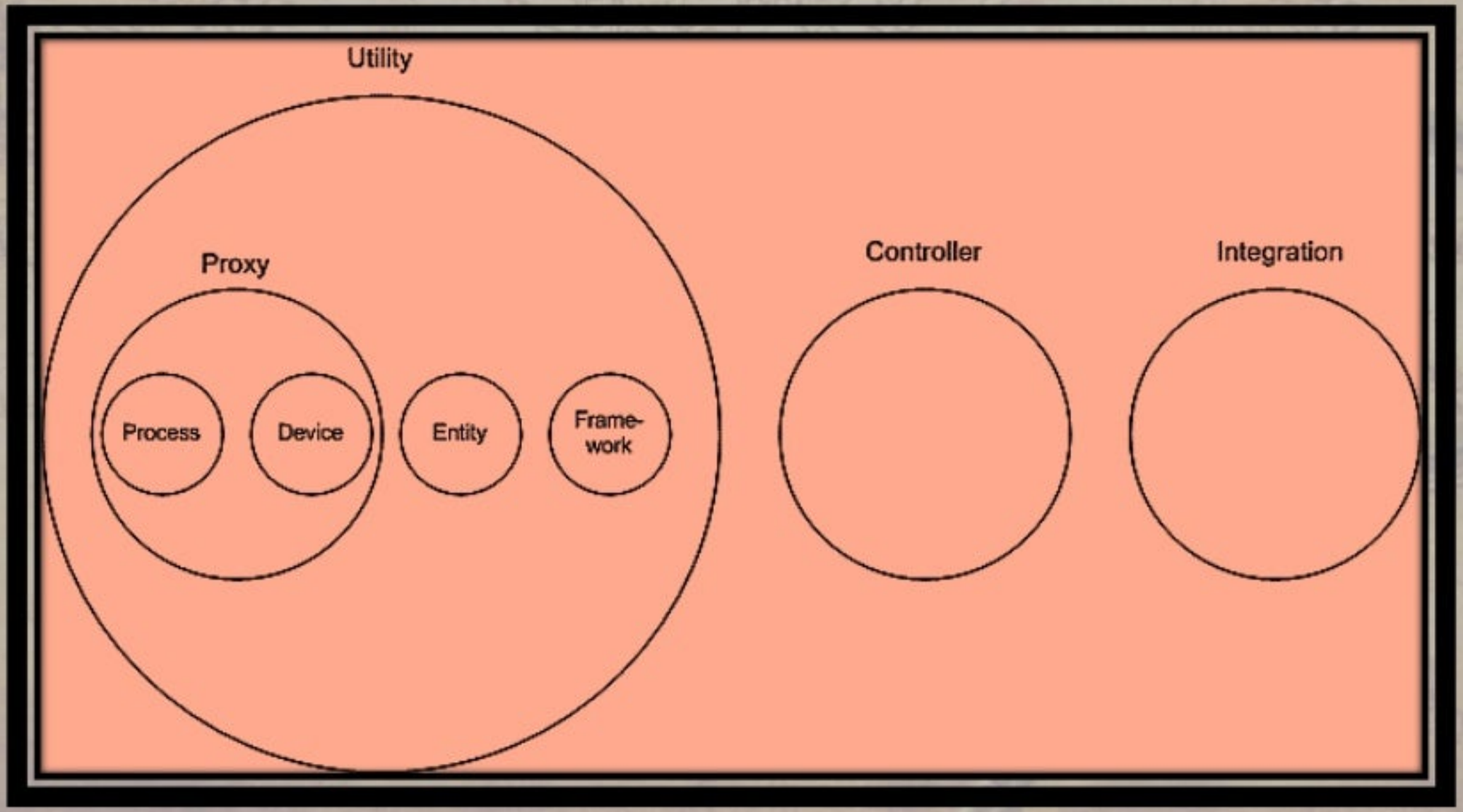
Consist of 3

components:

- **Service Registry**
- **Service Requestor**
- **Service Provider**



TYPES OF SERVICES THAT MAKE UP A SOA



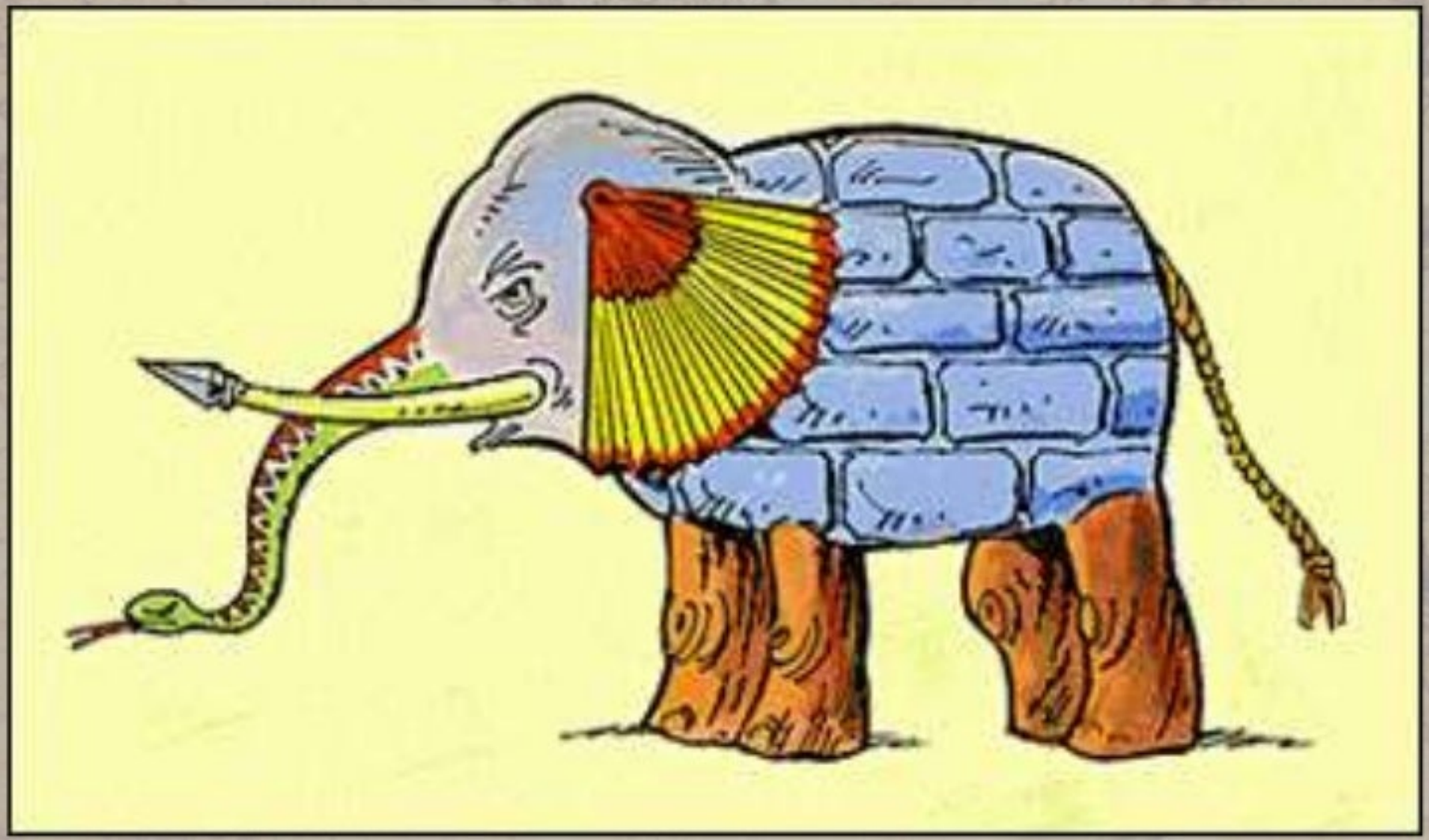
WHAT CAN SERVICES DO?

- Perform business logic
- Transform data
- Route messages
- Query databases
- Apply business policy
- Handle business exceptions
- Prepare information for use by a user interface
- Orchestrate conversations between multiple services

SERVICES OR COMPONENTS?

- A service can be defined as:
  - *A loosely-coupled, reusable software component that encapsulates discrete functionality which may be distributed and accessed. A web service is a service that is accessed using standard Internet and XML-based protocols.*

- A critical distinction between a service and a component as defined in component-based software engineering is that services are independent.
  - *Services do not have a 'requires' interface.*
  - *Services rely on message-based communication with messages in XML.*

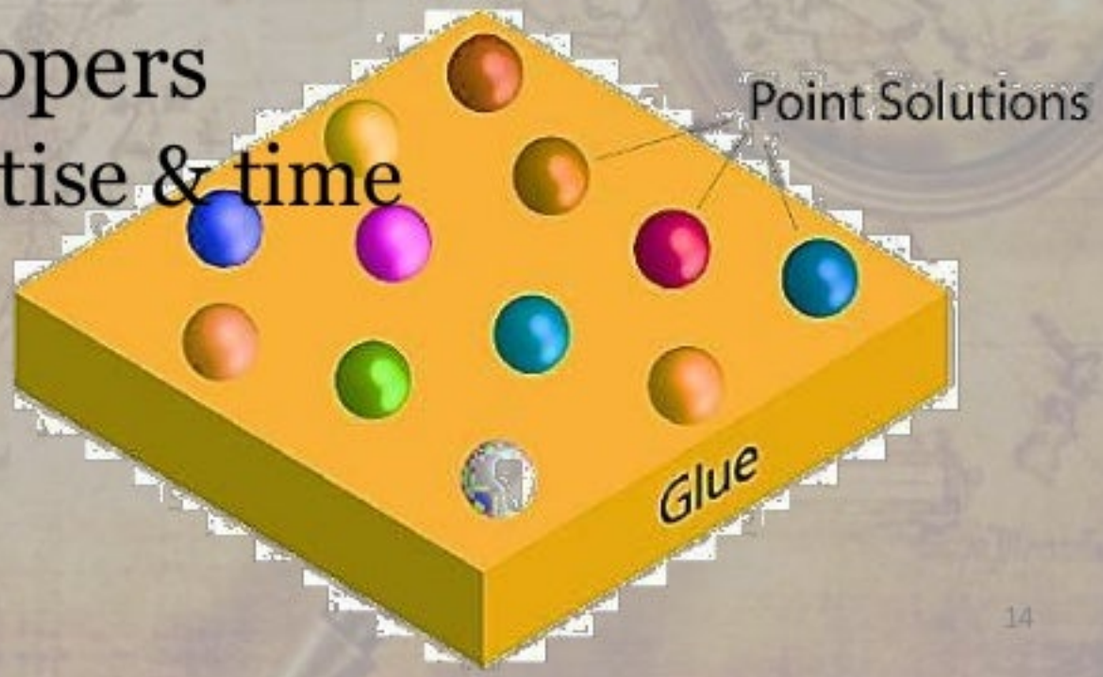


SOA

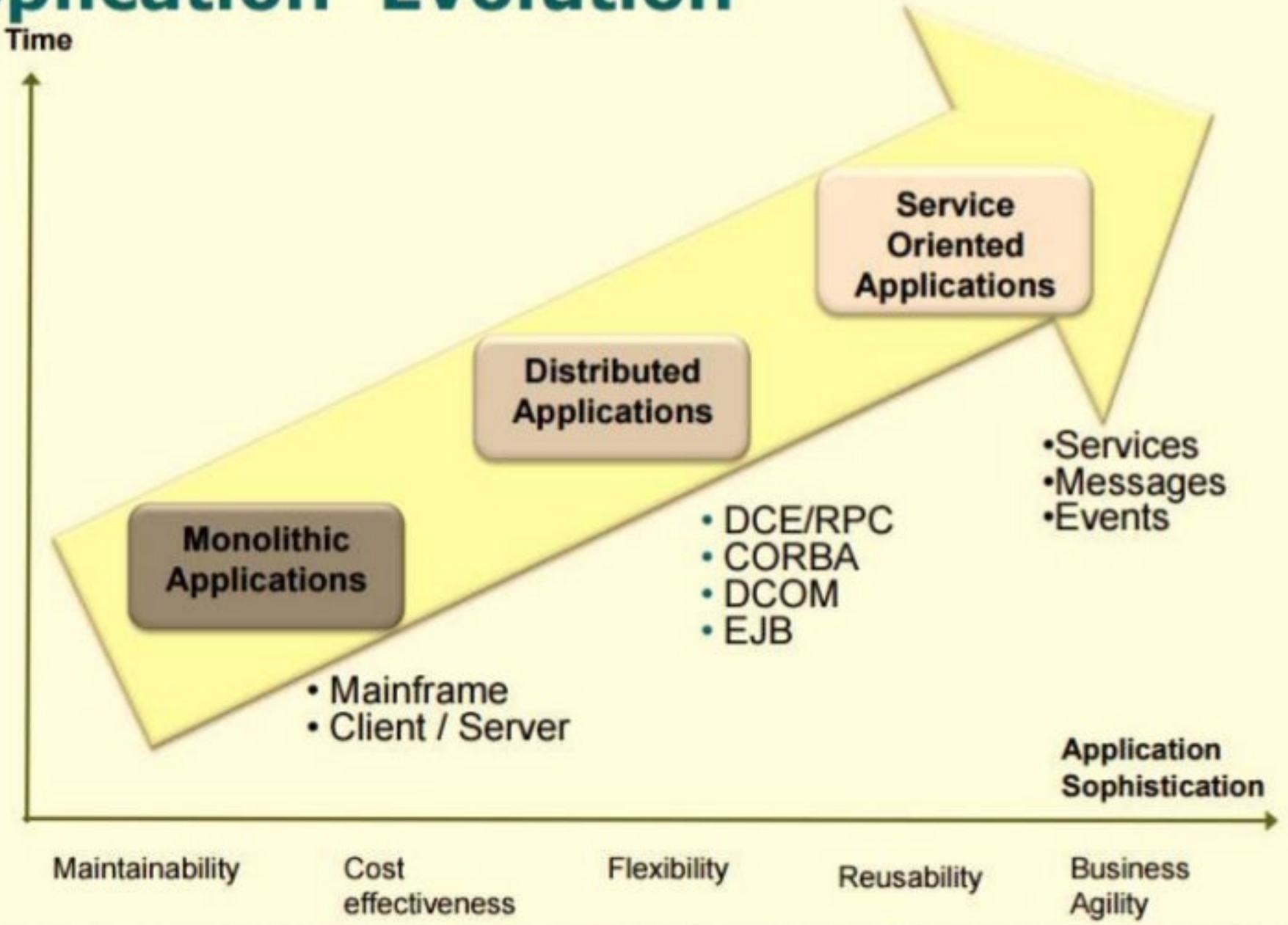
TRADITIONAL APPLICATION DEVELOPMENT

- Point technologies, products, and APIs
  - For example: EJB, Spring, Hibernate, JSF, Servlets, Struts, etc.

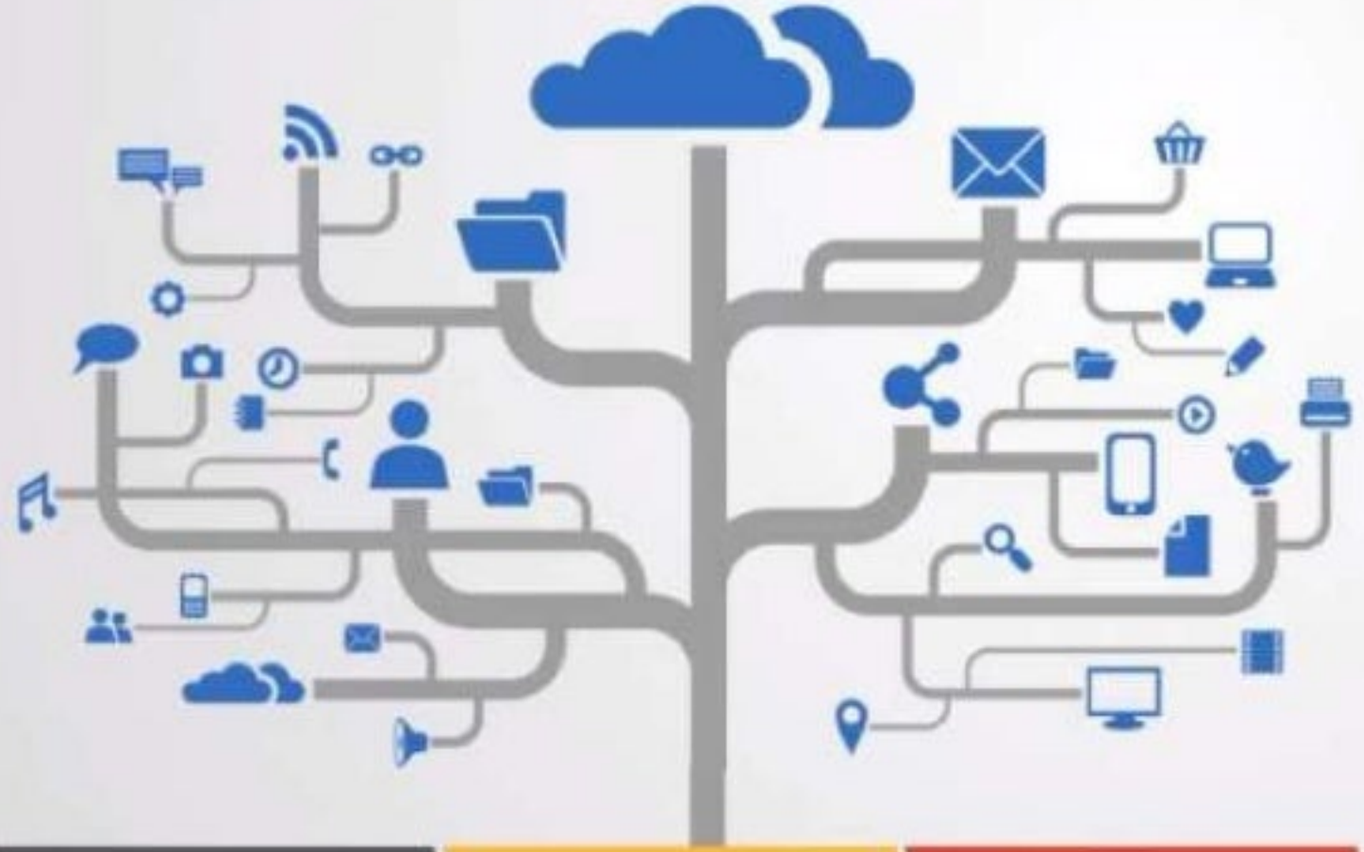
- Lots of glue written by developers
  - Requires a great deal of expertise & time
  - Inflexible



Application Evolution



Services orientated architecture



1970 1980 1990 2000 2010

WHAT IS SOA?

- **Gartner** – A software architecture that starts with an interface definition and builds the entire application topology as a topology of interfaces, interface implementations and interface calls.
- **IBM** – An application framework that takes everyday business applications and breaks them down into individual business functions and processes, processes, called services.
- **Microsoft** – A world-wide mesh of collaborating services that are published and available for invocation on a Service Bus.
- **BearingPoint** – A software design & implementation approach ("Architecture") of loosely coupled, coarse grained, reusable artifacts ("Services"), which can be integrated with each other, through a wide variety of platform independent service interfaces.

- **An architectural framework that enables us to:**
  - Create reusable services that are highly interoperable through the use of broadly-supported standards.
  - Support a new generation of agile composite applications applications assembled from business, application, and and technical services.
  - Create external / business partner interfaces to streamline streamline inter-enterprise integration, replacing EDI, EDI, VANs, etc.
  - Aligns business requirements with IT technology assets


Service Provider
Service Request
Service Response
Service Consumer

Services over components

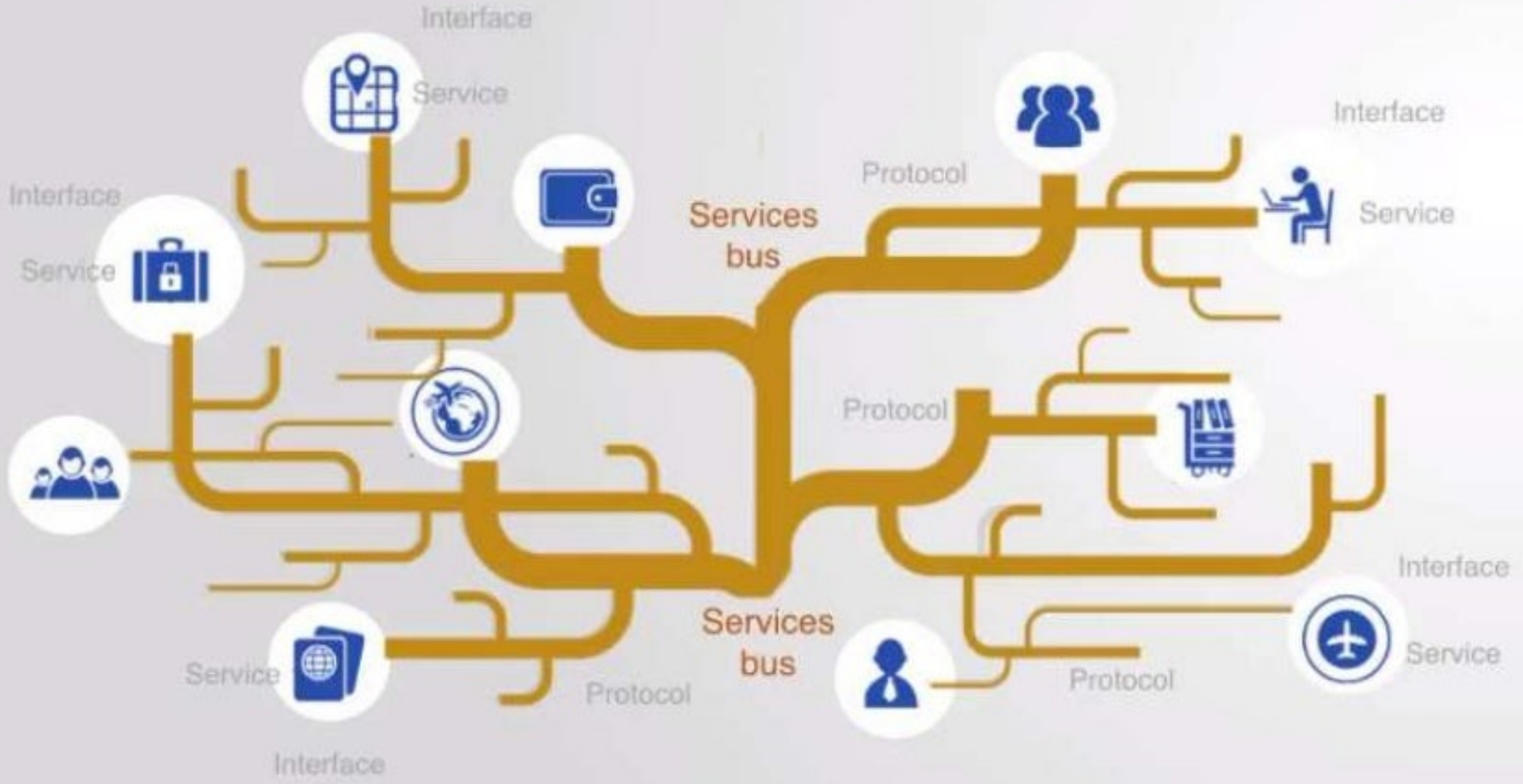
Interoperability & cross platform

Loose coupling & distributed

Abstraction again complexity

WHY IS SOA DIFFERENT?

- **Terminology**: Both IT people and business people know what a service is.
- **Interoperability**: The interfaces and the wire protocols are based on standards.
- **Extension** and **Evolution** not rip and replace.
- **Reuse** of both functionality and machine resources.


Interface
Service
Interface
Service
Services bus
Protocol
Interface
Service
Protocol
Services bus
Interface
Service
Service
Protocol
Protocol
Interface

SOA-ENABLED SCENARIO

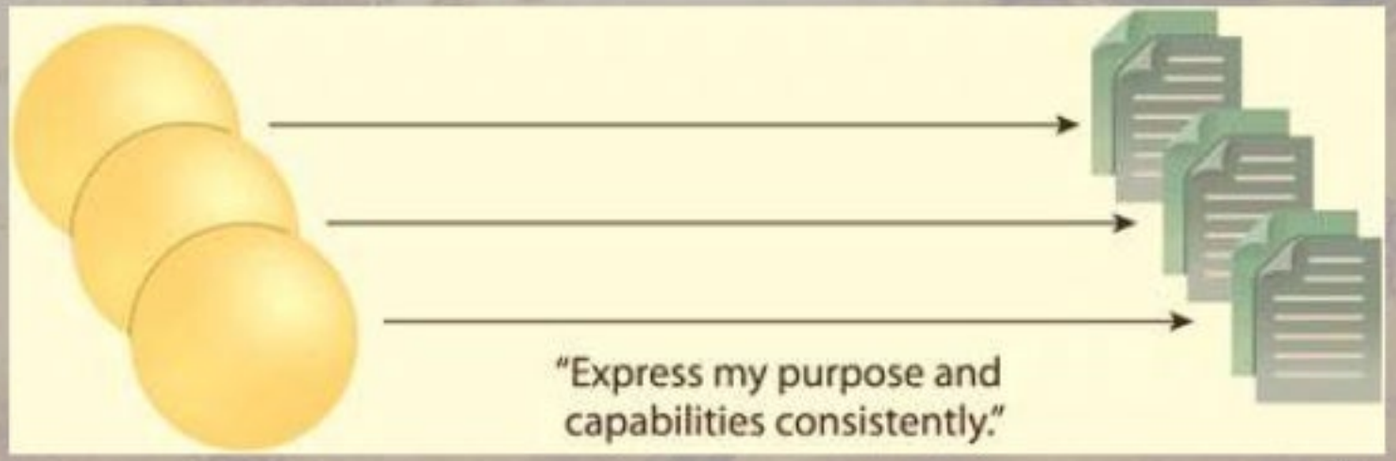
SOA ARCHITECTURE CHARACTERISTICS

SOA ARCHITECTURAL CHARACTERISTICS/PRINCIPLES

- Standardized Service Contracts
- Loose Coupling
- Abstraction
- Reusability
- Autonomy
- Statelessness
- Discoverability
- Composability

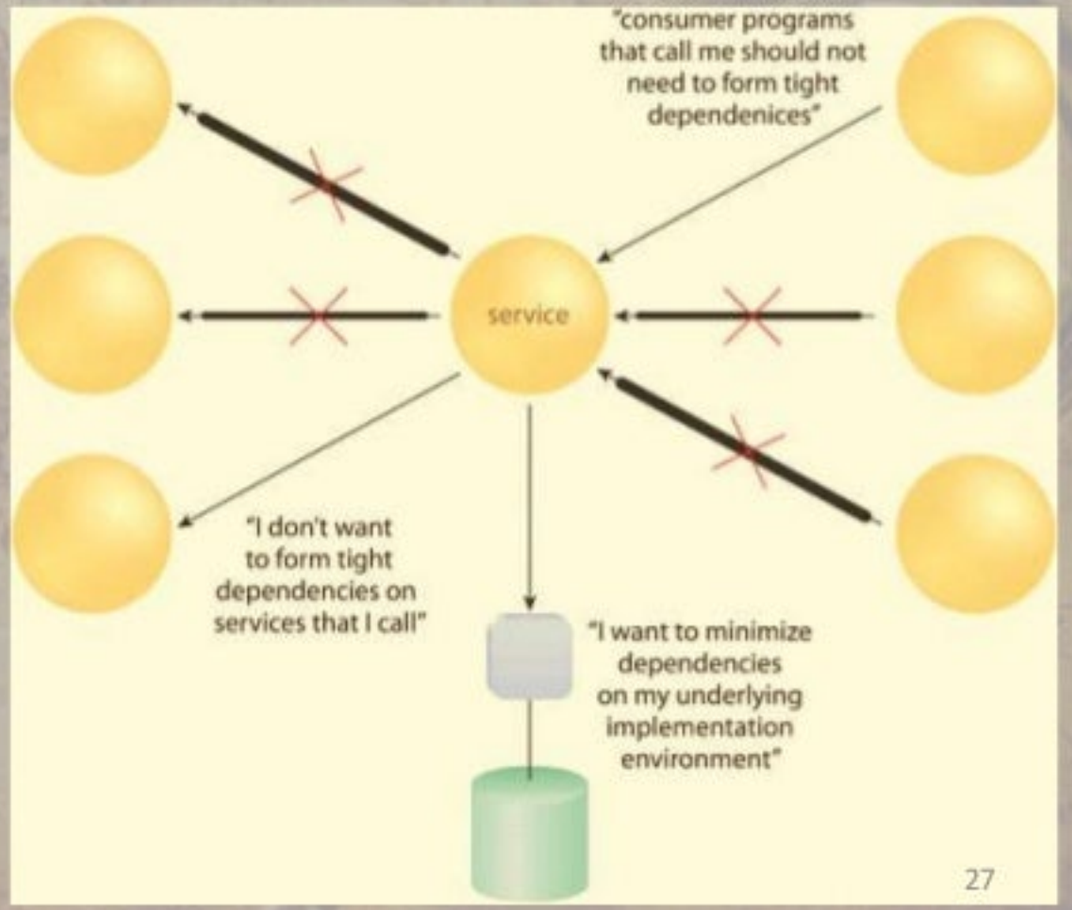
Standardized Service Contracts

- Services adhere to a service-description.
- Services use service contract to
  - Express their purpose
  - Express their capabilities
- Use formal, standardized service contracts



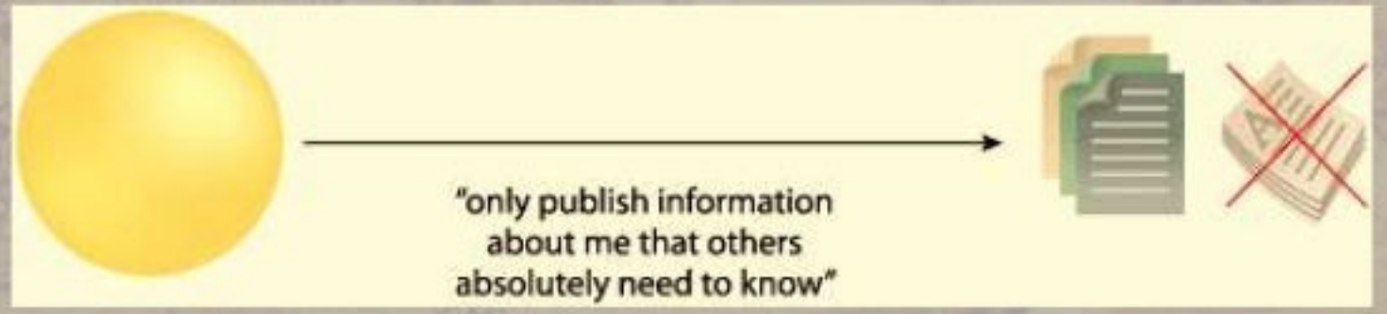
Loose Coupling

- Services minimize dependencies on each other.
- Create specific types of relationships within and outside of service boundaries with a constant emphasis on reducing ("loosening") dependencies dependencies between
  - Service contract
  - Service implementation
  - Service consumers



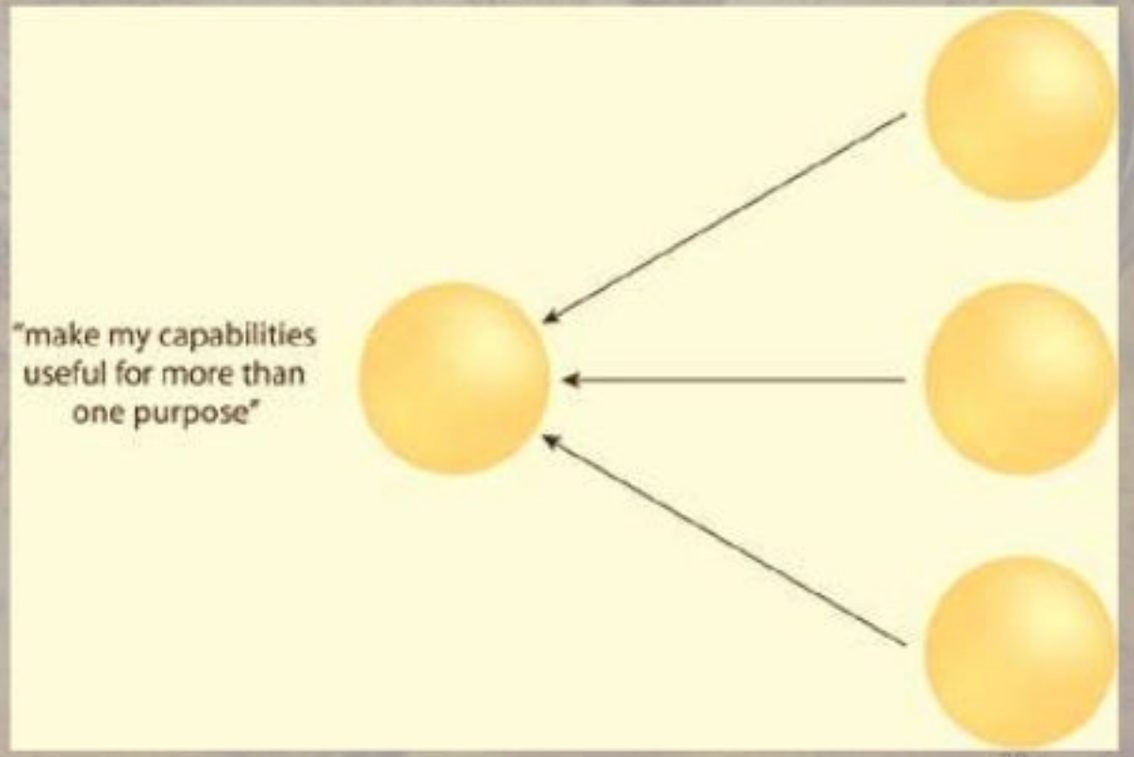
Abstraction

- Services hide the logic they encapsulate from the outside world.
- Avoid the proliferation of unnecessary service information, meta-data. data.
- Hide as much of the underlying details of a service as possible.
  - Enables and preserves the loosely coupled relationships
  - Plays a significant role in the positioning and design of service compositions compositions



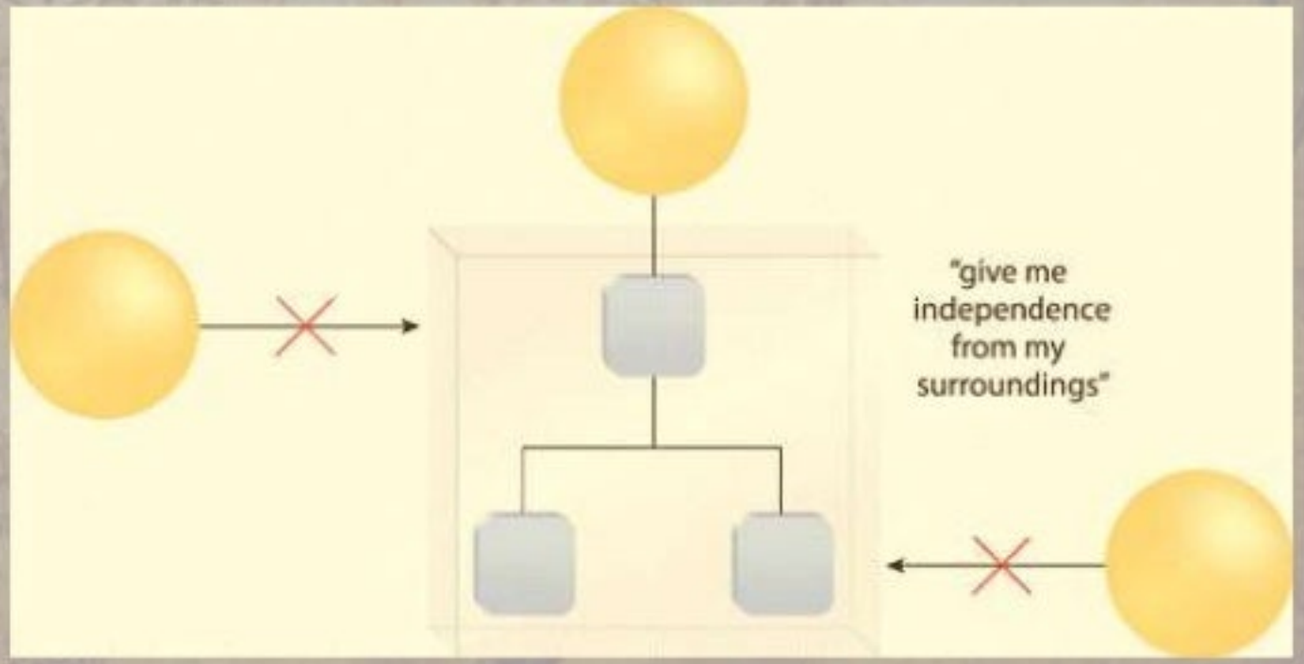
Service Reusability

- Logic is divided into services with the intent of maximizing reuse



Autonomy

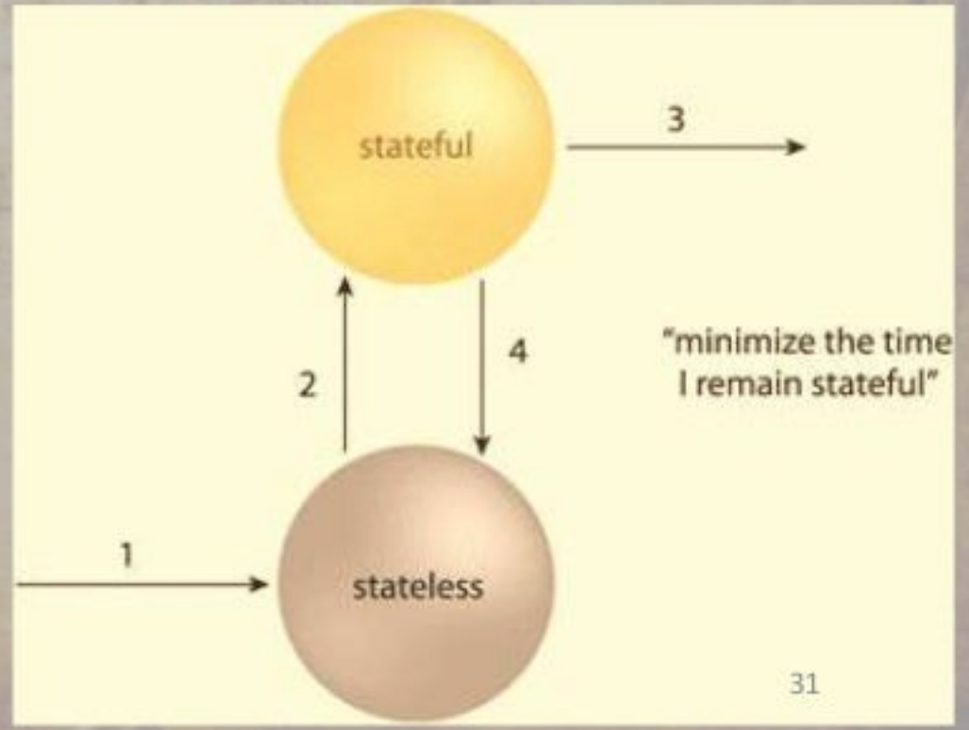
- Services should have control over the logic they encapsulate
- Represents the ability of a service to carry out its logic independently independently of outside influences
- To achieve this, services must be more isolated
- Primary benefits
  - Increased reliability
  - Behavioral predictability



Statelessness

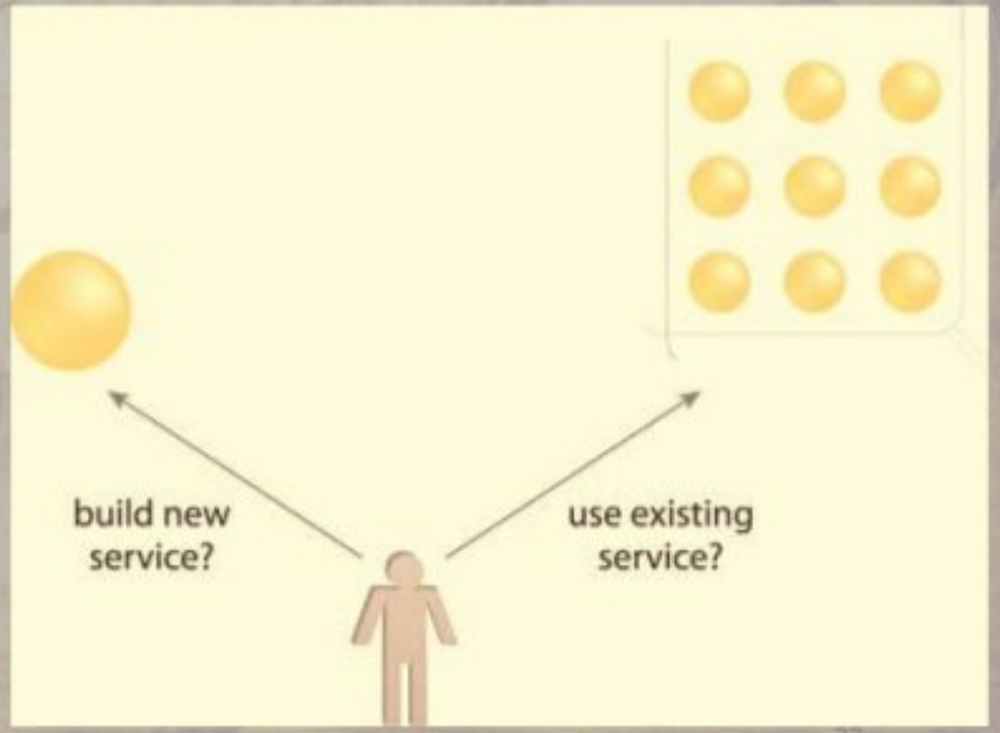
- Ideally, services should be stateless.

- Incorporate state management deferral extensions within a service service design Goals
  - Increase service scalability
  - Support design of agnostic logic and improve service reuse



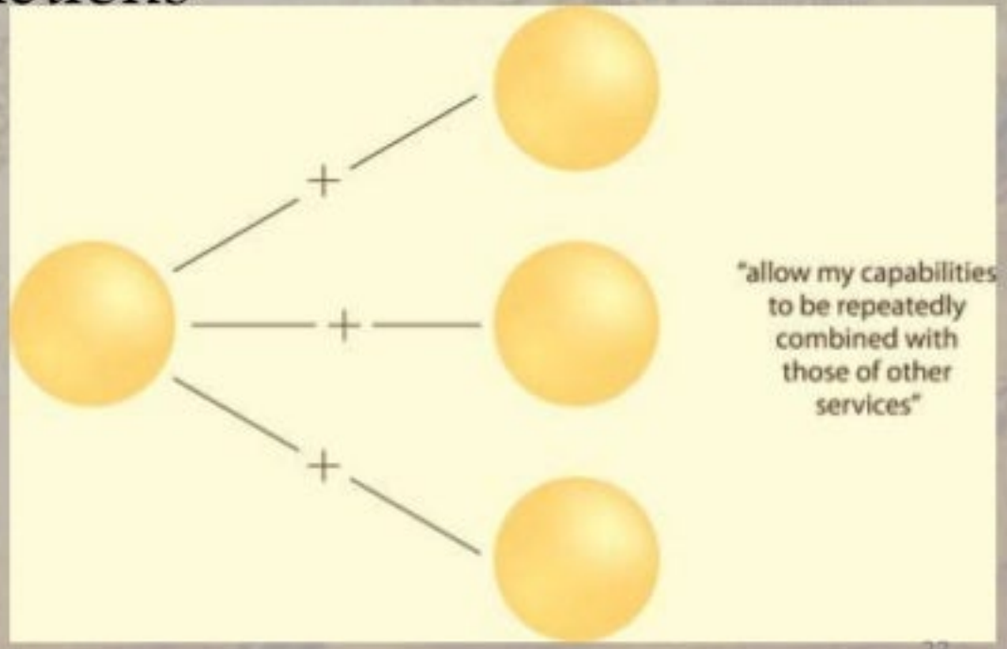
Discoverability

- Services can be discovered (usually in a service registry).
- Service contracts contain appropriate meta data for discovery which which also communicates purpose and capabilities to humans
- Store meta data in a service registry or profile documents



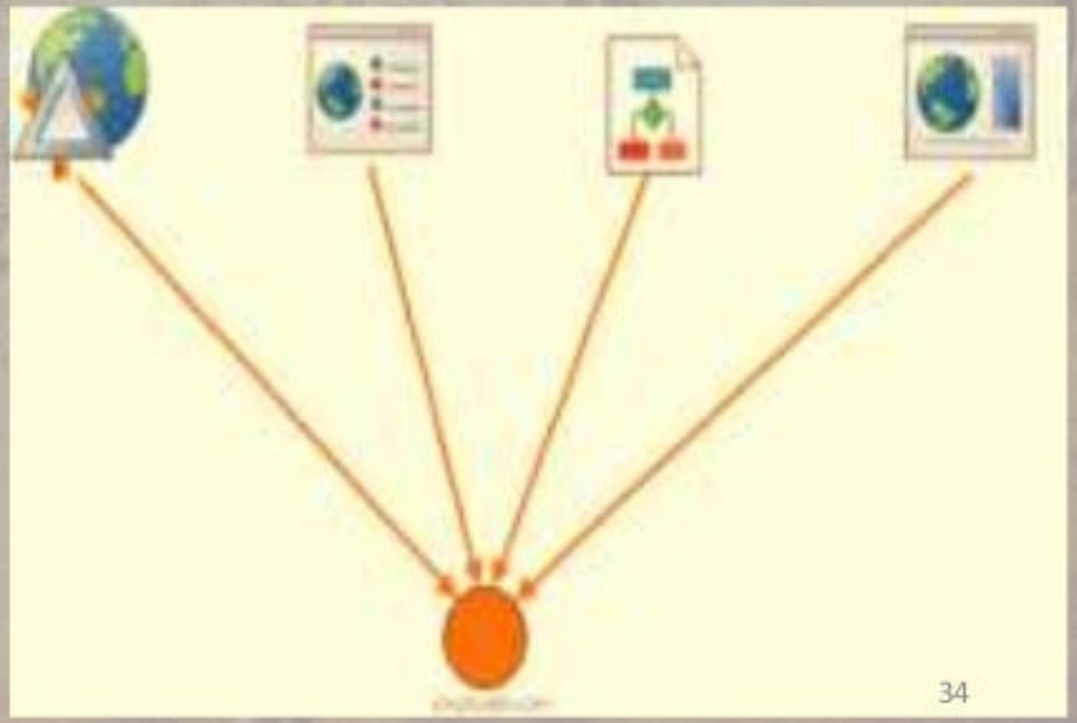
Composability

- Services break big problems into little problems.
  Related to Reusability principle
- Service execution should efficient in that individual processing should be highly tuned
- Flexible service contracts to allow different types of data exchange exchange requirements for similar functions

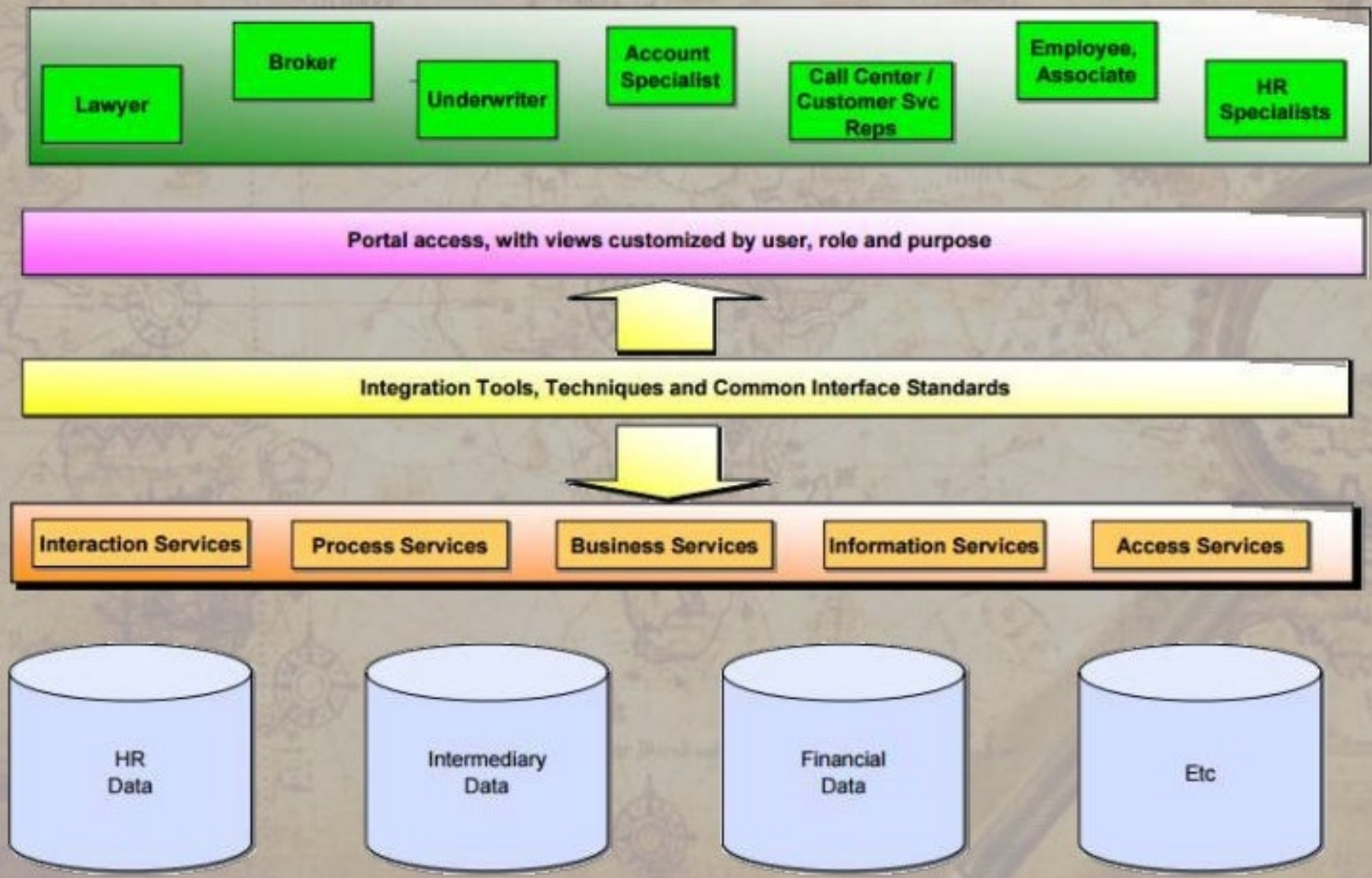


Interoperability

- Services should use standards that allow diverse subscribers subscribers to use the service.
- This is considered so obvious these days that it is often dropped as a principle.



SOA CONCEPTUAL MODEL



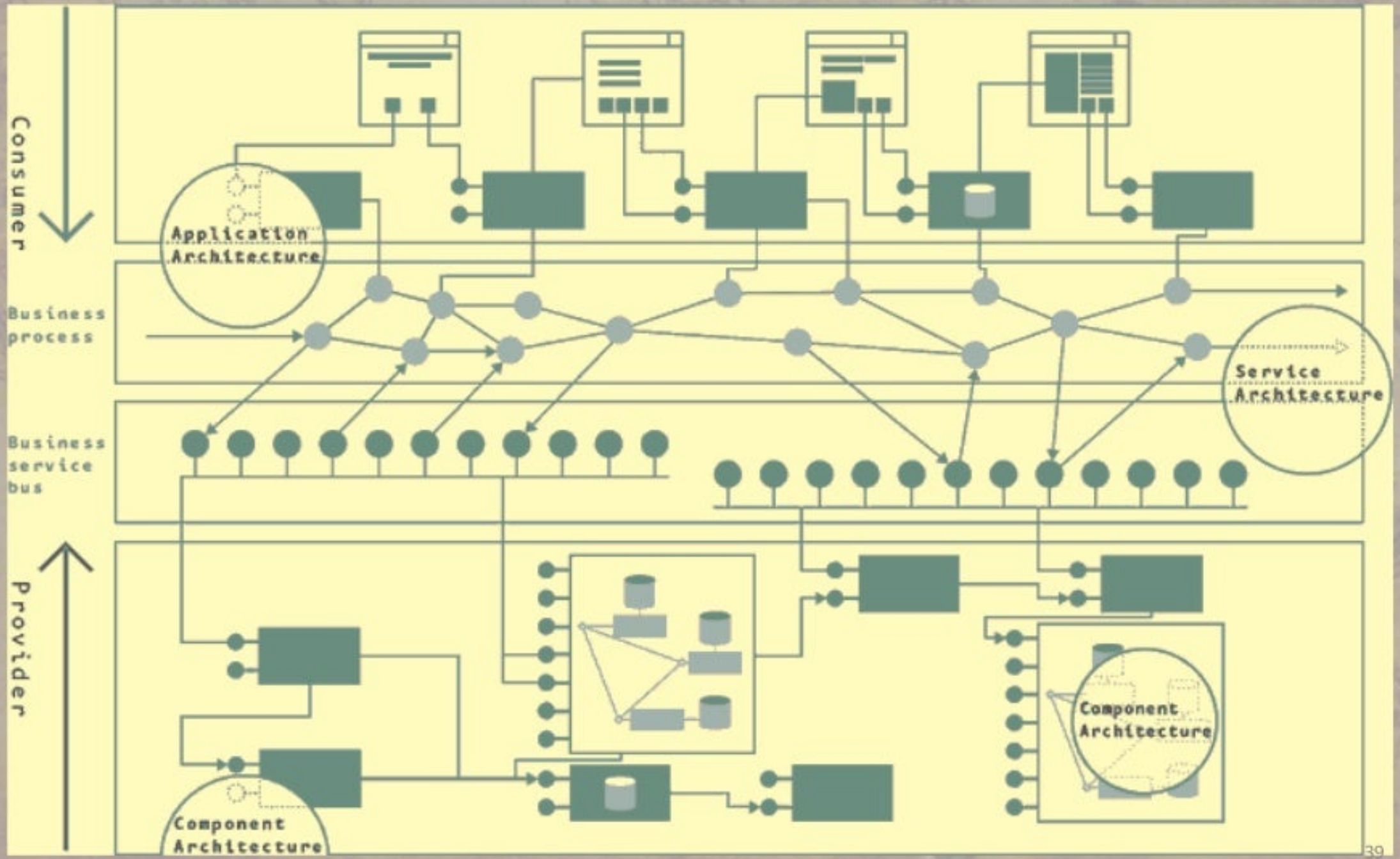
SOA ARCHITECTURE

SOA ARCHITECTURE

For SOA there are three important architectural perspectives:

- The Application Architecture
- The Service Architecture
- The Component Architecture

- These architectures can be viewed from either the consumer or provider perspective.
- Key to the architecture is that the consumer of a service should not be interested in the implementation detail of the service—just the service provided.
- The consumer is focused on their application architecture, the services used, but not the detail of the component architecture.
- Similarly, the provider is focused on the component architecture, the service architecture, but not on the


Consumer
Application Architecture
Business process
Service Architecture
Business service bus
Provider
Component Architecture
Component Architecture

SERVICE ARCHITECTURE

- At the core of the SOA is the need to be able to manage services as first order deliverables.
- It is the service that we have constantly emphasized that is the key to communication between the provider and consumer.
- So we need a Service Architecture that ensures that services don't get reduced to the status of interfaces, rather they have an identity of their own, and can be managed individually and in sets.
- CBDI developed the concept of the Business Service Bus

BUSINESS SERVICE BUS

- The BSB is a logical view of the available and used services for a particular business domain, such as Human Resources or Logistics.
- The purpose of the BSB is so that common specifications, policies, etc can be made at the bus level, rather than for each individual service
- It also facilitates the implementation of a number of common, lower-level business infrastructure services that can be aggregated into other higher level business services on the same bus.

- A key question for the Service Architecture is

'What is the scope of the service that is published to the Business Service Bus?'

- A simplistic answer is

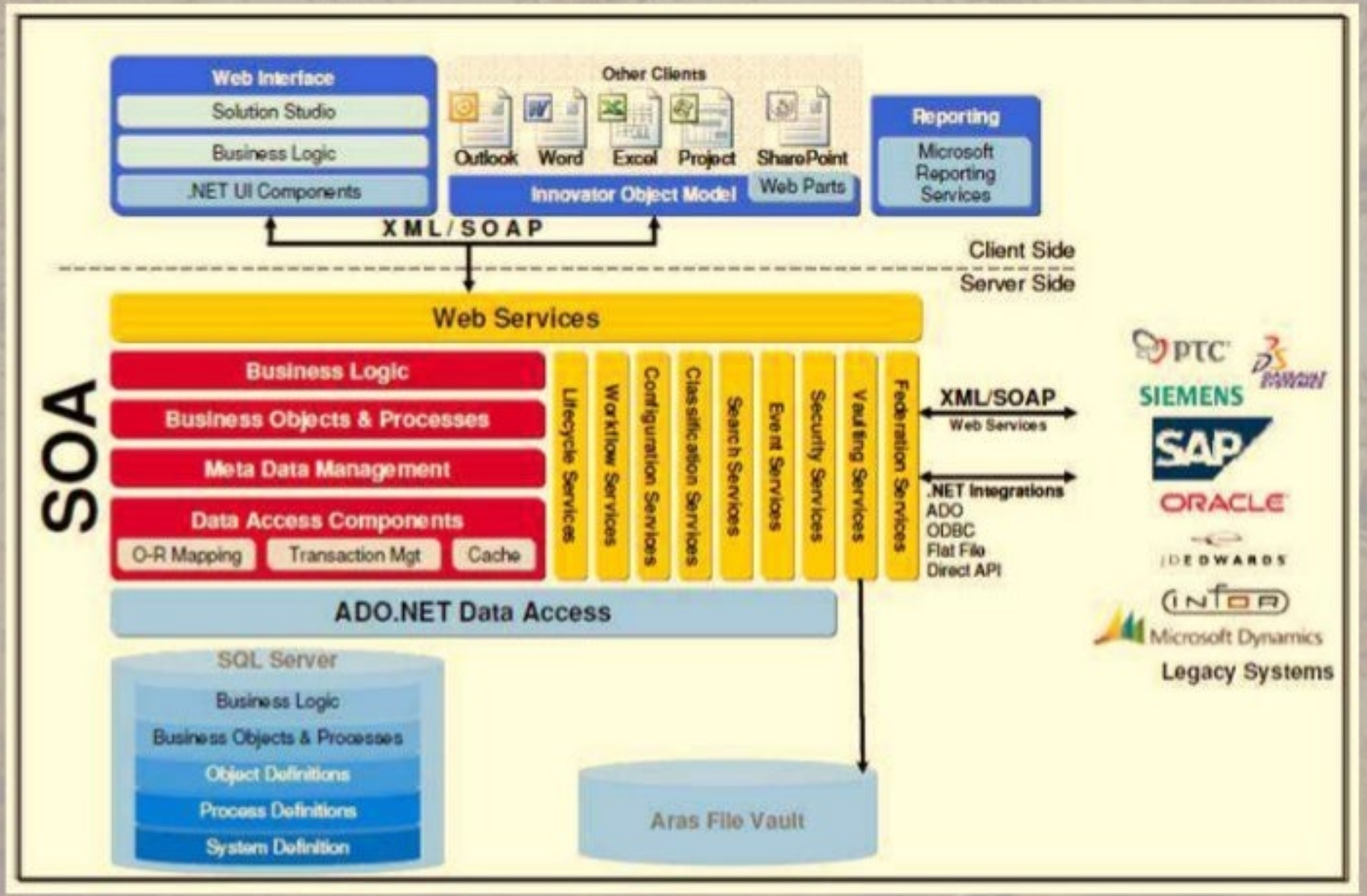
'At a business level of abstraction'

- In principle, the level of abstraction will be developed such that services are at a level that is relevant and appropriate to the consumer.

- The level might be one or all of the following:
  - Business Services
  - Service Consumer Oriented
  - Agreed by both Provider and Consumer
  - Combine low-level implementation-based services into something meaningful to business
  - Coarser Grained
  - Suitable for External Use
  - Conforms to pre-existing connection design

THE SOA PLATFORM

THE SOA PLATFORM



- The key to separation is to define a **virtual platform** that is equally relevant to a number of real platforms.

- The objective of the virtual platform is to enable the separation of services from the implementation to be as complete as possible and allow components built on various implementation platforms to offer services which have no implementation dependency.

- The virtual SOA platform comprises a blueprint which covers the development and implementation platforms.

- The blueprint provides guidance on the development and implementation of applications to ensure that the published services conform to the same set of structural principles that are relevant to the management and consumer view of the services.

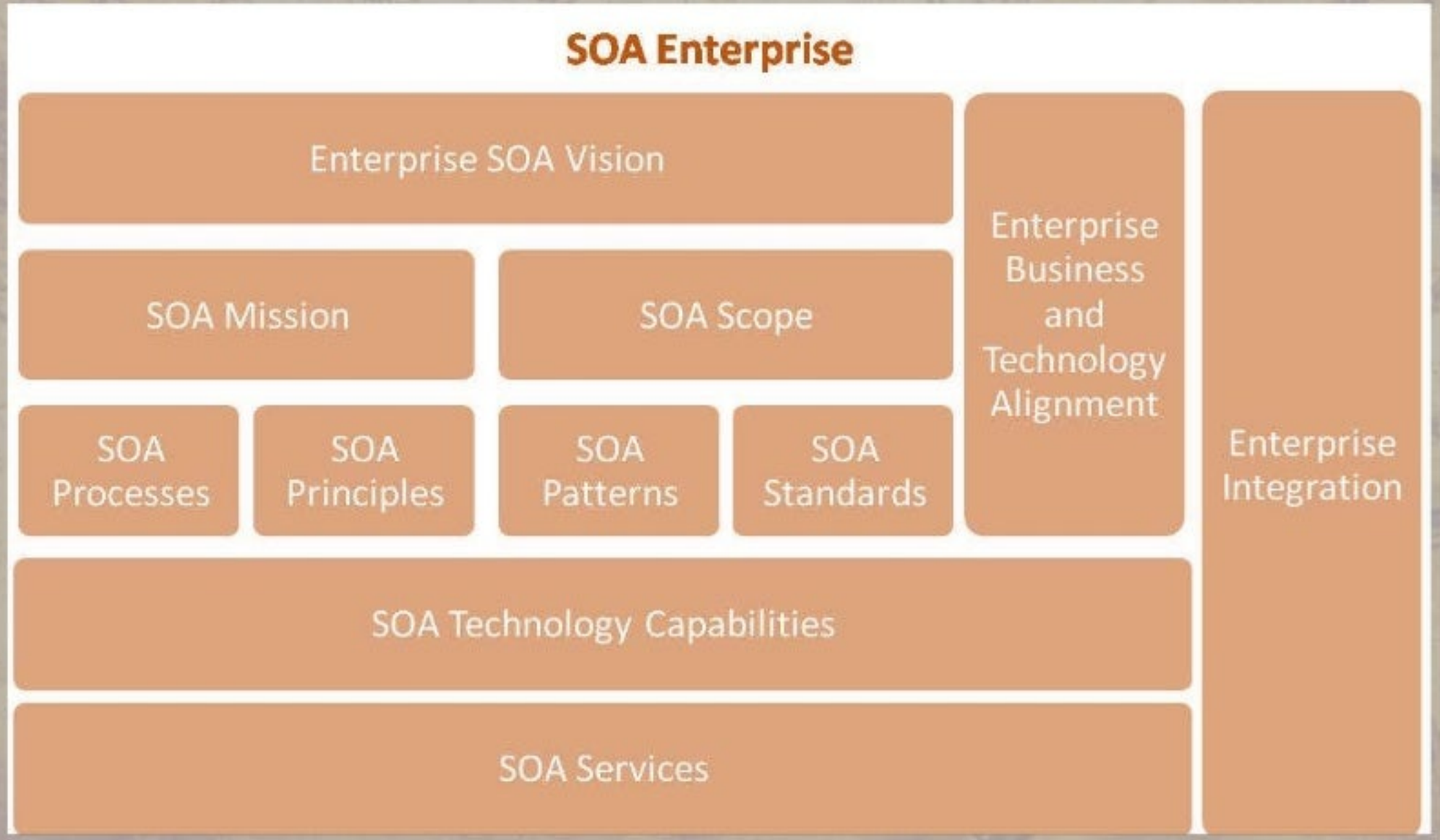
- When a number of different applications can all share the same structure, and where the relationships between the parts of the structure are the same, then have what might be called a common architectural style.

- The style may be implemented in various ways;
  - it might be a common technical environment
  - a set of policies, frameworks or practices
- Example platform components of a virtual platform include:

- Host environment
- Consumer environment
- Middleware
- Integration and assembly environment
- Development environment
- Asset management
- Publishing & Discovery
- Service level management

- Security infrastructure
- Monitoring & measurement
- Diagnostics & failure
- Consumer/Subscriber management
- Web service protocols
- Identity management
- Certification
- Deployment & Versioning

THE ENTERPRISE SOA



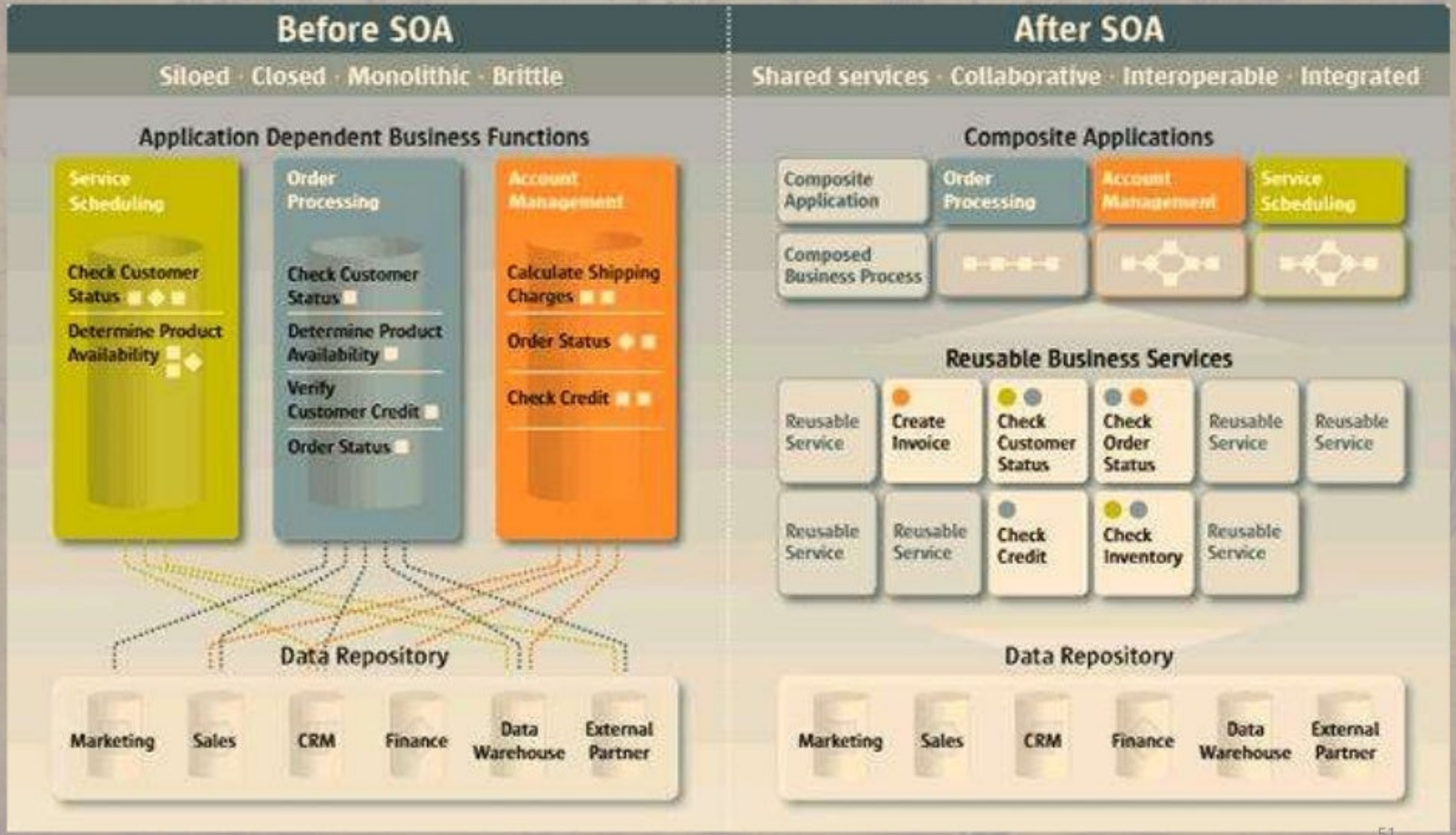
THE SOA PLATFORM

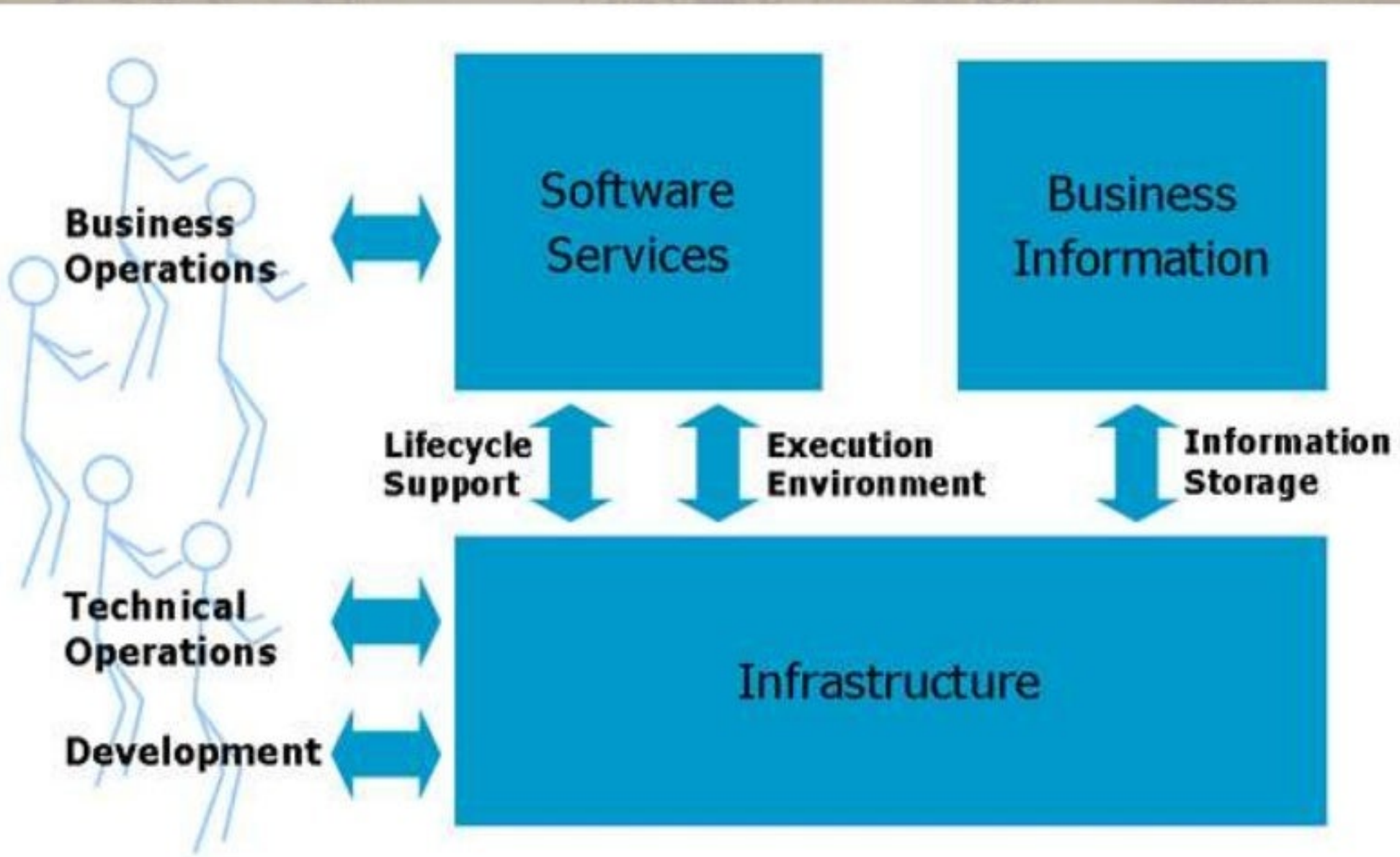
- The optimum implementation architecture for SOA is a component-based architecture.

- Many will be familiar with the concepts of process and entity component, and will understand the inherent stability and flexibility of this component architecture, which provide a one to one mapping between business entities and component implementations.

- Enterprise SOA (ESOA) brings the two main threads
  - Web services
  - CBD (or CBSE)—together

- The result is an enterprise SOA that applies to both Web services made available externally and also to core business component services built or specified for internal use.


Before SOA
Siloed · Closed · Monolithic · Brittle
Application Dependent Business Functions
Service Scheduling
Check Customer Status
Determine Product Availability
Order Processing
Check Customer Status
Determine Product Availability
Verify Customer Credit
Order Status
Account Management
Calculate Shipping Charges
Order Status
Check Credit
Data Repository
Marketing
Sales
CRM
Finance
Data Warehouse
External Partner
After SOA
Shared services · Collaborative · Interoperable · Integrated
Composite Applications
Composite Application
Order Processing
Account Management
Service Scheduling
Composed Business Process
Reusable Business Services
Reusable Service
Create Invoice
Check Customer Status
Check Order Status
Reusable Service
Reusable Service
Reusable Service
Reusable Service
Check Credit
Check Inventory
Reusable Service
Data Repository
Marketing
Sales
CRM
Finance
Data Warehouse
External Partner



Overview of a Service-Oriented Architecture

SOA IMPLEMENTATION STEPS

SOA IMPLEMENTATION STEPS

Step 1: Establish an SOA Vision.

Step 2: Building an SOA Roadmap

- Services needed
- What their interfaces should look like
- Scope of each service
- Granularity of each service

Step 3: Establish an SOA Methodology

For instance, exposing a specific function in an application as a Web service may be possible. But, will it serve the needs of the overall SOA? Your IT organization needs to establish core principles surrounding the SOA then consider potential applications. It may be useful to build a set of best practices over a period of time. These practices will become the core of a proven methodology.

Step 4: Link SOA with Key Business Initiatives

- Efficiency: Business processes can be transformed from isolated silo and replicated processes into highly leveraged, shared services that cost less to maintain.
- Responsiveness: Rapid adaptation and delivery of key business services can meet market demands for increased service levels to customers, employees and partners.
- Adaptability: Changes can be made throughout the business with minimal complexity and effort, saving time and money.

Step 5: Create Architectural Blueprints

Blueprints may include:

- Common security model
- Service orchestration model
- Metadata management
- Process integration model
- Web services compliance model

Step 6: Assess Risks

Key risk of SOA,

- Security
- Interoperability
- Approaches to failure

Step 7: Create an SOA Risk Mitigation Strategy

Step 8: Process-Driven Integration

Patterns for Aligning Business and IT supplies detailed guidance on how to design and build software architectures that follow the principles of business-IT alignment. It illustrates the design process using proven patterns that address complex business/technical scenarios, where integrated concepts of service-oriented architecture (SOA), Business Process Management (BPM), and Event-Driven Architecture (EDA) are required.

WEB SERVICES AND SOA

- *Web Services are the set of protocols by which Services can be published, discovered and used in a technology neutral, standard form.*

- In fact Web services are not a mandatory component of a a SOA, although increasingly they will become so.

- SOA is potentially much wider in its scope than simply defining service implementation, addressing the quality of of the service from the perspective of the provider and the

- Web services are purely the implementation. SOA is the approach, not just the service equivalent of a UML component packaging diagram.

- *SOA is not just an architecture of services seen from a perspective, but the policies, practices, and frameworks by we ensure the right services are provided and consumed.*

- It is important that if a service is to be used by multiple consumers, the specification needs to be generalized, the service needs to be abstracted from the implementation.

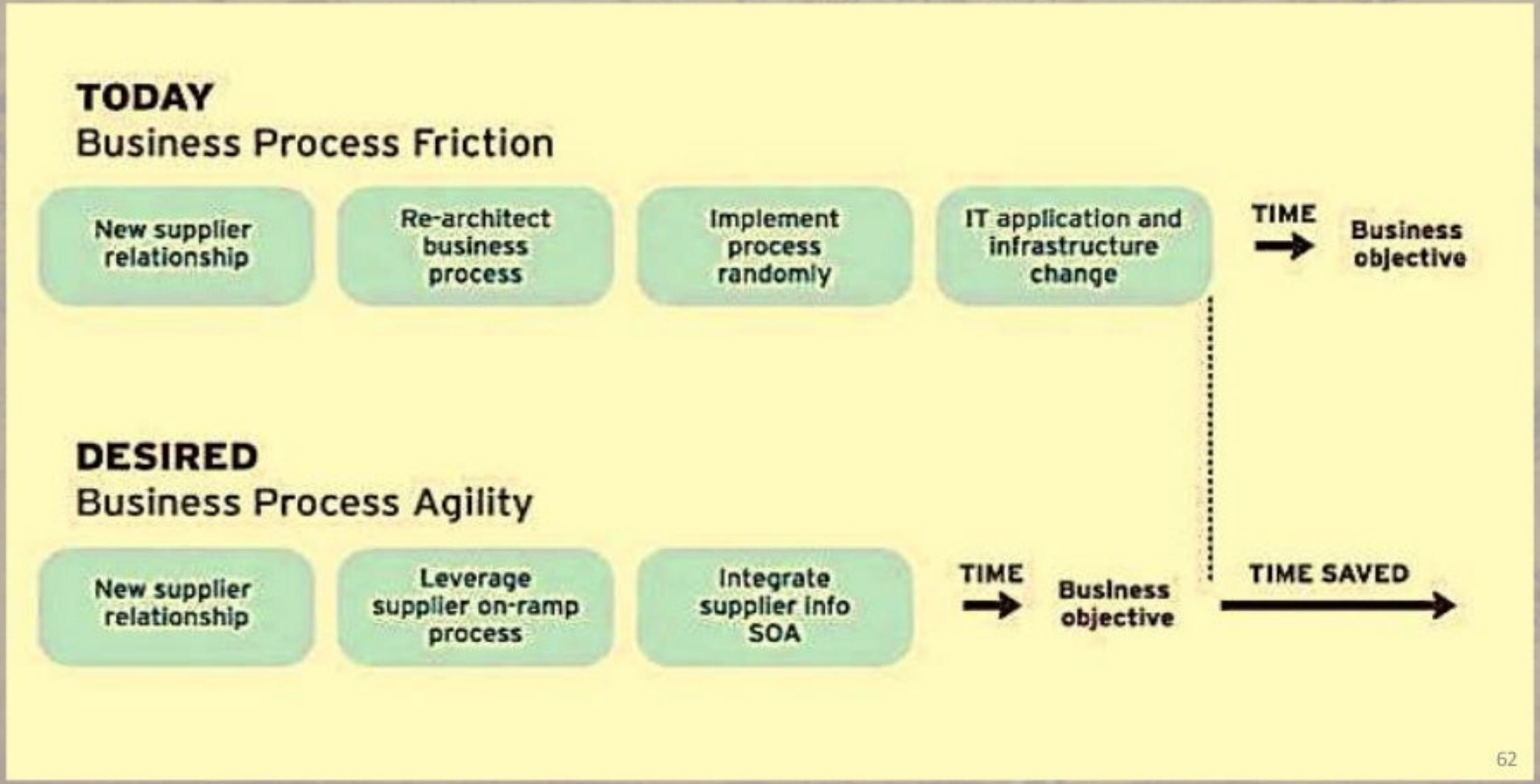
ENABLED BY WEB	**Technology neutral**	**Endpoint platform independence.**
SERVICES	Standardized	Standards-based protocols.
	Consumable	Enabling automated discovery and usage.
	Reusable	Use of Service, not reuse by copying of code/implementation.
	Abstracted	Service is abstracted from the implementation.
ENABLED BY SOA	Published	Precise, published specification functionality of service interface, not implementation.
	Formal	Formal contract between endpoints places obligations on provider and consumer.
	Relevant	Functionality presented at a granularity recognized by the user as a meaningful service.

VALUE PROPOSITION - BUSINESS BENEFITS

VALUE PROPOSITION - BUSINESS BENEFITS

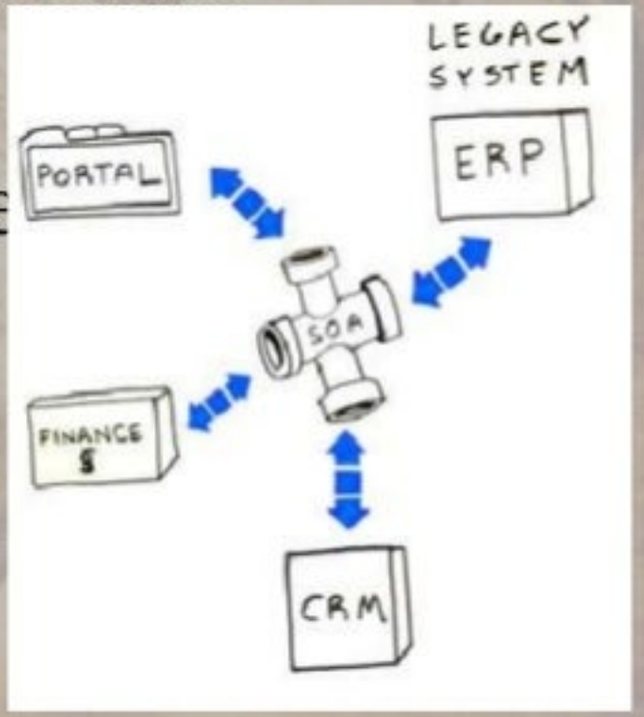
- Agile business and technology environment
- Reduced business, IT, and maintenance costs
- Manageable project size
- Technology autonomy and platform-agnostic systems
- Facilitates automated business process
- Evolutionary approach

VALUE PROPOSITION - BUSINESS BENEFITS



VALUE PROPOSITION - TECHNOLOGY BENEFITS

- Efficient development process
- Reduced number of interfaces and development time
- Platform-agnostic applications
- Design time discovery, introspection and usage
  services
- Standards-based architectures
- Several enablers - Portals, BPM, BI, ERP



SOA –BENEFITS PERSPECTIVE

CEO Perspective

- Budget Strategy

  Agile & Reactive IT environments.
- Short Term Planning

  SOA enables step-by-step approaches
- Budget Reduction
- Technology & Vendor Agnostic

CIO Perspective

- Independence from technology
- Positive Role of IT Department

  IT – Business bridging
- Cost Reduction
- Increase of Influence
- Manageable Project Size

IT Architect Perspective

- Disentanglement
- Loose Coupling
- Code Reuse

Project Manager Perspective

- Smaller & Shorter Projects
- Technology Independence
- Parallel Development
- Reduced Project Risk
- Easier Testing & Integration

Developer Perspective

- Reduction of Dependency
- Rapid Prototyping
- Better Define Requirements
- Simplified Testing

Business Department Perspective

- Independence from Technology
- Shorter Time to Market
- Reduction of Development Costs

ADVANTAGES OF SOA

- Service Reusability
- Easy Maintainability
- Greater Reliability
- Location Independence
- Improved Scalability & Availability
- Improved Software Quality
- Platform Independence
- Increased Productivity

DISADVANTAGES OF SOA

- Increased Overhead
- Complex Service Management
- High Investment Cost

SOA IS NOT RECOMMENDED FOR THE FOLLOWING TYPE OF APPLICATIONS.

1) *Homogenous:*

- Implementing SOA for applications that use the technologies of a single vendor will not be cost-effective.

e.g. ***- if an application is built in Java, then it would be better to use methods of Java rather than using component communications.***

2) *GUI-Based:*

- SOA would not be suitable for applications with GUI functionality,

e.g. **a map manipulation application.**

- Such applications require heavy data exchange, which in turn would increase the complexity of the application if SOA is used.

Cont…

Real-time:

- SOA is not desirable to be used with strictly-enforced response times times since the services communicate asynchronously.

Stand-alone:

- It would be futile to invest in SOA for stand-alone non-distributed distributed applications, which do not require request and response-response-based calls.

SUMMARY

- The goal for a SOA is a **world wide mesh of collaborating services**, which are published and available for invocation on the Bus.

- Adopting SOA is essential to deliver the **business agility** and **IT flexibility** promised by Web Services.

- These benefits are delivered not by just viewing service architecture from a technology perspective and the adoption of Web Service protocols, but **require the creation of a Service Oriented Environment that is based on the key principals**.

- Service is the important concept. Web Services are the set of protocols by which Services can be published, discovered and used in a technology neutral, standard form.

- SOA is not just an architecture of services seen from a technology perspective, but the **policies, practices, and frameworks by which ensure the *right* services are provided and consumed**.

- With SOA it is critical to implement processes that ensure that there are at least two different and separate processes — for **provider** and **consumer**.

- Rather than leaving developers to discover individual services and put them into context, the Business Service Bus is instead their starting point that guides them to a coherent set that has been assembled for their domain.

THANK YOU !