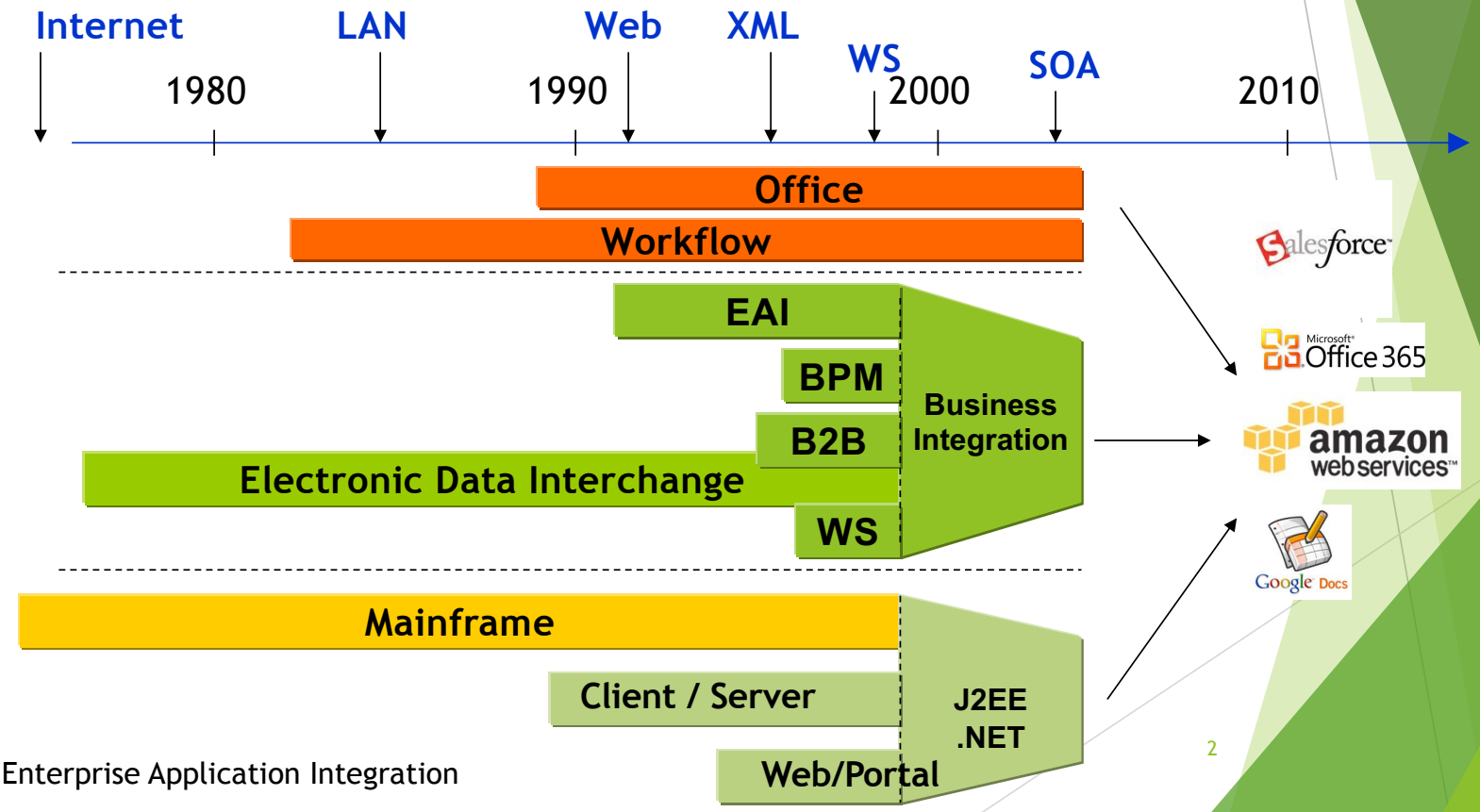
Service Oriented Architecture

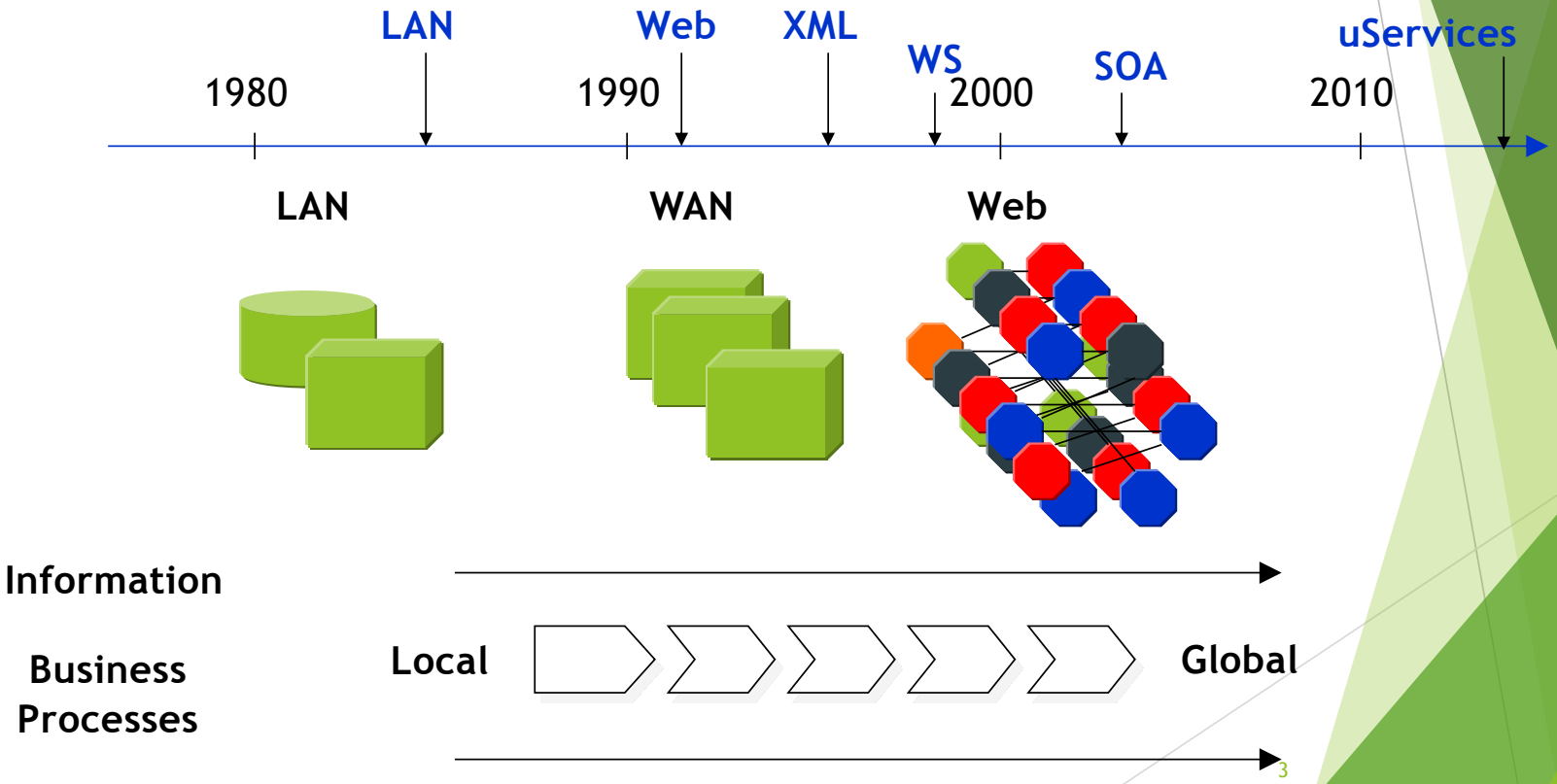
Diogo Gomes dgomes@ua.pt

Connectivity Drives the Emergence and Convergence of Technologies



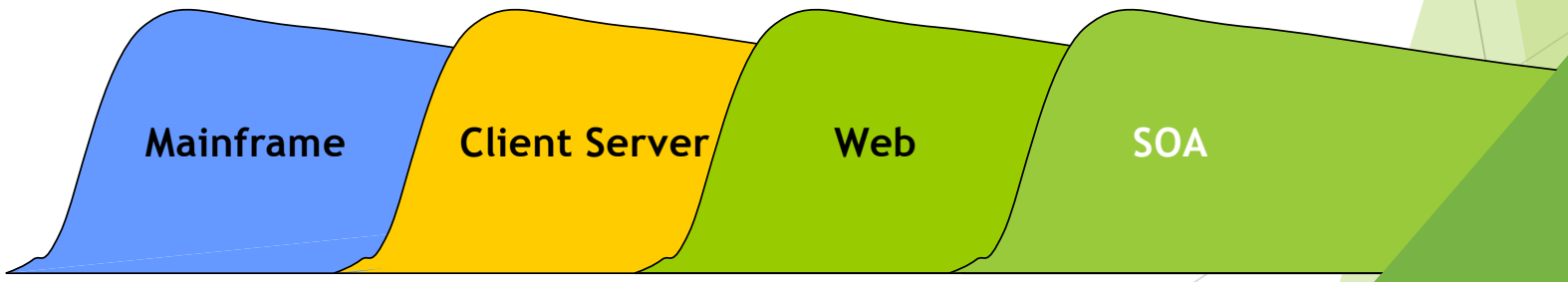
EAI- Enterprise Application Integration

Connectivity Enables Global Processes and Information Access



So…

- Today, the value is not defined as much by functionality anymore but by connectivity
  - However, we need a new programming model
- Why SOA today?
  - We are reaching a new threshold of connectivity and interrelation of distributed (knowledge) resources



SOA @ Amazon

- Jeff Bezos issued a mandate, sometime back around 2002 (give or take a year):
  - All teams will henceforth expose their data and functionality through service interfaces.
  - Teams must communicate with each other through these interfaces.
  - There will be no other form of inter-process communication allowed: no direct linking, no direct reads of another team's data store, no shared-memory model, no back-doors whatsoever. The only communication allowed is via service interface calls over the network.
  - It doesn't matter what technology they use.
  - All service interfaces, without exception, must be designed from the ground up to be externalizable. That is to say, the team must plan and design to be able to expose the interface to developers in the outside world. No exceptions.
- The mandate closed with:
  - "Anyone who doesn't do this will be fired. Thank you; have a nice day!"
- https://medium.com/slingr/what-year-did-bezos-issue-the-api-mandate-at-amazon-57f546994ca2

More than a **concept: a view**

- Service-orientation presents an ideal vision of a world in which resources are cleanly partitioned and consistently represented and used.
- SOA is a form of technology architecture that adheres to the principles of service-orientation.

- SOA and service-orientation are implementation-agnostic paradigms that can be realized with any suitable technology platform.

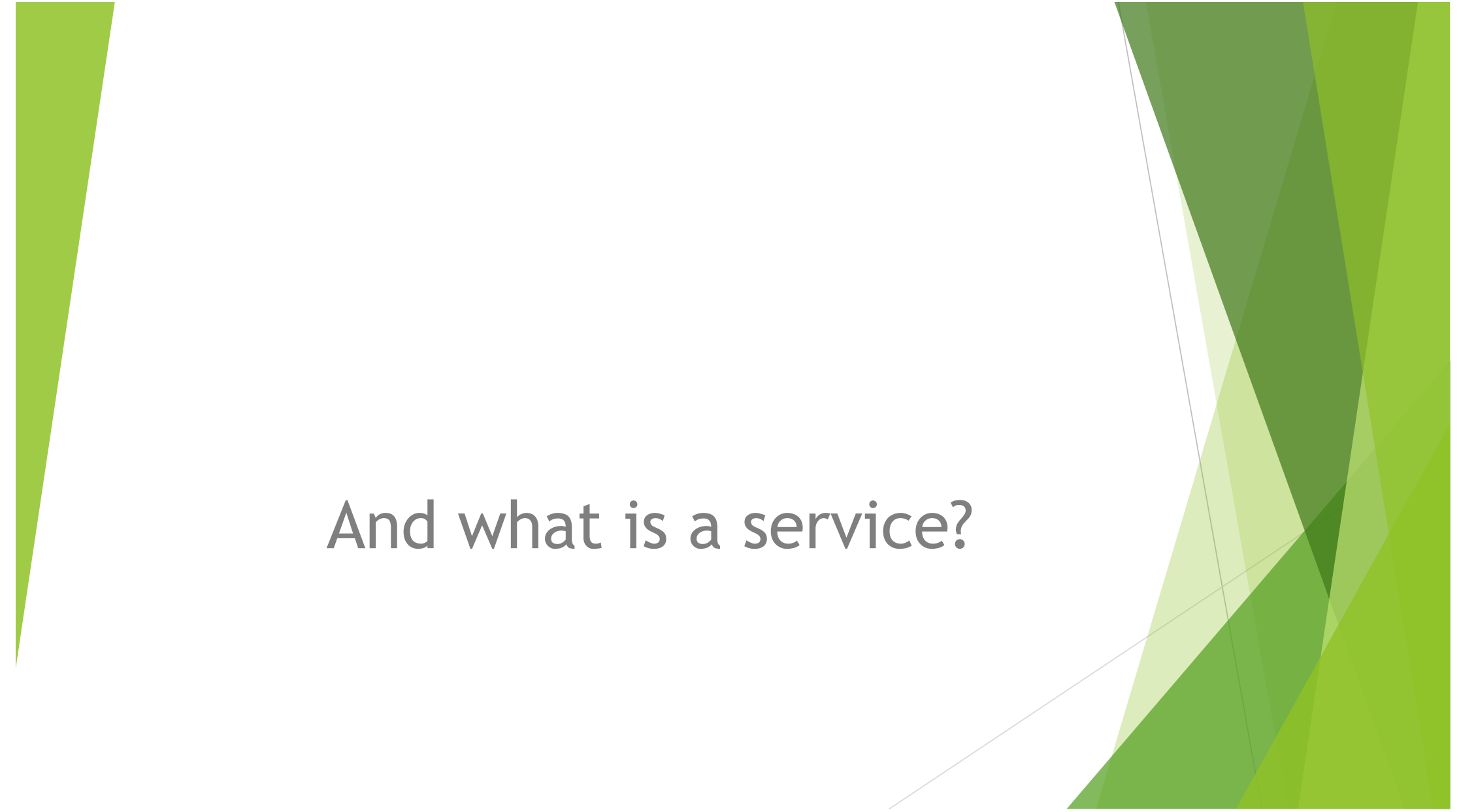
Definitions

- **Service-orientation** is a design paradigm intended for the creation of solution logic units that are individually shaped so that they can be collectively and repeatedly utilized in support of the realization of the specific strategic goals.
- **Enterprise Computing** is the combination of separate applications, services and processes into a unified system that is greater than the sum of its parts.

Service Orientation is a New Paradigm

- Not as a new name for
  - API
  - Component
- A genuine set of concepts with which we can construct new kinds of software in telecommunications
- Forces us to think about enabling the same piece of code to be leveraged
  - by large numbers of consumers
  - in unforeseen context
  - under controlled usage

And what is a service?



What is a service?

A **service** is a:

- set of related (software/business process) functionality, together with the policies that should control its usage.

- a mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is consistently imposed with constraints and policies specified by some description.

It is the application of service-orientation design principles (that is: consistent usage and policy control) that distinguish a unit of logic as a service compared to units of logic that may exist only as objects or components.

Policies

- Who can access?
- During what time/type of usage?
- To what type of feature/information?
- With what type of responsiveness?
- With what type of charge?

Policies inherently support **business models**!

Telecommunication operators favour controlled or walled garden business models.

Internet-style favour open *(?)* business models.

"open" is here a questionable concept

Federation

- **Federation** is the establishment of a well-defined trust relationship, which defines what is trusted, in what conditions, through which communication methods.
  - These relationships are usually potentially passive of liability issues.
- **Dynamic federation** is the real-time establishment of such trust relationships on a need-to basis.
- Most telecommunication systems currently rely on **static federation**, an off-line process for the establishment of such trust relationships.
  - An example: billing with pre-paid and post-paid in roaming, where you need real-time (or not) information from the home network.
    - **Note**: in reality, in telcos both cases are established with static federation. Only the user information is exchanged dynamically, trust relations are static.

Moving from Components to Services

- Requires a client library
- Client / Server
- Extendable
- Stateless
- Fast
- Small to medium granularity

- Loose coupling via
  - Message exchanges
  - Policies
- *Peer-to-peer (mostly)*
- Composable
- Context independent
- Some overhead
- Medium to coarse granularity

Service Interfaces

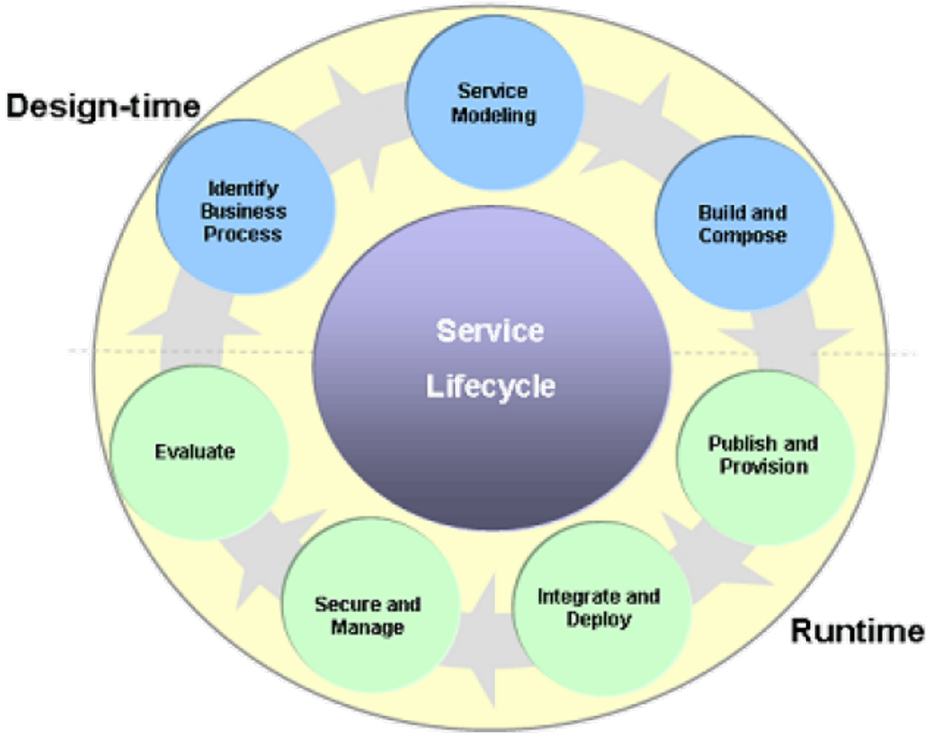
- *Non proprietary*
  - All service providers offer somewhat the same interface
- Highly Polymorphic
  - Intent is enough
  - Multiple usages
- Implementation can be changed in ways that do not break all the service consumers
  - Real world services interact with thousands of consumers
  - Service providers cannot afford to “break” the context of their consumers

Intents and Offers

- Service consumer expresses “intent”
- Service providers define “offers”

- Sometimes a mediator will:
  - Find the best offer matching an intent
  - Advertise an offer in different ways such that it matches different intent
- Request / Response is just a very particular case of an Intent / Offer protocol

Service development life-cycle



Services and SOA

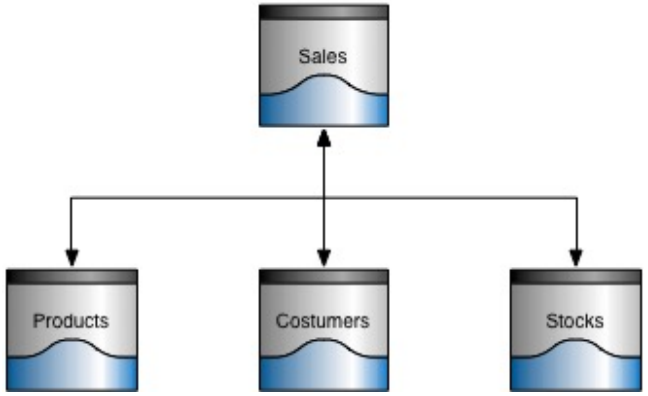
- Within SOA projects, a service is commonly conceptualized during the service-oriented **analysis** phase, at which time it is referred to as a service candidate.
- Subsequent service-oriented design and development stages **implement** a service as a physically independent software program with specific design characteristics that support the attainment of the strategic goals associated with service-oriented computing.
  - Each service is assigned its own distinct functional context and is comprised of a set of capabilities related to this context. Therefore, a service can be considered a container of capabilities associated with a common purpose (or functional context).

SOA

- **Service-Oriented Architecture** represents an architectural model that aims to enhance the agility and cost-effectiveness of an enterprise while reducing the burden of IT on the overall organization.
  - It accomplishes this by positioning services as the primary means through which solution logic is represented.
  - Services need to be discoverable and announced.
- SOA supports service-orientation in the realizations of the strategic goals associated with service-oriented computing.

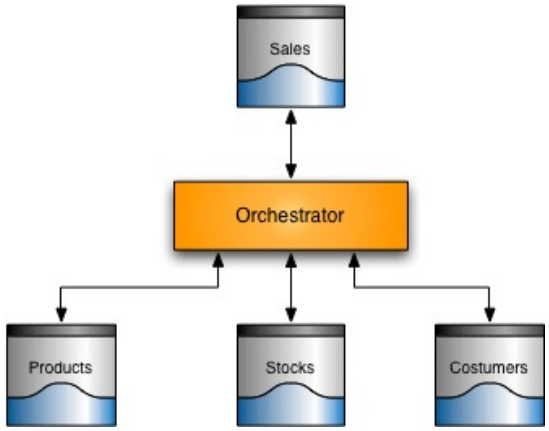
Service composition

- **Service Composition** is an aggregate of services collectively composed to automate a particular task or business process.
  - To qualify as a composition, at least two participating services plus a composition initiator need to be present. Otherwise, the service interaction only represents a point-to-point exchange



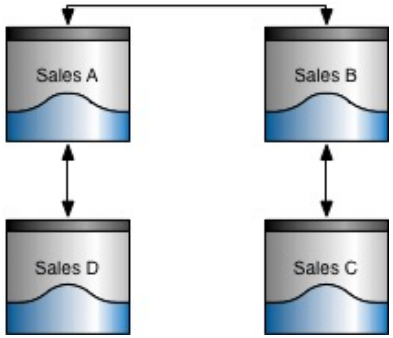
Orchestration

- An Orchestration establishes a business protocol that formally defines a business process.
  It can centralize and control inter application logic through a standardized service model.
  - Workflow logic is broken down into a series of basic and structured activities that can be organized into sequences and flows.



Choreography

- **Choreography** is complex activity comprised of a service composition and a series of Message Sequence Patterns. They involve multiple participants that can assume different roles and that have different relationships.



So …

- Orchestration
  - is concept that formally describe processes underlying business applications so that they can be exposed and linked to processes in other applications
- Choreography
  - Is a concept that specifies how these processes are linked together across the enterprise
  - Choreography can be « active » when mapping and routing are necessary
- They are both components for effective Coordination of multiple services

Individual principles of Service-Orientation

- **Loose coupling** – Services maintain a relationship that minimizes dependencies and only requires that they retain an awareness of each other
- **Service contract** – Services adhere to a communications agreement, as defines collectively by one or more service descriptions and related documents
- **Autonomy** – Services have control over the logic they encapsulate
- **Abstraction** – Beyond what is described in the service contract, services hide logic from the outside world.

Individual principles of Service-Orientation (2)

- **Reusability** – Logic is divided into services with the intention of promoting reuse
- **Composability** – Collections of services can be coordinated and assembled to form composite services
- **Statelessness** – Services minimize retaining information specific to an activity.
- **Discoverability** – Services are designed to be outwardly descriptive so that they can be found and assessed via available discovery mechanisms.

Primary characteristics

- At the core of service-oriented computing platforms
- Increases quality of service
- Fundamentally autonomous
- Based on Open Standards

  In telecom, “open” may be only for a trusted group
- Supports vendor diversity

Primary characteristics (2)

- Fosters intrinsic interoperability, reusability and extensibility
- Promotes discovery, federation and architectural composability
- Promotes loose coupling and organizational agility throughout the Telcos and Enterprise
- Implements layers of abstraction
- Is a building block
- Is an evolution

Service Orientation is a New Paradigm

- Not as a new name for
  - API
  - Component
- A genuine set of concepts with which we can construct new kinds of software in telecommunications
- Forces us to think about enabling the same piece of code to be leveraged
  - by large numbers of consumers
  - in unforeseen context
  - under controlled usage

SOA – key points

- SOA and Service-Orientation are implementation-agnostic paradigms that can be realized with any suitable technology platform.
- SOA is about leveraging existing Enterprise and Telecommunications Applications by combining them through new business processes.
- It’s an architectural evolution rather than a revolution