What is Services?

- Service is
  - component of distinctive functional meaning that typically encapsulate a high-level business concept
  - Lego block
- Service contains
  - Contract – message type def, constraint, description (comment)
  - Interface – set of operations
  - Implementation – Logic and data

Examples of a Service

- Creating a Purchase Order inside a mainframe application
- Requesting and reserving a room in a hotel
- Applying for a loan by filling out a loan request form
- Search books/music based on keywords

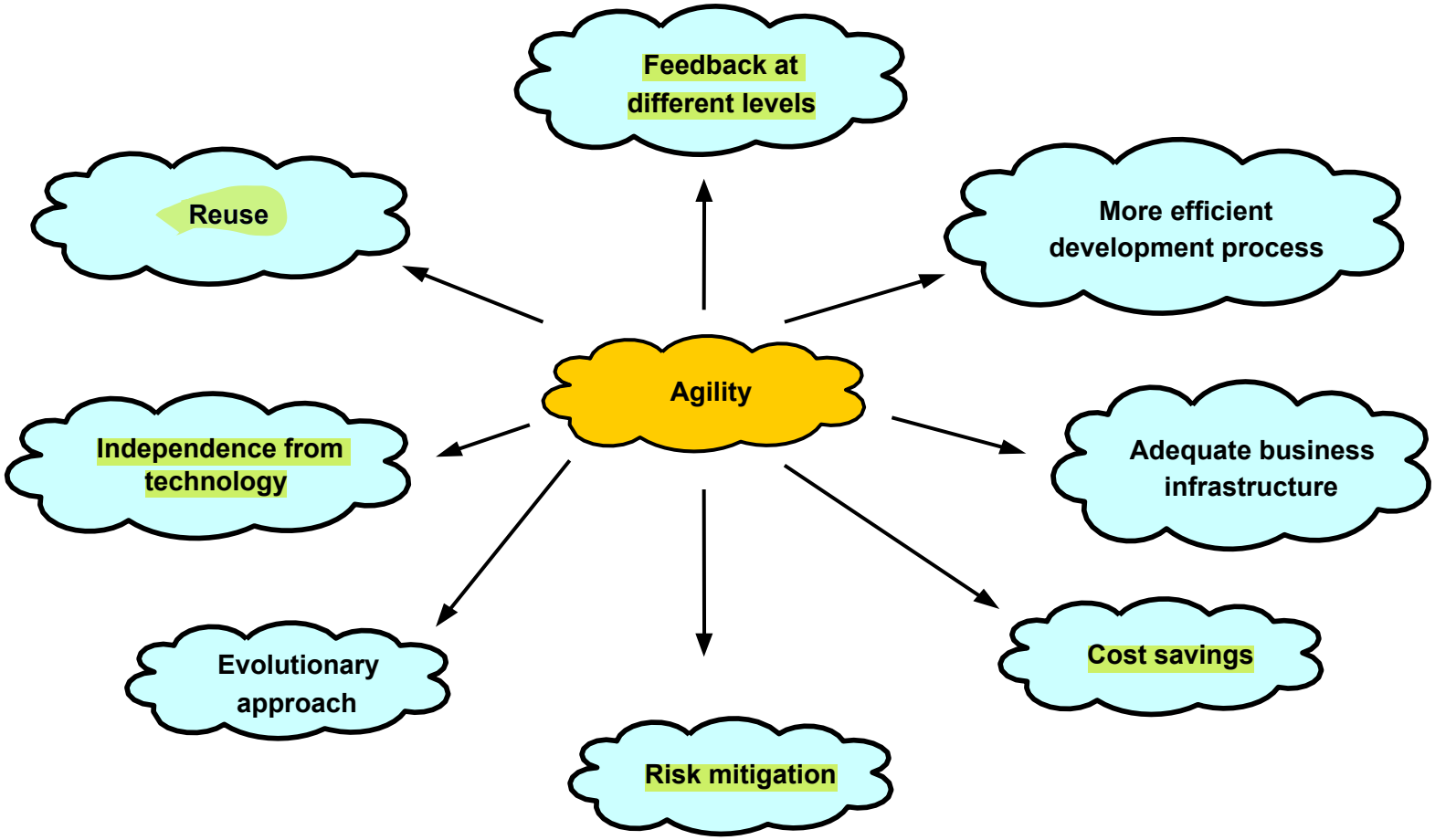
What is SOA?

- A set of *components* which can be invoked, and whose *interface* description can be published and discovered (W3C).
- Service-oriented architecture is a *client/server* design approach in which an application consists of software services and software service consumers (also known as clients or service requesters). SOA differs from the more general client/server model in its definitive emphasis on *loose coupling* between software *components*, and in its use of separately standing *interfaces* (Gartner).

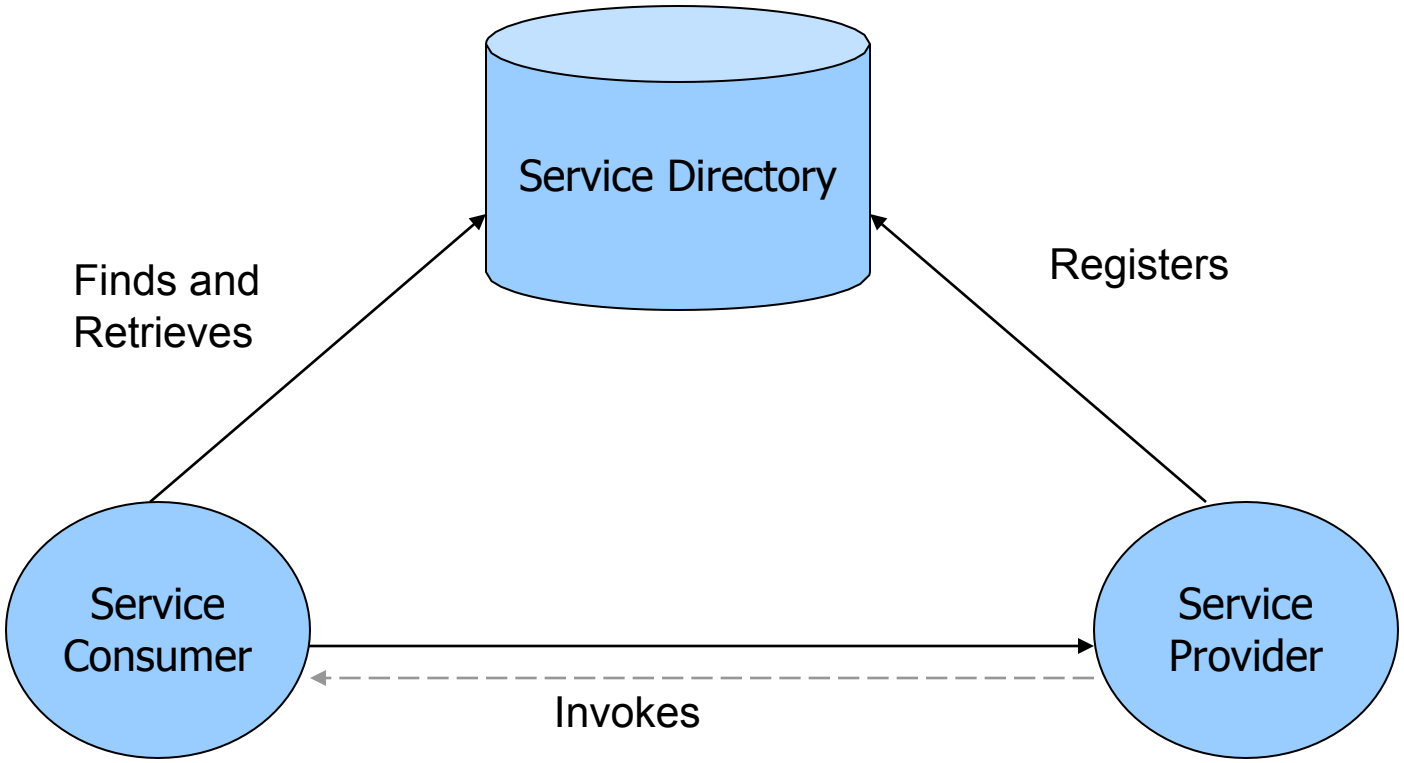
What is SOA?

- Service-Oriented Architecture is a business-driven IT architecture approach that supports integrating your business as linked, repeatable business tasks, or services. SOA helps today's business innovate by ensuring that IT systems can adapt quickly, easily and economically to support rapidly changing business needs. SOA helps customers increase the flexibility of their business processes, strengthen their underlying IT infrastructure and reuse their existing IT investments by creating connections among disparate applications and information sources. (IBM)

Potential Benefits of SOA



SOA architecture

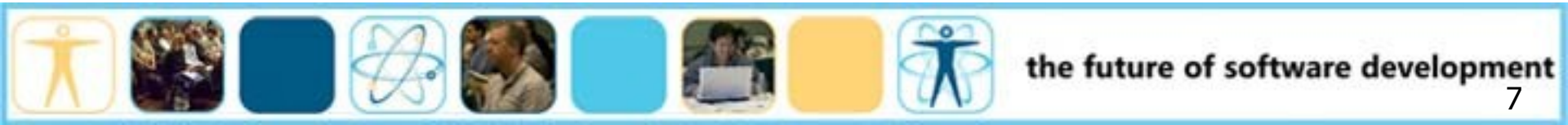


Traditional Architecture Vs Service Oriented Architecture

Traditional Architecture	Service Oriented Architecture
ARCHITECTURE	ARCHITECTURE
• Components are tightly coupled • Interface between subsystems is explicitly defined in terms a stack of protocols	• Loose coupling by means of services with standardized interfaces • Application components communicate only through services and can be plugged in to any infrastructure that implements the standardized service
• Known implementation • Components are not independent of implementation attributes	• Uses abstraction and is based on XML over SOAP • Largely independent of implementation attributes
• Tends to be closed architecture – Difficult to replace, or reuse components from one system to another	• Loosely coupling between interaction software components – leads to re-use of software components • Designed to follow publically accessible models for consumption

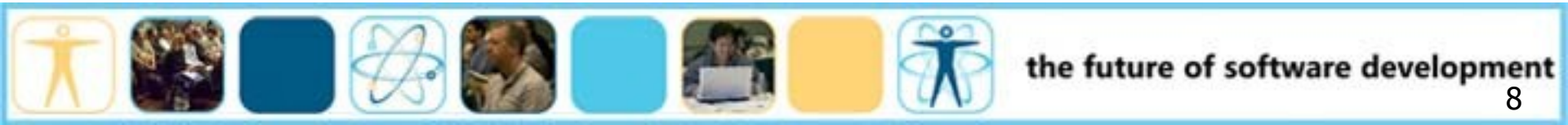
• Commonly, functions are accessible with the help of point-point connections over the network
• Tends to be confined to a single organization
	• Meant for enabling participation of multiple organizations
	• Requires additional layers
	  • Business layer => Service and business model / components
	  • Service Bus / Service Facade
	  • BPM

• Based on standard set of layer – presentation, business, data access, Database	



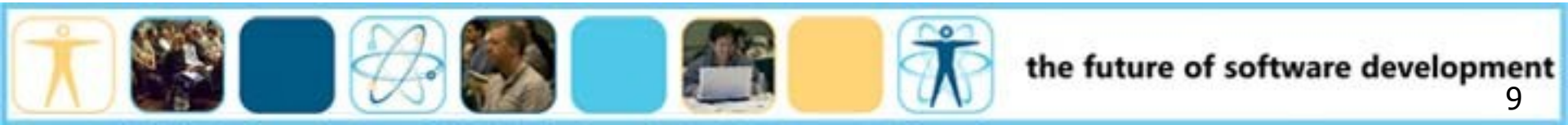
Traditional Architecture Vs Service Oriented Architecture

Traditional Architecture	Service Oriented Architecture
STANDARDS	STANDARDS
• Involves only traditional J2EE and Web related standards • Uses only HTTP • Uses HTTPS for security • More or less stable set of standards	• Includes standards related to Web Service • Builds a messaging layer above HTTP using SOAP • Prefer WS-Security for end-to-end security • Implementations must deal with evolving set of standards
USAGE	USAGE
• Process centric • Known context of usage	• Workflow centric • To a large extent, future context of usage unknown at the time of design i.e unknown users and usage platforms



Challenges of SOA

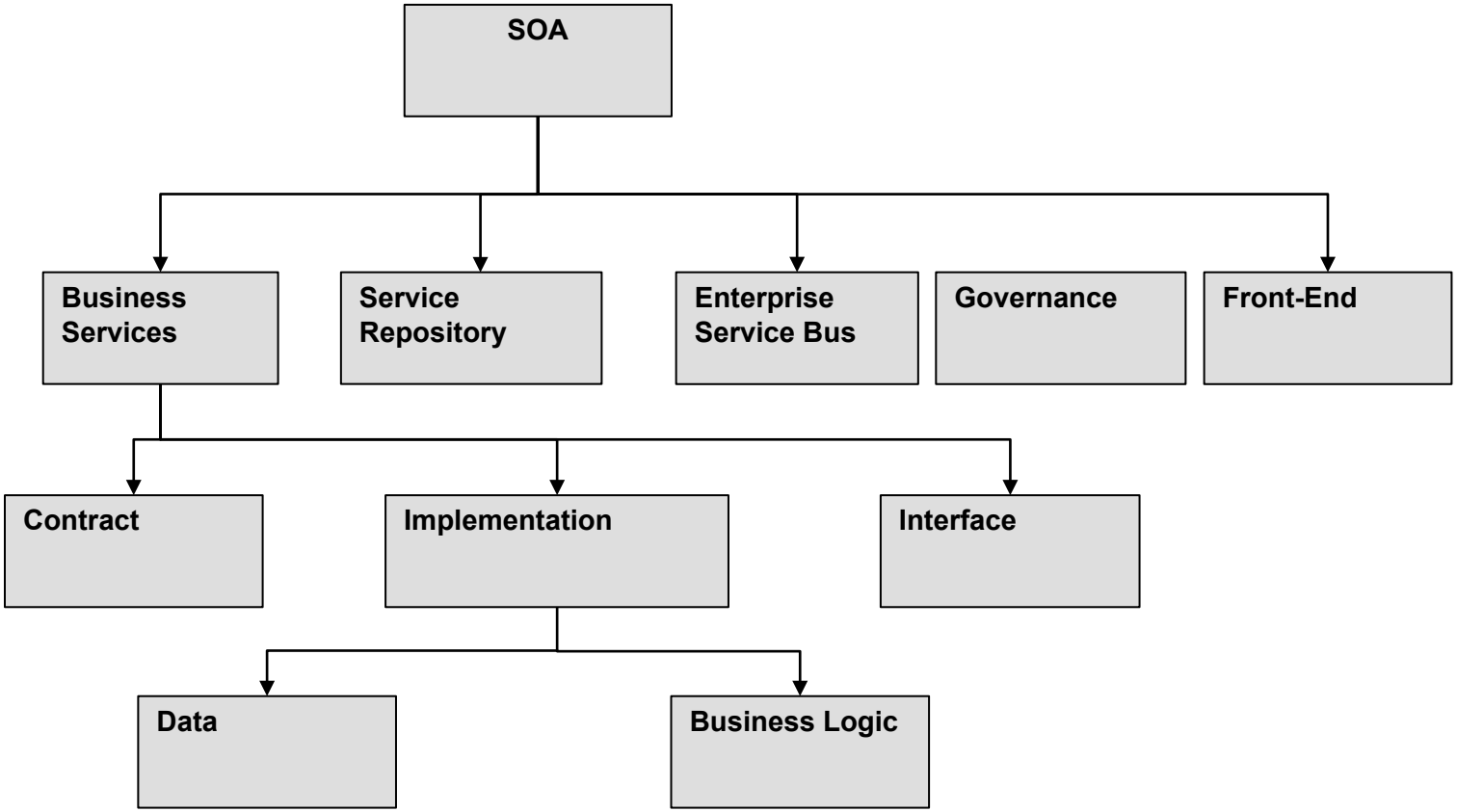
- Technical Challenges
  - Security challenges - loosely coupled environment
  - Performance - XML brings robustness not speed
  - Optimization
  - Organizing the services – registry & repository
  - Finding the right services and right interfaces
  - Transaction management is complex in interactions between logically separate system



Key components of SOA

- Services (common denominator)
- Service Description
- Advertising and Discovery
- Specification of associated data model
- Service contracts

Key components of SOA



Enterprise service bus (ESB): An ESB manages the flow of messages across different applications, orchestrating communications and allowing, for instance, an MDM hub to access application messages and data. It's not for data integration; rather, it's a messaging mechanism.

Service registry: A service registry is critical for an SOA environment in order to track and publish services to applications developers, business partners and exchange members so they know which services exist and how they should be used in the form of service metadata.

Business processes: Without business processes, SOA is just a framework comprised of the above pieces. A business process can be in the form of a Web service (e.g., "Update the customer's address" or "Change the product's name"). The point of SOA is to unify these processes across systems and make them repeatable.

MDM hub: MDM hub provides a means for reconciling common master data across systems and applications, providing an operational single version of the truth for customer, product, location or other reference data. Master data management ensures that the meaning and format of the data being accessed is unified across all the applications that access the hub; the data is understood by all the services that access it.

Data management: If your services are information-rich then data management is a must. Since the data that services access might not necessarily come from an MDM system but from a range of applications and databases, managing, tracking and maintaining that data at the enterprise level is a critical component of SOA. This means having business rules, policies and metadata used and enforced.