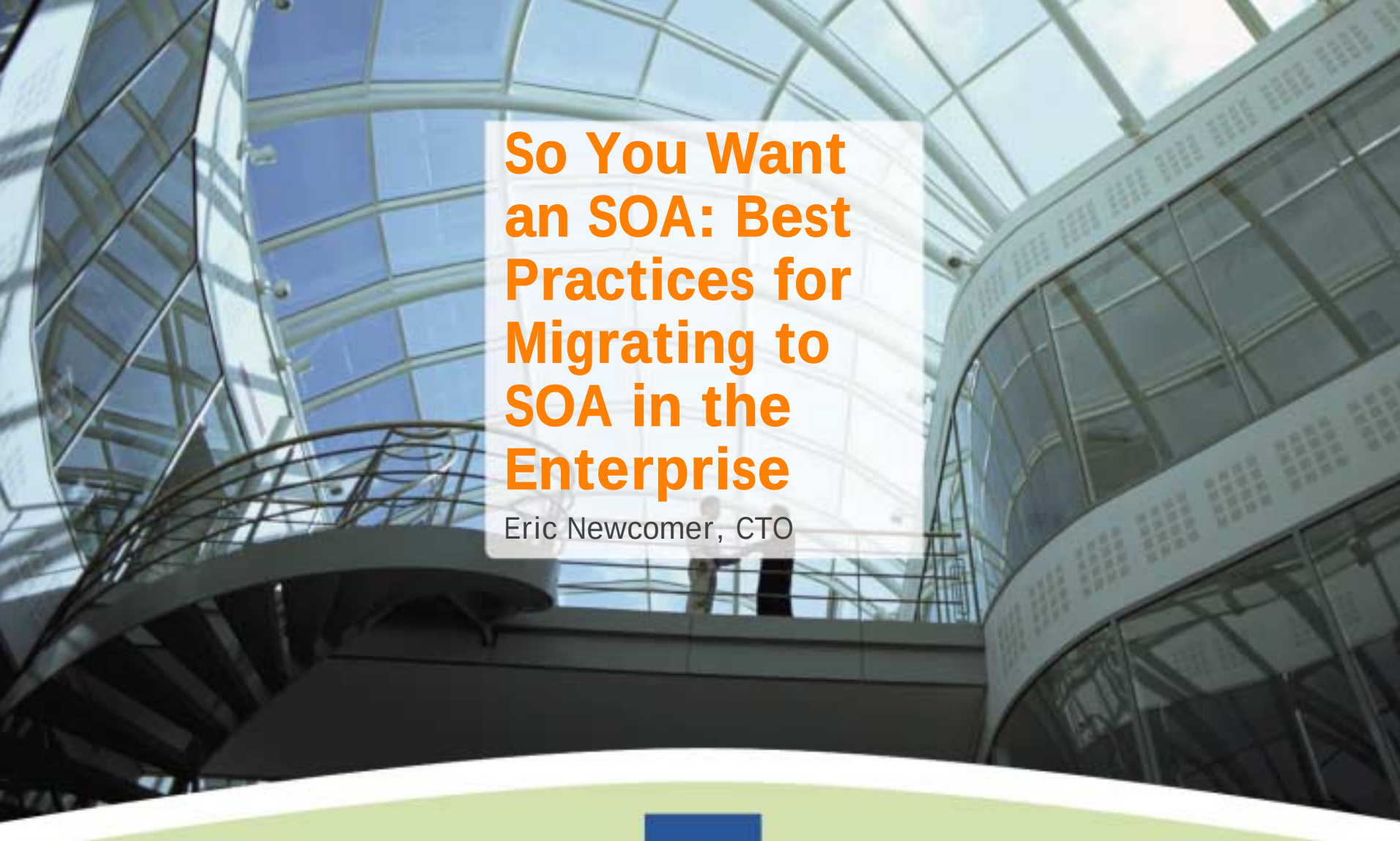


So You Want an SOA: Best Practices for Migrating to SOA in the Enterprise

Eric Newcomer, CTO

IONA

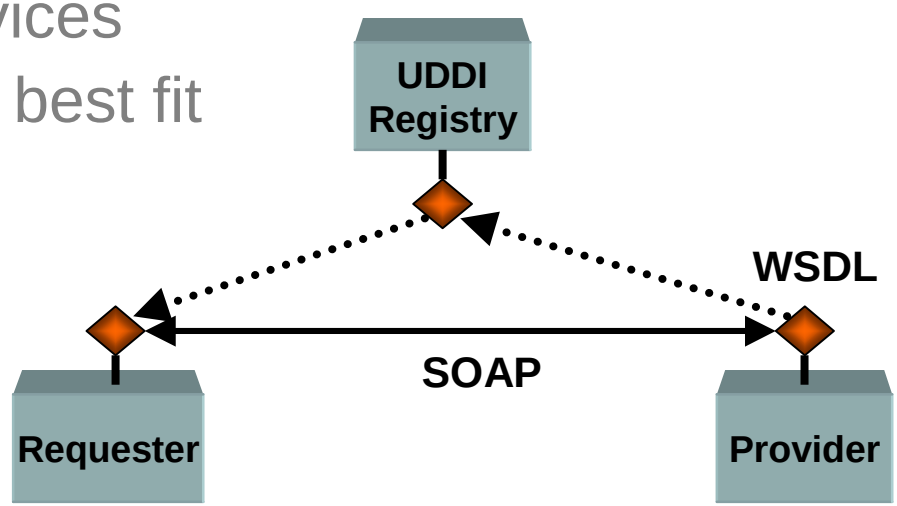
Making Software Work Together™

Overview

- First of all: concepts and definitions
- Change your thinking about your IT environment
  - Including organization and skill set
  - Embrace heterogeneity and complexity
- Support businesses agility to:
  - Reduce complexity
  - Improve integration and data sharing
  - Become more competitive
- Move toward services and SOA adoption:
  - Map IT services to business services
  - Start with incremental change
  - Understand and apply architectural best practices

Concepts and definitions

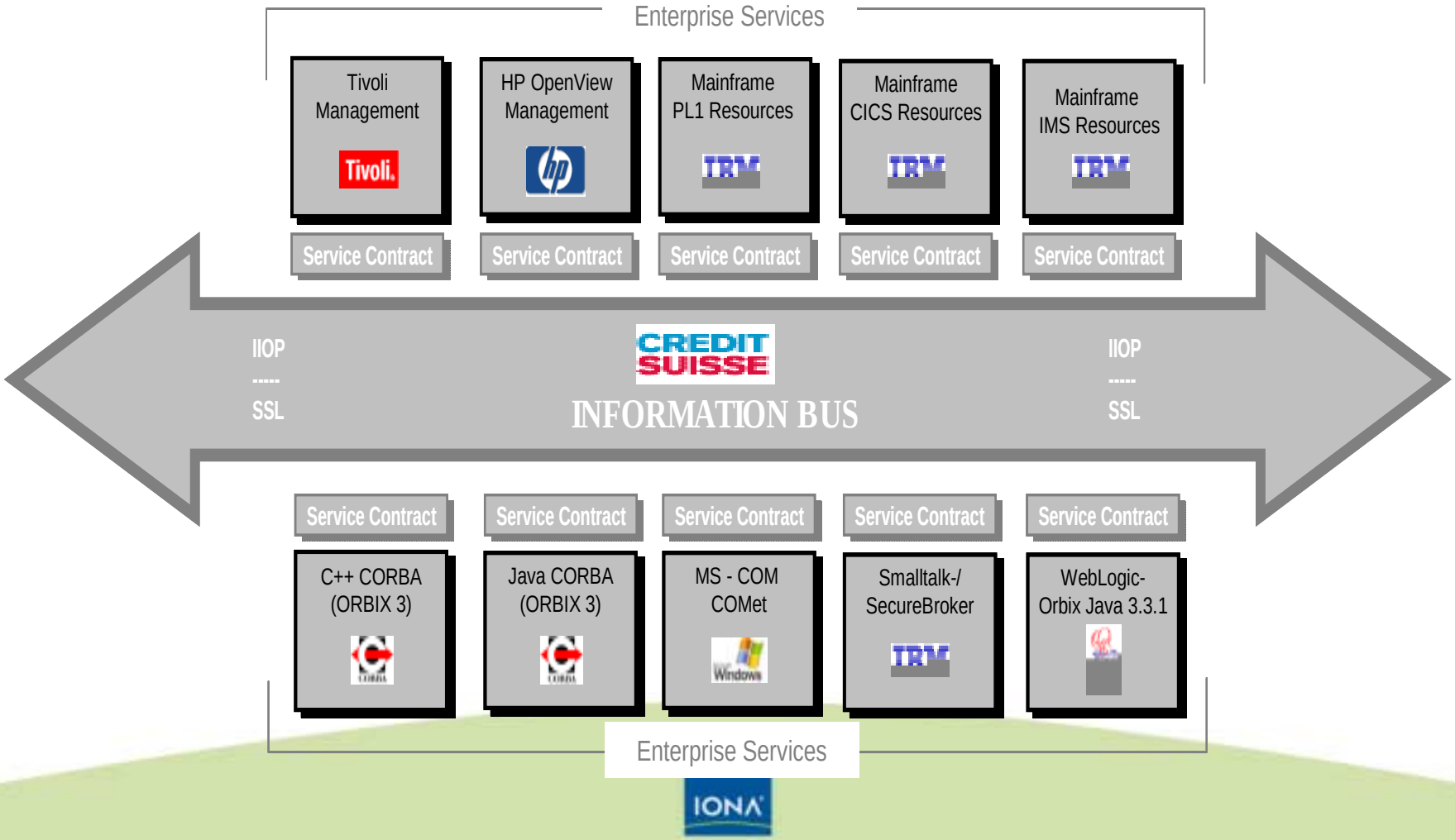
- Services and SOA are technology independent
  - They represent “styles of design”
  - Have been mapped successfully to CORBA, WebSphere MQ, Web services
  - N-tier technologies are the best fit
  - CORBA, Web services
- A service includes:
  - A *requester*
  - A *provider*
  - One or more *message exchange patterns*



Concepts and definitions (cont)

- Services model human and business *functions*
  - Not software artifacts such as databases or files
  - Unlike objects, services are not “things”
- Services are defined by the messages they exchange
  - *Request* list of customers
  - *Update* the customer’s order
  - *Notify* the call center operator
  - *Transfer* my funds
- The design concept is the message *schema*

Credit Suisse: SOA pioneer



Making Software Work Together™

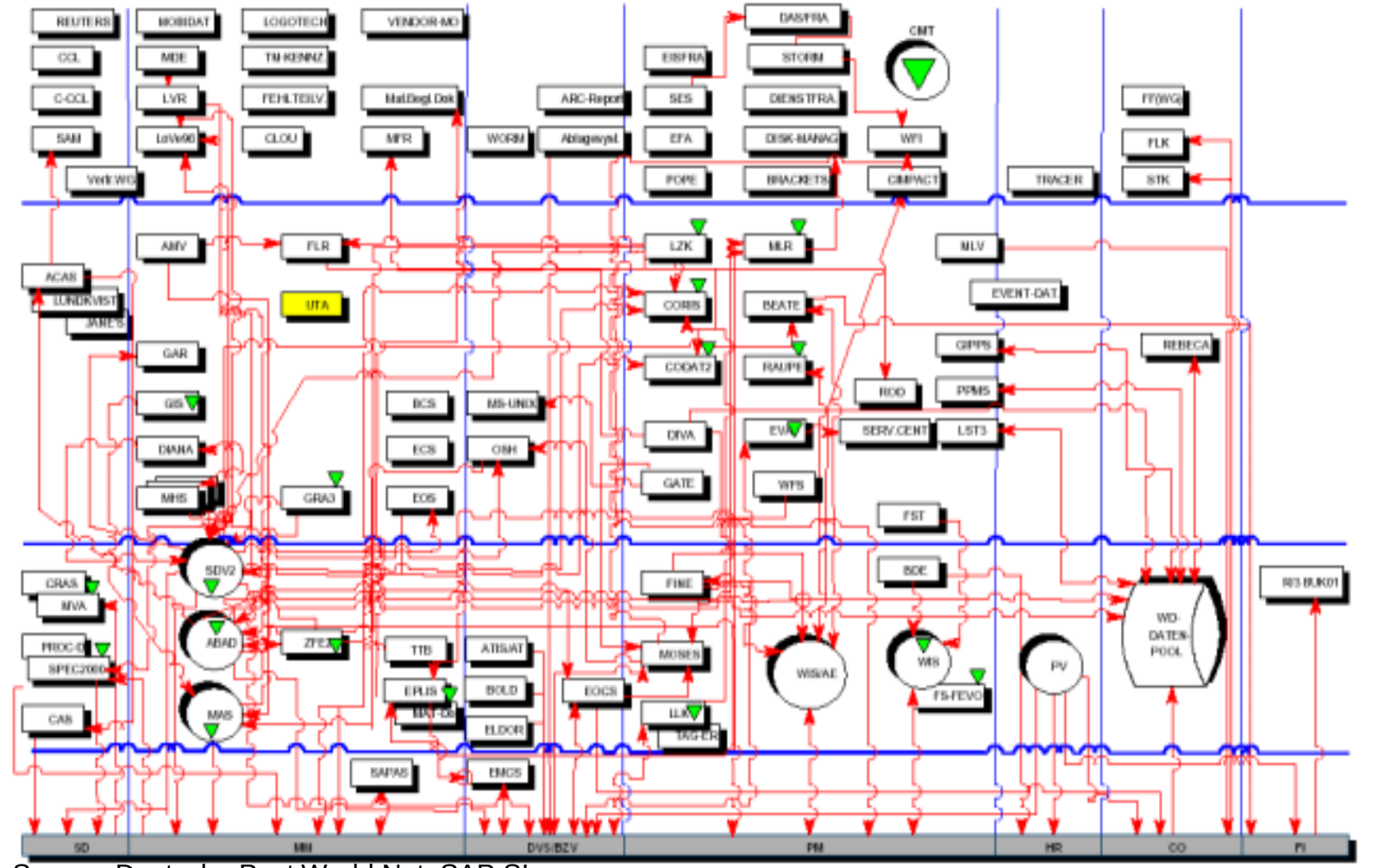
Services change the organization

- A *service architect* determines
  - The overall plan and design for services
  - Which services fit the model, which don't
  - Design principles for services, especially *reuse*
- Developers update their skills
  - Learn to develop for future reuse
  - Work with multiple technologies (Web services plus an execution environment – J2EE, CORBA, .NET)
- Business analysts may *compose* services
  - Given right levels of abstraction and toolset

Services embrace heterogeneity

- Heterogeneity and complexity are constant
  - Services can abstract across any software system
  - Apply “veneer” using XML and Web services
  - Extend for enterprise qualities of service
- Unique characteristics are valuable
  - Make complexity work for you
  - J2EE, .NET, CORBA, and Mainframe all have benefit
- Existing applications have functions with value
  - Modernize without the expense of replacement
  - Maintain enterprise QoS – reliability, transactions, security

Typical IT complexity map



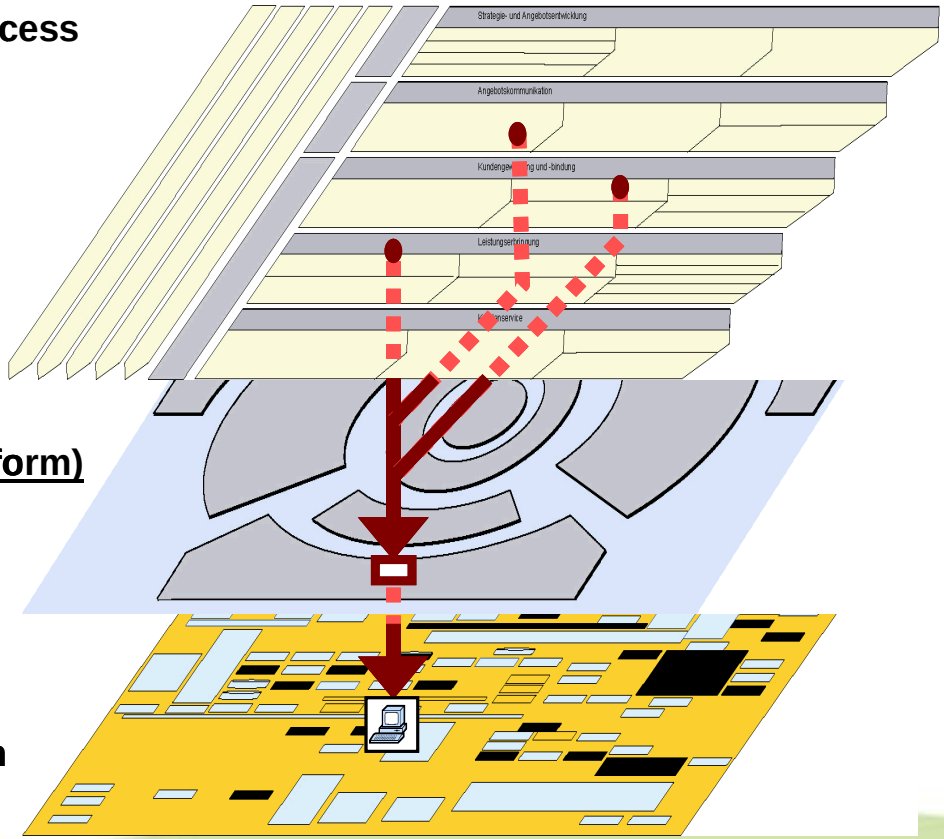
Source: Deutsche Post World Net, SAP SI

IONA

Making Software Work Together™

Desired goal

Business Process Architecture



SOP (Service Oriented Platform)

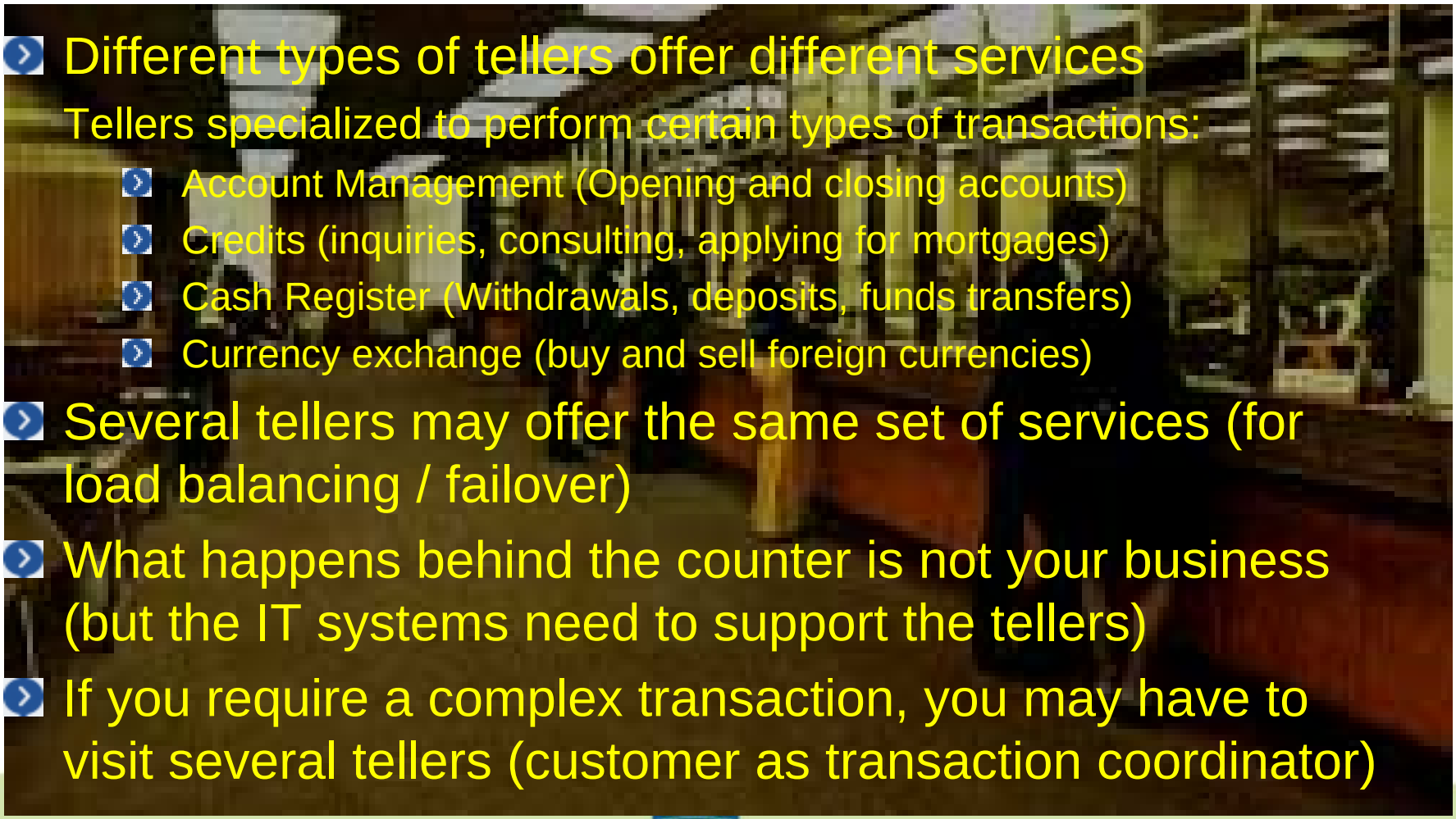
IT Application Landscape

- A “plug and play” view of IT architecture and systems
- Native interoperability and integration capabilities now available for every system end-point via a standards-base service interface
- Orchestration of core business processes, independent of underlying systems and adaptable to changing business conditions
- Investments in standards to help drive down long term operating costs and as a mechanism for insuring future interoperability between infrastructure and systems

IONA

Making Software Work Together™

Bank Teller analogy



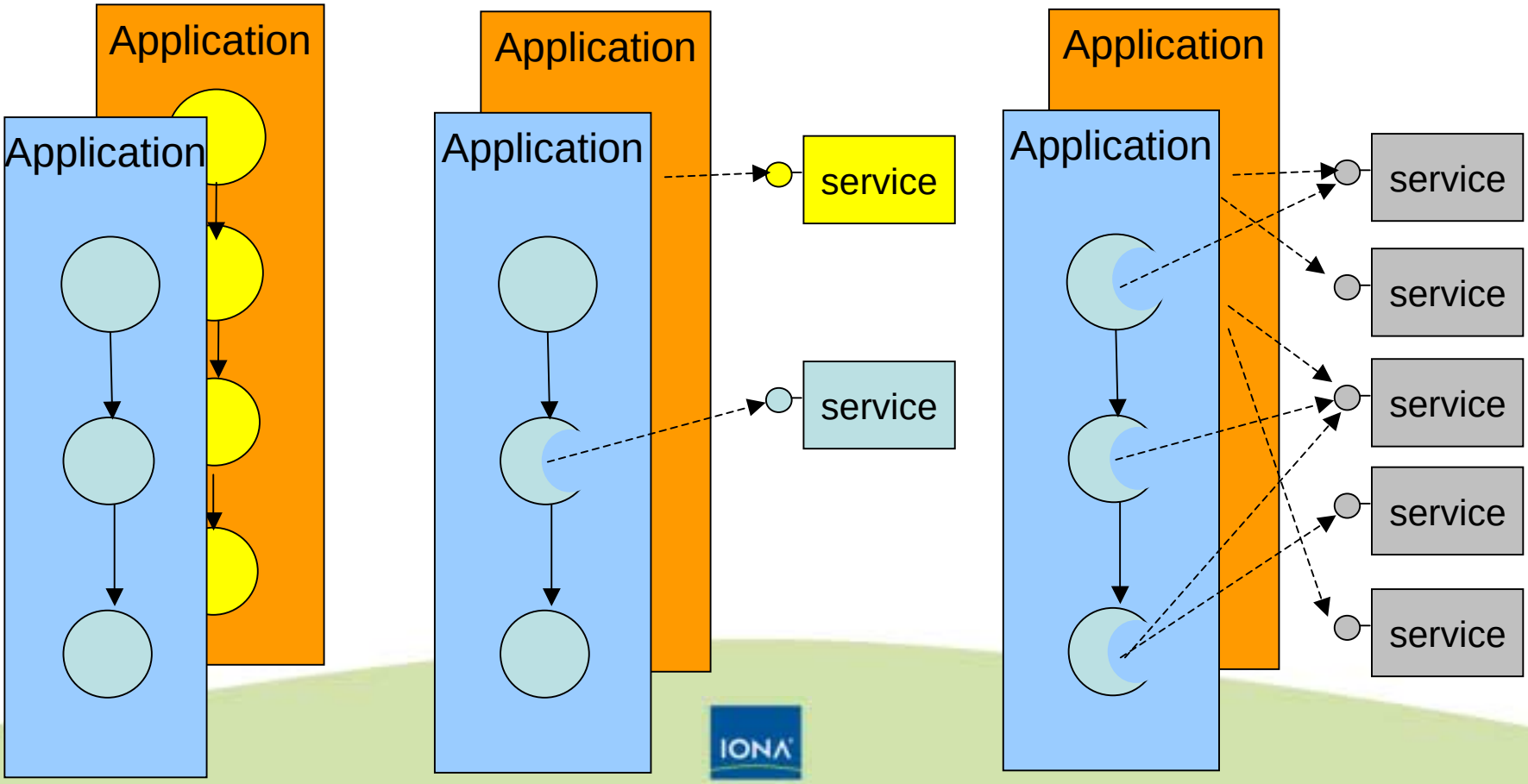
- Different types of tellers offer different services
  - Tellers specialized to perform certain types of transactions:
    - Account Management (Opening and closing accounts)
    - Credits (inquiries, consulting, applying for mortgages)
    - Cash Register (Withdrawals, deposits, funds transfers)
    - Currency exchange (buy and sell foreign currencies)
- Several tellers may offer the same set of services (for load balancing / failover)
- What happens behind the counter is not your business (but the IT systems need to support the tellers)
- If you require a complex transaction, you may have to visit several tellers (customer as transaction coordinator)

Move from monolithic applications in steps

Monolithic applications

Intermediate stage: Break out individual services

Goal: Service-oriented architecture



IONA

Making Software Work Together™

Reasons to develop services

- Not for the sake of the individual applications
  - Services don’t make sense in a mono-technology environment
  - Focus on reusability, component design, data sharing
- For the sake of the next level developers
  - Office workers, business analysts
  - Can share data without worrying about details of underlying implementation
  - Lay the foundation for future applications
  - Infrastructure investment in “agility”

Service Contracts Are the Key

- The key principles of service-oriented architecture (SOA)
  - Services should be business-oriented
  - Services should have well-defined interfaces (aka service contracts)
  - Service contracts should separate interface from implementation
- Service contracts are critical to achieve *reuse* and *abstraction*

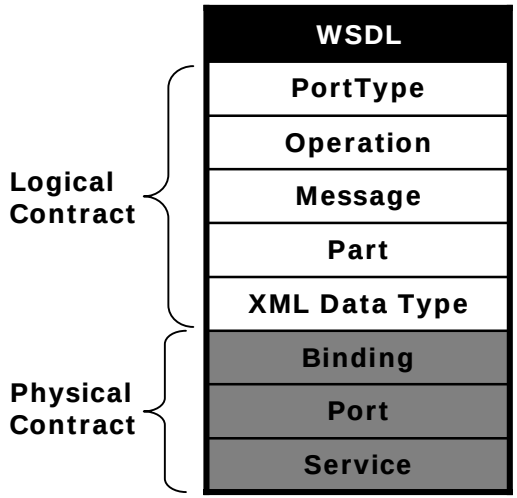
WSDL is best for service contracts

WSDL is very flexible:

- Import existing WSDL contracts
- Create new WSDL contracts using XML Schema
- Create new WSDL contracts from an external metadata source such as CORBA IDL
- Annotate with *policy* metadata

Benefits

- Abstraction
- Encapsulation
- Loose-Coupling
- Separation of Concerns



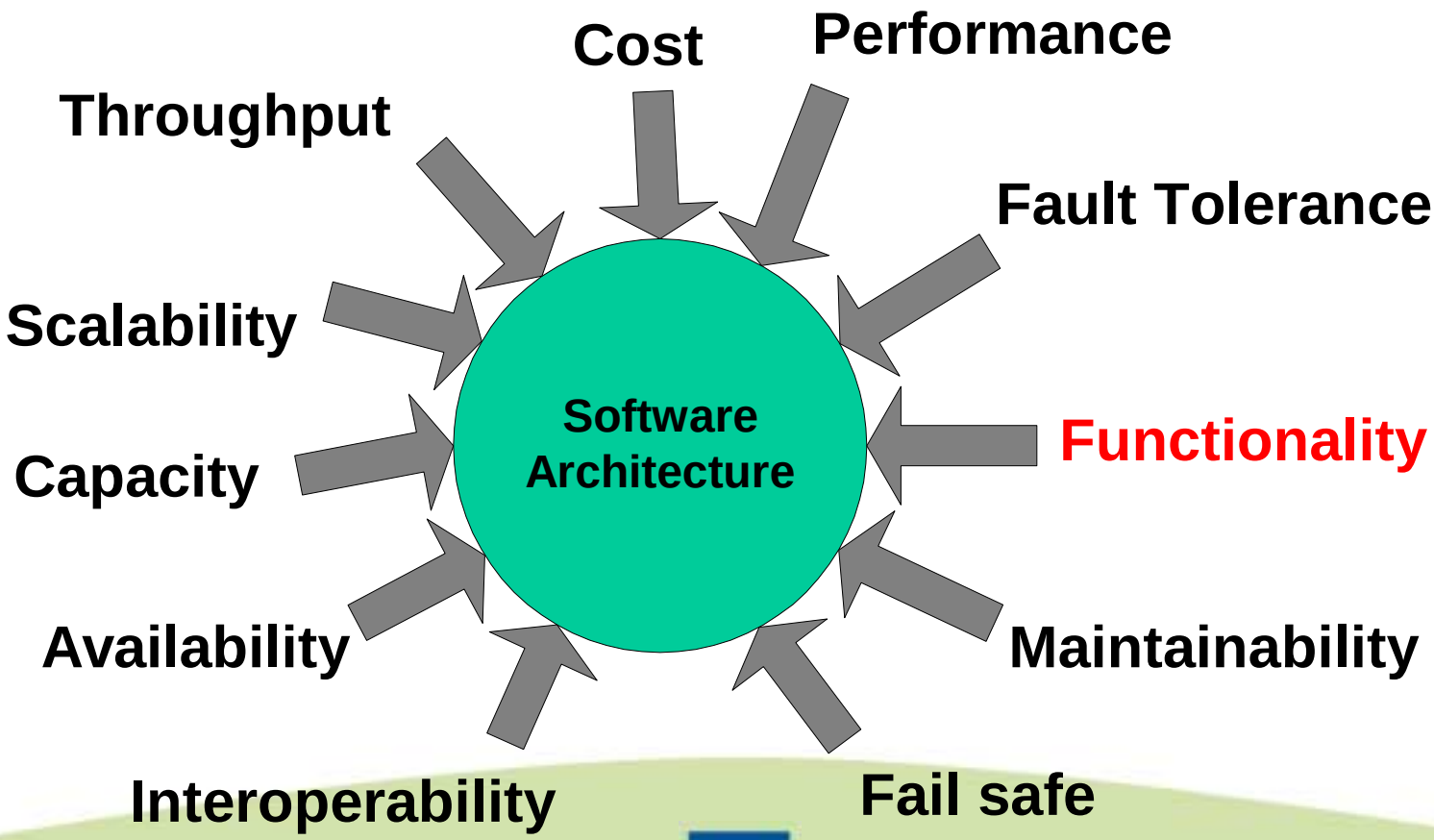
- *Logical Contract* is what other applications care about
- *Physical Contract* is extensible to support any middleware binding
- *XML Schema* provides independent type system

An important question: *How to decompose services?*

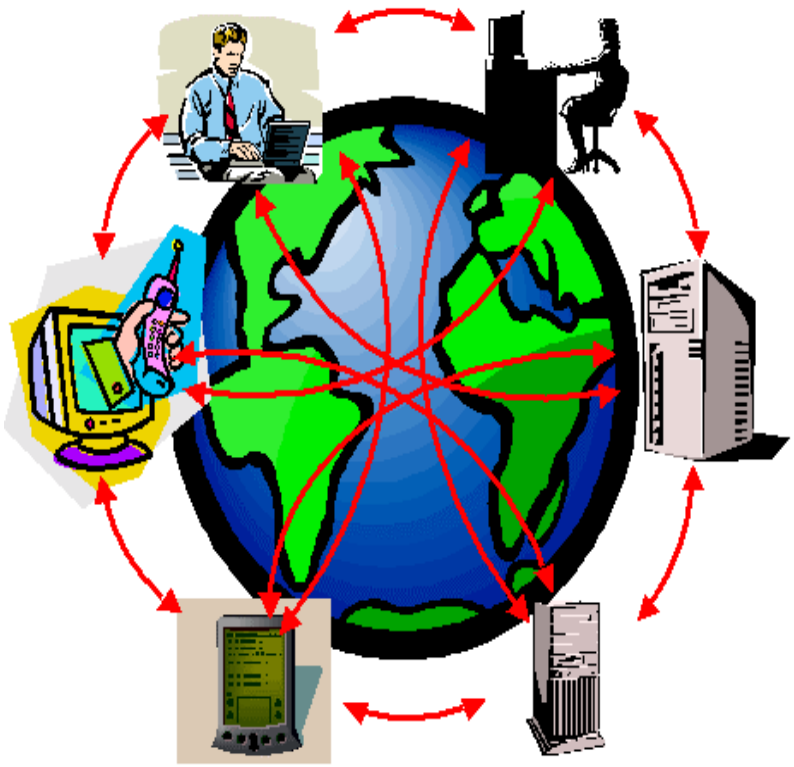
- Quality criteria used to measure the result:

1. Derived from top rank use cases *(adequateness)*
2. Balanced with existing assets: platform technology, frameworks, components
3. Balanced with requirements (trade-offs, performance vs. security,…)
4. Compliance with (domain-specific) industry standards and reference models *(interoperability, readiness for merging)*
5. Designed to make the system more resilient to future changes (20 years?) (*maintainability*)
6. Designed for substantial reuse, and with substantial reuse
7. Intuitively understandable (people buy-in!) *(usability)*

Considerations for SOA design

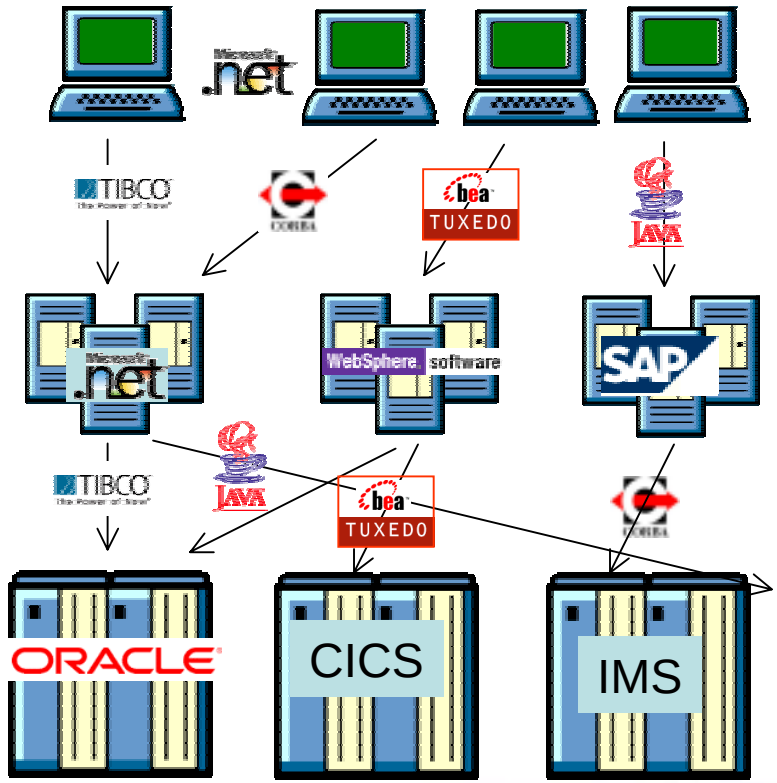


Take the example of the Web



- Human to computer interactions resolved
- Standards in place for programming (HTML) and interoperability (HTTP)
- Huge value adds
- Highly productive, low cost
- *Standardization at the network endpoints*

Web Services: Composites for IT



C, C++, COBOL, PL/I, Java, C#

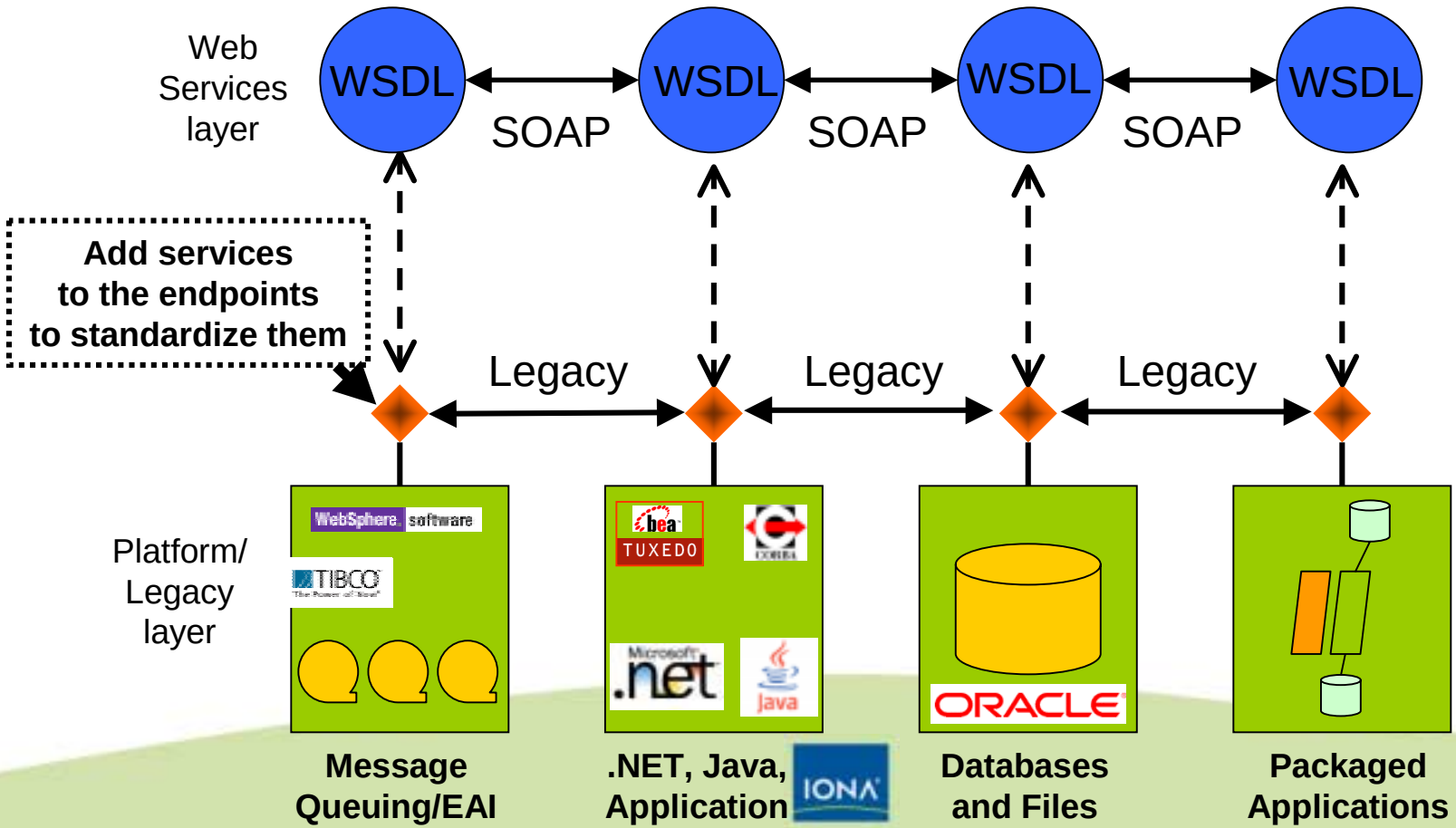
- Standards (WSDL & SOAP) and approach (SOA) need to be deployed – *at the endpoints*
- Open source will help drive adoption
- Join Web 2.0 mashups to enterprise data

IONA

Making Software Work Together™

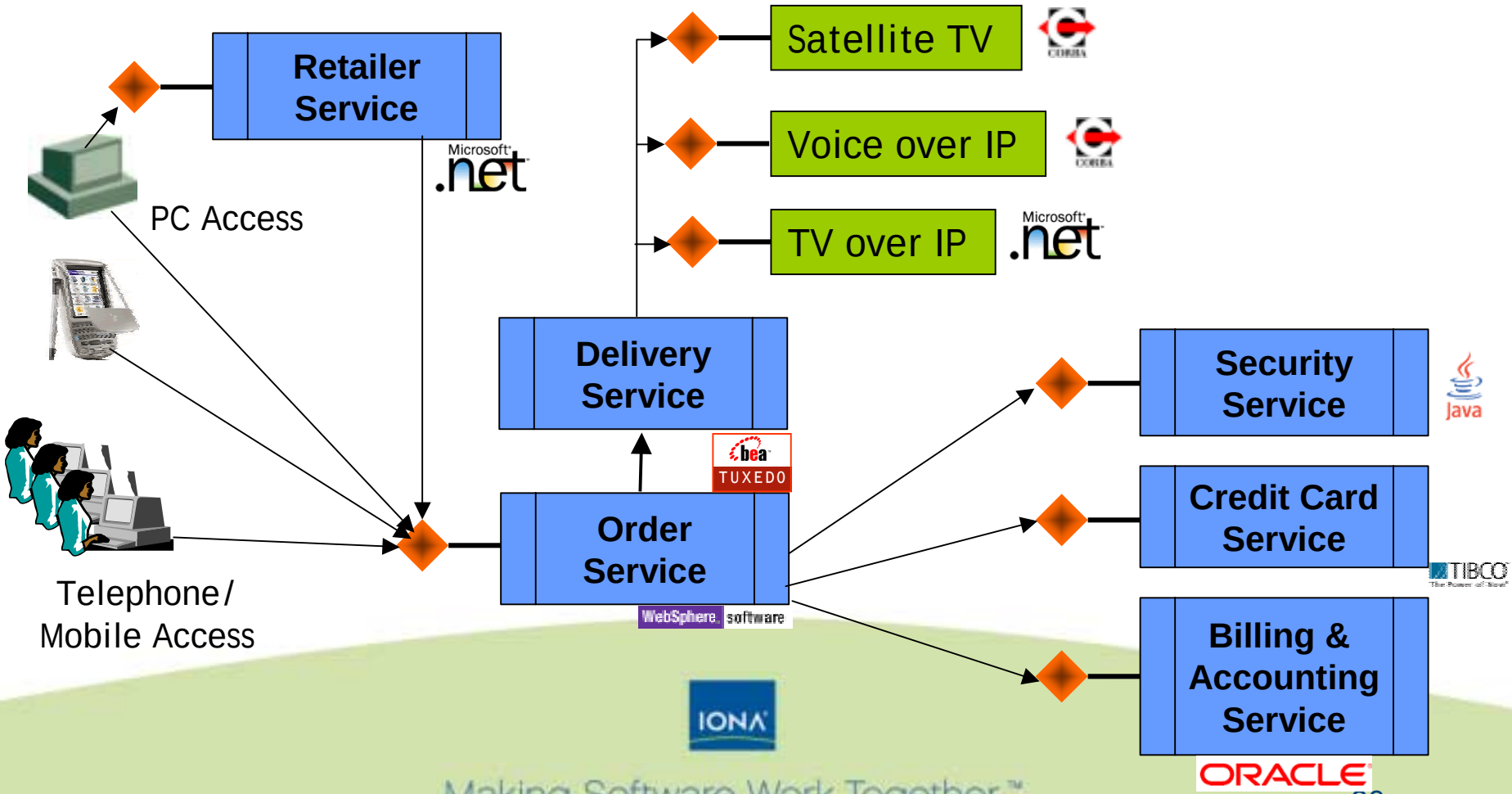
Leverage the power of the endpoints

Applications have a common interface/protocol for composites



Result – Order Entry/Service Delivery

Services can be anywhere on the network, running on any platform, hosted by anyone



Summary

- Enterprises want *business agility*
- They become agile by investing in services
- Key benefits include:
  - Rapid application integration
  - Multi-channel access
  - Obtaining service functionality over the Internet
  - Creating new business models
- SOA requires a change in thinking and organization
- The agile business embraces heterogeneity
  - Resolving differences via services and SOA
  - Best practices involve considering many capabilities