Service Oriented Architecture

Adam Belloum
Faculty of Science, Institute of Informatics,
Section of Scientific Computing
University of Amsterdam
Email: a.s.z.belloum@uva.nl

What is Software Architecture

- **collection** of the **fundamental decisions** about a software product/ solution designed to meet the project ‘s quality attributes
- **Includes** the main **components**, their main **attributes**, and their **collaboration**
- Expressed in **several levels of abstraction** (depending on the project's size).
- Architecture is communicated from multiple viewpoints

Why Architecture?

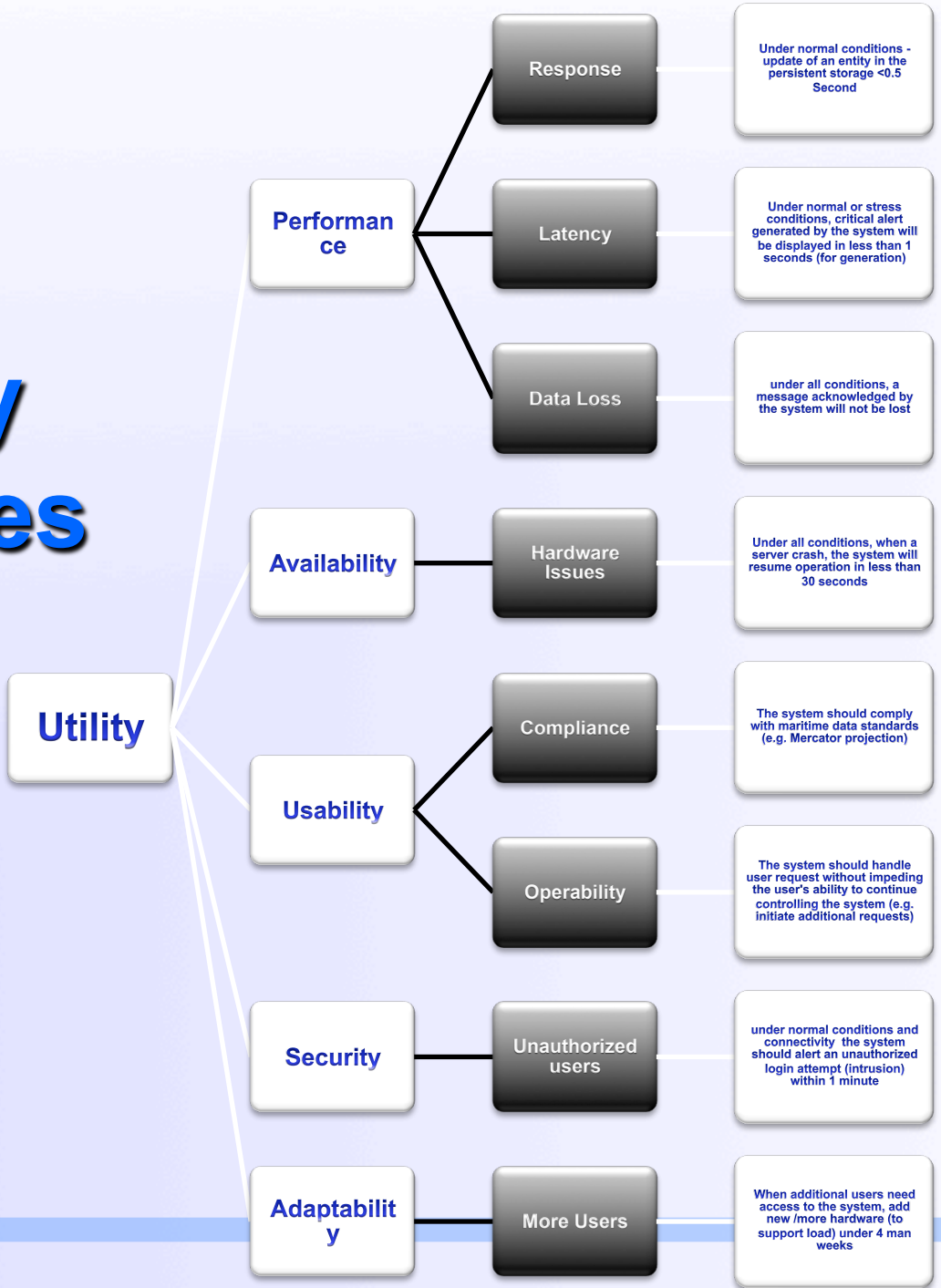
- Architecture serves as the **blueprint** for the **system** but also the **project**:
  - Team structure
  - Documentation organization
  - Work breakdown structure
  - Scheduling, planning, budgeting
  - Unit testing, integration
- Architecture establishes the communication and coordination mechanisms among components

Architecture is Early

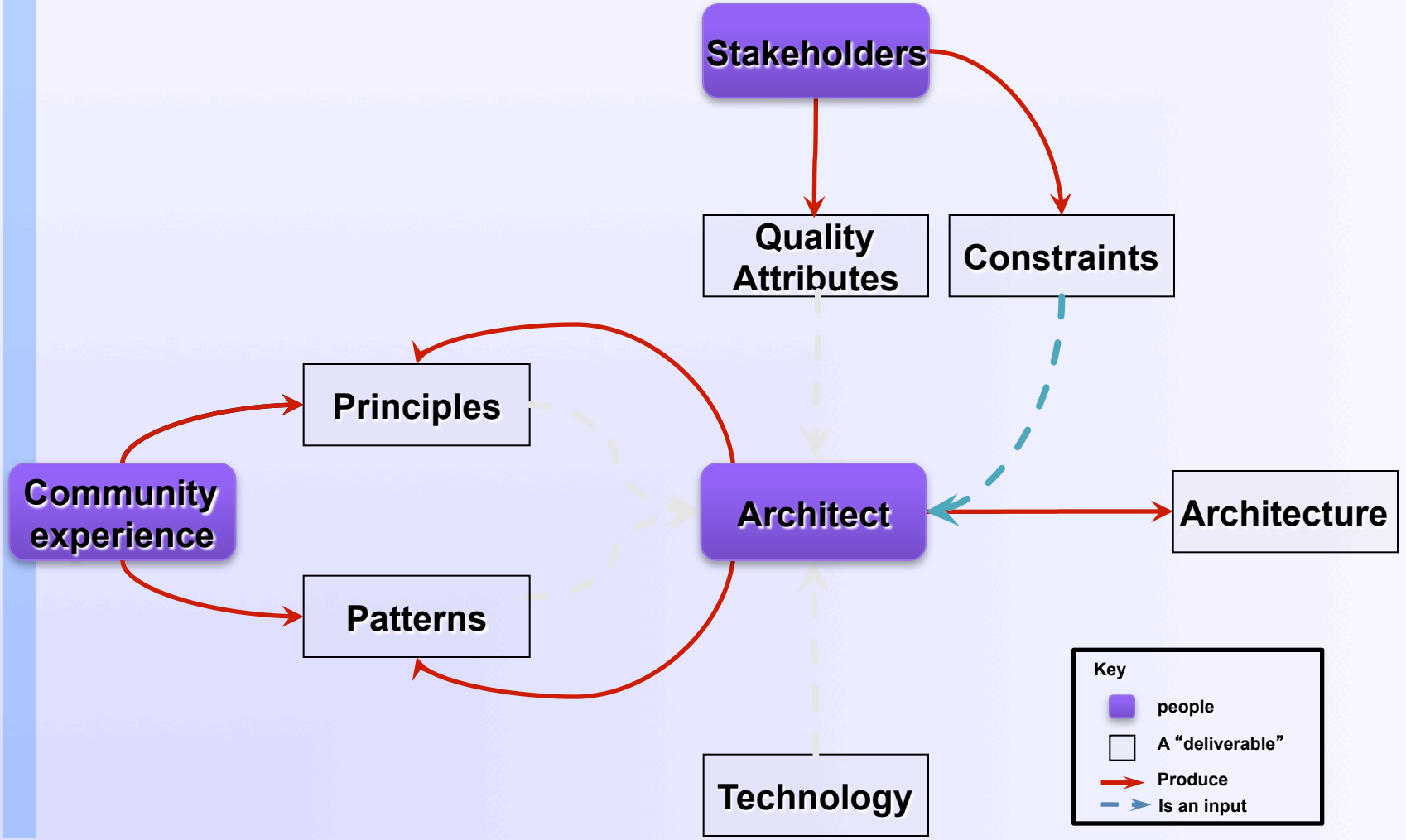
- Architecture represents the set of earliest design decisions
  - Hardest to change
  - Most critical to get right

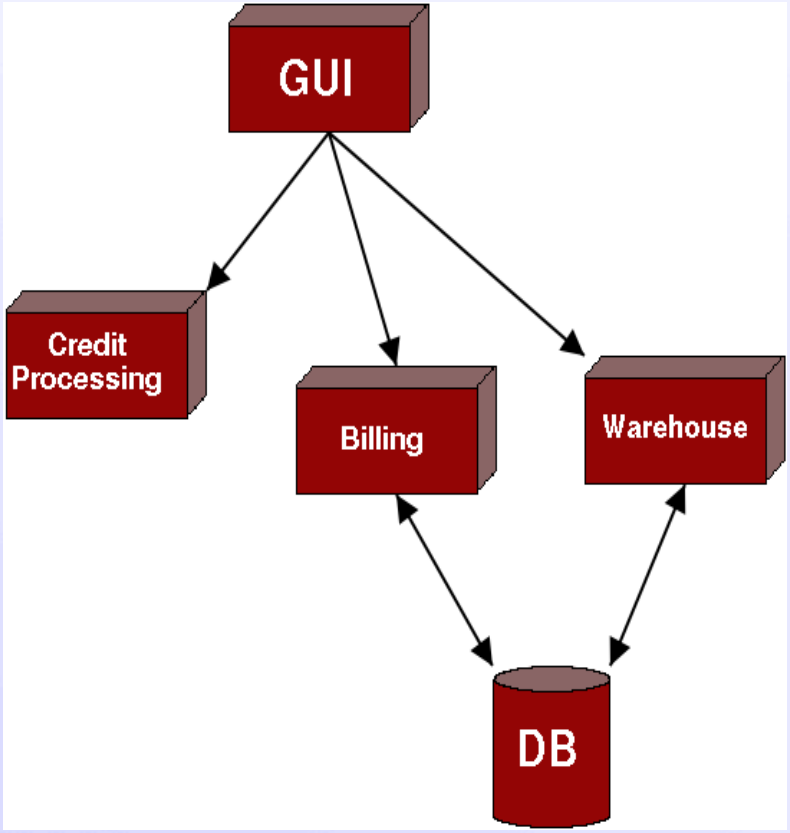
- Architecture is the **first design artifact** where a system's **quality attributes** are addressed

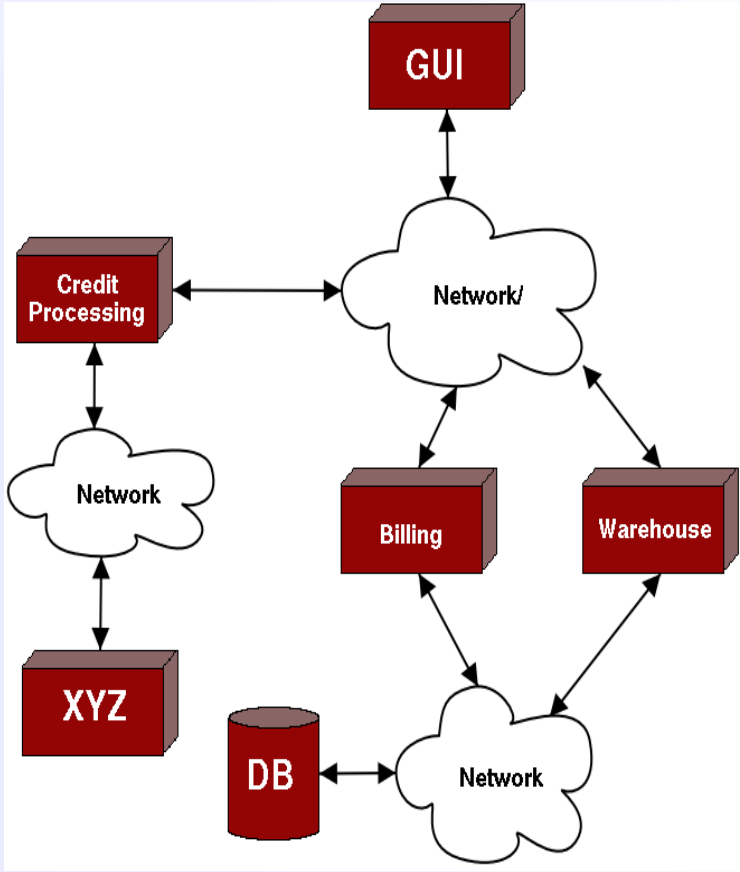
Quality Attributes



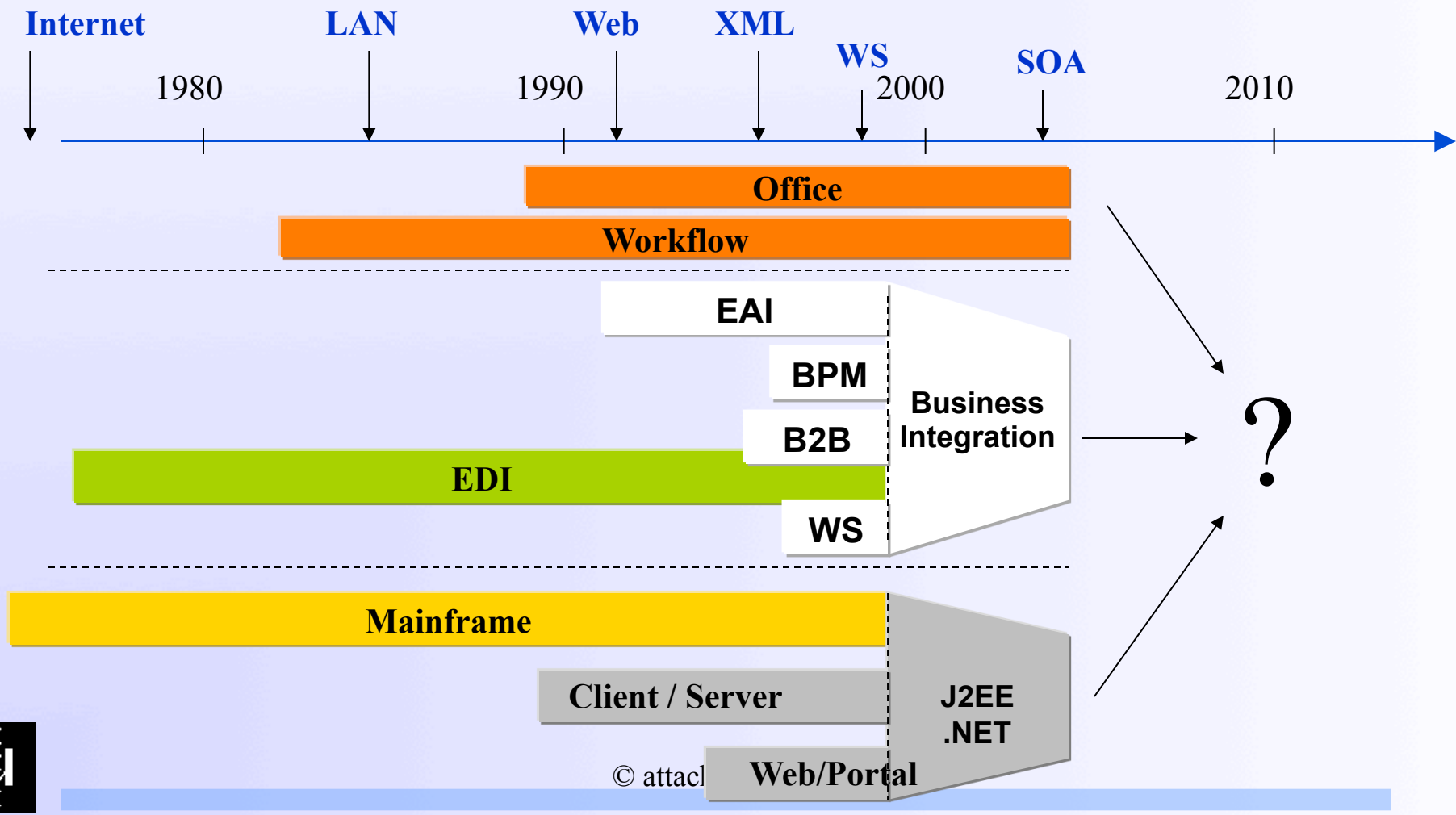
How Architecture



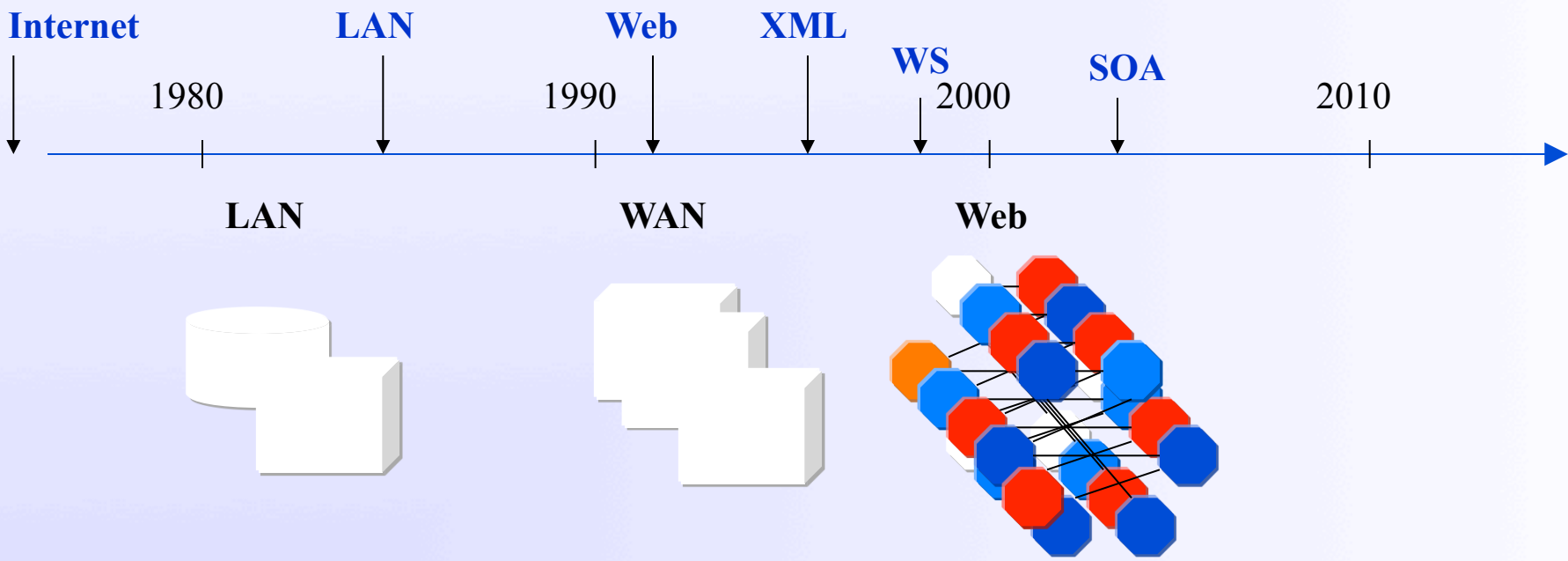

GUI
Credit Processing
Billing
Warehouse
DB

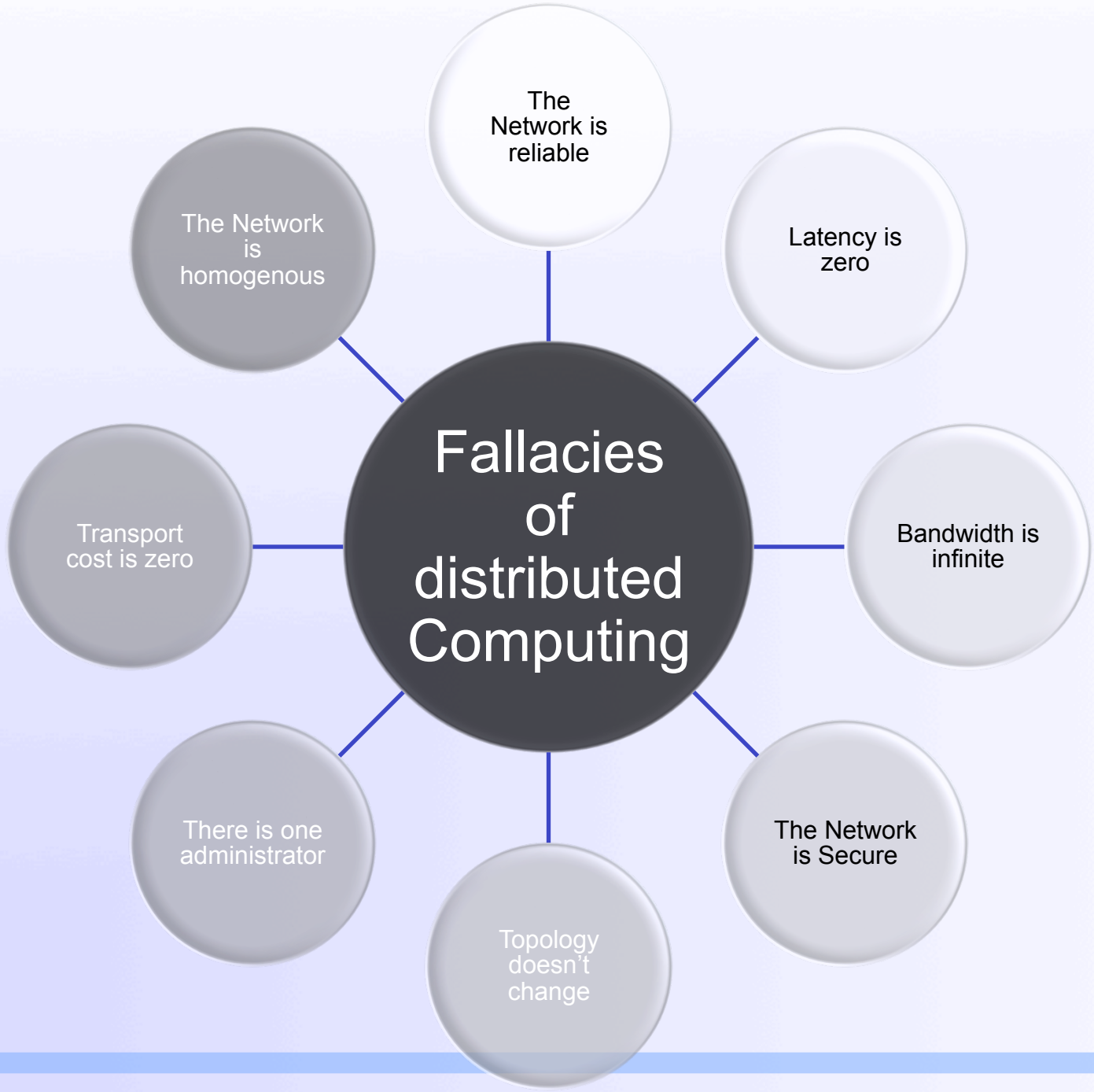

GUI
Credit Processing
Network/
Network
Billing
Warehouse
XYZ
DB
Network

Connectivity Drives the Emergence and Convergence of Technologies



Connectivity Enables Global Processes and Information Access

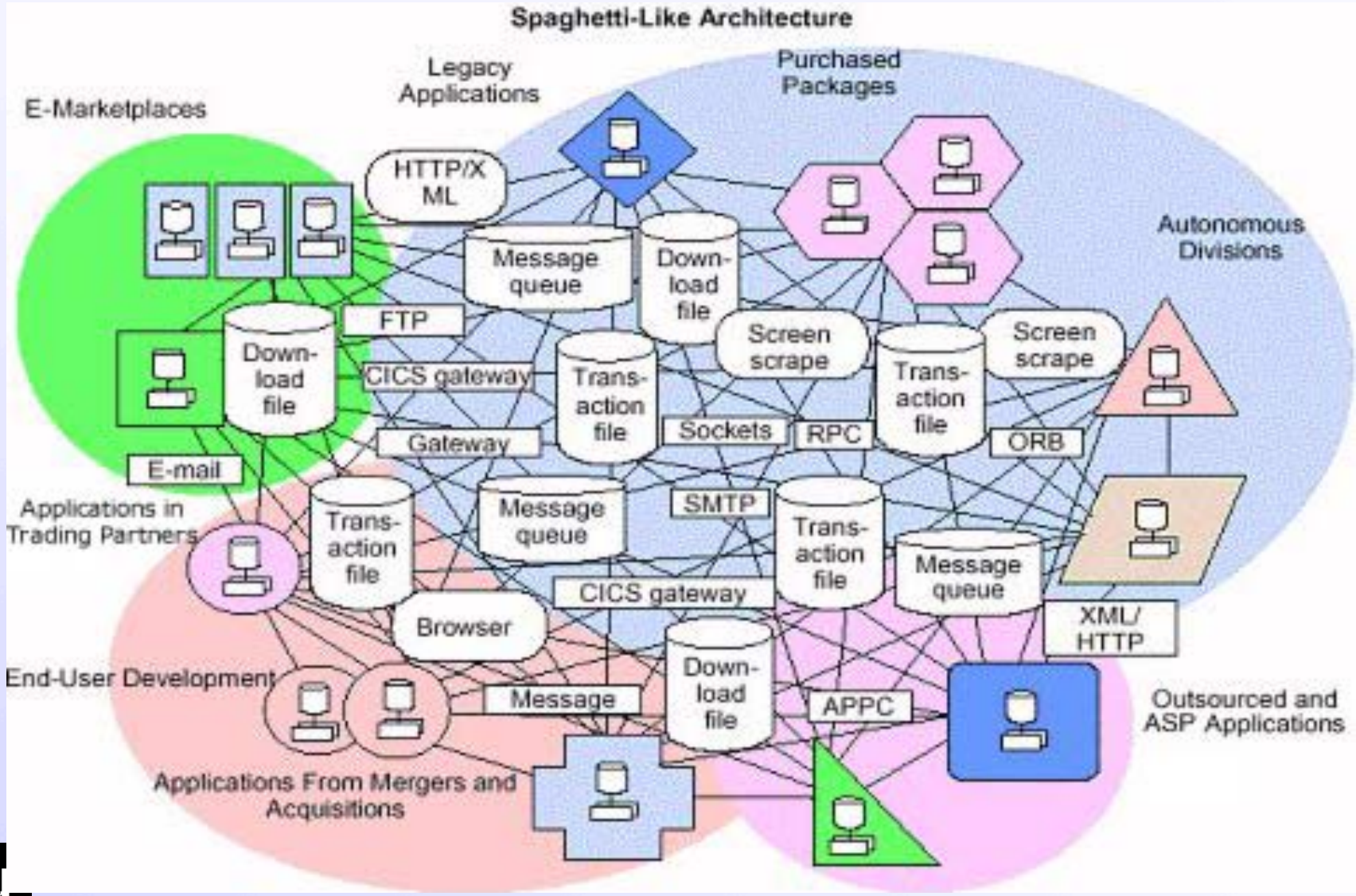



Fallacies of distributed Computing
The Network is reliable
Latency is zero
Bandwidth is infinite
The Network is Secure
Topology doesn't change
There is one administrator
Transport cost is zero
The Network is homogenous

The case for developing SOA

- Level of **Software complexity** continues to **increase**, and traditional architectures seem to be reaching the limit of their ability
- Need to **respond quickly** to new requirements of the Application
- Need to **continually reduce** the cost of IT for the application
- Ability to **absorb** and **integrate** new partners, new users and applications

Problems


Spaghetti-Like Architecture
Legacy Applications
Purchased Packages
E-Marketplaces
HTTP/XML
Message queue
Down-load file
Autonomous Divisions
FTP
Down-load file
Screen scrape
Screen scrape
CICS gateway
Trans-action file
Trans-action file
Sockets
RPC
ORB
Gateway
E-mail
Applications in Trading Partners
Trans-action file
Message queue
SMTP
Trans-action file
Message queue
CICS gateway
XML/HTTP
Browser
End-User Development
Down-load file
Message
APPC
Outsourced and ASP Applications
Applications From Mergers and Acquisitions

Requirement for a SOA

- Leverage existing assets.
  - Existing systems can rarely be thrown away, and often contain within them great value to the enterprise/search group.

- Support all required types of integration.
  - User Interaction
  - Application Connectivity
  - Process Integration
  - Information Integration
  - Build to Integrate

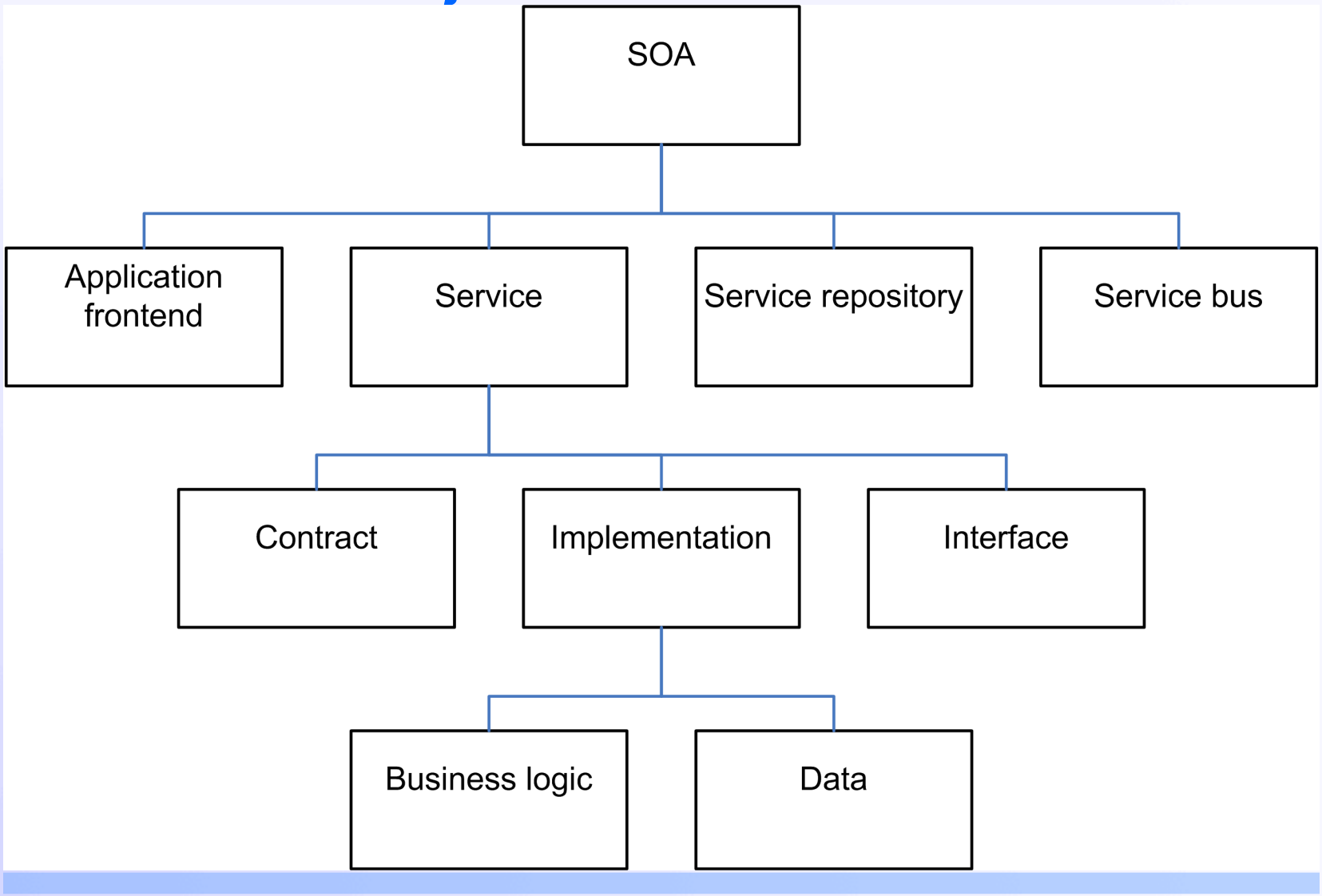
Requirement for a SOA

- Allow for incremental implementations & migration of assets
  →Include a development environment that will be built
    - around a **standard component** framework,
    - promote better **reuse** of **modules** and **systems**,
    - allow legacy assets to be **migrated** to the framework,
    - allow for the **timely** implementation of new technologies.

- Allow implementation of new computing models;
  – specifically, new portal-based client models, Grid computing, and on-demand computing

A service-oriented architecture

-- not just Web services



A service-oriented architecture
-- not just Web services

- All functions are defined as services.
- All services are **independent**.
  - Operate as "black boxes";
    - external components neither know nor care how boxes are executed
- The interfaces are **invocable**;
  - At an architectural level, it is irrelevant whether they are local or remote
    - what interconnect scheme or protocol is used to effect the invocation,
    - what infrastructure components are required to make the connection.

A service-oriented architecture
-- not just Web services

- **Interface** is the key, & the focus of the calling application.
  - It defines the required parameters and the nature of the result
- It is the system's responsibility to effect and manage the invocation of the service,
- This allows two critical characteristics to be realized:
  - Services are truly independent,
  - They can be managed: Security, Deployment, Logging, Dynamic rerouting, and Maintenance

The Nature of a Service

- In a business environment
  - **Service means business functions & transactions, and system services.**
- In a research environment
  - **Service means application functions, and system services**
- The difference in the types of services.
  - **Business functions are from the application's perspective, non-system functions that are effectively atomic.**
  - **Services might be low-level or complex high-level (fine-grained or course grained) functions**

What is a Service (1)

- A facility supplying some public demand
- the work performed by one that serves HELP, USE, BENEFIT
- In economics and marketing, a service is the non-material equivalent of a good. Service provision has been defined as an economic activity that does not result in ownership, and this is what differentiates it from providing physical goods.
- It is claimed to be a process that creates benefits by facilitating either a change in customers, a change in their physical possessions, or a change in their intangible assets.

What is a service (2)

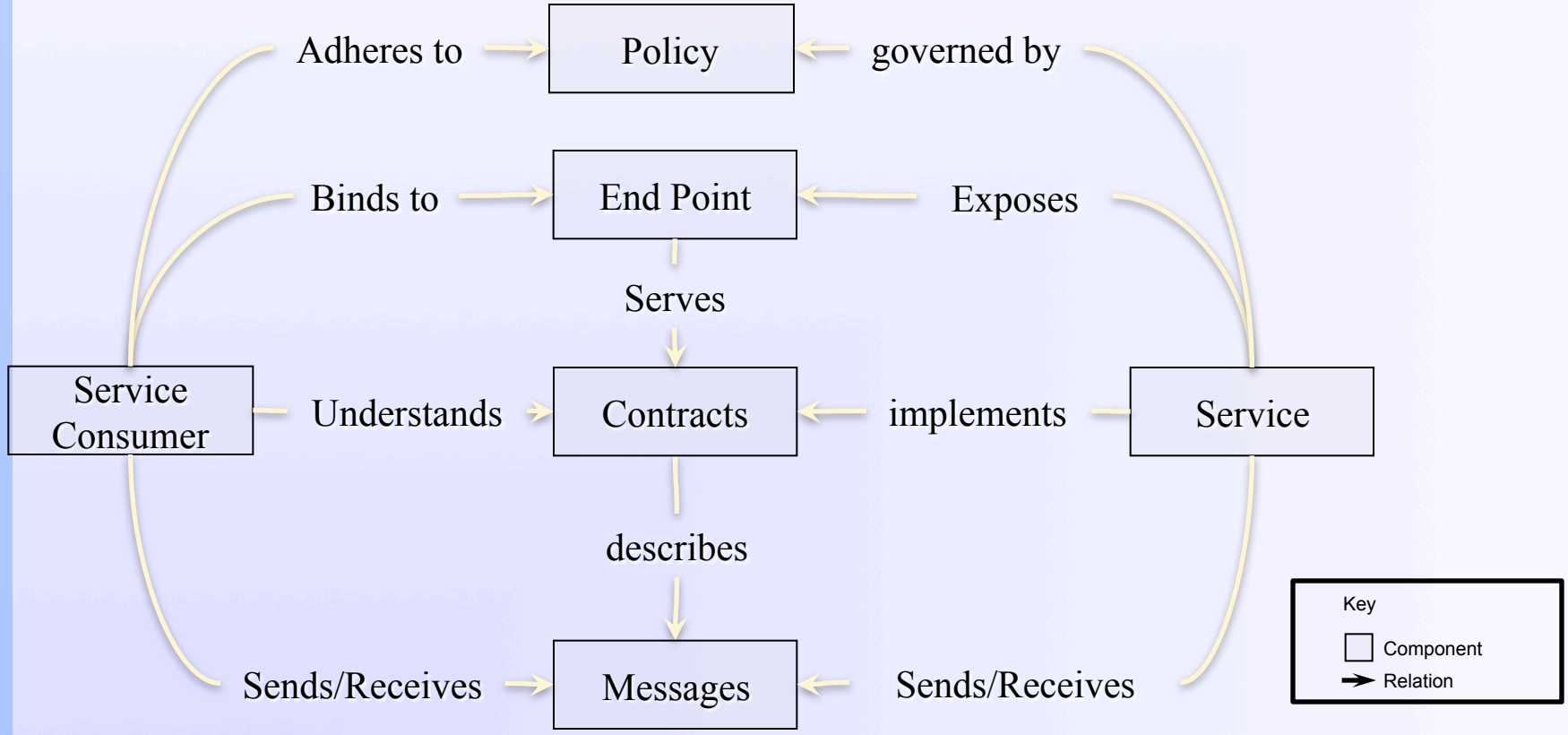
- A Windows Service?
  - RPC Locator, EventLog, DHCP Client,
- Software Service?
  - Distribution Service, Alert Service
  - Security Service, Log Service
- Business Service?
  - Common Operational Picture, Navigation
  - Accounts Receivable, Customers

SOA isn't a solution to world hunger

- Nor is it:
  - A specific Technology
  - The Ultimate answer to reuse
  - A New name for EAI
  - A New way to do RPC

Little SOA

- Architectural Style
- For building distributed systems
- Loosely coupled components
- Our focus from here onward…


Policy
Adheres to
governed by
End Point
Binds to
Exposes
Serves
Service Consumer
Understands
Contracts
implements
Service
describes
Messages
Sends/Receives
Sends/Receives
Key
Component
Relation

Services and Systems

- A **service** is a program you interact with via message exchanges
- A **system** is a **set** of **deployed services cooperating** in a given task

An SOA - Constituent Parts

- To determine what the constituent parts of an SOA are it is first necessary to break down the question into the design-time and run-time requirements.

- The idea that SOA covers both design-time and run-time is critical to understanding SOA

  – SOA is really about both physical and logical architectures.

SOA- Design-time requirements

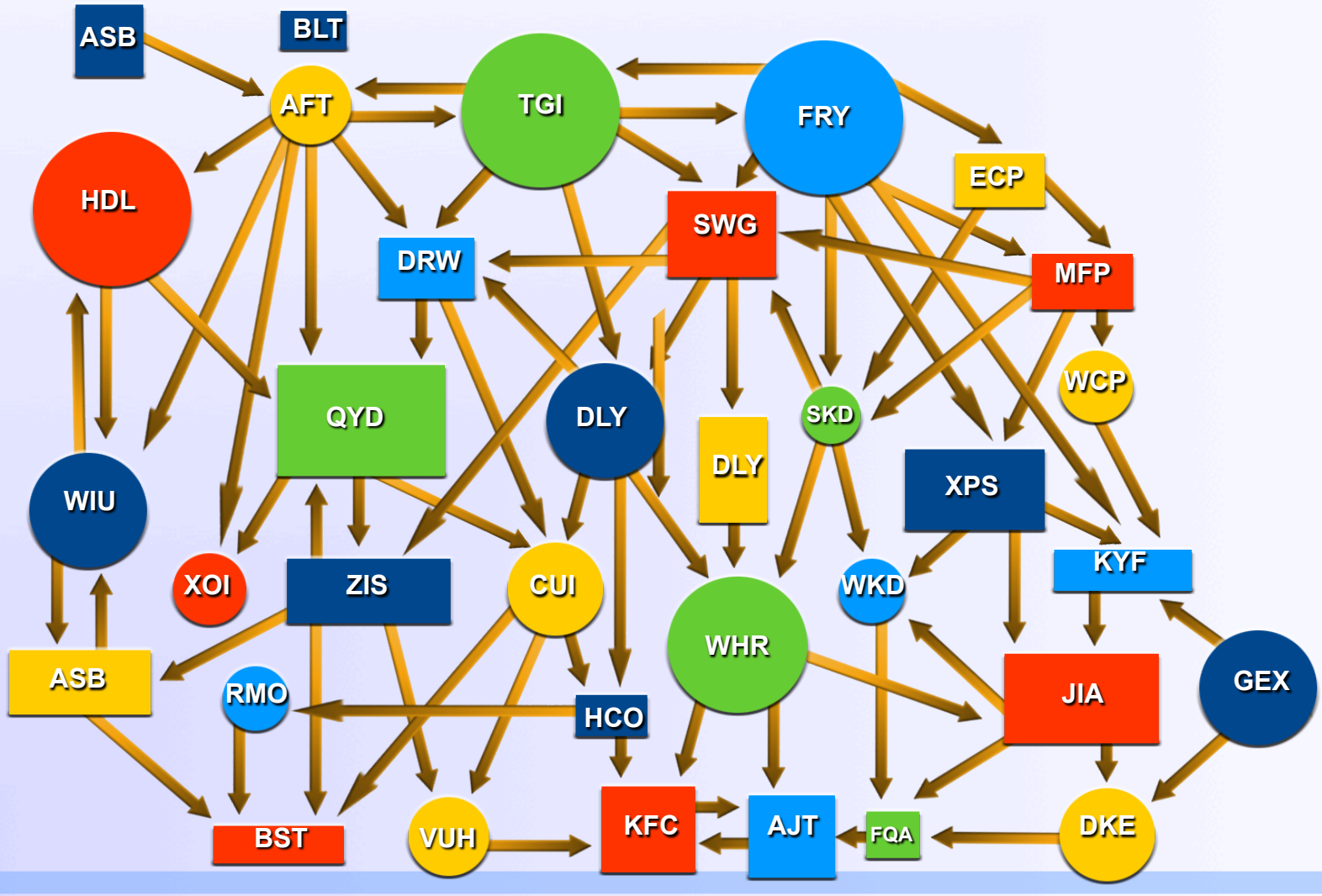
- **Re-use available** existing **services** when designing new processes/application
  - **Directory** of services
    - provides the definition of a set of services
    - indicates whether the web service is available.
- Agile Design Methodology which is oriented towards re-use,
  - methodology needs to emphasize the requirement for cross-project information and working.

SOA - Design-time requirements

- Process Driven Development
  - based upon the modeling or re-modeling of processes.
  - The **start point** should be the **expansion** or **re-working** of the set of modeled processes.
- Workflow Oriented Development
  - A key paradigms for SOA development is that **application processes are seamless**.
    - Each step in each process should be linked, as an automatic next step

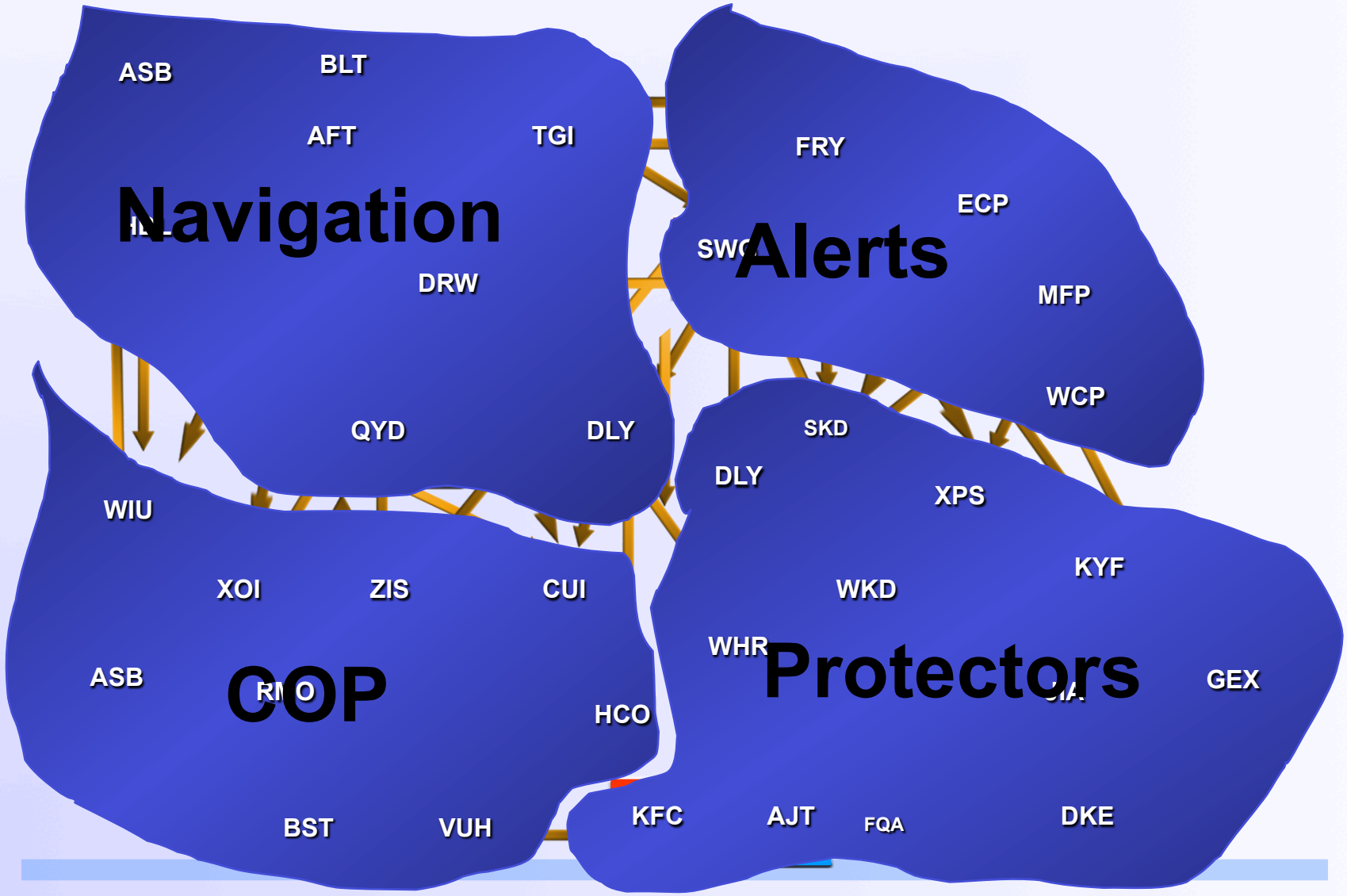
Big SOA

Analyze the business



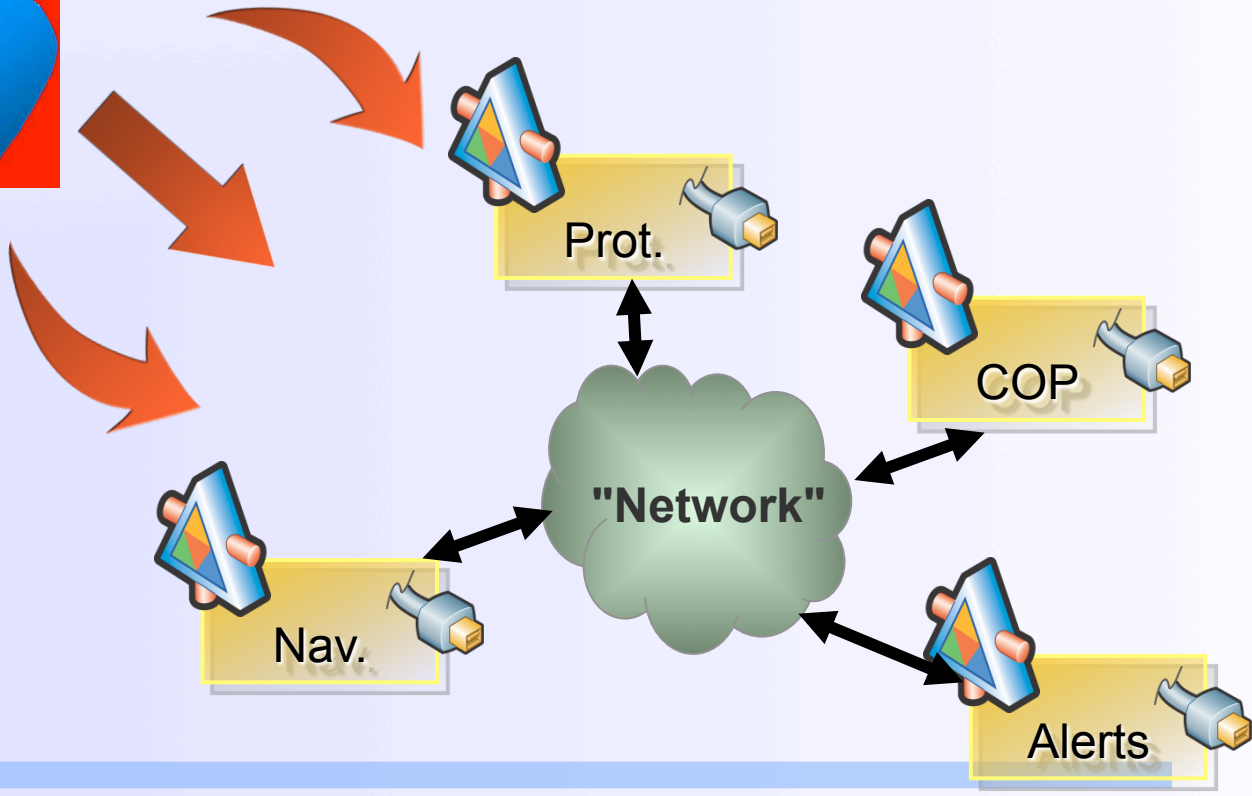
Big SOA

Identify Business Areas



Big SOA

Map to software



SOA - Design-time requirements

- Information Routing Modeling
  - the need to integrate & deliver information to the right people at the right time.
  - SOA solution must be able to **model the flows of information** across the VOs and the extended supply chain.
- Agile Toolset For SOA Development
  - abstraction of existing functionality into new services
  - Debugging And Simulation Capability
  - Multi-Language Capability

SOA – Run-time requirements

- Consolidated Process Management
  - ability to present transactional and information flows visually by application process, organizational unit and server.
- Process Oriented Monitoring & Administration Tools
  - Run Time env should display information at the process level and allow activation/de-activation of any process (stopping the process at a specific step) as a means of handling problems/ implementation.

SOA – Run-time requirements

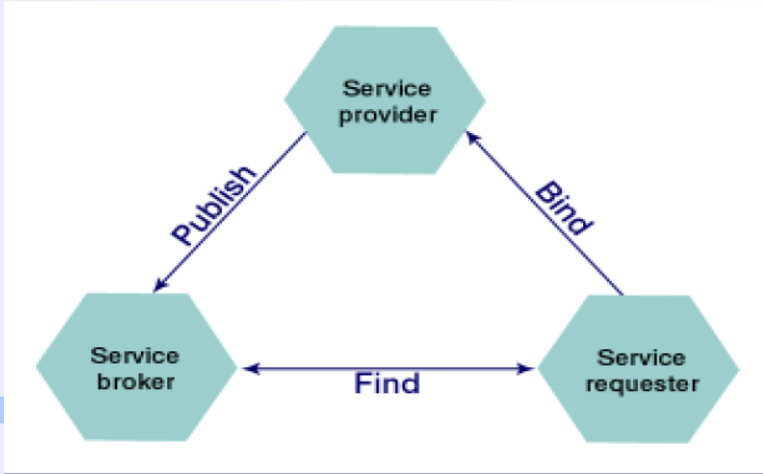
- Persistence Of Message-Based Asynchronous Process Data
  - SOA requires a data store external to the applications that provide the underlying functionality, akin to an Operational Data Store, to store potentially long-term but essentially transient process related data
- Scalability Of The Environment
  - Scaleable means that the toolset supports the deployment of further servers, the assignment of specific processes or organizational units to servers and the management of software across servers.
- Resilience
  - must provide sufficient resilience to support the application
- User Access And Security
  - SOA solution offers a browser-based, role-oriented experience for the user which incorporates task lists based on the users' roles and the relevant collaboration and knowledge content as well as links to the key web sites for the role.

SOA – Run-time requirements

- Workflow- Availability of workflow functionality in any SOA solution
  - facilitates the Easy linking of processes/process parts or Browser-based task lists for the users
- Event driven
  - The link between processes (or between a process and the external world) will often be in the form of an event.
- Error Management
  - A key criterion for any SOA run-time environment is its error management. The criteria for error management are;
  - Visibility of errors, Re-start capability, Error notification, Workflow linking
- Simulation capability
  - The ability to simulate traffic across any process is very useful when reviewing performance and scalability questions.

SOA Model

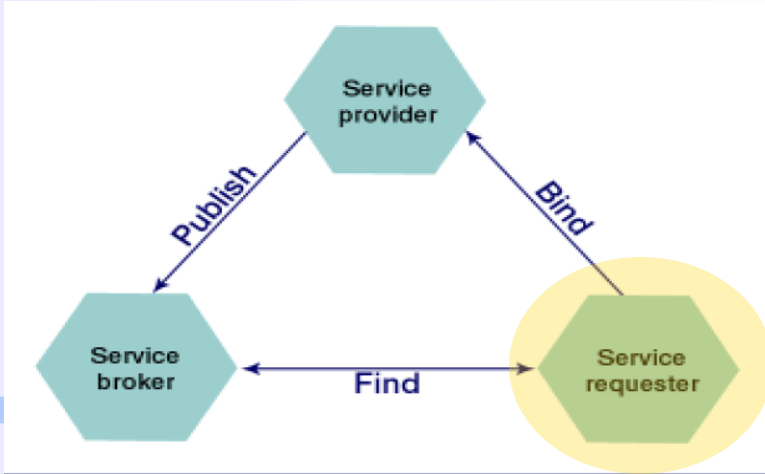
- A *service provider*
  - provides a service interface for a software asset that manages a specific set of tasks.
- A *service requester*
  - discovers and invokes other software services to provide a business solution..
- A *service broker*;



Service Requester

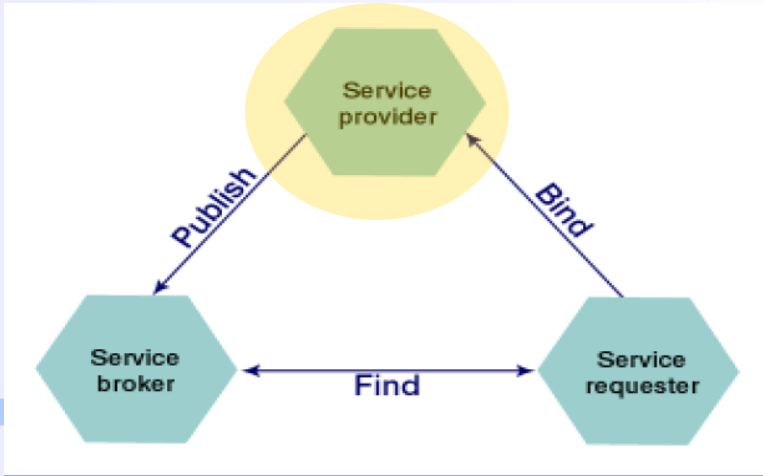
- *Content Aggregation*
  - Activity where an entity interacts with a variety of content providers to process/reproduce such content in the desired presentation format of its customers.

- Service Aggregation
  - Activity where an entity interacts with a variety of service providers to re-brand, host, or offer a composite of services to its customers.



Service Provider

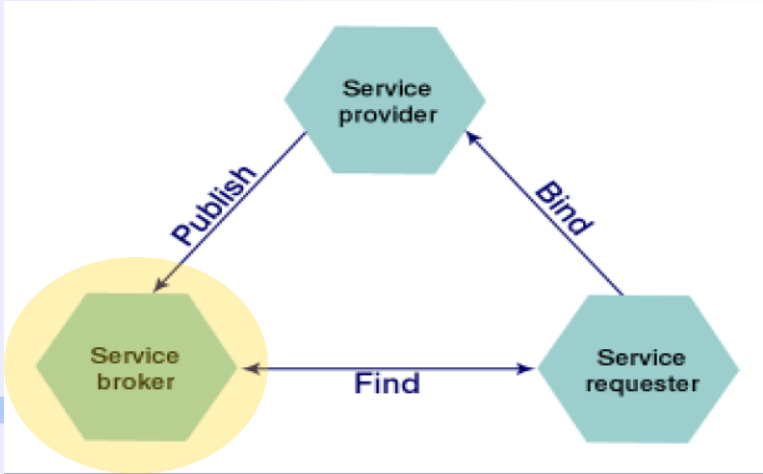
- Independent software vendors are prime examples of potential service providers.
  - They own and maintain a software asset that performs tasks.
  - Software assets could be made available as an aggregation of services or broken down into distinctly separate software service.
- Processes that are proven and generalized for a diverse set of applications would be good candidates for service providers.
  - For example, if a bank felt that its business process for loan processing was a strong enough asset to be made publicly available and was willing to support it as a business offering, then that bank could view itself as a loan processing service provider.



Registry

- Is an entity that collects and catalogs data about other entity and then providing that data to others (a form of SOA Broker.)
- Usually, a registry would collect data such as
  - Entity name,
  - Description, and contact information.

In UDDI terms, this Registry role is often referred to as the *White Pages*.




It’s all about the
Message!


Messages
travel in
no man’s land!


LOST
Is he Idempotent?
Messages Get Retransmitted
Messages Arrive More than once

Idempotence

- Idempotent Means It's OK to Arrive Multiple Times
  - As Long as the Request Is Processed at Least Once, the Correct Stuff Occurs
- In Today's Internet, You Must Design Your Requests to Be Idempotent

Naturally Idempotent
Sweeping the Floor

Idempotent
Baking a Cake Starting from the Shopping List (If Money Doesn't Matter)

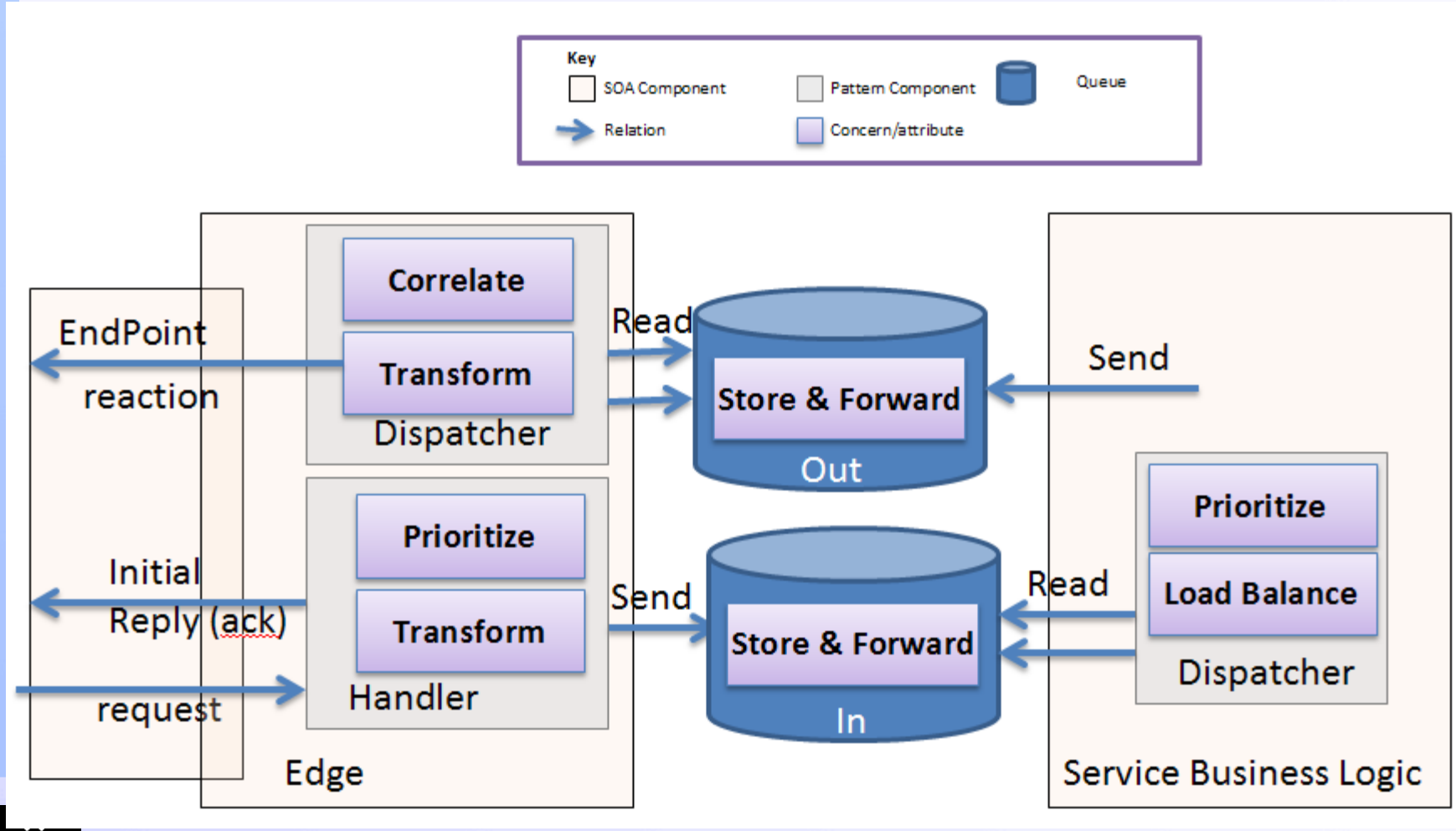
Not Idempotent
Withdrawing $1 Billion

Idempotent
If Haven't Yet Done Withdrawal #XYZ for $1 Billion, Then Withdraw $1 Billion and Label as #XYZ

Not Idempotent
Baking a Cake Starting from Ingredients

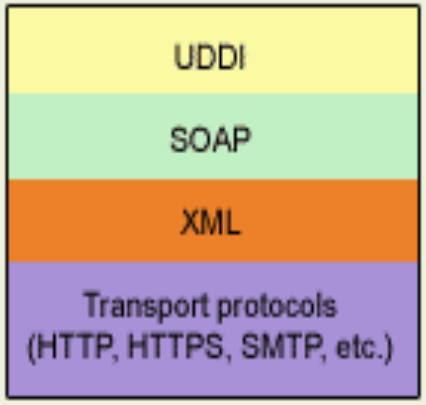
Naturally Idempotent
Read Record "X"

Decoupled Invocation Pattern



Enabling technologies

- XML: The Extensible Markup Language
- SOAP:
  - Simple Object Access Protocol is an XML-based lightweight protocol for the exchange of information in a decentralized,
- WSDL:
  - The Web Services Description Language is an XML vocabulary that provides a standard way of describing service IDLs.
- UDDI:
  - The Universal Description, Discovery, and Integration specification provides a common set of SOAP APIs that enable the implementation of a service broker.



Web Services Protocol Stack

Source: Java™ Web Services Unleased

12 Steps to implement a SOA

1. Understand the functional objectives and define success.
2. Define your problem domain.
3. Understand all application semantics in your domain.
4. Understand all services available in your domain.
5. Understand all information sources and sinks available in your domain.
6. Understand all processes in your domain.

12 Steps to implement a SOA

7. Identify and catalog all interfaces outside of the domain you must leverage (services and simple information).
8. Define new services/information bound to the services.
9. Define new processes, services, and information bound to the processes.
10. Select your technology set.
11. Implement & Deploy SOA technology.
12. Test and evaluate