Introduction to Cloud Computing

Sanjay Chaudhary
Dean of Students, Ahmedabad University
Professor and Associate Dean,
School of Engineering and Applied Science

Presentation Outline

- Background
  - Distributed Computing
    - Definition, RPC, J2EE …
  - Grid Computing
- Service-Oriented Architecture
- Cloud Computing

Distributed Computing

High-Performance Computing

- It generally refers to traditionally known as super computing.
- Key parallel processing algorithms have already been developed to support execution of programs on different, but co-located processors.
- A list of industries in which HPC systems are deployed can be found at www.top500.org

Industry area

- Telecommunication: Sprint, Duetsche Telekom
- Automotive: BMW, GM
- Electronics: Cisco, Motorola
- Chemistry: Bayer

Cluster Computing

- Cluster Computing came about as a response to the high prices of supercomputers.
- Clusters are high performance, massively parallel computers built primarily out of commodity hardware components, running a free-software operating system such Linux and interconnected by a private high-speed network.
- It consists of cluster of PCs, workstations, dedicated to running high-performance computing tasks.
- A cluster is usually connected to the outside world through only one single node.
- Cluster computing has been around since 1994 when the first Beowulf clusters were developed and deployed.
- Platform Computing developed many of the early load-balancing tools for clusters.
- Industries: life sciences, digital entertainment, finance etc.

Peer-to-Peer Computing

- Recent growth of user-friendly file-sharing networks, such as Napster or Kazaa
- Centralized model vs. Decentralized model

Internet Computing

- The explosion of the Internet and increasing power of the home computer prompted computer scientists and engineers to apply techniques learned in HPC and cluster-based distributed computing to utilize the vast processing cycles available at users' desktops.
- Large compute intensive projects are coded so that tasks can be broken down into smaller subtasks and distributed over Internet for processing.
- Volunteer users then download a lightweight client onto their desktop, which periodically communicates with the central server to receive tasks.
- The client initiates the task when the desktop CPU is not in use.
- Upon completion of the task, it communicates results back to the central server. The central server aggregates the information received from all the different desktops and compiles results.

Internet Computing Projects

Internet Computing projects, although not profitable, have allowed companies to understand large-scale distributed computation projects.

Science

- SETI@Home
- Analytical Spectroscopy Research Group
- Evolutionary Research
- eOn
- Climateprediction.com
- Distributed Particle Accelerator Design

Internet Computing Projects

Life Sciences

- Folderoi
- Folding@Home
- Genome@Home
- FightAIDS@Home
- Ubero
- Drug Design Optimization Lab
- The Virtual Laboratory Project
- Distributed Folding, Community TSC
- Find-a-Drug

Cryptography

- Distributed, net
- ECCp-109

Internet Computing Projects

Mathematics

- Great Internet Mersenne Prime Search
- Proth Prime Search
- ECMNET
- n!+1 and n!-1 Prime Search
- Minimal Equal Sums of Like Powers
- GRISK
- MM61 Project
- 3x +1 Problem

Internet Computing Projects

- Pi(x) Project
- Distributed Search for Format Number Divisors
- PCP@Home
- Generalized Woodal! Numbers
- Generalized Fermat Prime Search
- ZetaGrid, Strong Pseudoprime Search
- Wilson Prime Search
- Largest Proth Prime Search

Internet Computing Projects

- Search for Primes of the Form k.2n -1
- Weiferich Prime Number Search
- Seventeen or Bust
- Factorizations of Cyclotomic Numbers

Source: Grid Technology Partners

Grid Computing

Grid Computing: Definition

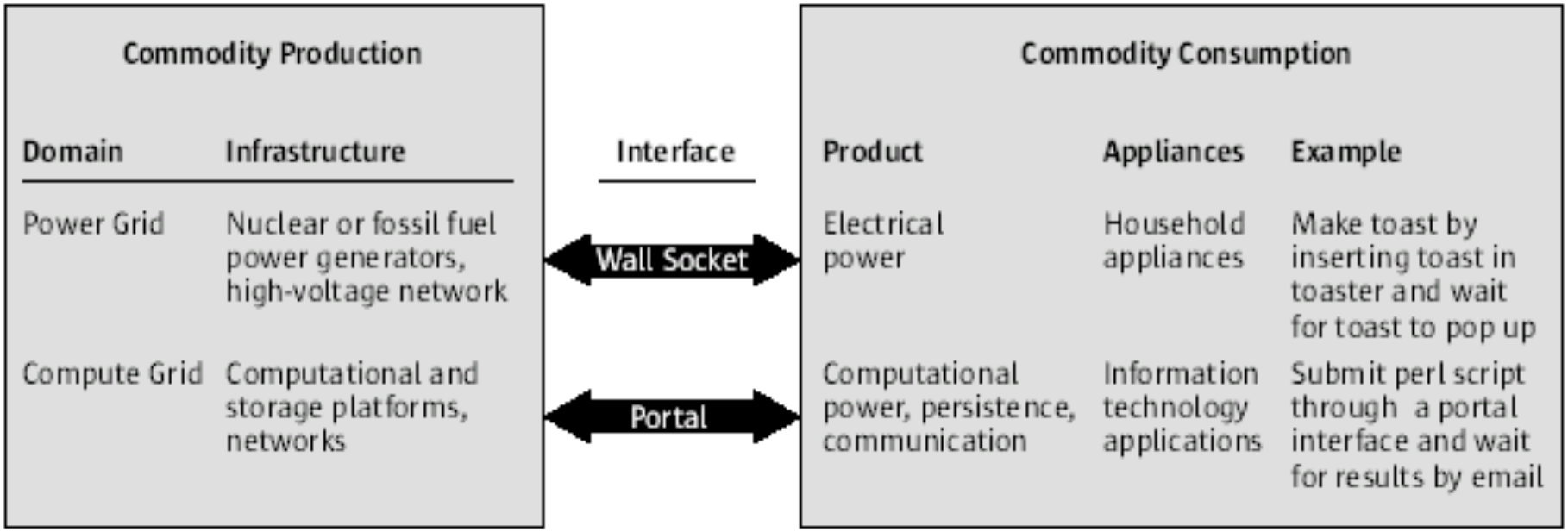
- Grid computing tries to bring, under one definitional umbrella all the work being done in the high performance, cluster, peer-to-peer and Internet computing arenas.
- In 1969 Len Kleinrock suggested: "We will probably see the spread of 'computer utilities', which, like present electric and telephone utilities, will service individual homes and offices across the country.“
- Book: "The Grid: Blueprint for a New Computing Infrastructure", 1998, "A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities.”

Grid Computing: Definition

- Another definition by Dr. Rajkumar Buyya: “Grid is a type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed "autonomous" resources dynamically at runtime depending on their availability, capability, performance, cost, and users' quality-of-service requirements.”
- “Grid Computing enables virtual organizations to share geographically distributed resources as they pursue common goals, assuming the absence of central location, central control, omniscience, and an existing trust relationship.” www.globus.org
- Over a period of time, there will be — academic grids, enterprise grids, research grids, entertainment grids, community grids and so on.

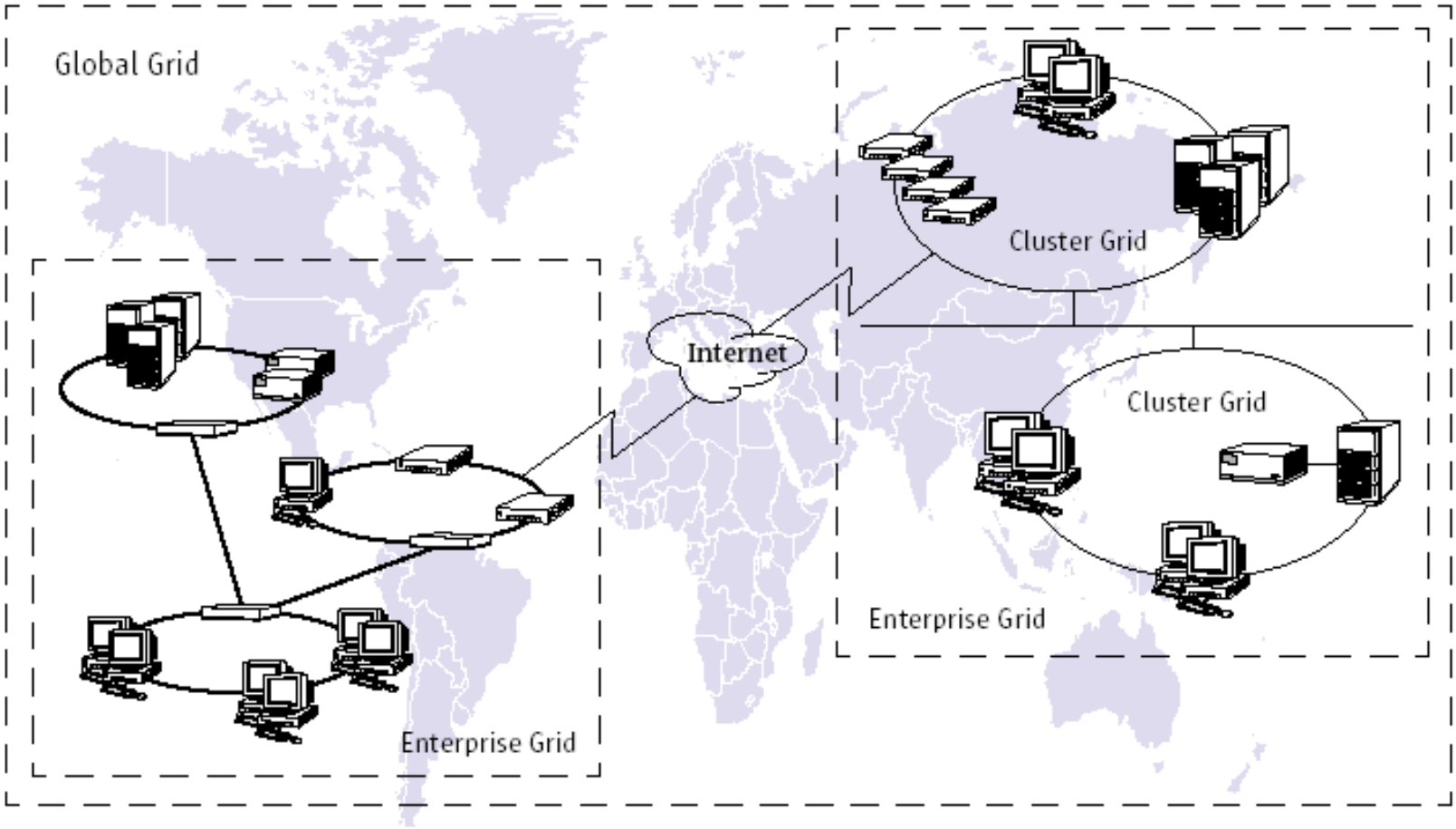
Grid Computing: Definition

- Grid computing is concerned with “coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations.” by Ian Foster
- The key concept is the ability to negotiate resource-sharing arrangements among a set of participating parties (providers and consumers) and then to use the resulting resource pool for some purpose.
- The sharing that we are concerned with is not primarily file exchange but rather direct access to computers, software, data, and other resources, as is required by a range of collaborative problem-solving and resource brokering strategies emerging in industry, science, and engineering.


Commodity Production
Domain
Infrastructure
Power Grid
Nuclear or fossil fuel power generators, high-voltage network
Compute Grid
Computational and storage platforms, networks
Interface
Wall Socket
Portal
Commodity Consumption
Product
Appliances
Example
Electrical power
Household appliances
Make toast by inserting toast in toaster and wait for toast to pop up
Computational power, persistence, communication
Information technology applications
Submit perl script through a portal interface and wait for results by email

A Grid Checklist, suggested by Ian Foster

1. Coordinates resources that are not subject to centralized control …
- (A Grid integrates and coordinates resources and users that live within different control domains—for example, the user's desktop vs. central computing; different administrative units of the same company; or different companies; and addresses the issues of security, policy, payment, membership, and so forth that arise in these settings. Otherwise, we are dealing with a local management system.)
2. … using standard, open, general-purpose protocols and interfaces
- … (A Grid is built from multi-purpose protocols and interfaces that address such fundamental issues as authentication, authorization, resource discovery, and resource access. As I discuss further below, it is important that these protocols and interfaces be standard and open. Otherwise, we are dealing with an application specific system.)
3. … to deliver nontrivial qualities of service.
- (A Grid allows its constituent resources to be used in a coordinated fashion to deliver various qualities of service, relating for example to response time, throughput, availability, and security, and/or co-allocation of multiple resource types to meet complex user demands


Global Grid
Cluster Grid
Internet
Cluster Grid
Enterprise Grid
Enterprise Grid

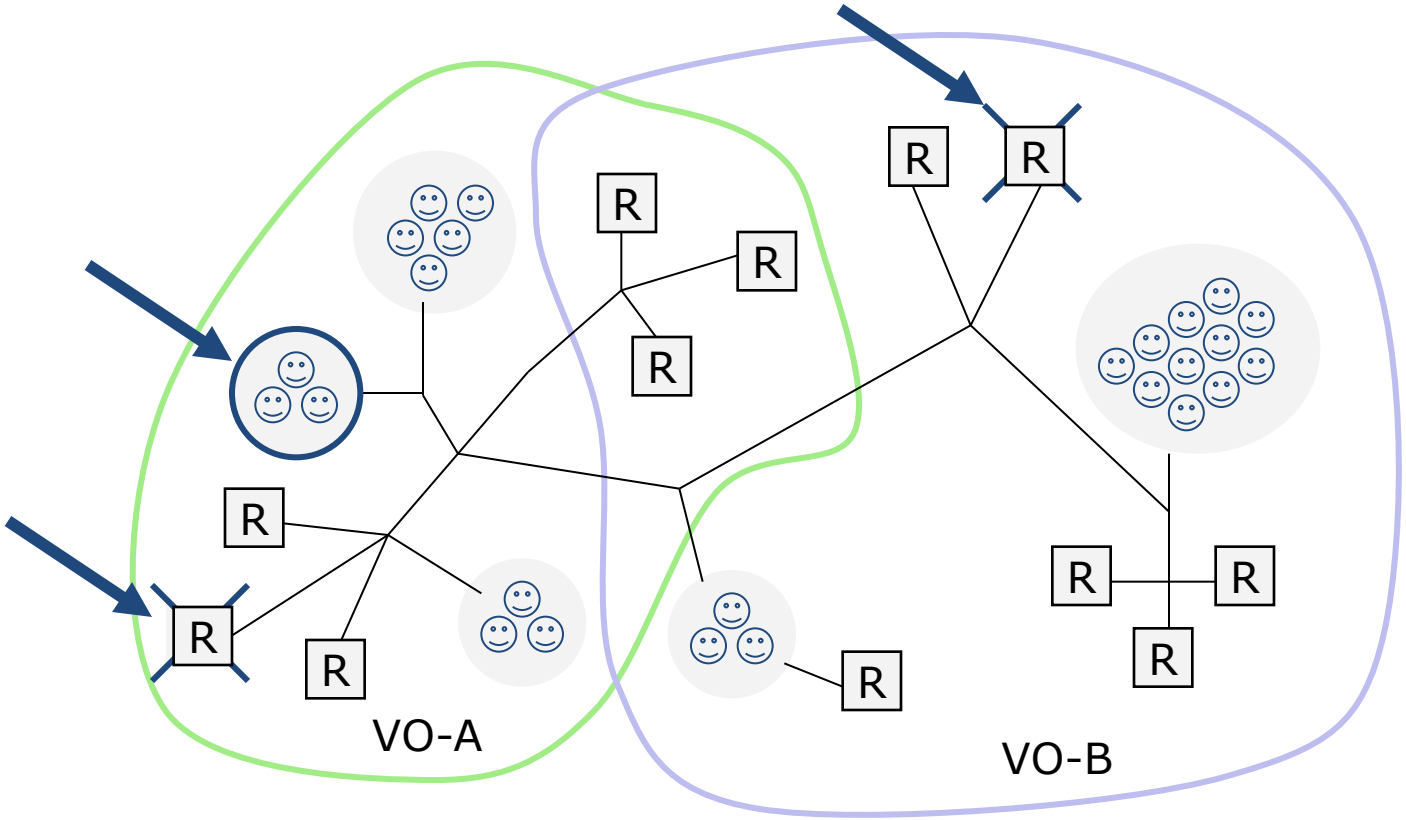
Characteristics	Cluster	Grid	P2P
Resource Management (i.e. memory, objects, storage, network access, etc)	Centralized	Distributed	Distributed
Resource Ownership	Singular (Often locked to a single node to prevent data corruption)	Singular or multiple, varies from platform to platform	Singular, multiple, or distributed, depending on circumstance and architecture
Method of Resource Allocation / Scheduling	Centralized, allocated according configuration	Decentralized	N/A, there is no single permanent host for centralized data or resource management. Everything is transient.
External Representation	Single Image	Single or multiple image(s)	Unknown, it is circumstantial
Inter-Operability	Guaranteed within a cluster	Enforced within a framework	Multiple competing standards
Suggested Equipments	Mostly high-end, high capability systems	High-end or commodity systems	Any type, including wireless device and embedded systems.
Scaling	2- 16 way (Although, theoretically 128+ is possible)	Two to thousands units connection	Theoretically, infinite (In actuality, it depends on network backbone transmission speed, number of clients, type of transmission protocol, etc.)
Discovery Mechanism	Defined membership (Static or Dynamic)	Centralized index, as well as, multiple decentralized mechanisms.	Always decentralized discovery mechanism.

Virtual Organization (VO)

- This sharing is, necessarily, highly controlled, with resource providers and consumers defining clearly and carefully just what is shared, who is allowed to share, and the conditions under which sharing occurs.
- Virtual organizations: from small departments within same physical location to large groups of people from different organizations that are spread across the globe.
- A set of individuals and/or institutions defined by such sharing rules form what we call a *virtual organization* (VO).

Virtual Organizations

- Distributed resources and people
- Linked by networks, crossing admin domains
- Sharing resources, common goals
- Dynamic



Source [1]

Examples of Virtual Organizations

- The application service providers, storage service providers, cycle providers
- Consultants engaged by a car manufacturer to perform scenario evaluation during planning for a new factory
- Members of an industrial consortium bidding on a new aircraft
- A crisis management team and the databases and simulation systems that they use to plan a response to an emergency situation
- and members of a large, international, multiyear high energy physics collaboration.
- Each of these examples represents an approach to computing and problem solving based on collaboration in computation- and data-rich environments.

Resource

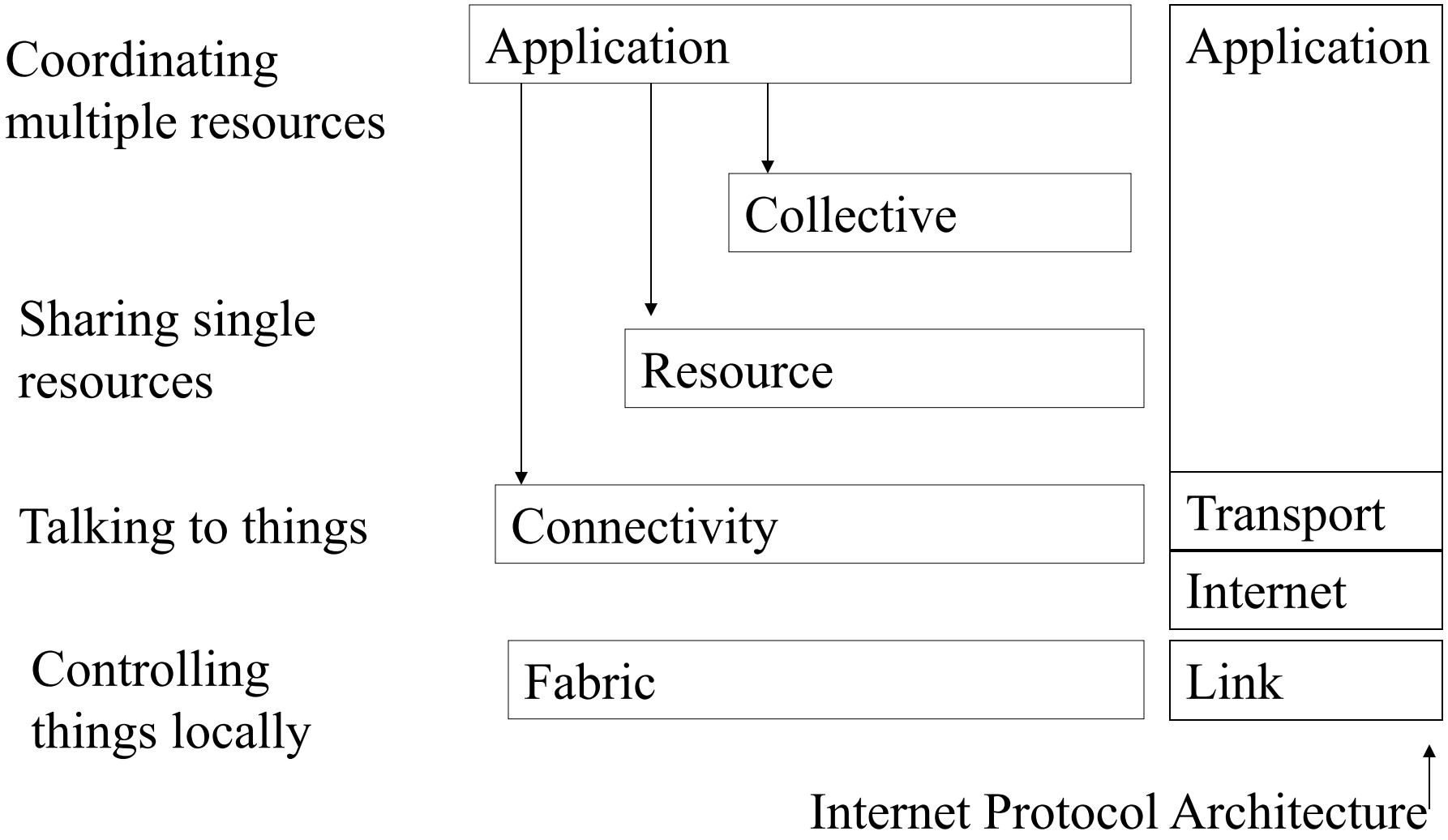
- A resource is an entity that is to be shared. It can be
  - computational such as PDA, laptop, workstation, server, cluster, and supercomputer or
  - a storage resource such as hard disk in a desktop,
  - Sensors,
  - Bandwidth,
  - Software etc.

Resource

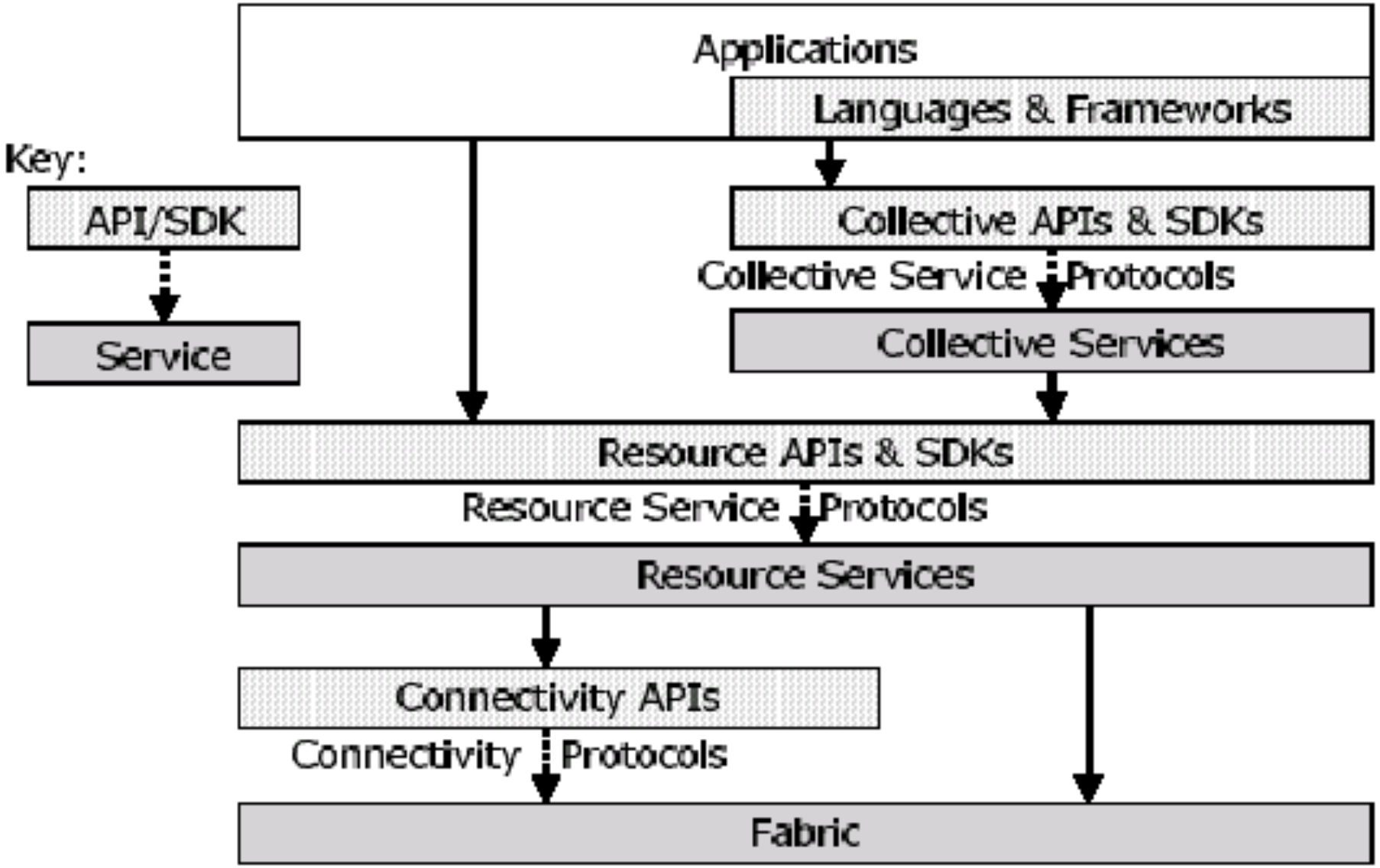
- Absence of central location and central control implies that grid resources do not require a particular central location for their management.
- Resources do not require a particular central location for their management.
- Resources do not have prior information about each other not do they have pre-defined security relationships.

Applications of Grid Computing

- Electronic Design Automation (EDA) for simulation, verification and regression testing for software development and hardware design
- Life Sciences for genetic sequencing and proteomics
- Financial services to perform risk analysis
- Scientific research
- Disaster Management
- Manufacturing
- Automotive and aeronautical (MCAE)
- Oil and gas: gather seismic mammoth amounts of seismic data that must be analyzed for the presence of strata that signal oil and gas deposits.
- Digital Content Creation for movie and TV industry



Grid computing architecture model


Applications
Languages & Frameworks
Key:
API/SDK
Service
Collective APIs & SDKs
Collective Service Protocols
Collective Services
Resource APIs & SDKs
Resource Service Protocols
Resource Services
Connectivity APIs
Connectivity Protocols
Fabric

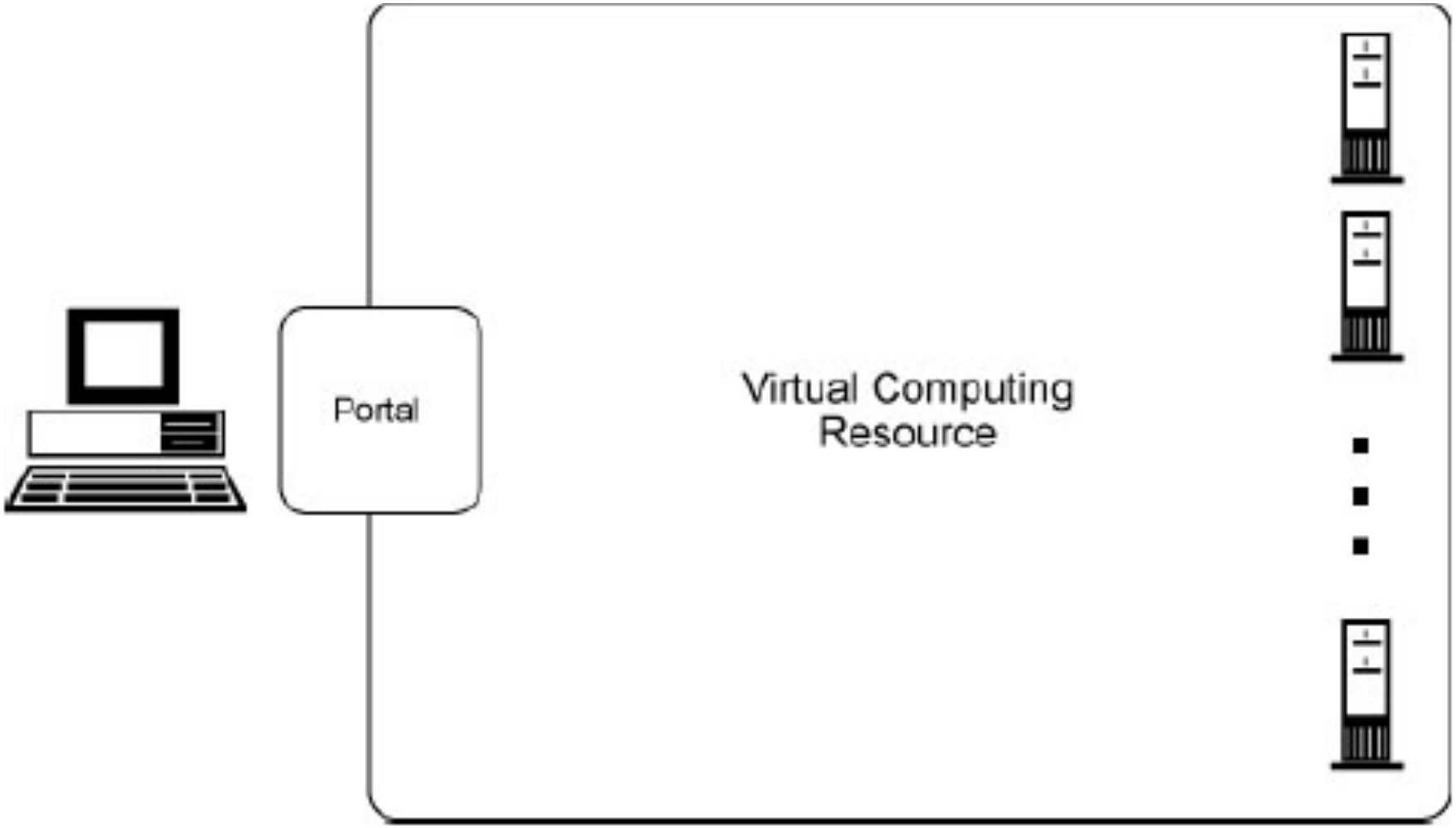
Collective (application-specific)	Solver coupler, distributed data archiver	Checkpointing, job management, failover, staging
Collective (generic)	Resource discovery, resource brokering, system monitoring, community authorization, certificate revocation	
Resource	Access to computation; access to data; access to information about system structure, state, performance.	
Connectivity	Communication (IP), service discovery (DNS), authentication, authorization, delegation	
Fabric	Storage systems, computers, networks, code repositories, catalogs	

Grid Computing: Major Components

- Security
- User interface
- Workload management
- Scheduler
- Data management
- Resource management

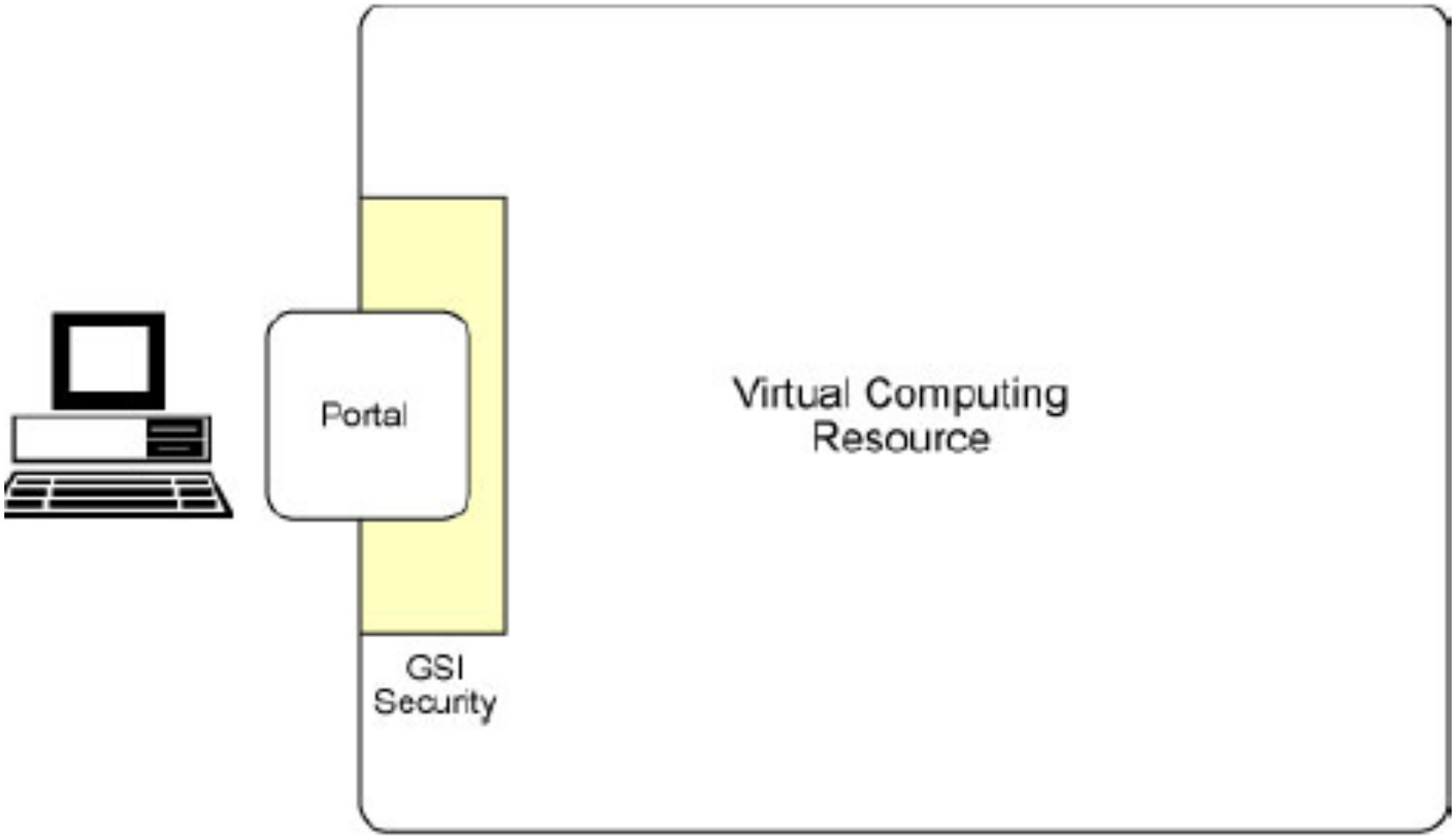
Portal/user interface

- A grid portal provides the interface for a user to launch applications that will use the resources and services provided by the grid.
- From this perspective, the user sees the grid as a virtual computing resource just as the consumer of power sees the receptacle as an interface to a virtual generator.
- The current Globus Toolkit does not provide any services or tools to generate a portal, but this can be accomplished with tools such as WebSphere Portal and WebSphere Application Server.


Portal
Virtual Computing
Resource

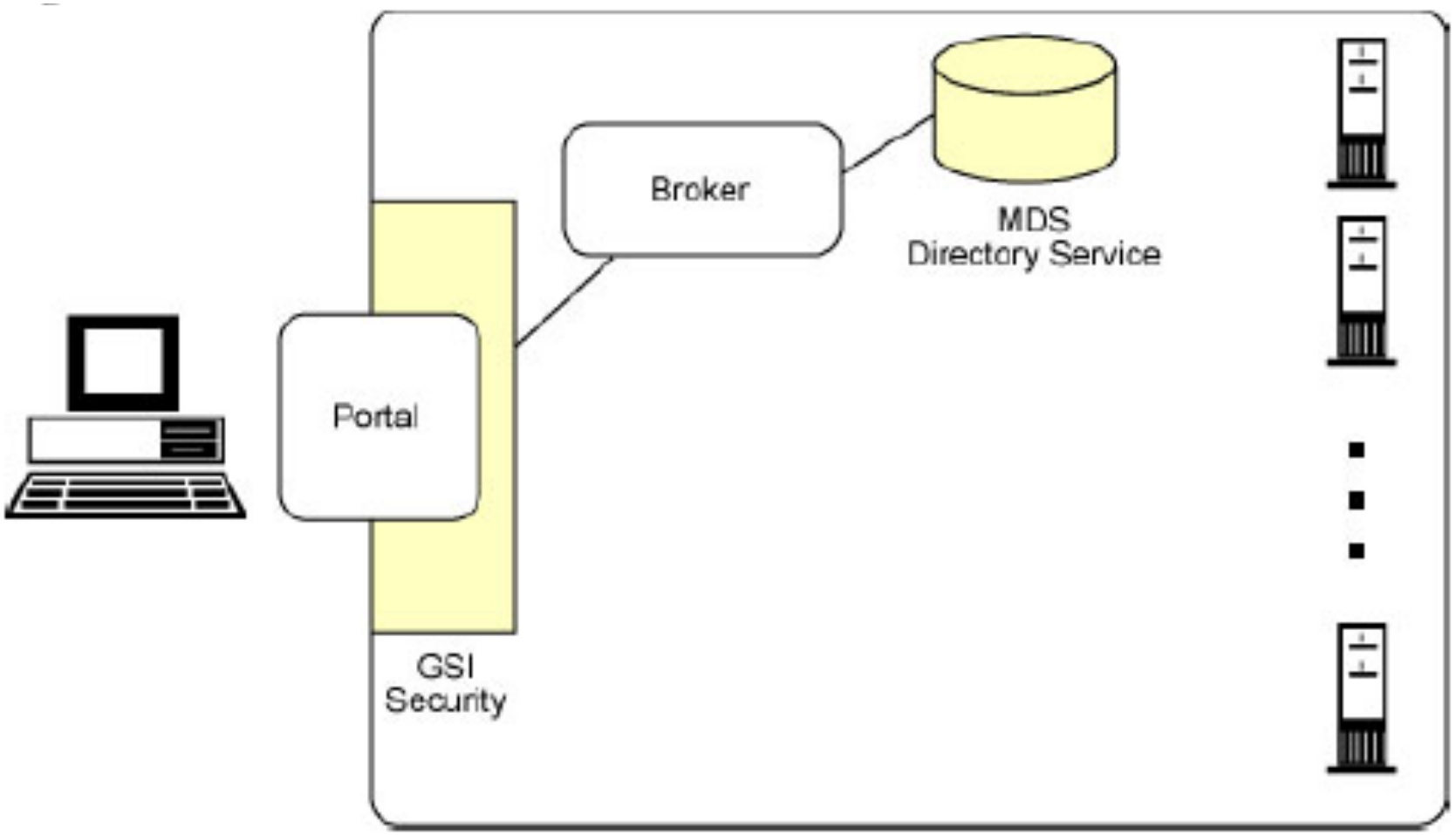
Security

- A major requirement for Grid computing is security. At the base of any grid environment, there must be mechanisms to provide security, including authentication, authorization, data encryption, and so on.
- The Grid Security Infrastructure (GSI) component of the Globus Toolkit provides robust security mechanisms.
- The GSI includes an OpenSSL implementation. It also provides a single sign-on mechanism, so that once a user is authenticated, a proxy certificate is created and used when performing actions within the grid.


Portal
Virtual Computing
Resource
GSI
Security

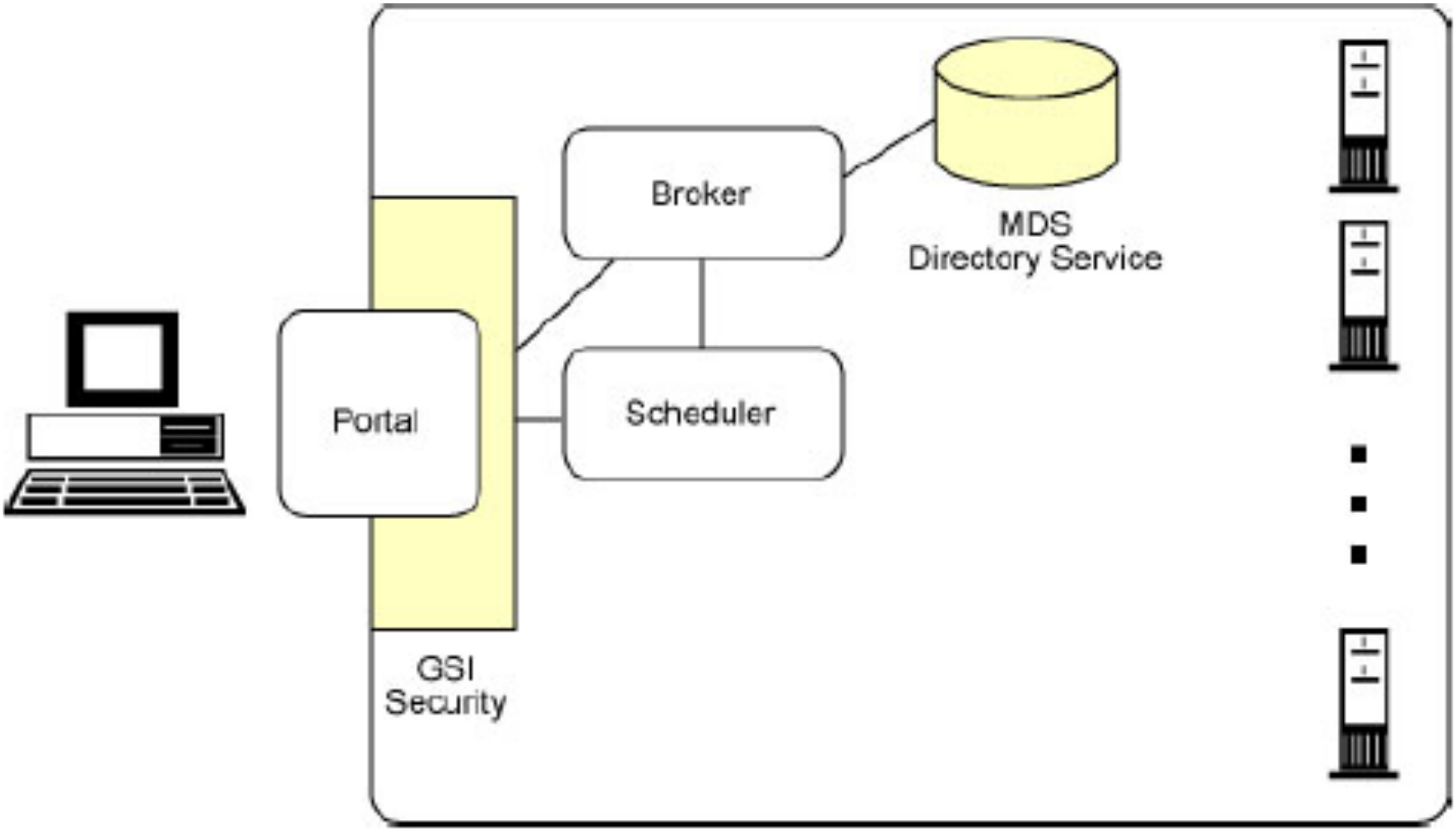
Broker

- Once authenticated, the user will be launching an application.
- Based on the application, and possibly on other parameters provided by the user, the next step is to identify the available and appropriate resources to use within the grid.
- This task could be carried out by a broker function. Although there is no broker implementation provided by Globus, there is an LDAP-based information service.
- This service is called the Grid Information Service (GIS), or more commonly the Monitoring and Discovery Service (MDS).
- This service provides information about the available resources within the grid and their status.
- A broker service could be developed that utilizes MDS.


Broker
MDS
Directory Service
Portal
GSI
Security

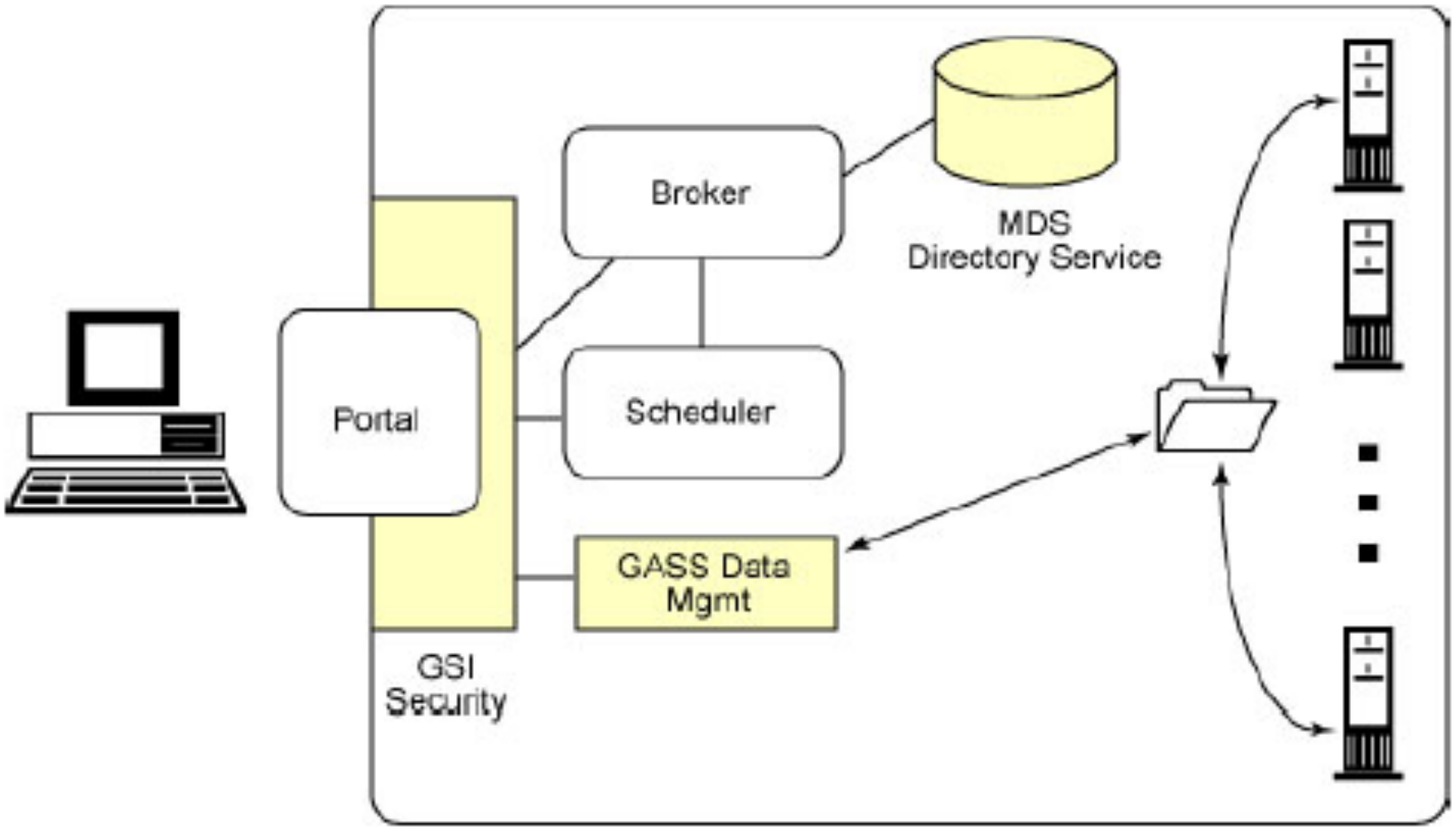
Scheduler

- Once the resources have been identified, the next logical step is to schedule the individual jobs to run on them.
- If a set of stand-alone jobs are to be executed with no interdependencies, then a specialized scheduler may not be required.
- However, if you want to reserve a specific resource or ensure that different jobs within the application run concurrently (for instance, if they require inter-process communication), then a job scheduler should be used to coordinate the execution of the jobs.


Broker
MDS
Directory Service
Portal
Scheduler
GSI
Security

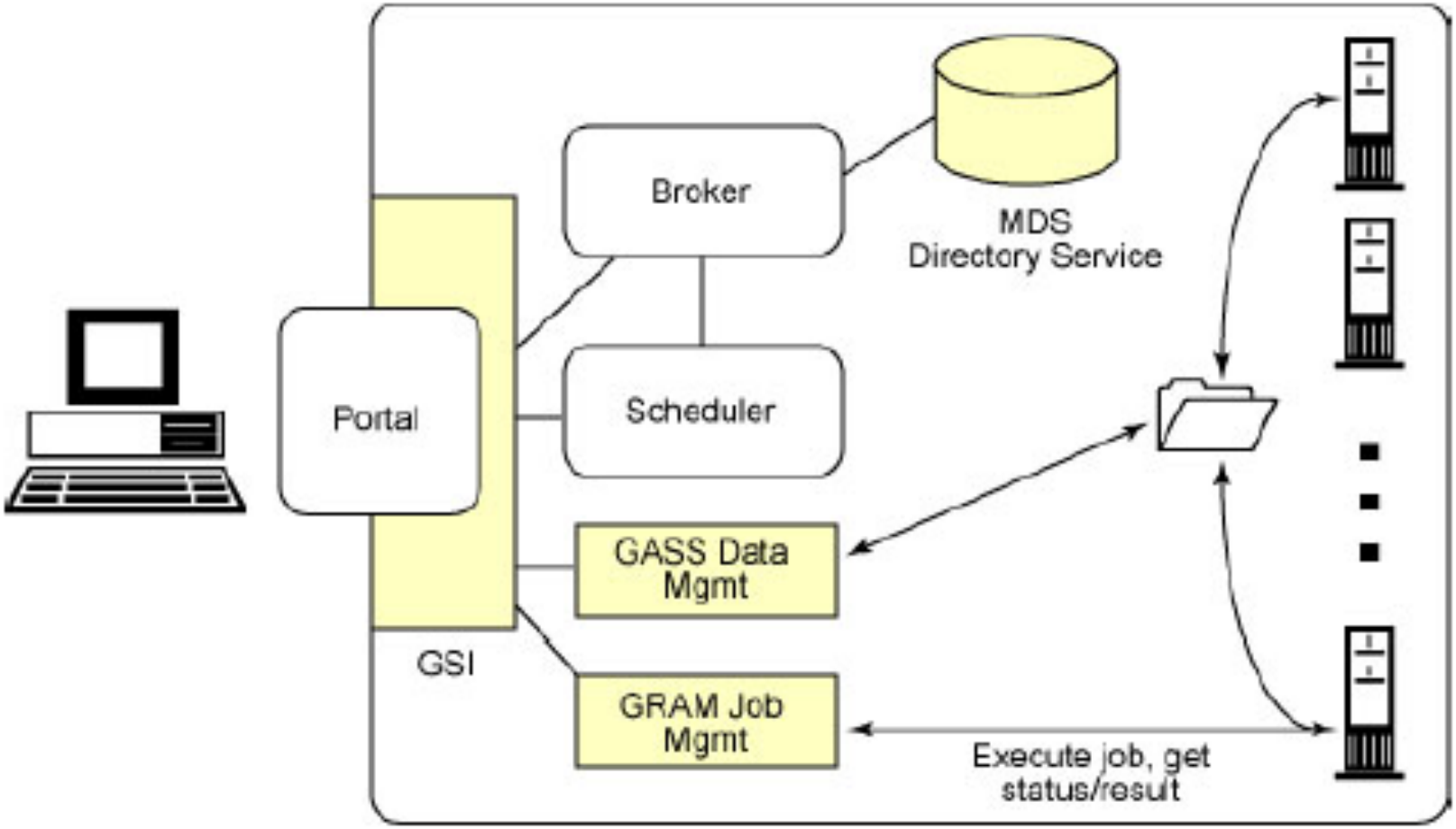
Data management

- If any data -- including application modules -- must be moved or made accessible to the nodes where an application's jobs will execute, then there needs to be a secure and reliable method for moving files and data to various nodes within the grid.
- The Globus Toolkit contains a data management component that provides such services.
- Grid Access to Secondary Storage (GASS), includes facilities such as GridFTP. GridFTP is built on top of the standard FTP protocol, but adds additional functions and utilizes the GSI for user authentication and authorization.

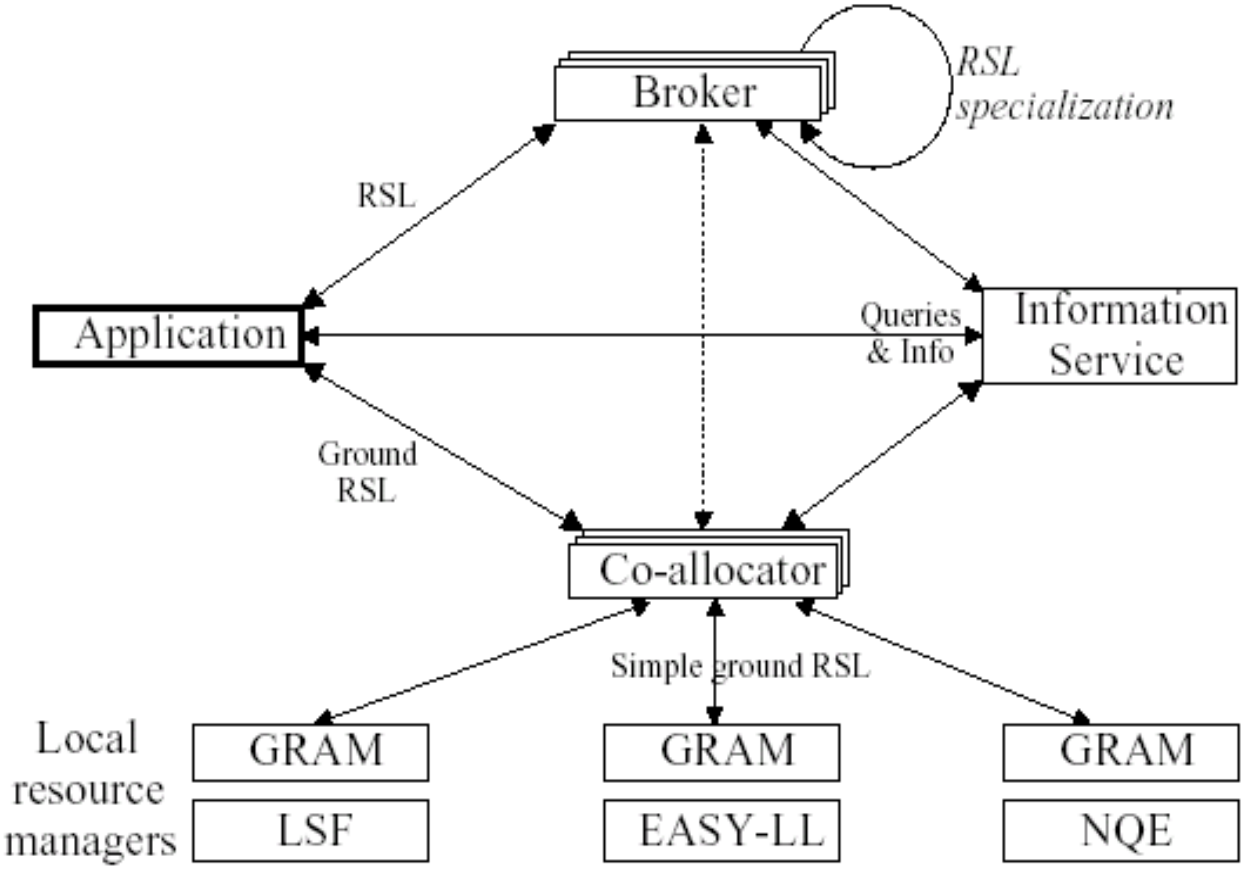

Broker
MDS
Directory Service
Portal
Scheduler
GASS Data
Mgmt
GSI
Security

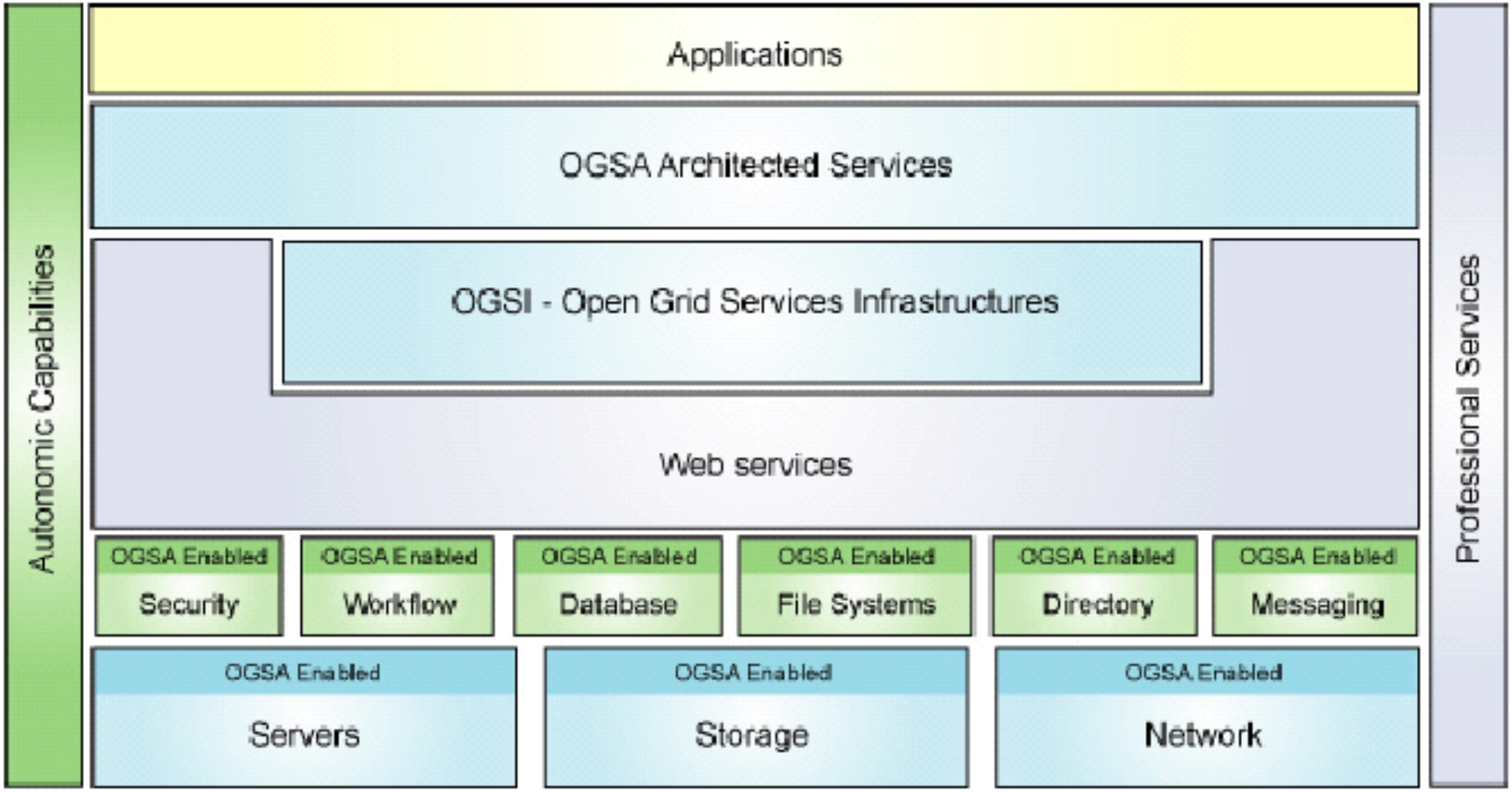
Job and resource management

- With all the other facilities we have just discussed in place, we now get to the core set of services that help perform actual work in a grid environment.
- The Grid Resource Allocation Manager (GRAM) provides the services to actually launch a job on a particular resource, check its status, and retrieve its results when it is complete.


Broker
MDS
Directory Service
Portal
Scheduler
GASS Data
Mgmt
GSI
GRAM Job
Mgmt
Execute job, get
status/result

Globus Resource Management Architecture




Autonomic Capabilities
Applications
OGSA Architected Services
OGSI - Open Grid Services Infrastructures
Web services
OGSA Enabled
Security
OGSA Enabled
Workflow
OGSA Enabled
Database
OGSA Enabled
File Systems
OGSA Enabled
Directory
OGSA Enabled
Messaging
OGSA Enabled
Servers
OGSA Enabled
Storage
OGSA Enabled
Network
Professional Services

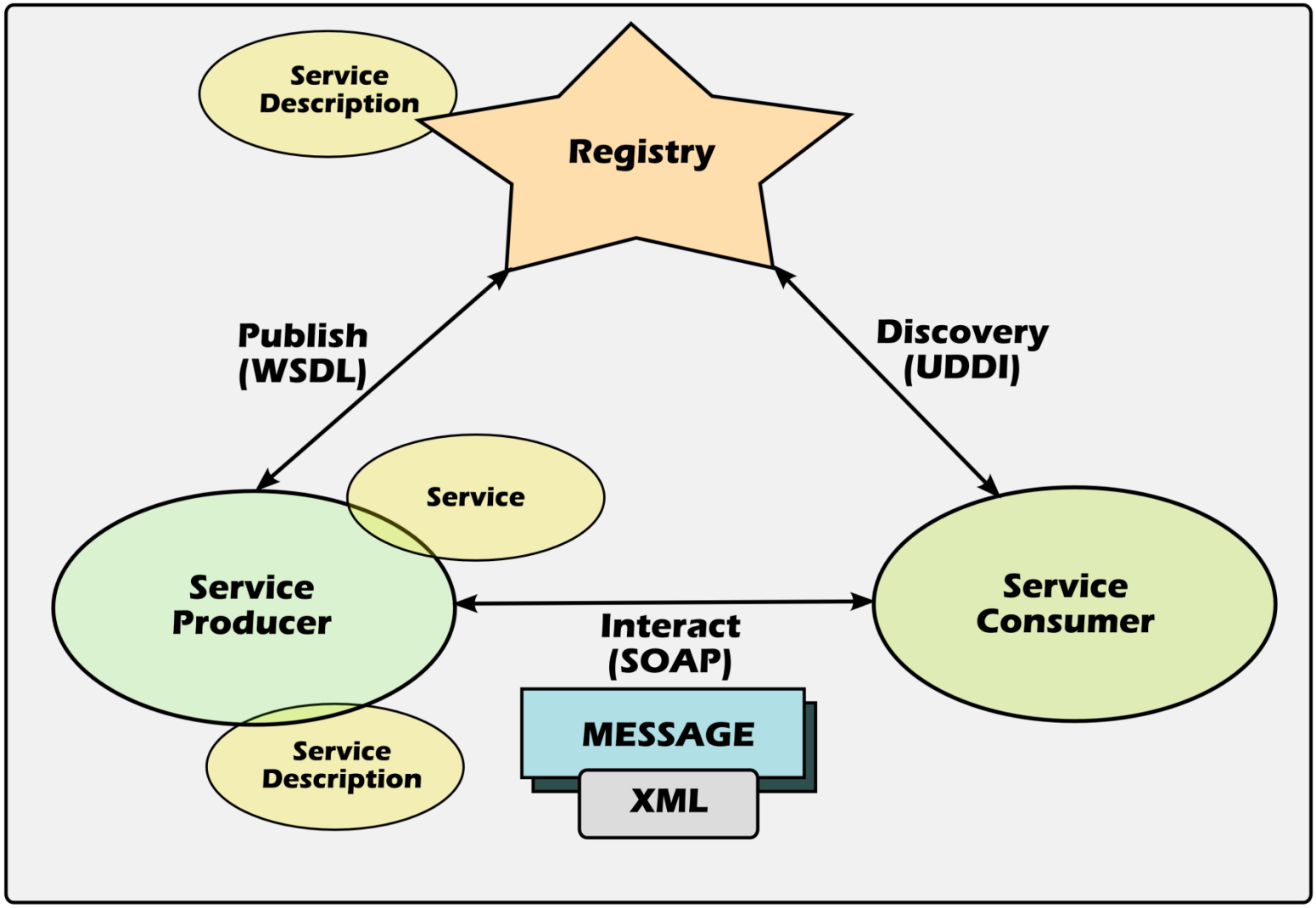
Service-Oriented Computing

Introduction

- Service-oriented architecture (SOA) is an approach to distributed computing:
  - that is loosely coupled,
  - that is protocol independent,
  - that is standards-based,
- where software is accessed as a service, where software resources interact over a network according to a contract
- *Contemporary SOA represents an open, extensible, federated, composable architecture that promotes service-orientation and is comprised of autonomous, QoS-capable, vendor diverse, interoperable, discoverable, and potentially reusable services, implemented as Web services.*

Defining SOA

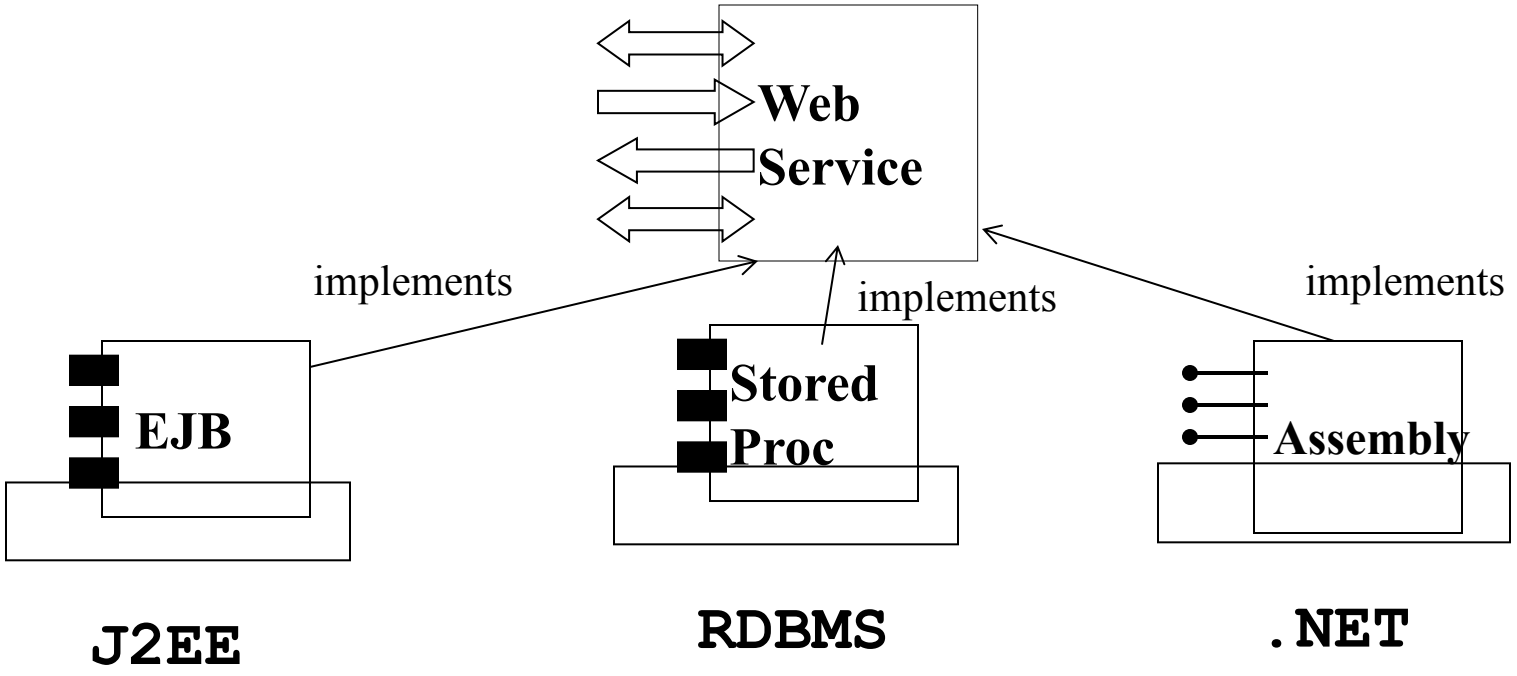
- Though accurate, this definition of contemporary SOA is quite detailed. For practical purposes, let's provide a supplementary definition that can be applied to both primitive and contemporary SOA.
- *SOA is a form of technology architecture that adheres to the principles of service-orientation. When realized through the Web services technology platform, SOA establishes the potential to support and promote these principles throughout the business process and automation domains of an enterprise.*

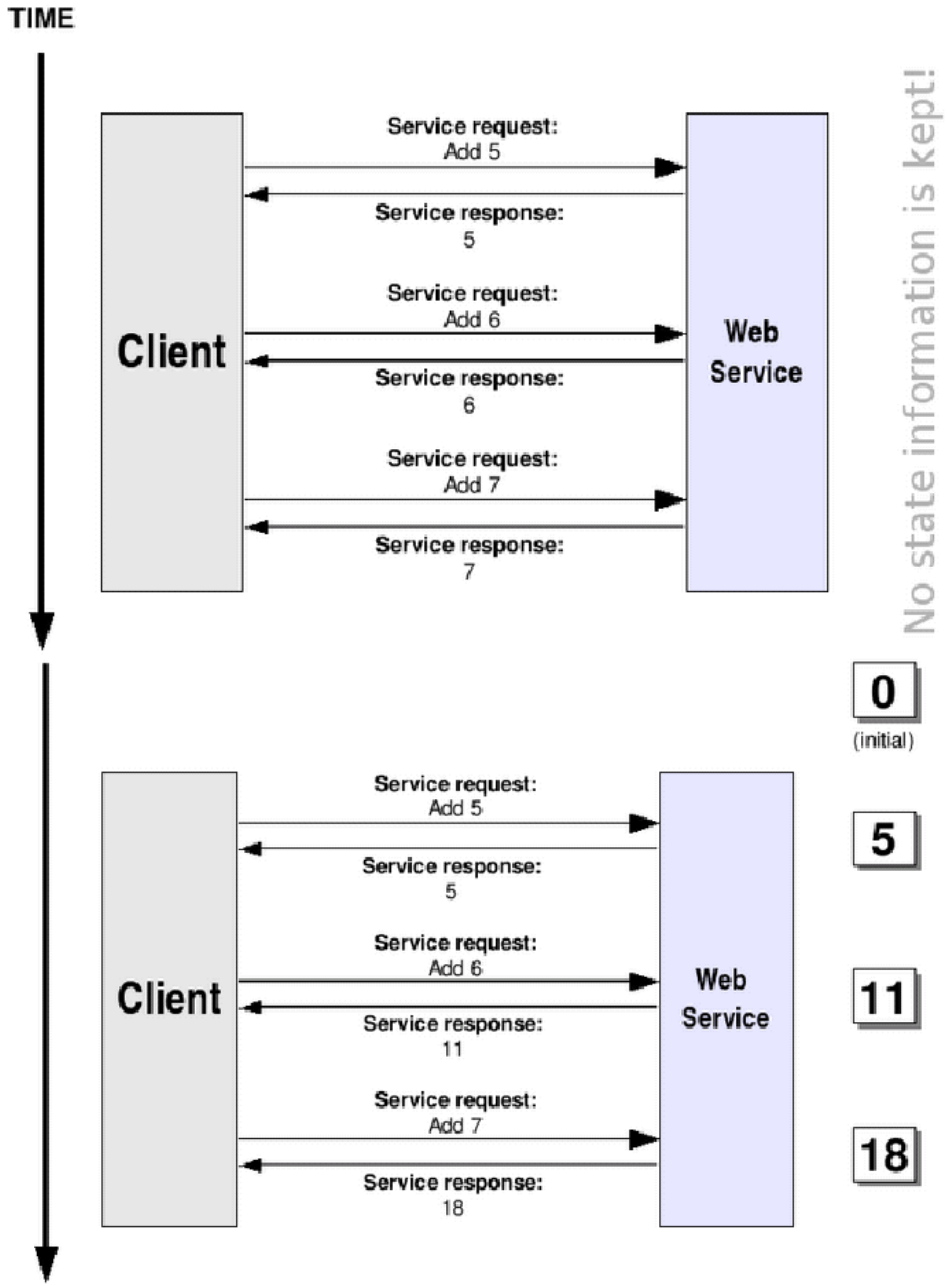


SOA Interaction Pattern

Definition of Web Services

- Web service is defined as “a standardized way of integrating web-based software components/applications using XML, SOAP,WSDL, and UDDI open standards over an Internet. XML is used to tag the data to support information interchange, SOAP is used as a communication protocol to transfer the data, WSDL is used for describing the service, and UDDI is used as a registry of published services.” [1]

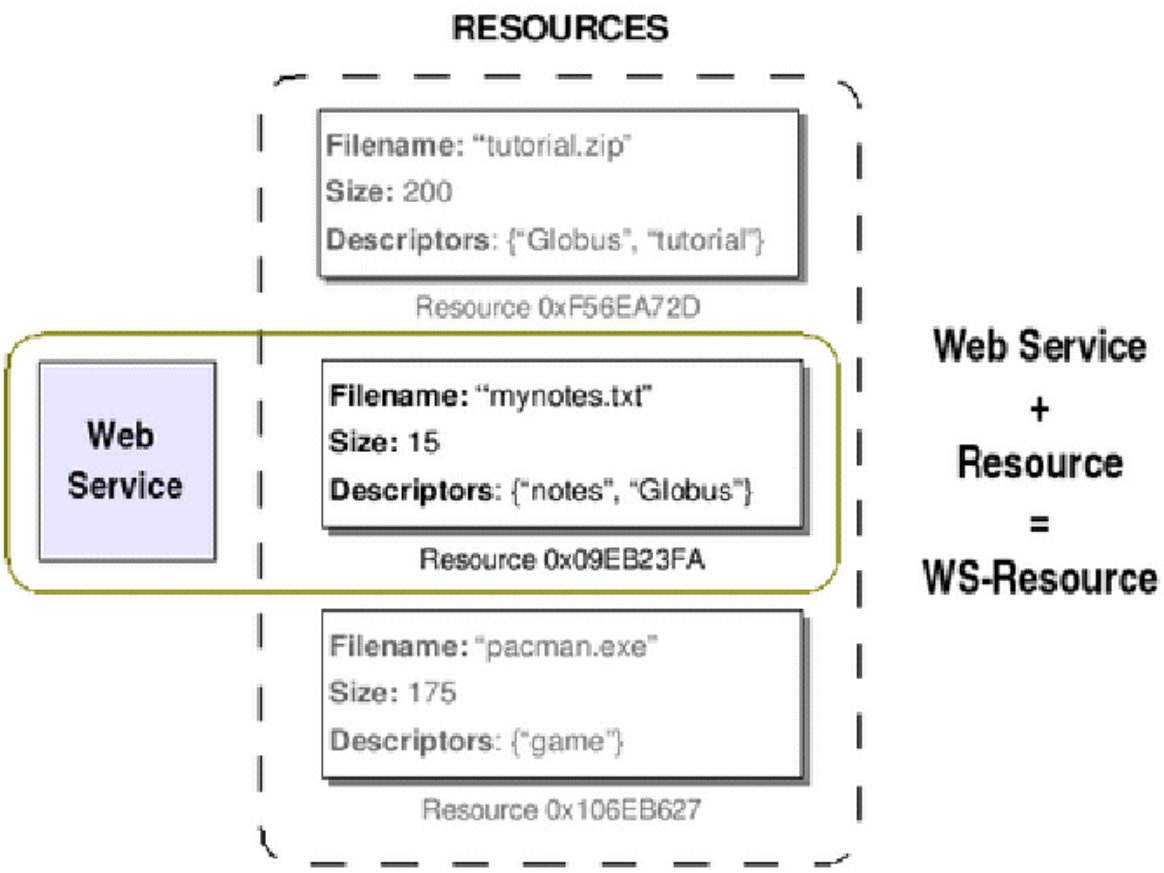




Stateless WS Invocation

Stateful WS Invocation

Source: [2]



Endpoint Reference

URI: http://www.example.com/myservice

Resource ID: 0x09EB23FA

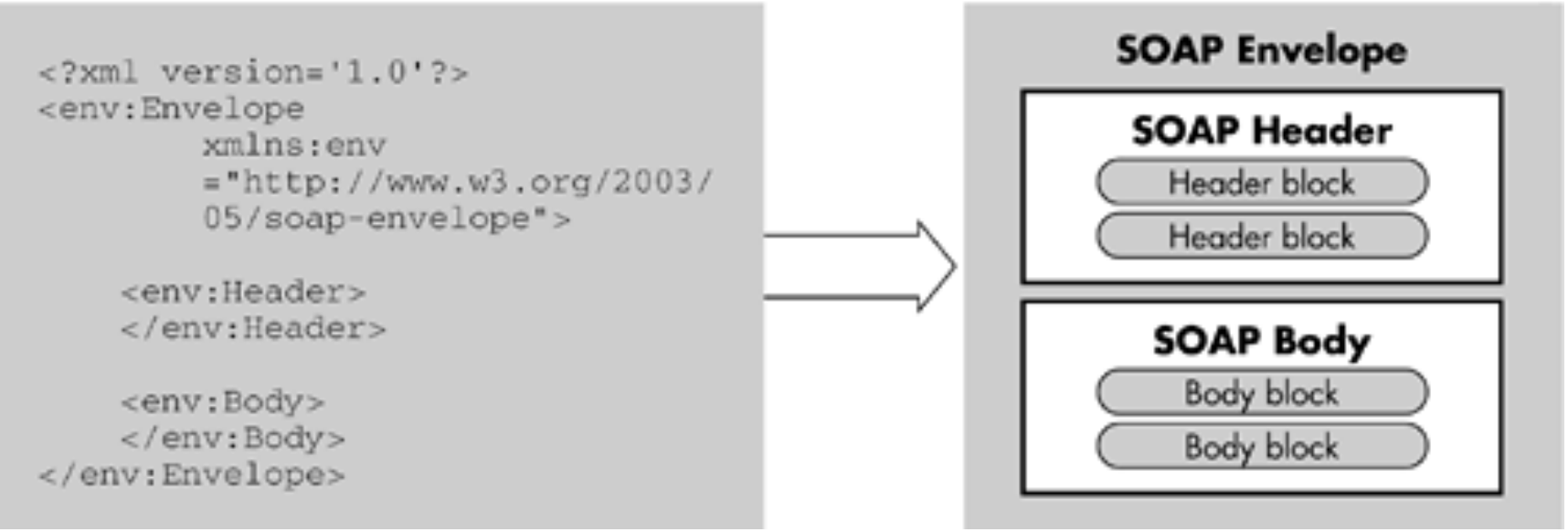
Pointer to a WS-Resource

Web Service

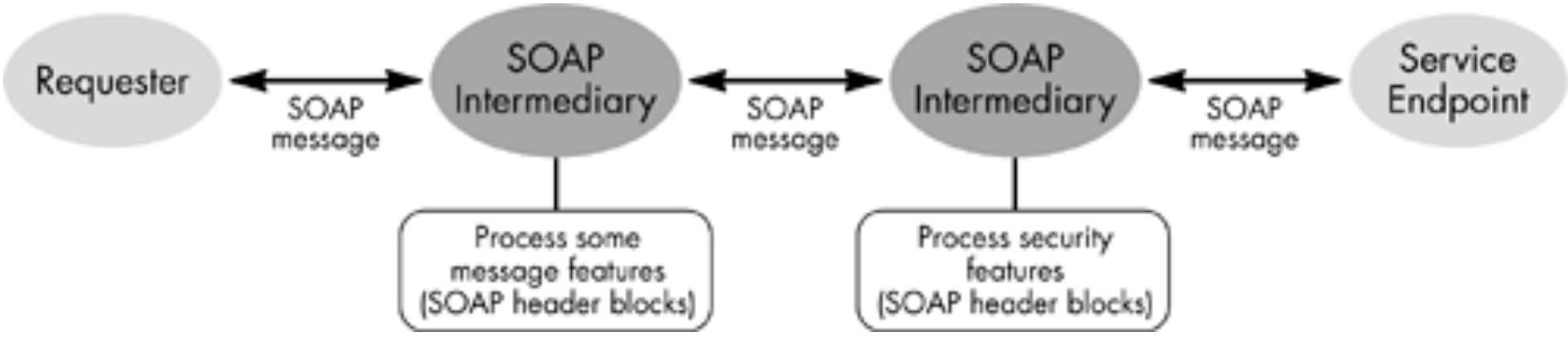
Filename:
Size:

Resource: 0x09EB23FA

```
<?xml version='1.0'?>
<env:Envelope
        xmlns:env
        ="http://www.w3.org/2003/
        05/soap-envelope">

    <env:Header>
    </env:Header>

    <env:Body>
    </env:Body>
</env:Envelope>
```



SOAP message formats



SOAP intermediaries

Envelope

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

Header

```
<soapenv:Header>
  <wsse:Security soapenv:mustUnderstand="1"
    xmlns:wsse="http://docs.oasis-open.org/oasis-wsswssecurity-secext-1.0.xsd">
      <wsse:UsernameToken>
        <wsse:Username>Karan</wsse:Username>
        <wsse:Password
            Type="http://docs.oasis-open.org/oasis-wssusername-token-profile-1.0#PasswordText">
             Karan
         </wsse:Password>
      </wsse:UsernameToken>
   </wsse:Security>
</soapenv:Header>
```

Body

```
<soapenv:Body>
   <customerCheck xmlns="http://CustomerLibrary/CustomerInterface">
     <CustomerID xmlns="">Karan</CustomerID>
     <Qty xmlns="">200</Qty>
     <Crop xmlns="">Rice</Crop>
   </customerCheck>
</soapenv:Body>
```

```
</soapenv:Envelope>
```

SOAP Envelope

WSDL

Abstract Part

```
<wsdl:definitions targetNamespace.....>
 <wsdl:types>  <schema elementFormDefault="qualified" targetNamespace= ...>
  <element name="calculatePrice">
   <complexType>     <sequence> <element name="order" type="xsd:string"/>
    </sequence>    </complexType>    </element>   <element name="calculatePriceResponse">
   <complexType>     <sequence>      <element name="calculatePriceReturn" type="xsd:float"/>
    </sequence>    </complexType>    </element>  </schema>  </wsdl:types>
  <wsdl:message name="calculatePriceResponse">
    <wsdl:part element="impl:calculatePriceResponse" name="parameters"/>   </wsdl:message>
  <wsdl:message name="calculatePriceRequest">
    <wsdl:part element="impl:calculatePrice" name="parameters"/>   </wsdl:message>
  <wsdl:portType name="CalculatePrice">     <wsdl:operation name="calculatePrice">
      <wsdl:input message="impl:calculatePriceRequest" name="calculatePriceRequest"/>
      <wsdl:output message="impl:calculatePriceResponse" name="calculatePriceResponse"/>
    </wsdl:operation>    </wsdl:portType>
```

Concrete Part

```
<wsdl:binding name="CalculatePriceSoapBinding" type="impl:CalculatePrice">
    <wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="calculatePrice">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="calculatePriceRequest">
        <wsdlsoap:body use="literal"/>        </wsdl:input>
      <wsdl:output name="calculatePriceResponse">
        <wsdlsoap:body use="literal"/>        </wsdl:output>
    </wsdl:operation>    </wsdl:binding>
  <wsdl:service name="CalculatePriceService">
    <wsdl:port binding="impl:CalculatePriceSoapBinding" name="CalculatePrice">
      <wsdlsoap:address location="http://localhost:8080/FirstWS/services/CalculatePrice"/>
    </wsdl:port>    </wsdl:service> </wsdl:definitions>
```

```
<?xml version="1.0"?>
<definitions name="Procurement"
   targetNamespace="http://example.com/procurement/definitions"
   xmlns:tns="http://example.com/procurement/definitions"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns="http://schemas.xmlsoap.org/wsdl/" >

   <message name="OrderMsg">
      <part name="productName" type="xs:string"/>
      <part name="quantity" type="xs:integer"/>
   </message>

   <portType name="procurementPortType">
      <operation  name="orderGoods">
         <input message = "OrderMsg"/>
      </operation>
   </portType>

   <binding name="ProcurementSoapBinding" type="tns:procurementPortType">
      <soap:binding style="document"
            transport="http://schemas.xmlsoap.org/soap/http"/>
      <operation name="orderGoods">
         <soap:operation soapAction="http://example.com/orderGoods"/>
         <input>
            <soap:body use="literal"/>
         </input>
         <output>
            <soap:body use="literal"/>
         </output>
      </operation>
   </binding>

   <service name="ProcurementService">
      <port name="ProcurementPort" binding="tns:ProcurementSoapBinding">
         <soap:address location="http://example.com/procurement"/>
      </port>
   </service>

</definitions>
```

abstract part

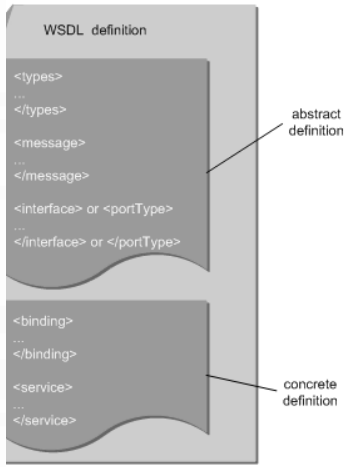
messages

operation and port type

concrete part

binding

port and service



WSDL Structure

WSDL

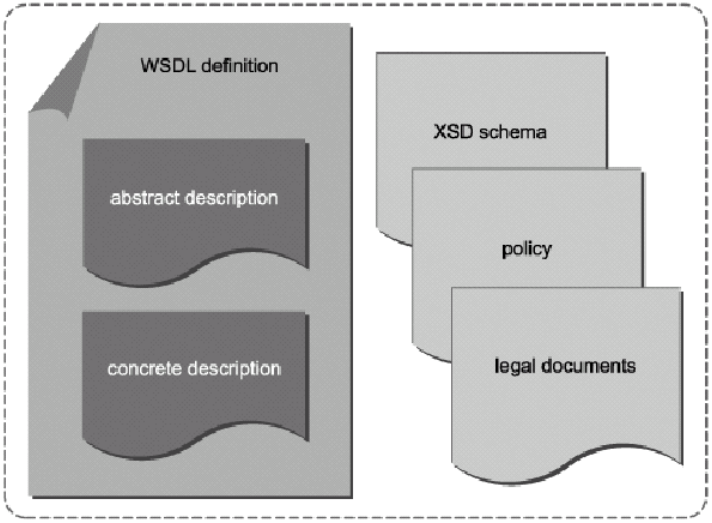
Abstract Part

```
<wsdl:definitions targetNamespace.....>
 <wsdl:types>  <schema elementFormDefault="qualified" targetNamespace= ...>
  <element name="calculatePrice">
   <complexType>     <sequence> <element name="order" type="xsd:string"/>
    </sequence>    </complexType>    </element>    <element name="calculatePriceResponse">
   <complexType>     <sequence>      <element name="calculatePriceReturn" type="xsd:float"/>
    </sequence>    </complexType>    </element>  </schema>  </wsdl:types>
  <wsdl:message name="calculatePriceResponse">
    <wsdl:part element="impl:calculatePriceResponse" name="parameters"/>    </wsdl:message>
  <wsdl:message name="calculatePriceRequest">
    <wsdl:part element="impl:calculatePrice" name="parameters"/>    </wsdl:message>
  <wsdl:portType name="CalculatePrice">     <wsdl:operation name="calculatePrice">
      <wsdl:input message="impl:calculatePriceRequest" name="calculatePriceRequest"/>
      <wsdl:output message="impl:calculatePriceResponse" name="calculatePriceResponse"/>
    </wsdl:operation>    </wsdl:portType>
```

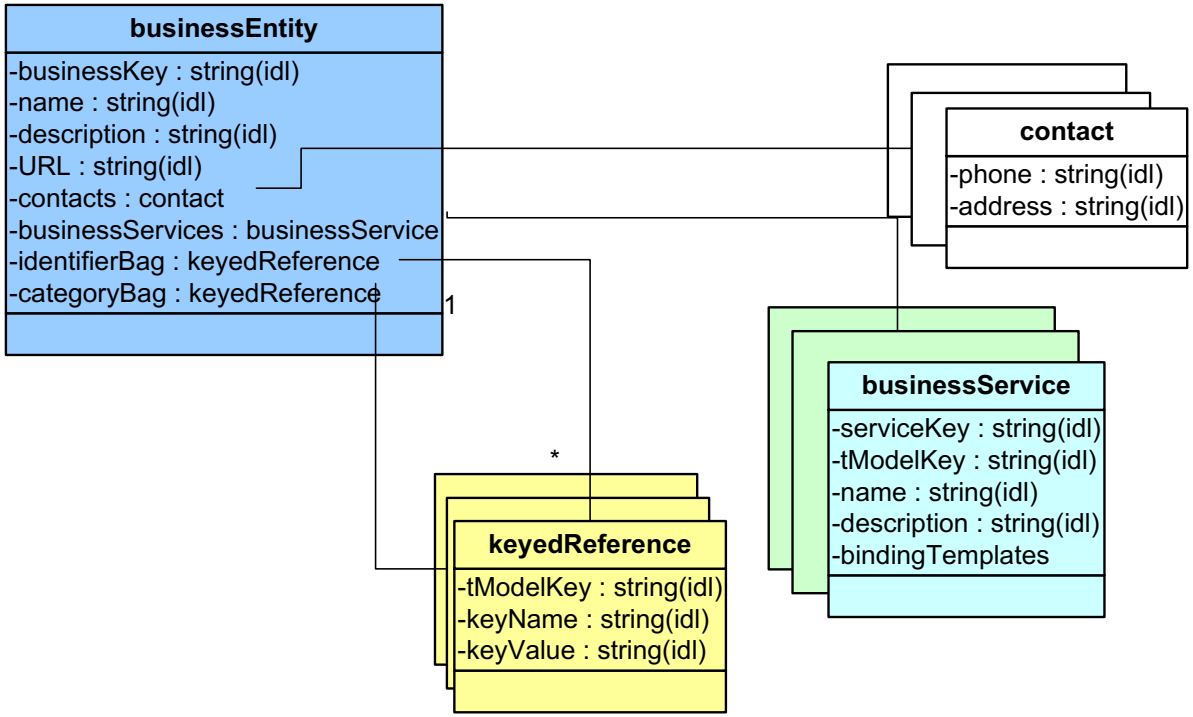
Concrete Part

```
<wsdl:binding name="CalculatePriceSoapBinding" type="impl:CalculatePrice">
   <wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
   <wsdl:operation name="calculatePrice">
     <wsdlsoap:operation soapAction=""/>
     <wsdl:input name="calculatePriceRequest">
       <wsdlsoap:body use="literal"/>         </wsdl:input>
     <wsdl:output name="calculatePriceResponse">
       <wsdlsoap:body use="literal"/>         </wsdl:output>
   </wsdl:operation>    </wsdl:binding>
 <wsdl:service name="CalculatePriceService">
   <wsdl:port binding="impl:CalculatePriceSoapBinding" name="CalculatePrice">
     <wsdlsoap:address location="http://localhost:8080/FirstWS/services/CalculatePrice"/>
   </wsdl:port>    </wsdl:service> </wsdl:definitions>
```

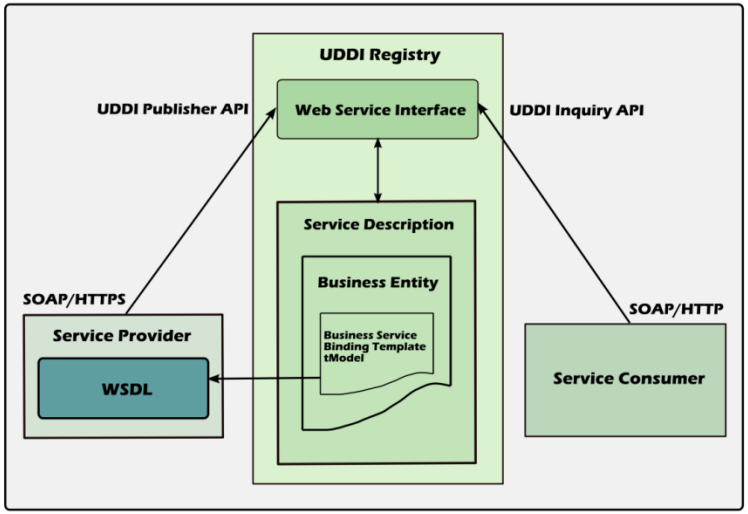
Sample WSDL document



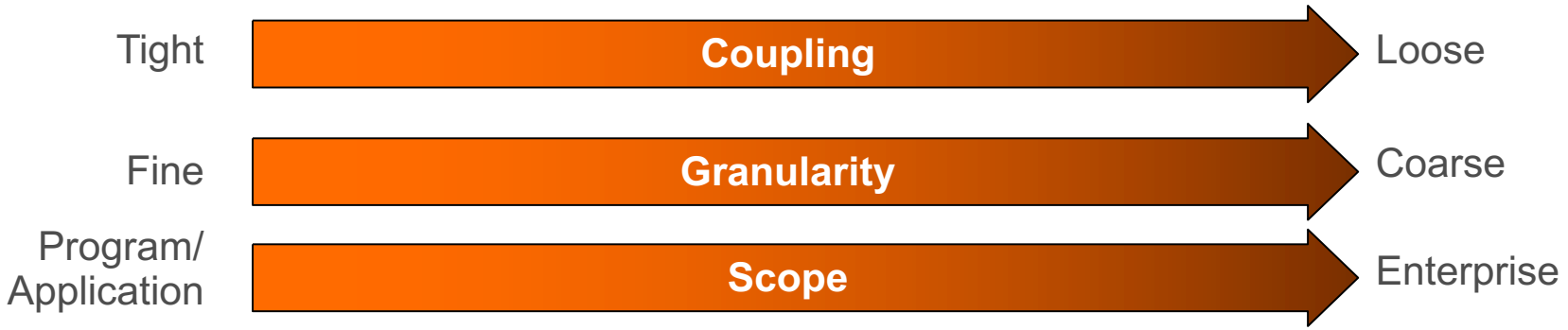
UDDI



SOAP, WSDL and UDDI Interaction



SOA – The next in architectural evolution



Object Oriented
(for e.g. Java Objects)

Component based
(for e.g. EJB's & .NET remoting)

Service Oriented
(for e.g. web services)

Objects

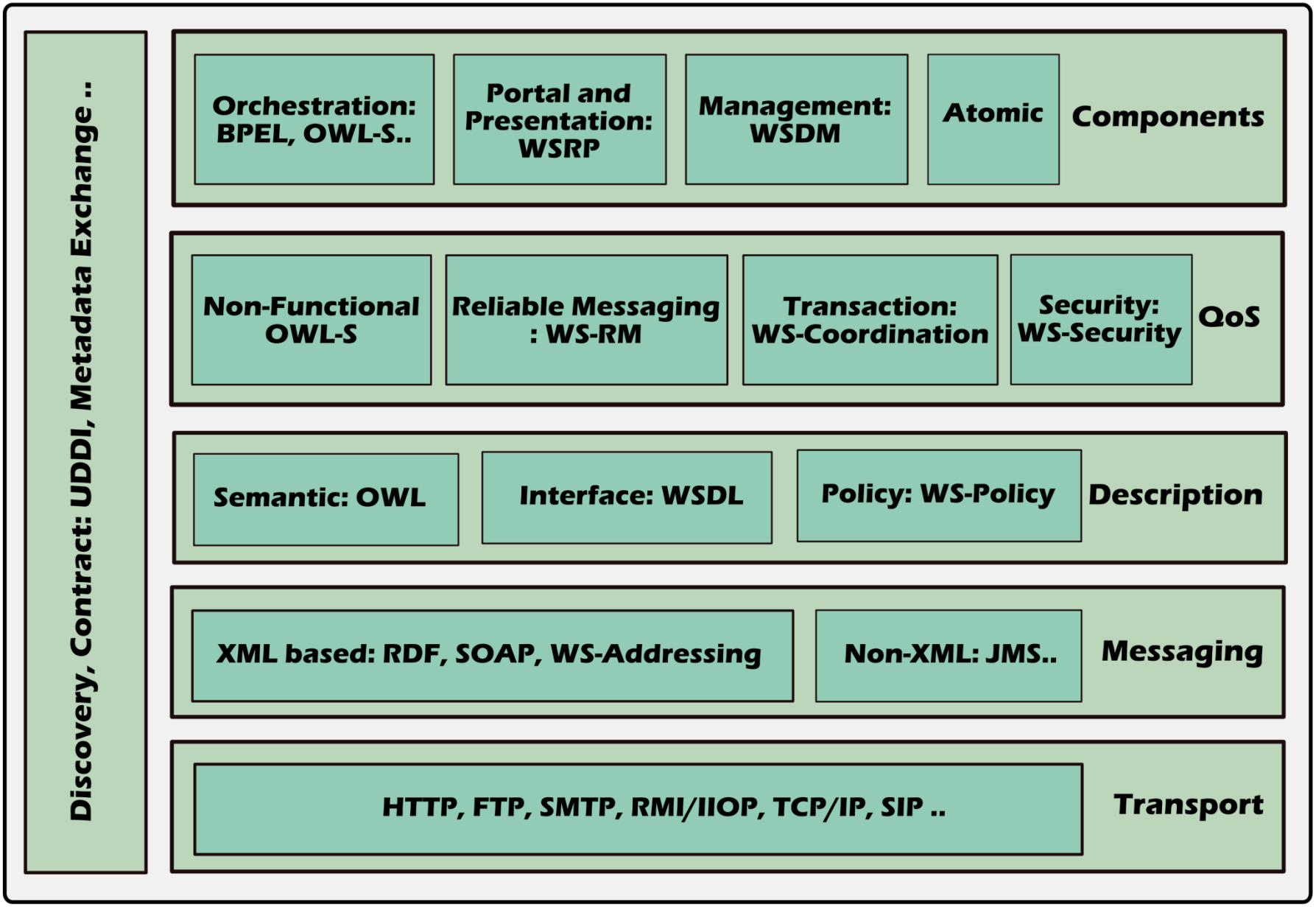
- Objects implement fine grained functionality
- Objects are accessed in memory
- Base unit – Objects

Components

- Defines well defined functionality from a set of objects
- Components are accessed via distributed middleware (RMI, .NET remoting)
- Base unit – Classes, jars, wars

Services

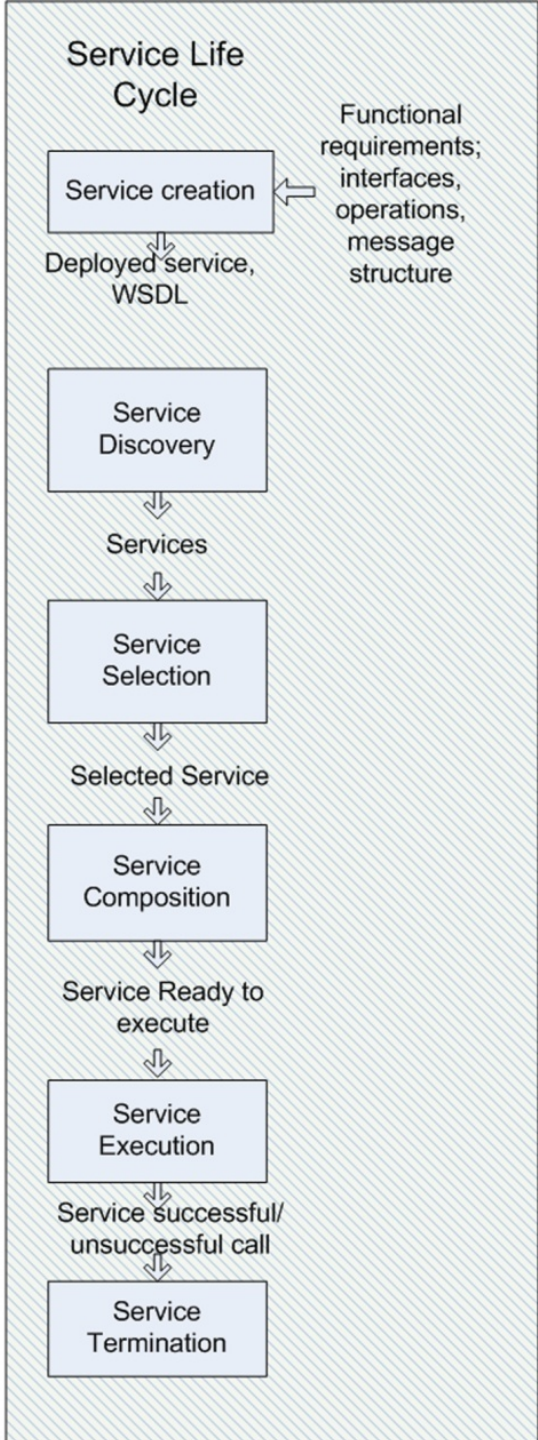
- Represents a higher-lever business function
- Predominantly asynchronous and stateless
- Services are accessed via the network (HTTP etc.)
- Base unit - schemas/interfaces



WS-* standards and specifications stack

Service Life Cycle

- Service Creation
  - Service Design and Build
  - Service Deployment
  - Publish Web service using UDDI
- Service Discovery
- Service Selection
- Service Composition
- Service Execution and Monitoring
- Service Termination

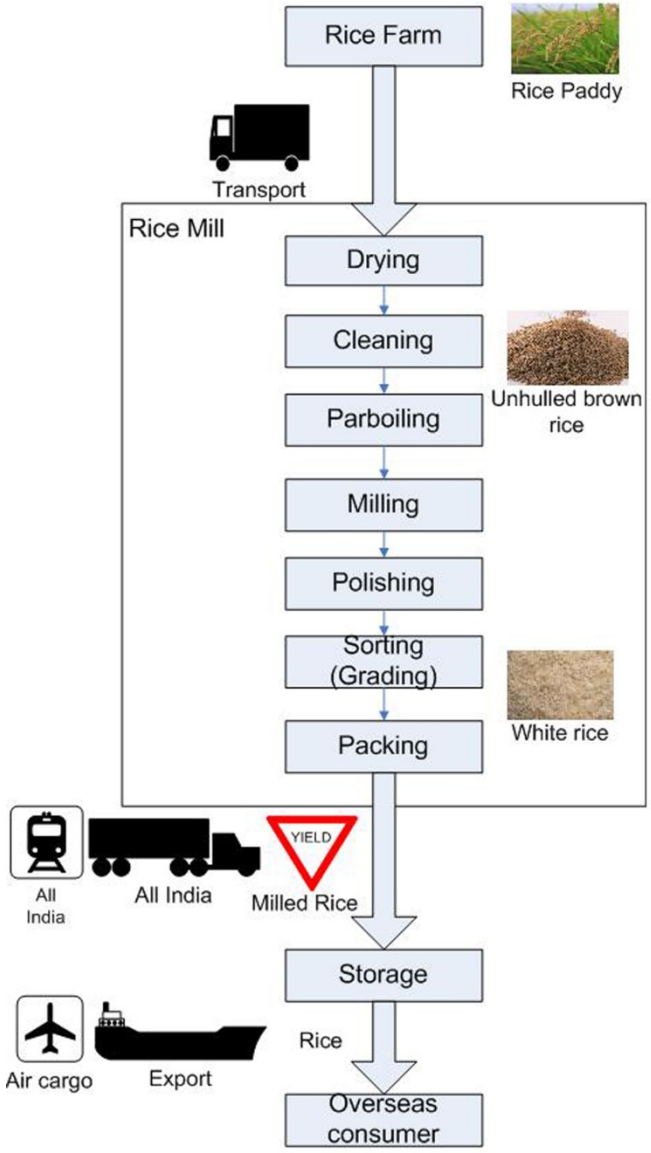


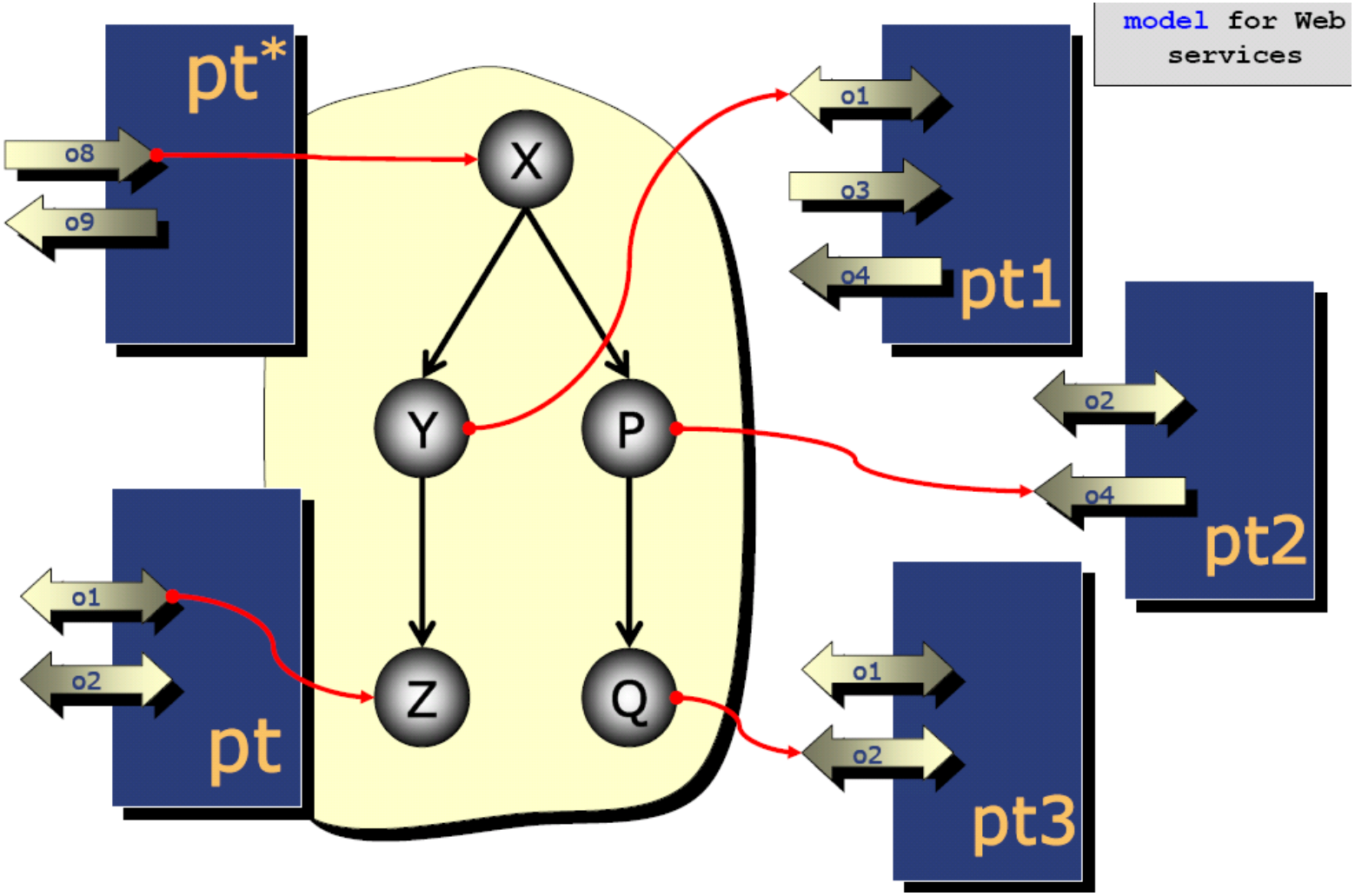
Composite Service

- If the implementation of a web service's logic involves the invocation of other web services, it is necessary to combine the functionality of several web services. It is referred as a composite service [1].
- The process of developing a composite service is called service composition.
- Service composition can be either performed by composing elementary or composite services.
- This is analogous to workflow management, where the application logic is realized by composing autonomous applications.

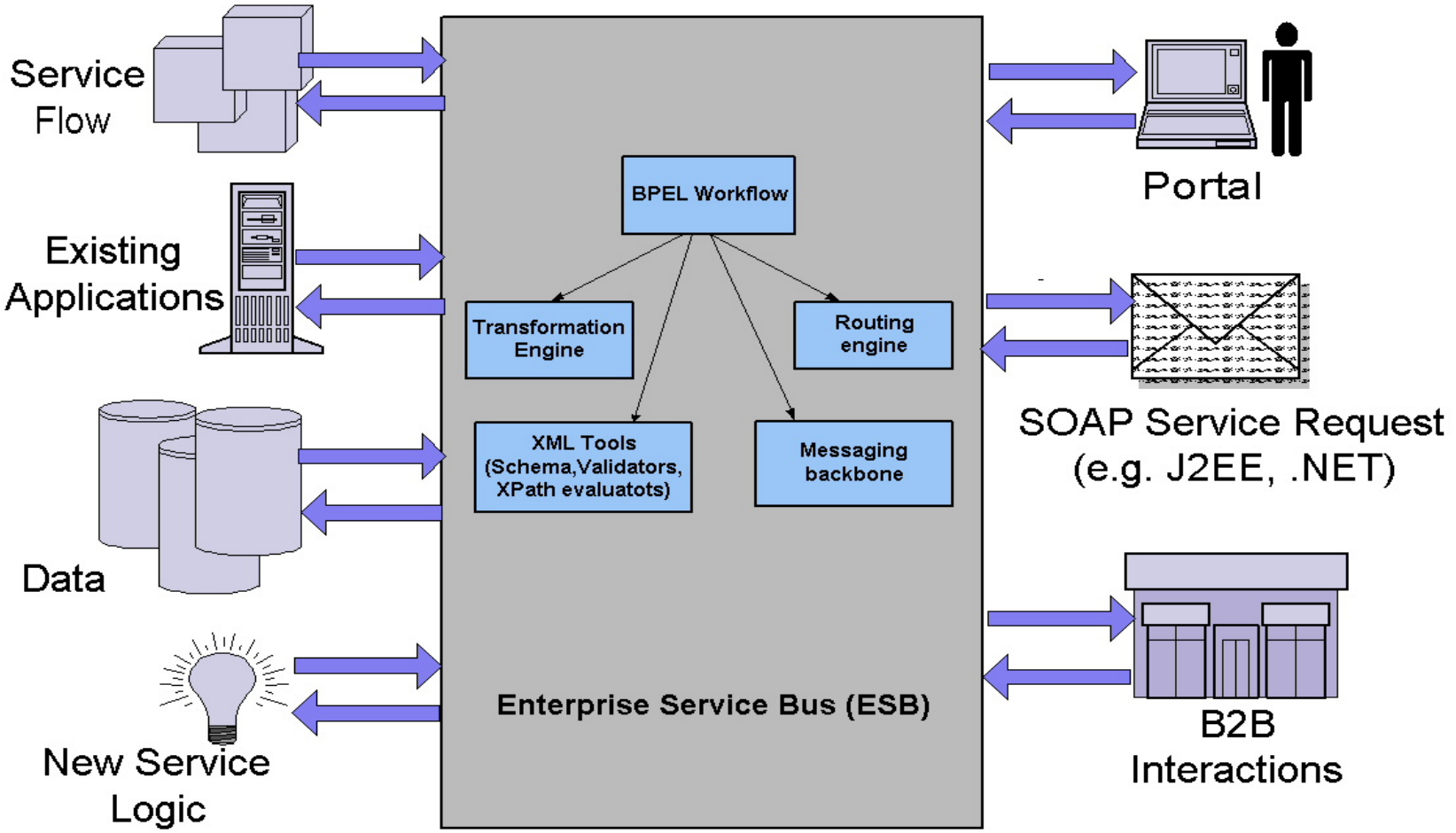
Rice Procurement Process

"To purchase quantity of rice (paddy) directly from the farmers (contract farming) and process rice at rice mill(s)"




model for Web services
pt*
o8
o9
X
Y
P
Z
Q
o1
o3
o4
pt1
o2
o4
pt2
o1
o2
pt
o1
o2
pt3

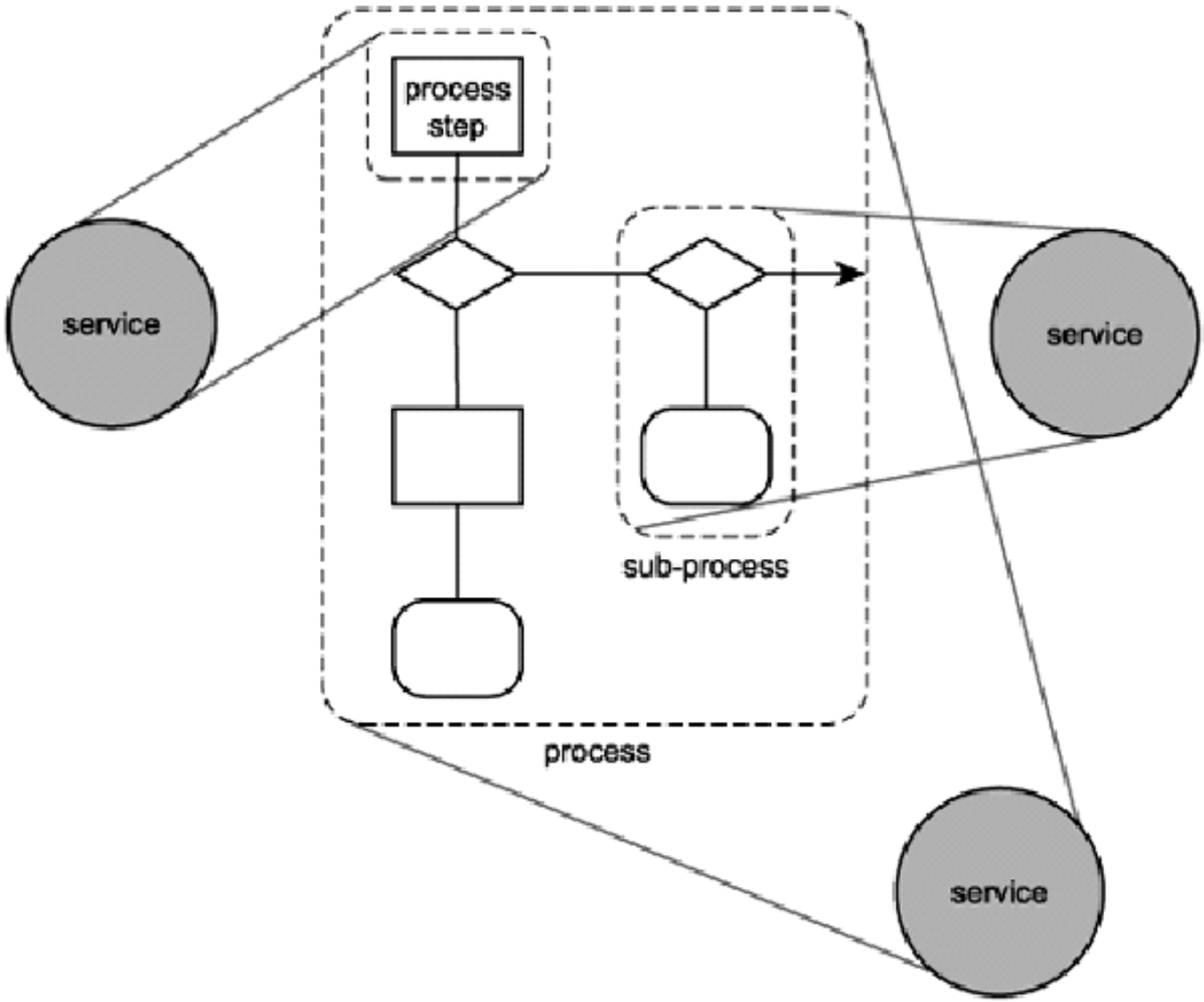
SOA: An EAI Perspective realized through ESB



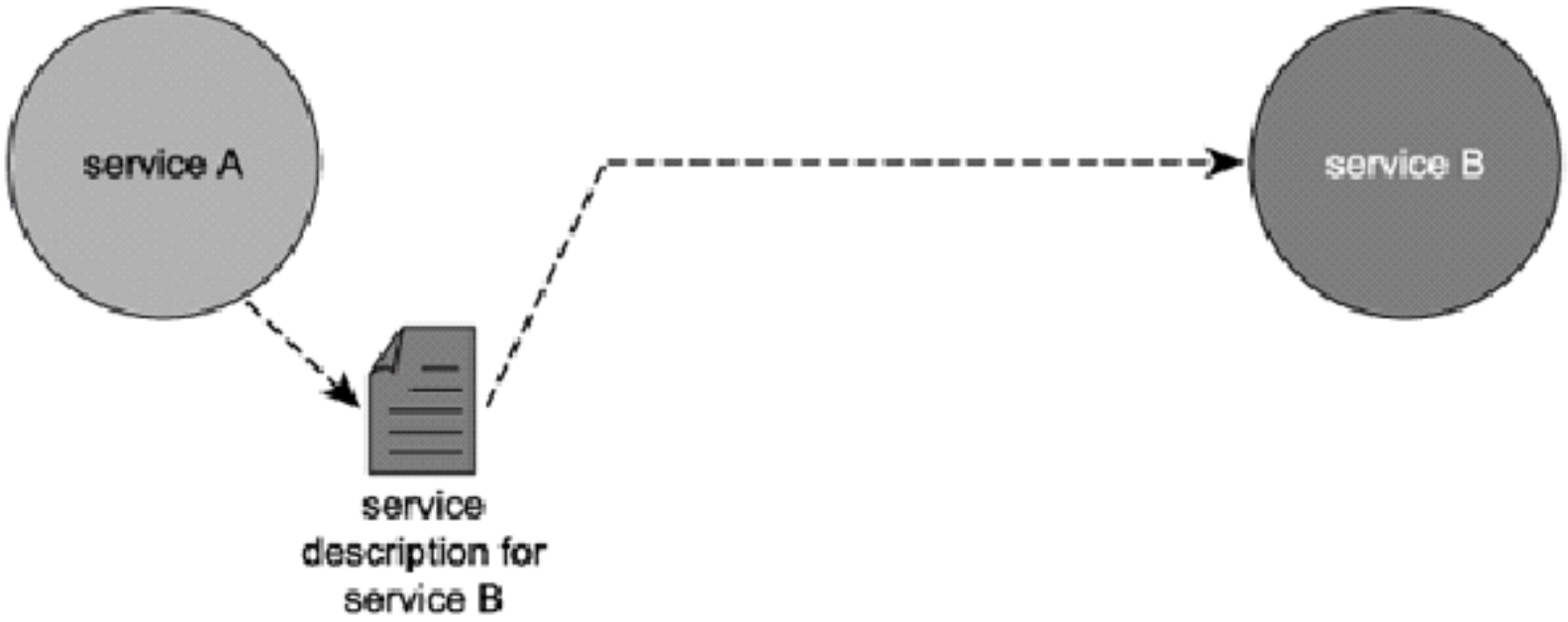
Myths about SOA

- SOA = Web Services
- Wrapping Services over Applications yields services
- SOA is a technology issue
- Shared Semantics is trivial
- SOA needs fully standardized mechanisms
- SOA can be done at an application level
- SOA can be implemented just by bringing an ESB

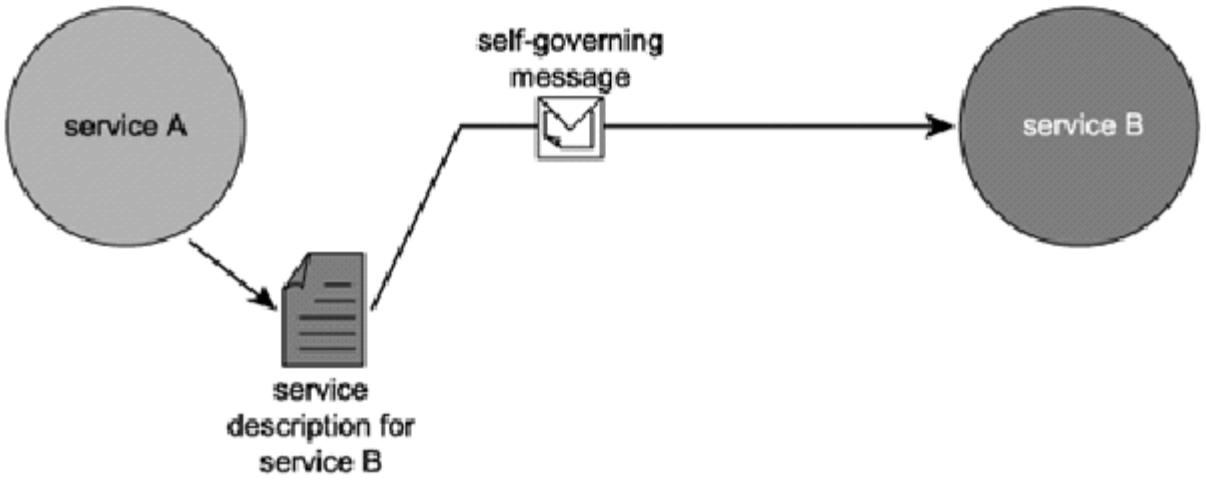
How services encapsulate logic



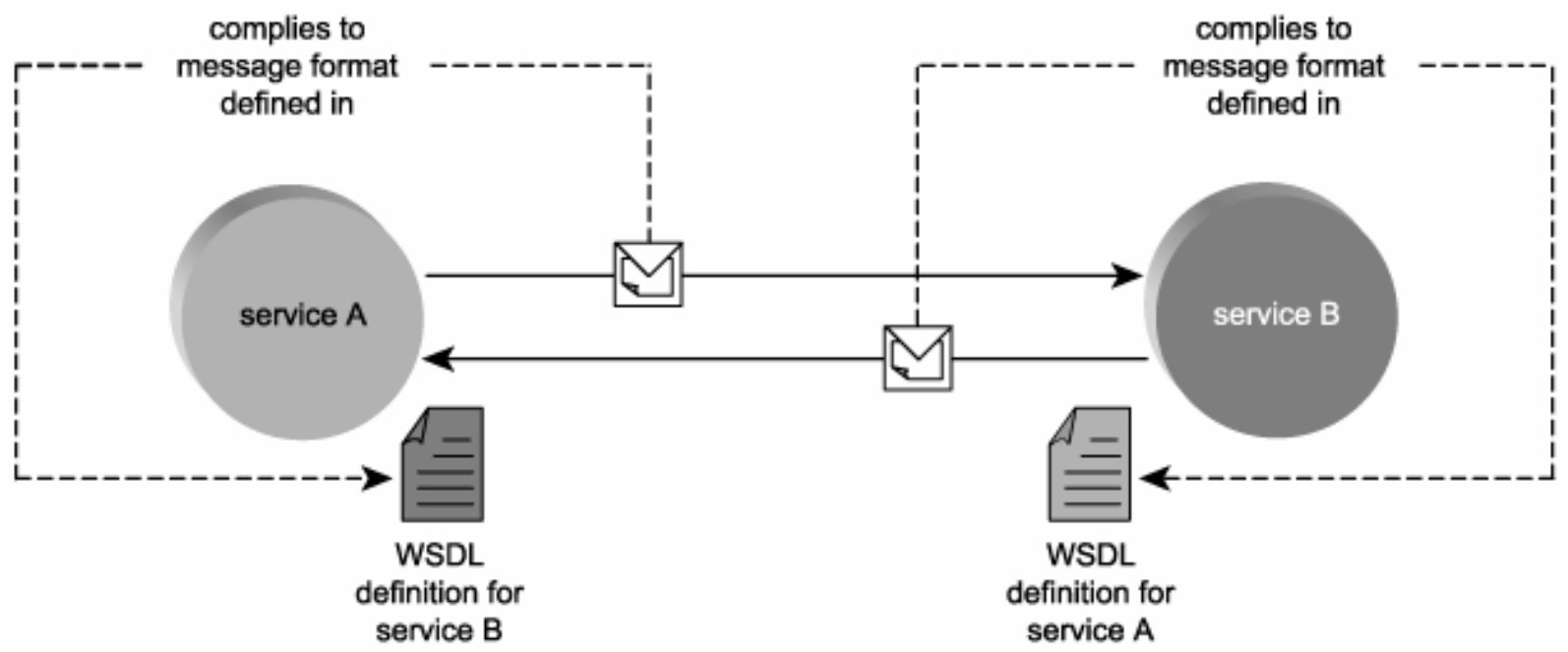
How services relate



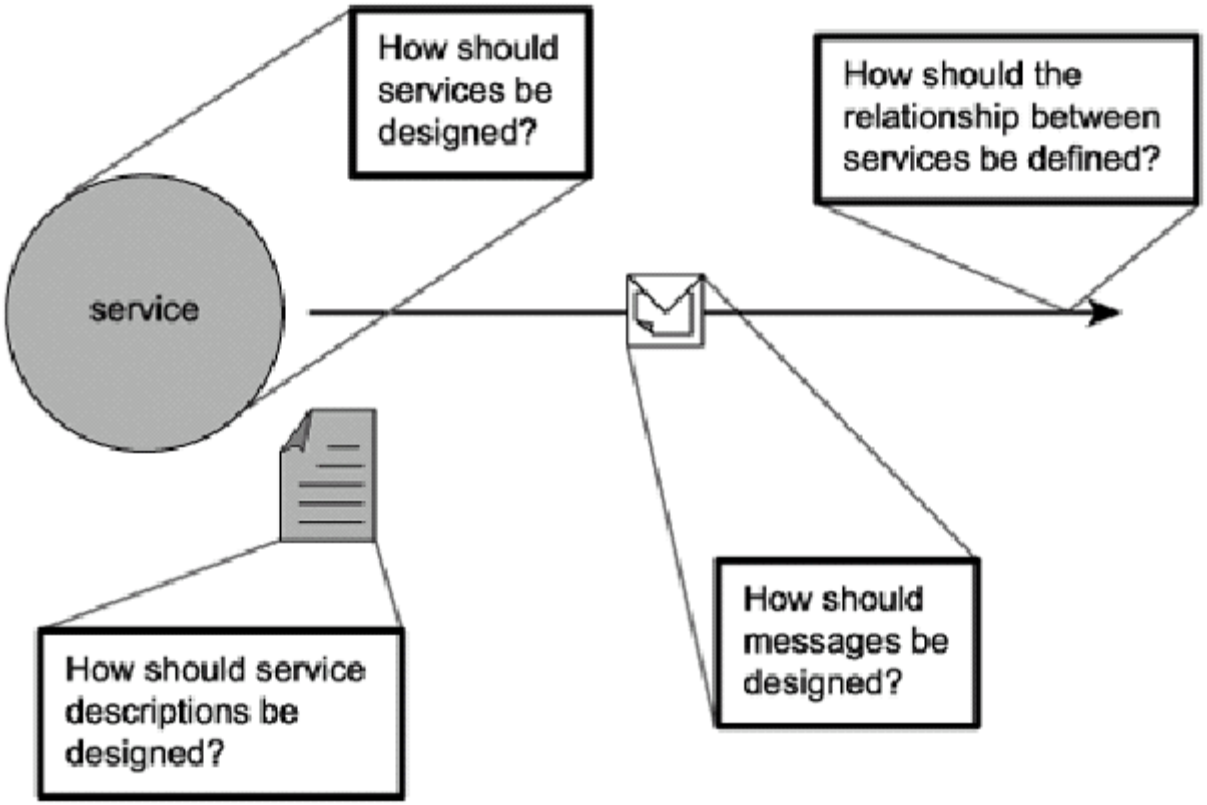
How services communicate



WSDL definitions enable loose coupling between services

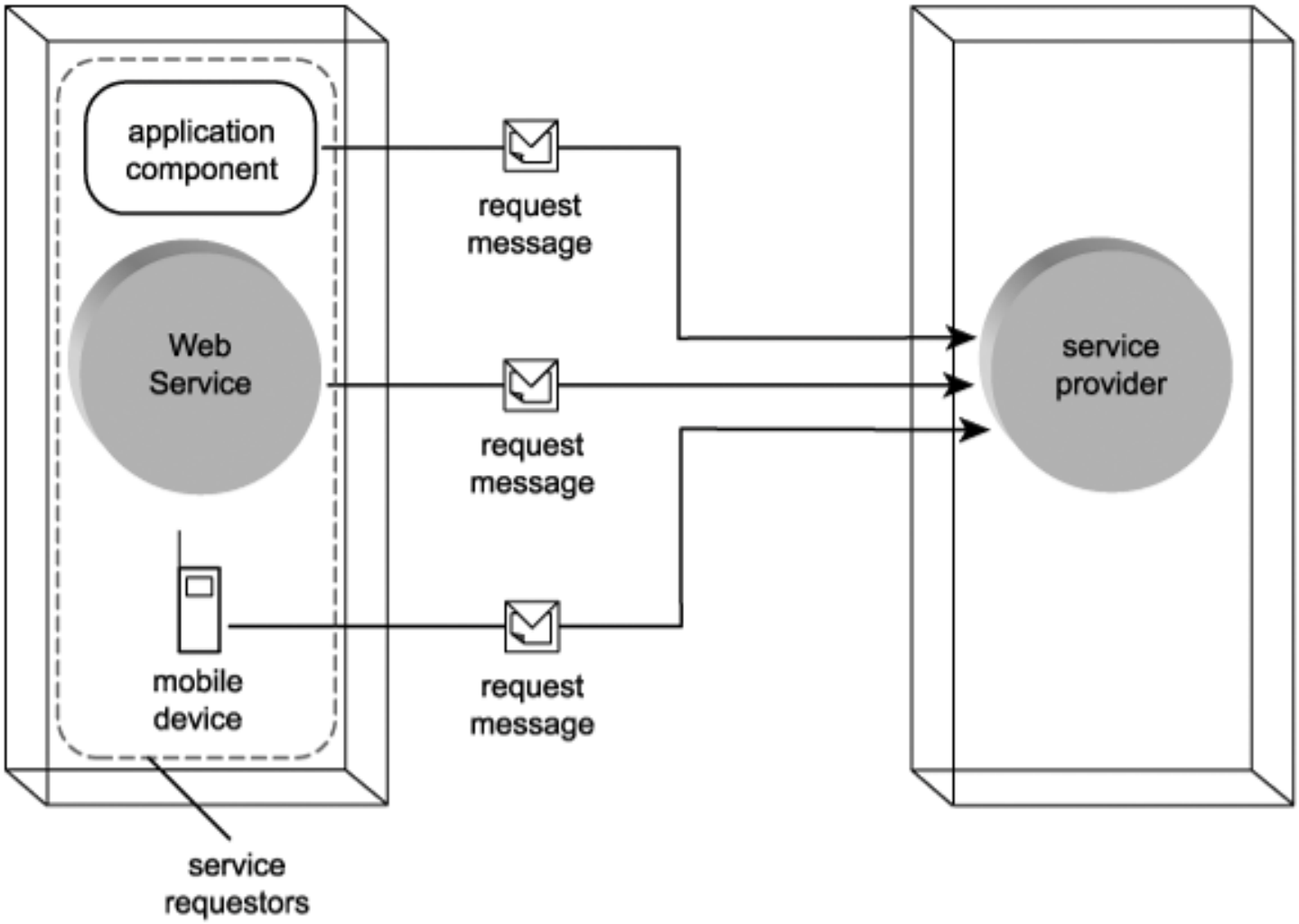


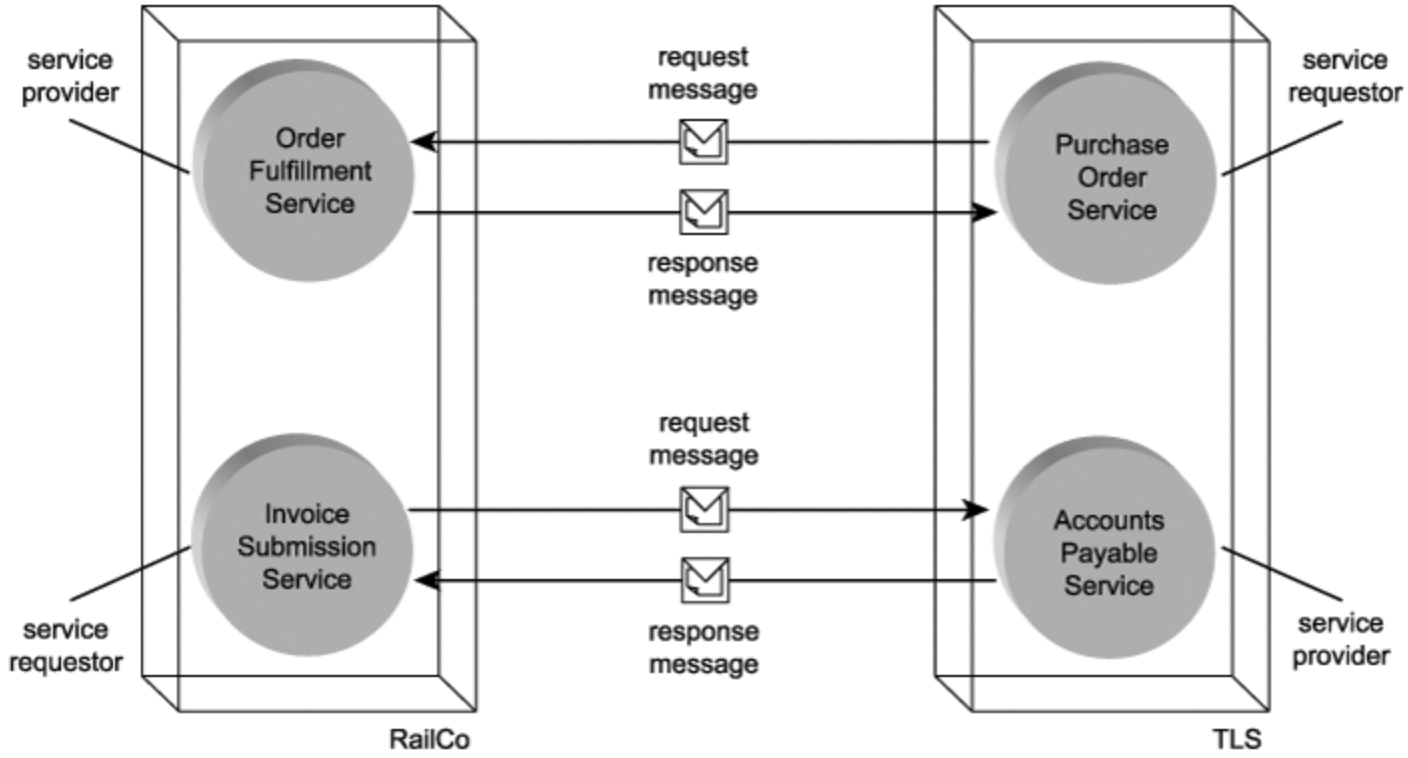
How services are designed

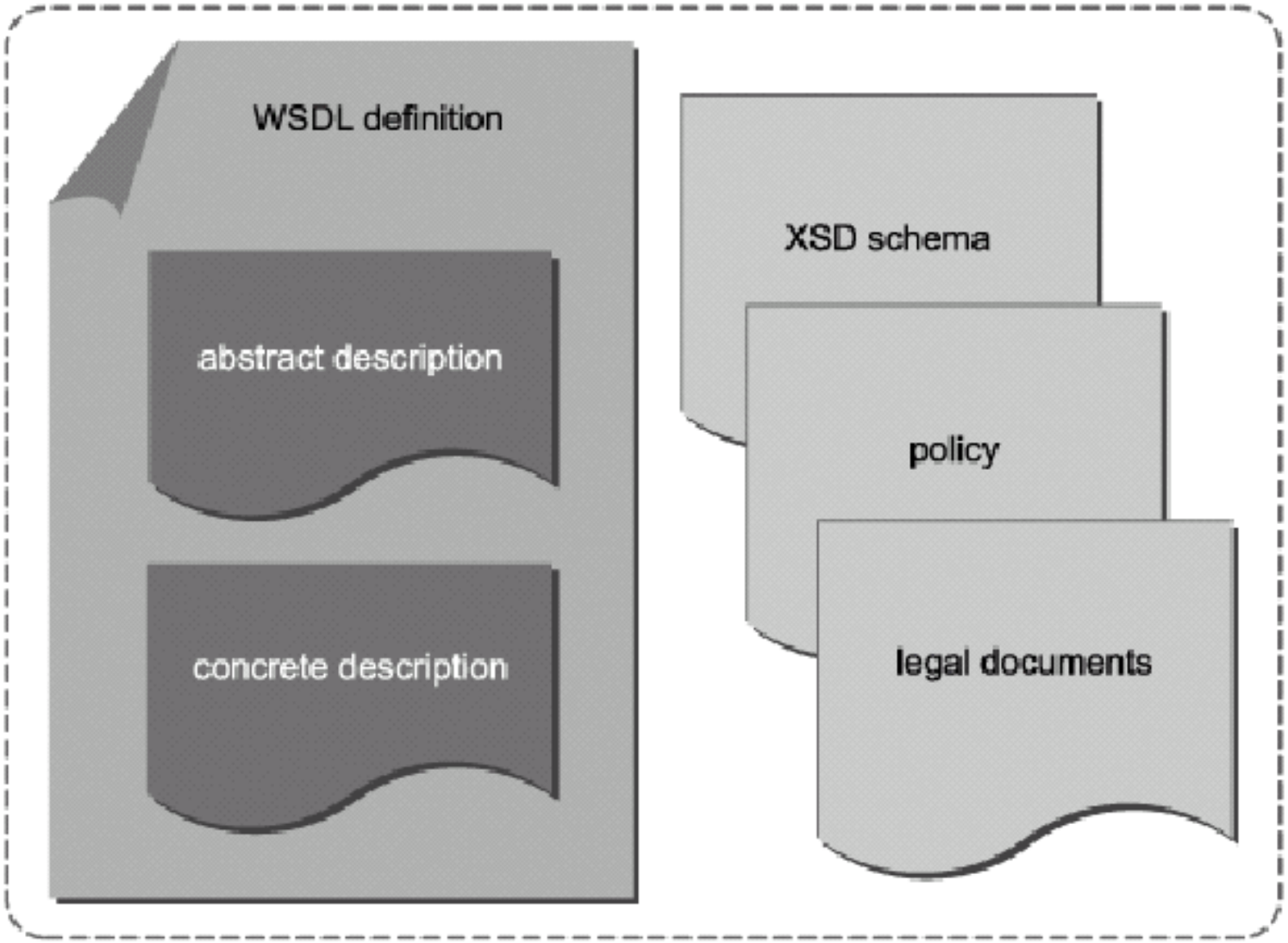


Key Principles of SOA

- Loose coupling
- Service contract
- Autonomy
- Abstraction
- Reusability
- Composability
- Statelessness
- Discoverability

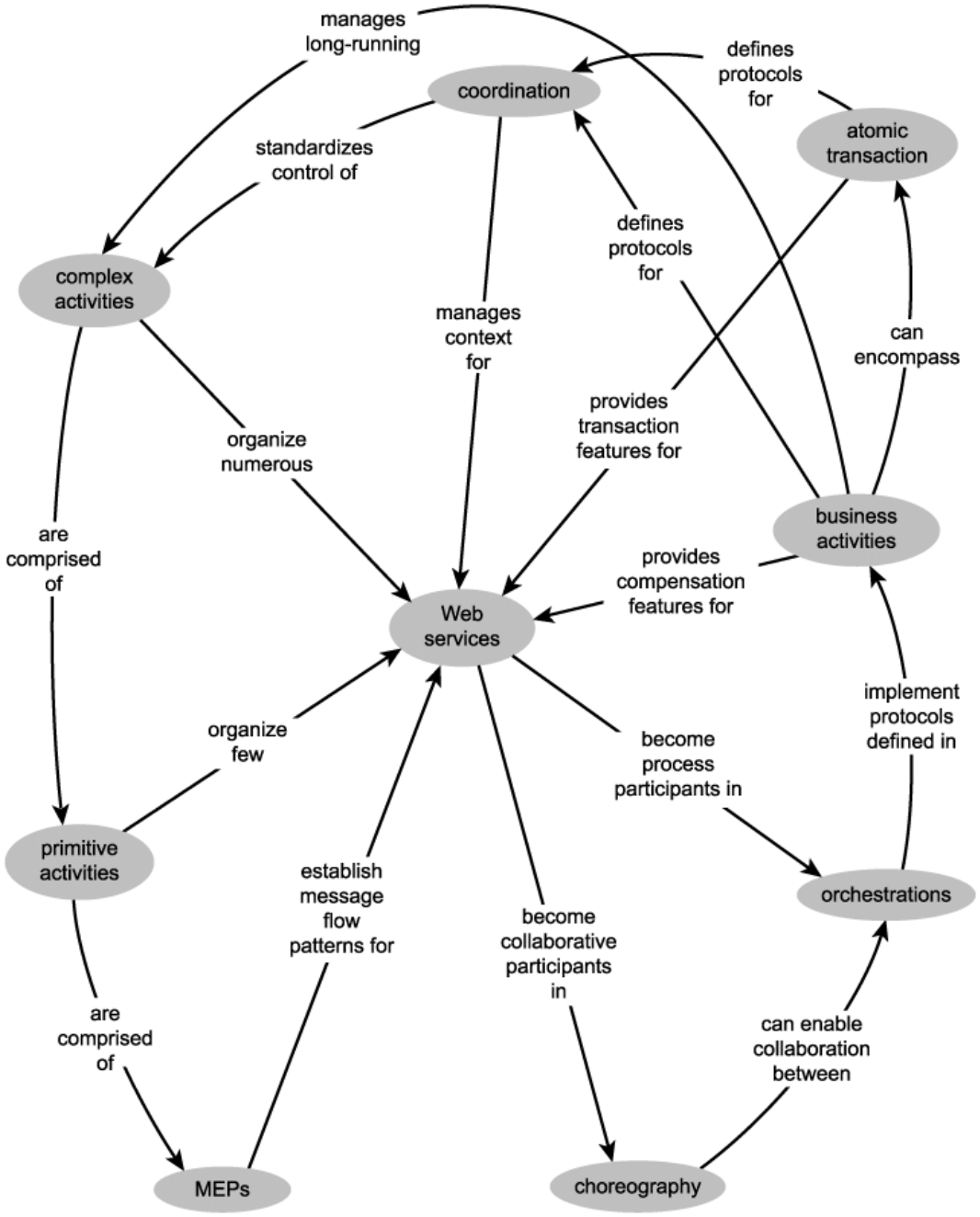

application component
Web Service
mobile device
service requestors
request message
request message
request message
service provider


service provider
Order Fulfillment Service
request message
response message
service requestor
Purchase Order Service
request message
Invoice Submission Service
service requestor
response message
Accounts Payable Service
service provider
RailCo
TLS

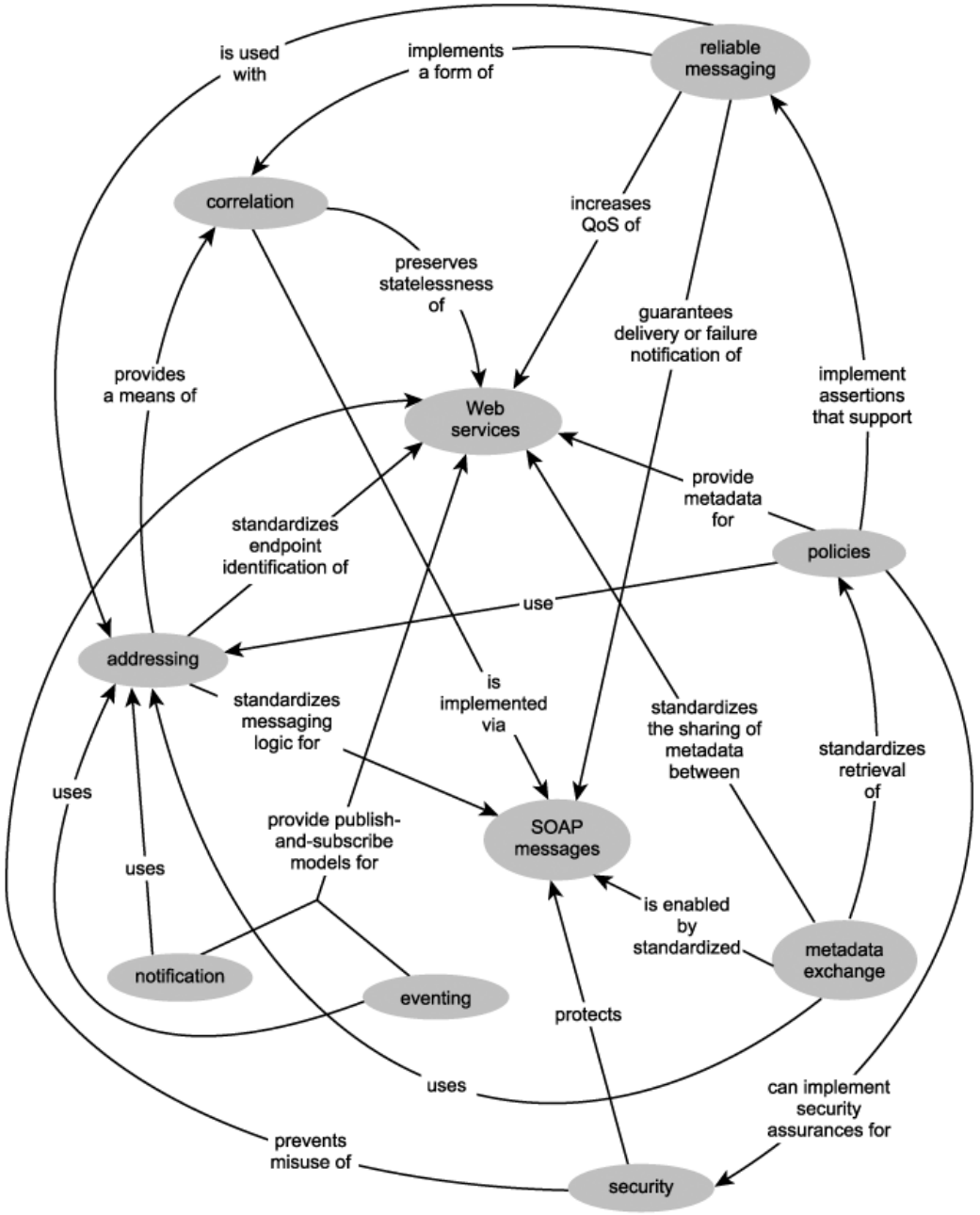


service contract

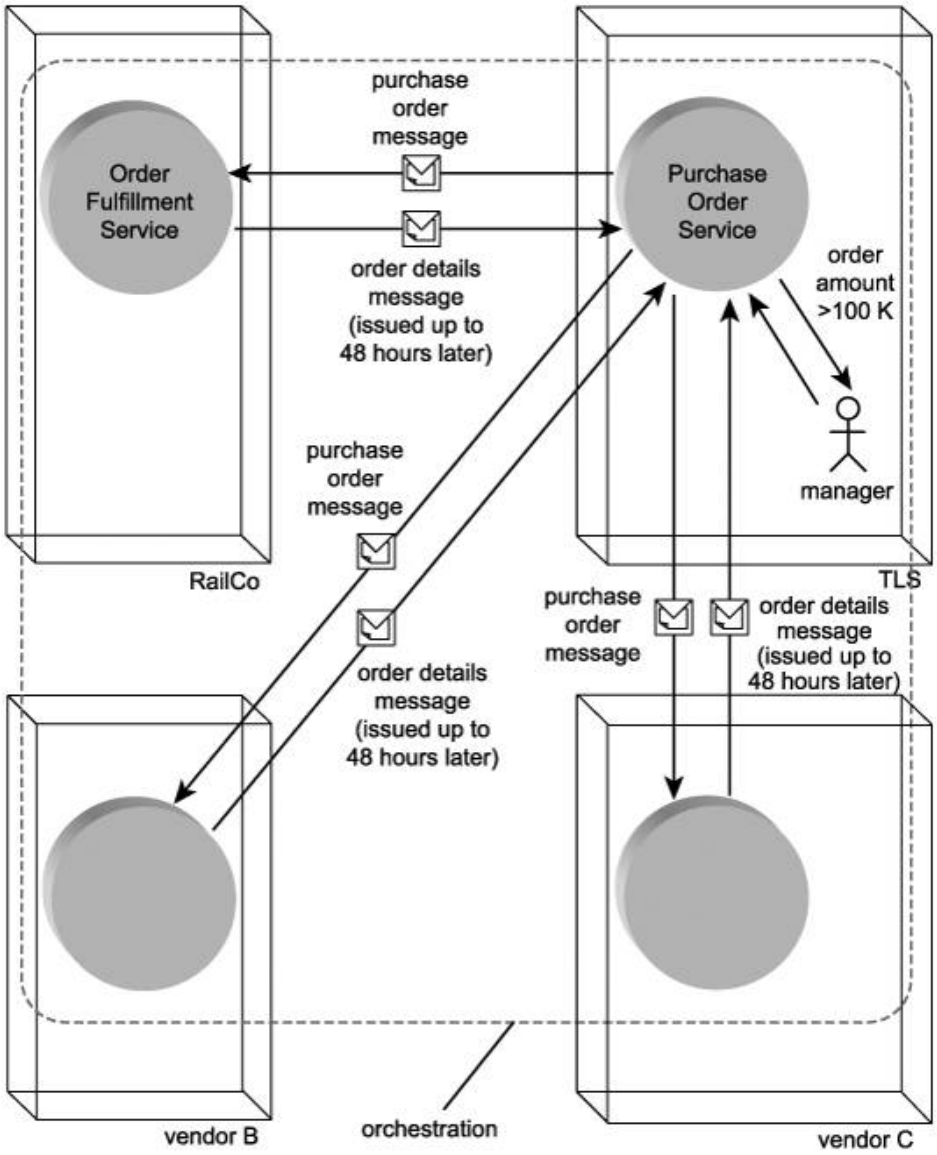
WS-* specifications



WS-* specifications



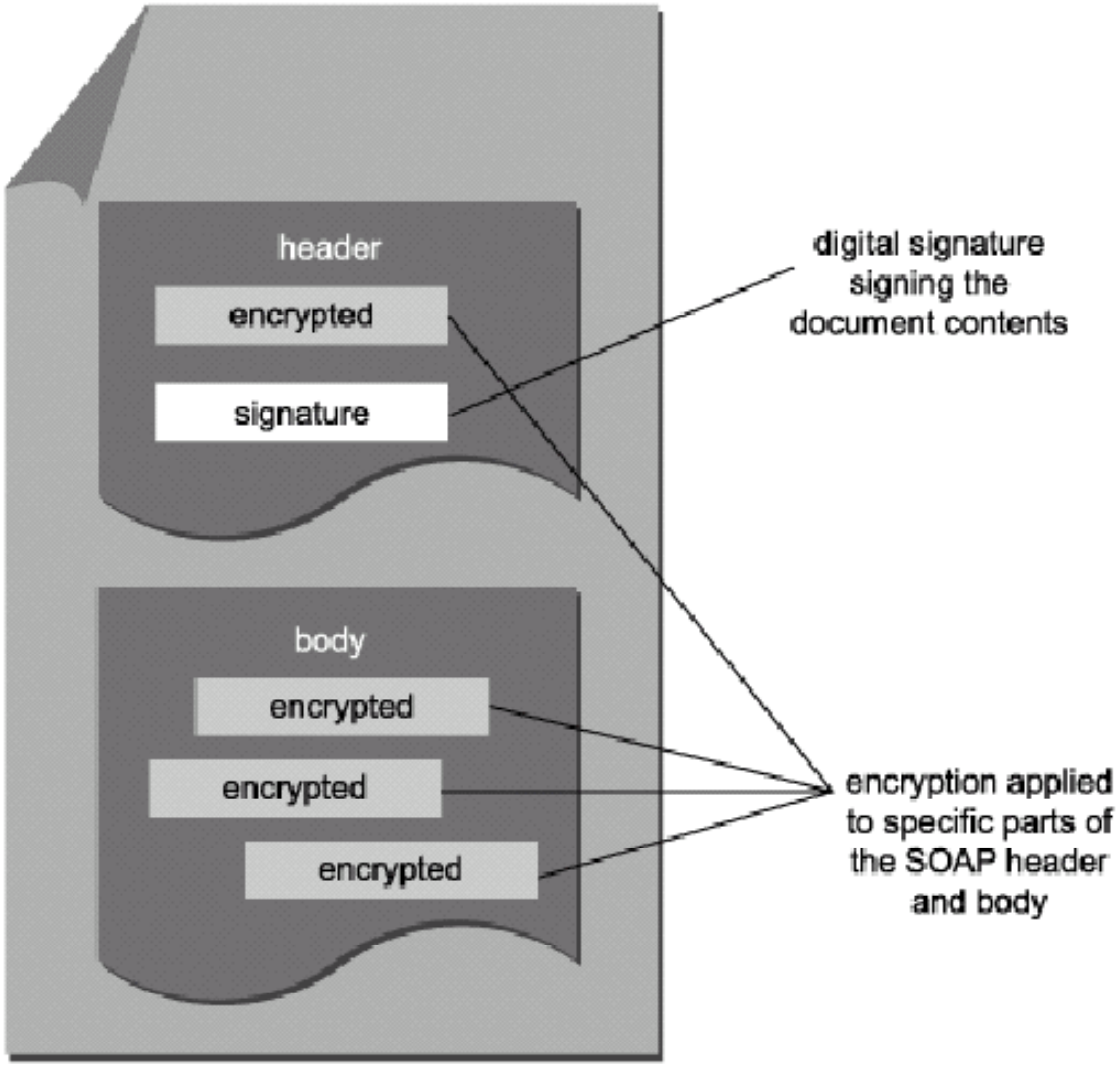
Orchestration of Services

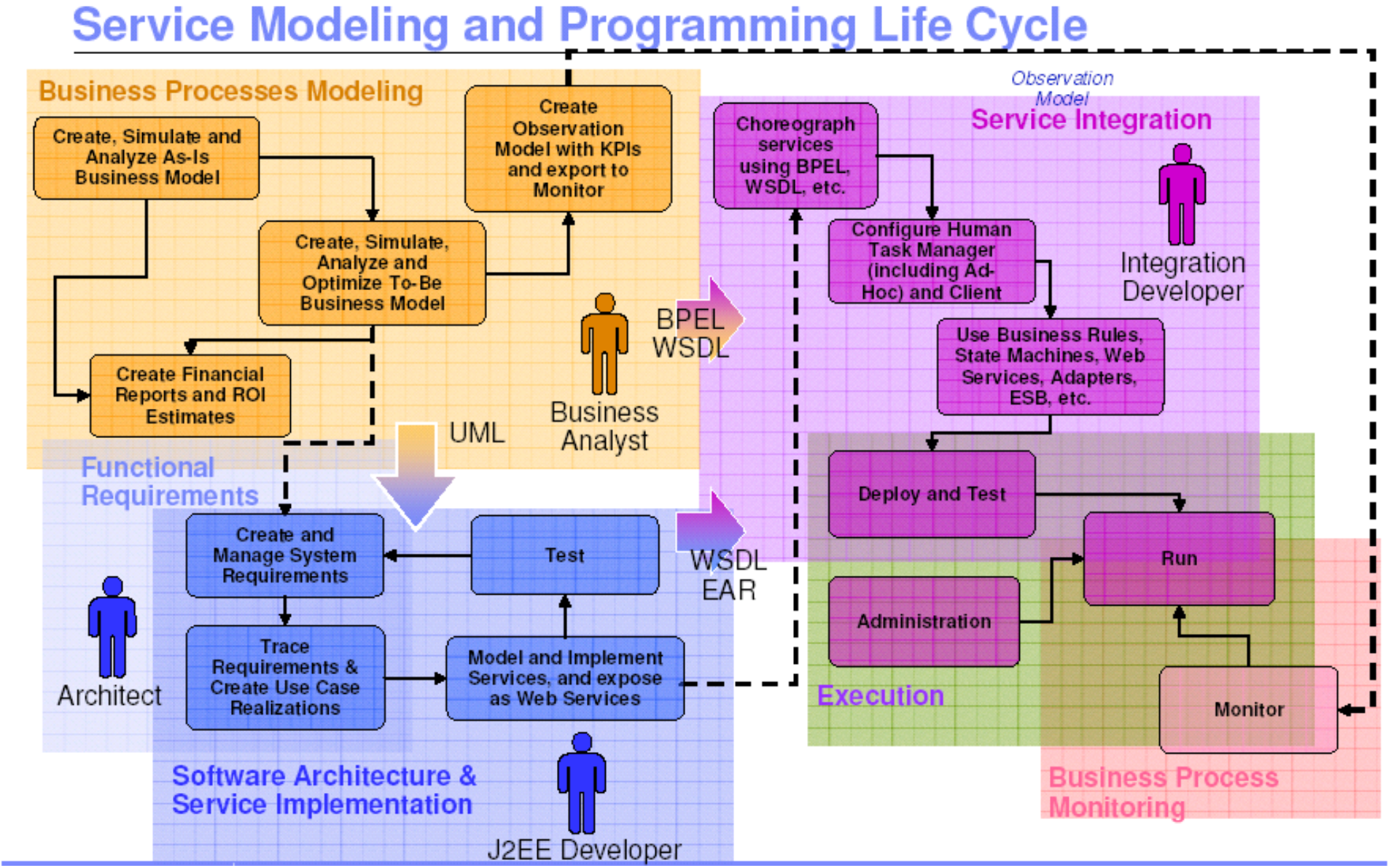


WS-Security specification

- A list of security specifications that may be used as part of SOA.
- WS-Security
- WS-SecurityPolicy
- WS-Trust
- WS-SecureConversation
- WS-Federation
- Extensible Access Control Markup Language (XACML)
- Extensible Rights Markup Language (XrML)
- XML Key Management (XKMS)
- XML-Signature
- XML-Encryption
- Security Assertion Markup Language (SAML)
- .NET Passport
- Secure Sockets Layer (SSL)
- WS-I Basic Security Profile

A digitally signed SOAP message containing encrypted data





Source: TRIC, IBMRL, India

Best Practices for SOA

SOA requires **strategic approach for maximum Reuse and flexibility**, with models for sharing Costs, benefits across the organization

Majority of implementations as we see today are **stopping at Integration+** owing to Governance issues

SOA **need not mean same thing** for everyone. Depending on the application domain it can be for EAI, Or for B2B integration or for even Infrastructure Virtualization

There is **Strong need to develop both Processes and People dimension** in addition to technology dimension

Implementing SOA is **not just about implementing an ESB** as product vendors may claim

Architects and Developers need to retrained to follow contract first development approaches

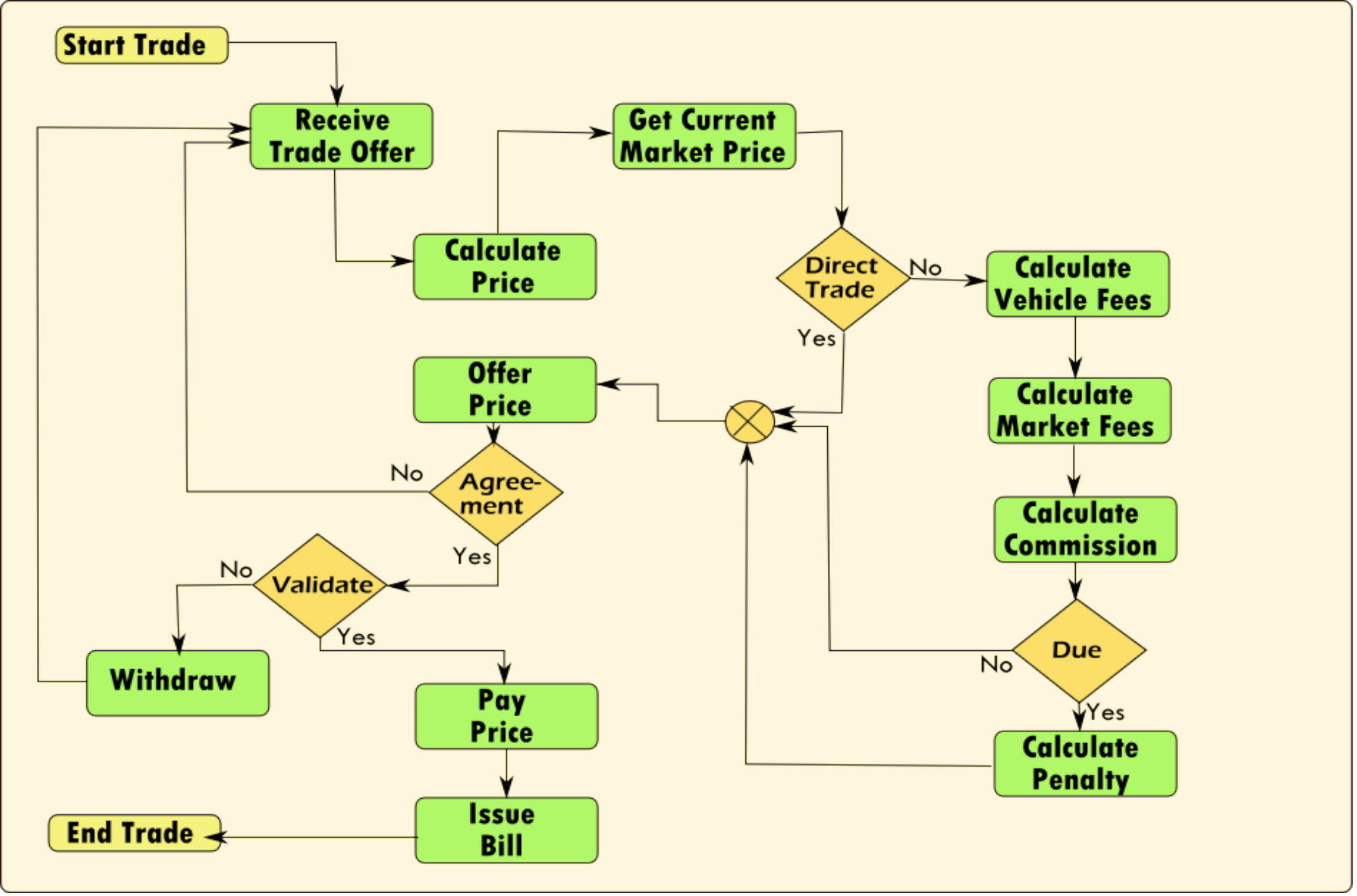
Initial investment is high, with payoff in the longer run, hence a **long term view is important**

Conclusions about SOA

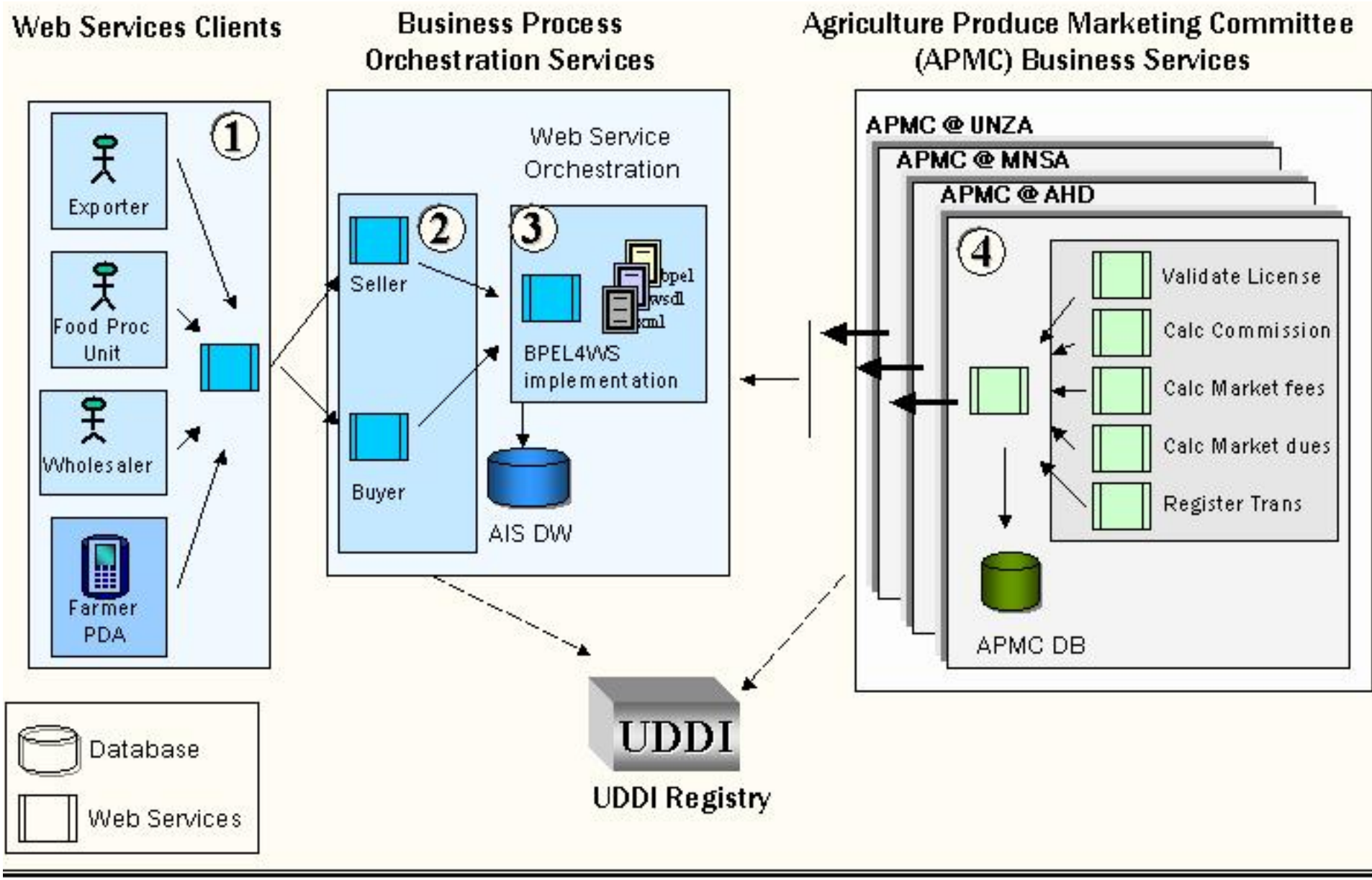
- SOA enables a loosely coupled model of distributed computing
- SOA is best applied with strategic intent on the scale of an enterprise
- A structured approach to SOA planning is vital
- SOA is not the same as web services though web services are the most popular means of achieving SOA
- SOA needs to looked at from the domain of its application
- Best practices right through life cycle of SOA planning will yield optimal results

Motivating Use Case: Marketing of Agro-Produce

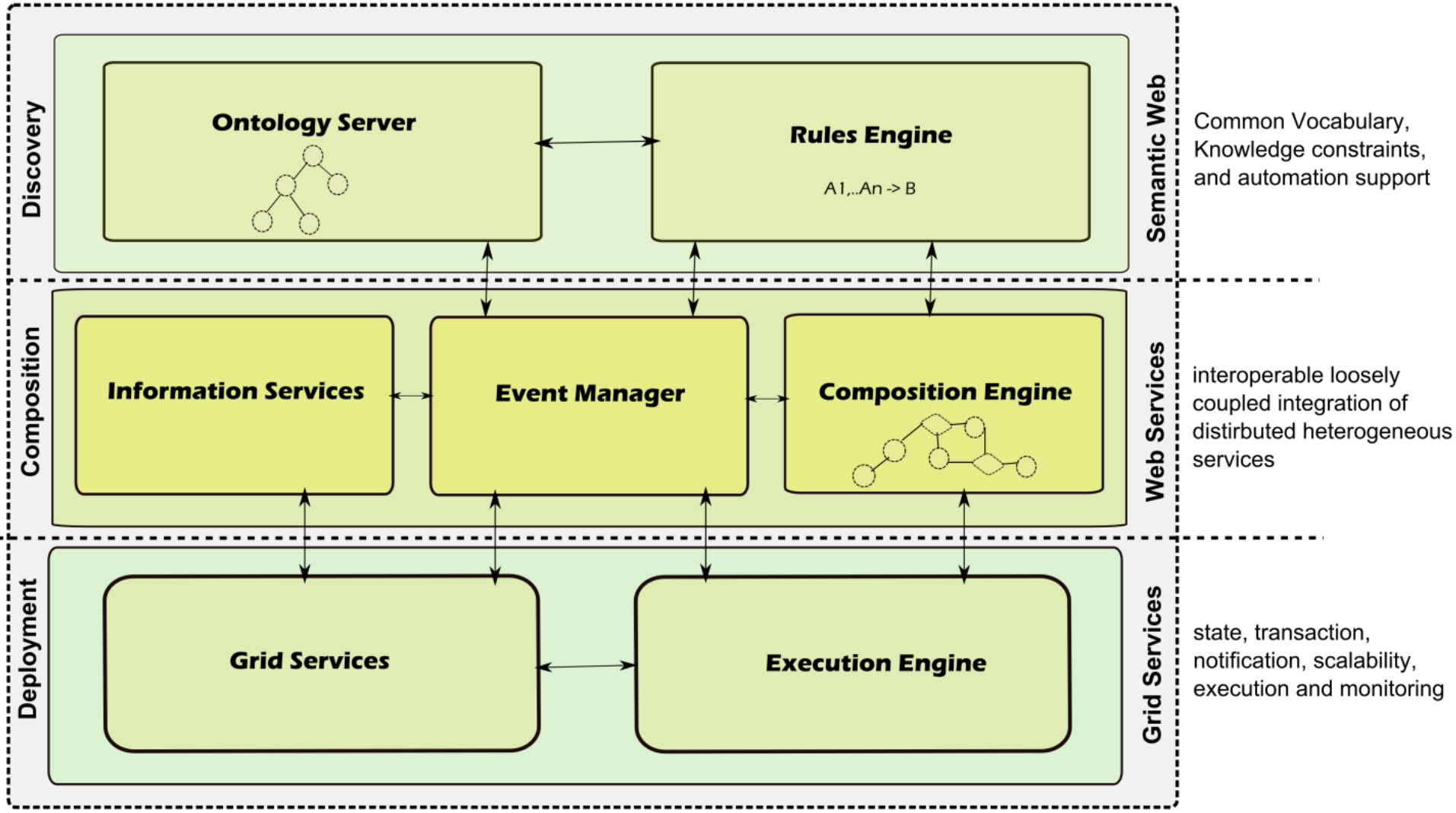
- The Model Act (Ministry of Agriculture, Govt. of India, 2003) is formulated to bring reforms in the Agricultural Marketing Process.
- Additional responsibilities are assigned to the existing Agricultural Produce Market Committee (APMC) to realize the reforms having following objectives:
  - To promote setting up of privately-owned markets
  - To promote direct sale and contract farming
  - To provide transparency in trading transactions
  - To provide market-led extension
  - To ensure payment on the same day
  - To enable value addition in agricultural produce by promoting processing



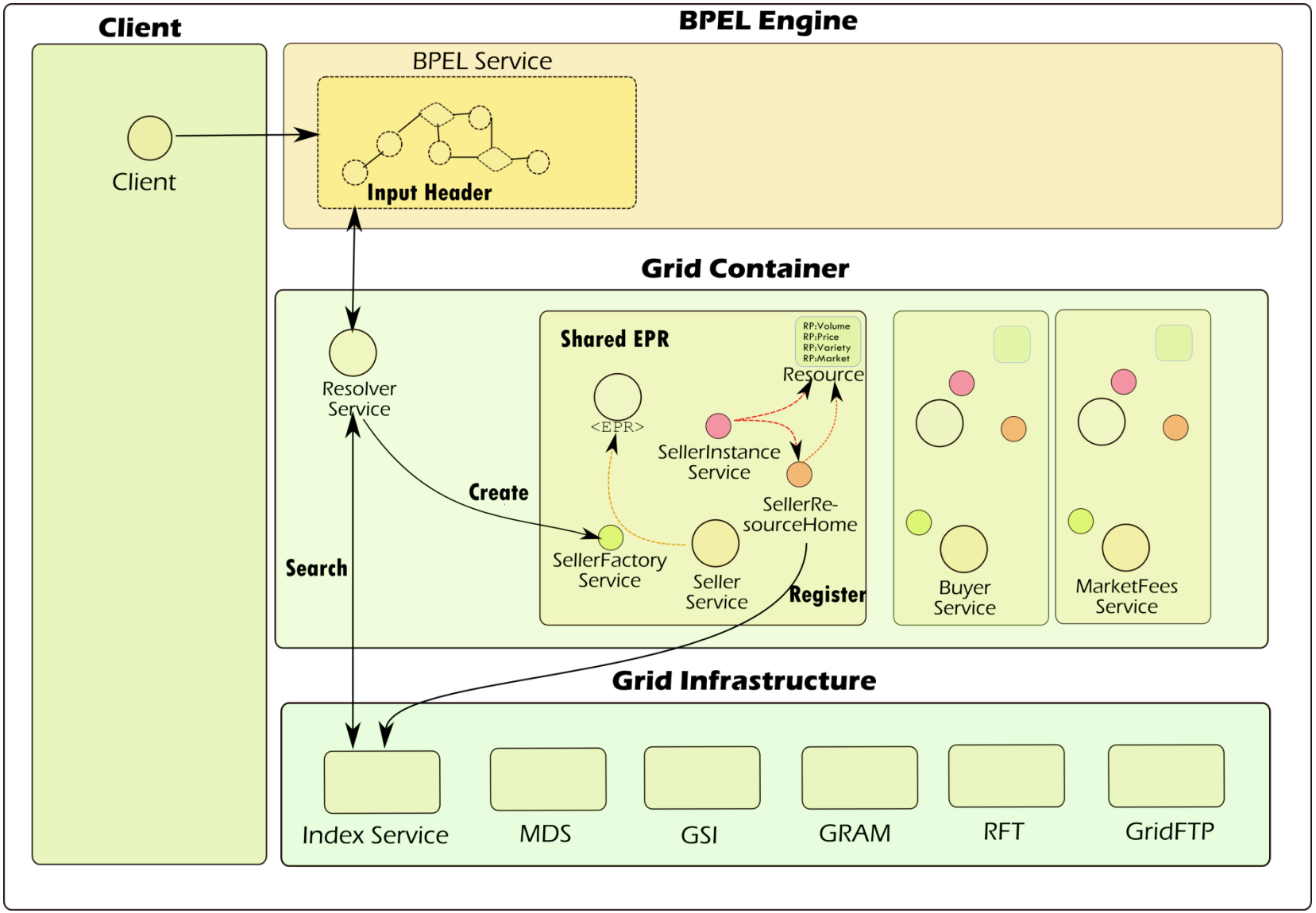
Workflow of Agro-produce Marketing Trading Process



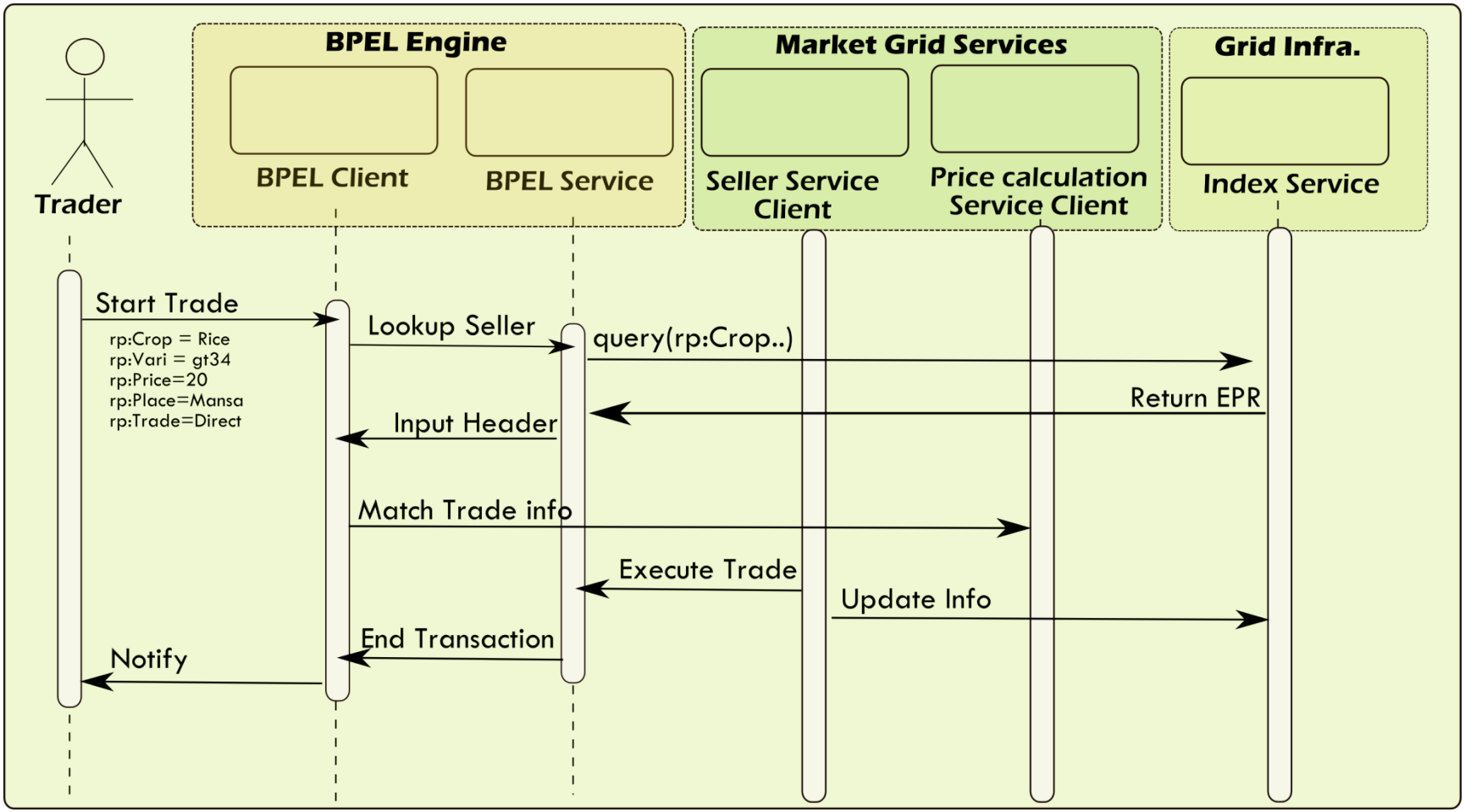
Proposed Architecture
(Static Composition of Services)



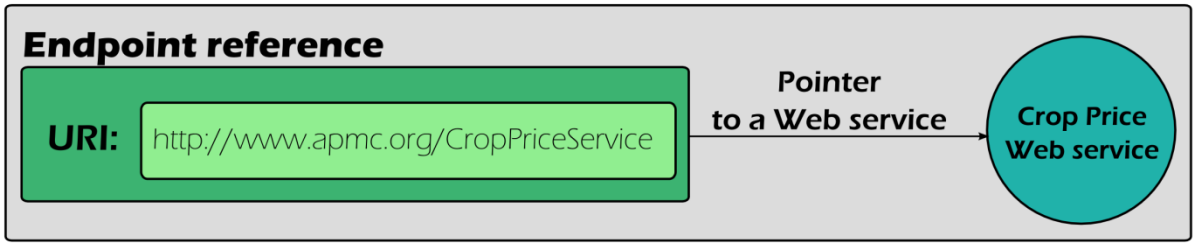
Event-Driven Service-Oriented Architecture



Component Interaction Diagram



Interactions among services

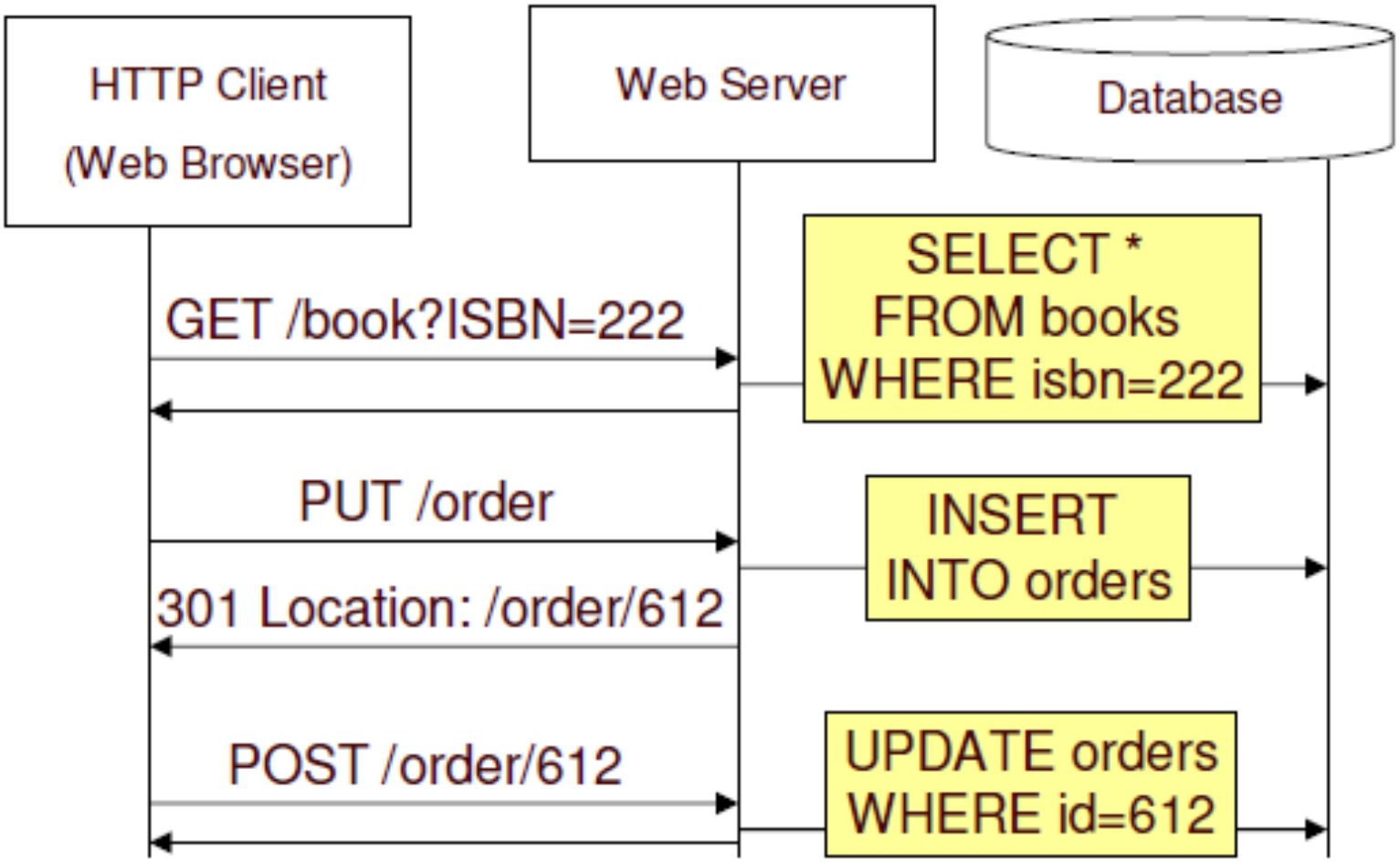


What is REST? (REpresentational State Transfer)

- REST is the architecture of the Web, its principles have been used to explain why the HTTP protocol scales so well (Roy Fielding PhD) (REST 1994, HTTP 1.0 1989)

1. Resource Identification through URI
2. **Uniform Interface** for all resources:
   - GET (Query the state, idempotent, can be cached)
   - POST (Modify, transfer the state)
   - PUT (Create a resource)
   - DELETE (Delete a resource)
3. "Self-Descriptive" Messages (Format/compression can be negotiated)
4. Hyperlinks between different media types

RESTful Web Application Example

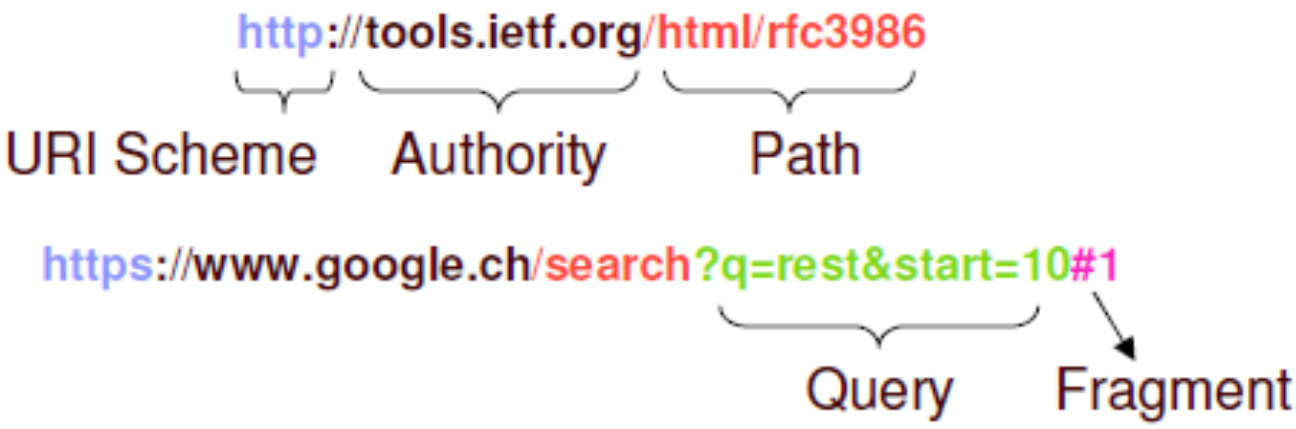


Uniform Interface Principle (CRUD Example)

	SQL	REST
CREATE	Insert	PUT
READ	Select	GET
UPDATE	Update	POST
DELETE	Delete/Drop	DELETE

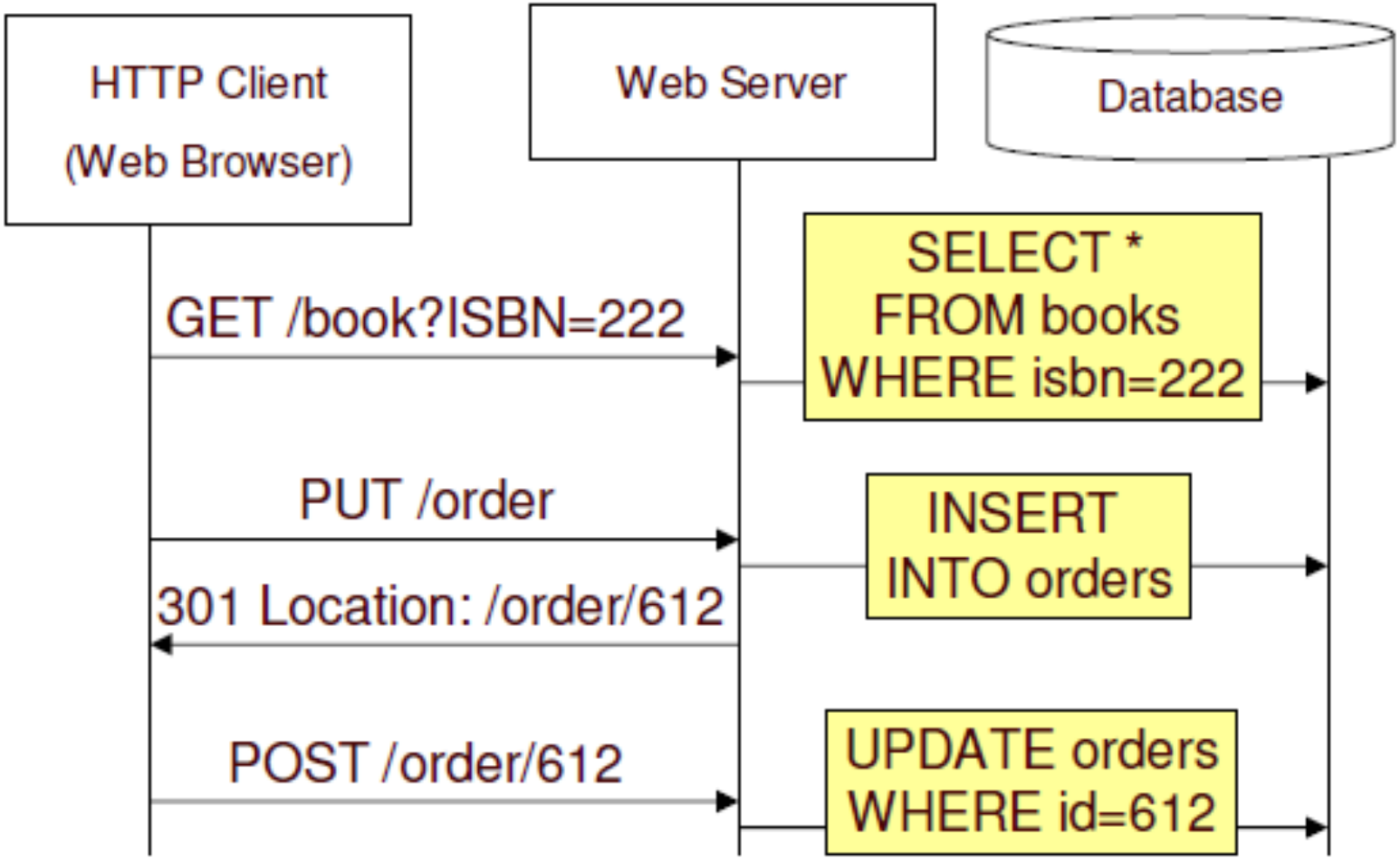
URI: Uniform Resource Identifier

- Internet Standard for resource naming and identification (originally from 1994, revised until 2005)
- Examples:

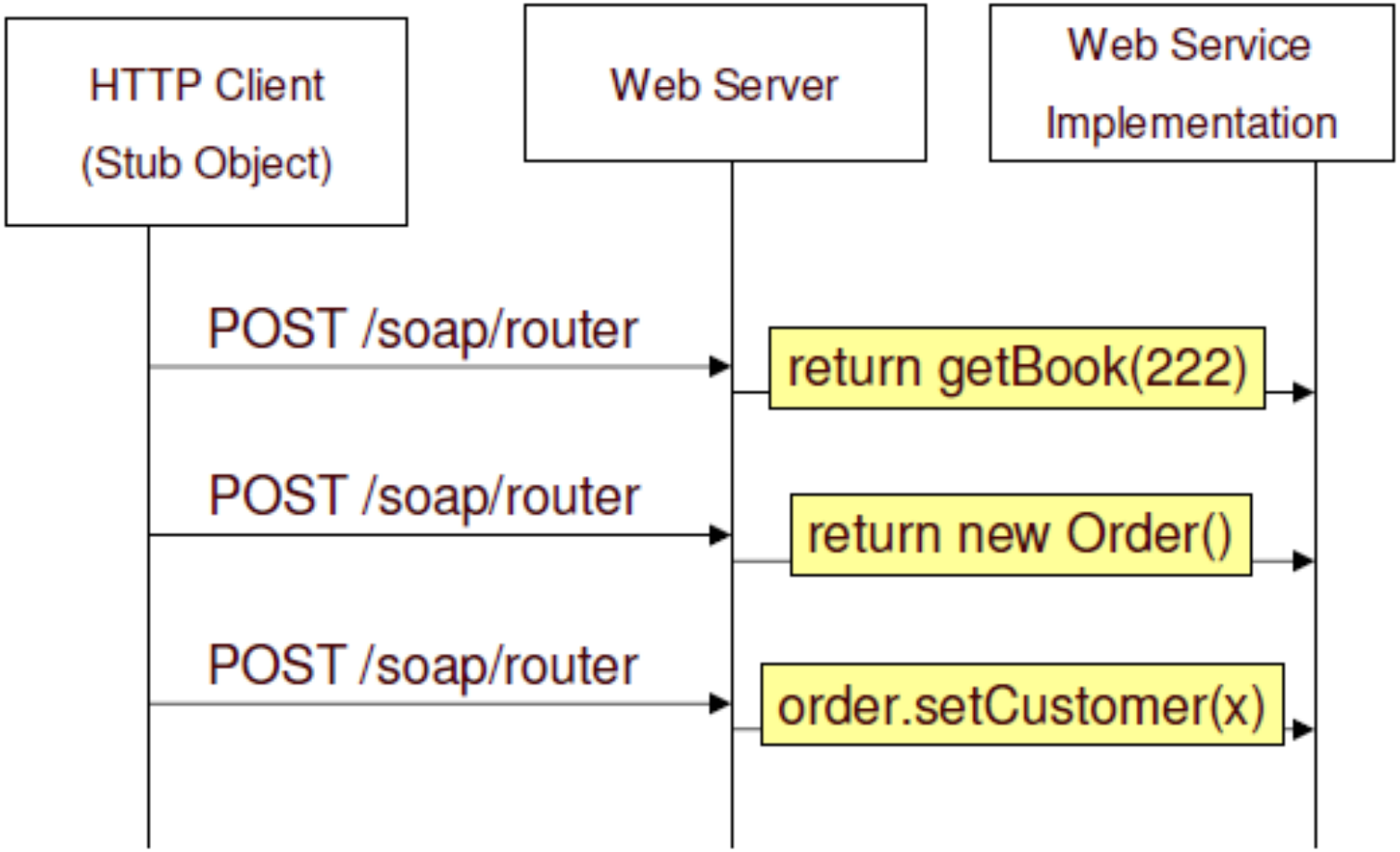


- RESTafarians advocate the use of “nice” URIs
- In most HTTP stacks URIs cannot have arbitrary length (4Kb)

RESTful Web Application Example

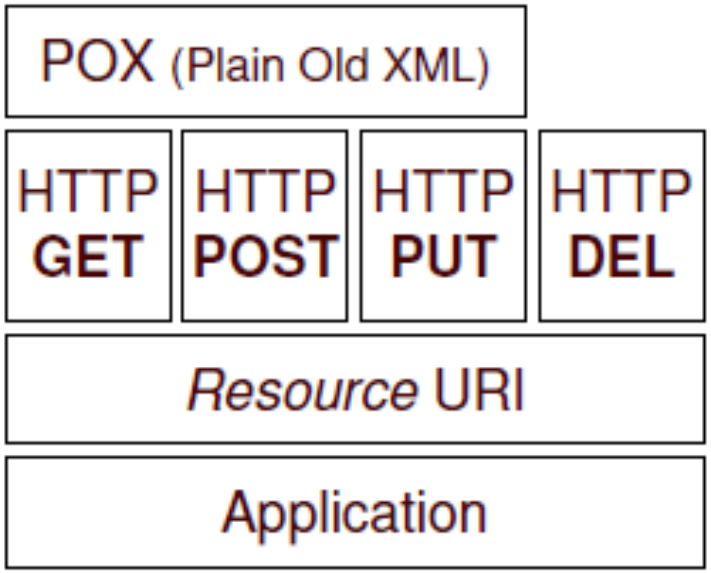


Web Service Example (from REST perspective)

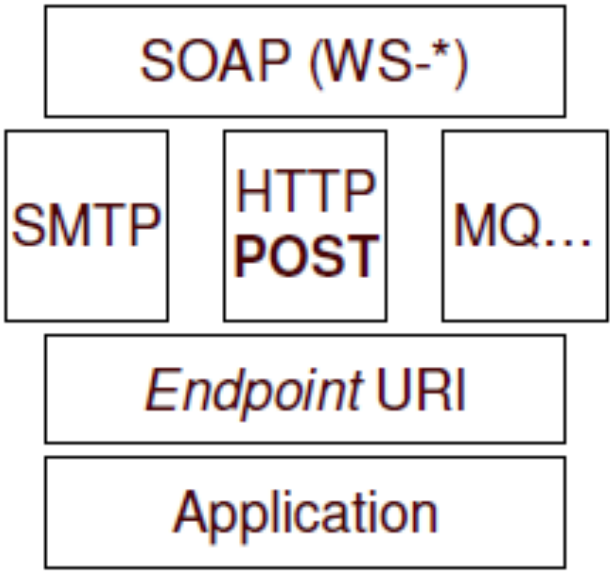


Main difference: REST vs. SOAP

- "The Web is the universe of globally accessible information" (Tim Berners Lee)
  - Applications should publish their data on the Web (through URI)



- "The Web is the universal transport for messages"
  - Applications get a chance to interact but they remain "outside of the Web"



REST vs. SOAP – Protocol Comparison

▪ XML in – XML out (with POST)	▪ SOAP in – SOAP out (with POST)
▪ URI in – XML out (with GET)	
▪ "Self-Describing" XML	▪ Strong Typing (XML Schema)
▪ HTTP only	▪ "Transport independent"
▪ HTTP/SSL is enough – no need for more standards	▪ Heterogeneity in QoS needs. Different protocols may be used
▪ HTTP is an application protocol	▪ HTTP as a transport protocol
▪ Synchronous	▪ Synchronous and Asynchronous
▪ Do-it-yourself when it comes to "reliable message delivery", "distributed transactions"	▪ Foundation for the whole WS* advanced protocol stack

What about security?

- REST security is all about HTTPS
- Proven track record (SSL1.0 from 1994)

- Secure, point to point communication (Authentication, Integrity and Encryption)

- SOAP security extensions defined by WS-Security (from 2004)
- XML Encryption (2002)
- XML Signature (2001)
- Implementations are starting to appear now
  - Full interoperability moot
  - Performance?

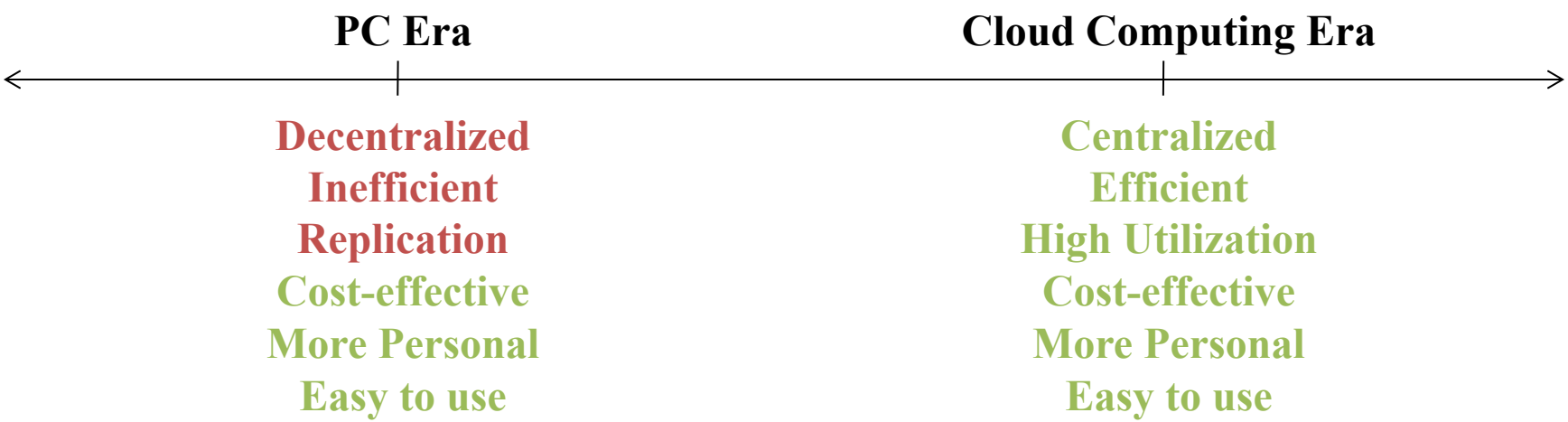
- Secure, end-to-end communication – Self-protecting SOAP messages (does not require HTTPS)

Cloud Computing

Cloud Computing: Definition

- “Cloud computing overlaps some of the concepts of distributed, grid and utility computing, however it does have its own meaning if contextually used correctly.
- Cloud computing really is accessing resources and services needed to perform functions with dynamically changing needs.
- An application or service developer requests access from the cloud rather than a specific endpoint or named resource.
- What goes on in the cloud manages multiple infrastructures across multiple organizations and consists of one or more frameworks overlaid on top of the infrastructures tying them together.
- The cloud is a virtualization of resources that maintains and manages itself."
- **- Kevin Hartig**

Why Cloud Computing?



What has been changed?

- Bandwidth is present in great quantity and cheaper
- Internet is being present everywhere
- Hardware has become more valuable

Cloud

- Large pool of easily usable and accessible virtualized resources (Such as hardware, development platforms and/or services).
- Resources can be dynamically reconfigured to adjust to a variable load (scale), allowing also for an optimum resource utilization.
- Pool of resources is typically exploited by a pay-per-use model in which guarantees are offered by the Infrastructure Provider by means of customized Service Level Agreements (SLAs).

Service Level Agreements (SLA)

- It is the relationship between the cloud provider and the cloud
- consumer that must be described
- Cloud consumers trust cloud providers to deliver some of their infrastructure services
- It is part of the contract between the service consumer and service provider and formally defines the level of service.
- Although managing and monitoring the quality levels of services rely heavily on automated tools
- At present the actual Cloud SLAs are typically plain-text documents, and sometimes an informative document published online.
- Examples: document with legal obligations is the Amazon S3 Service Level Agreement, Amazon EC2 Service Level Agreement.

Service Level Agreements (SLA)

SLA parameters can be considered as listed below:

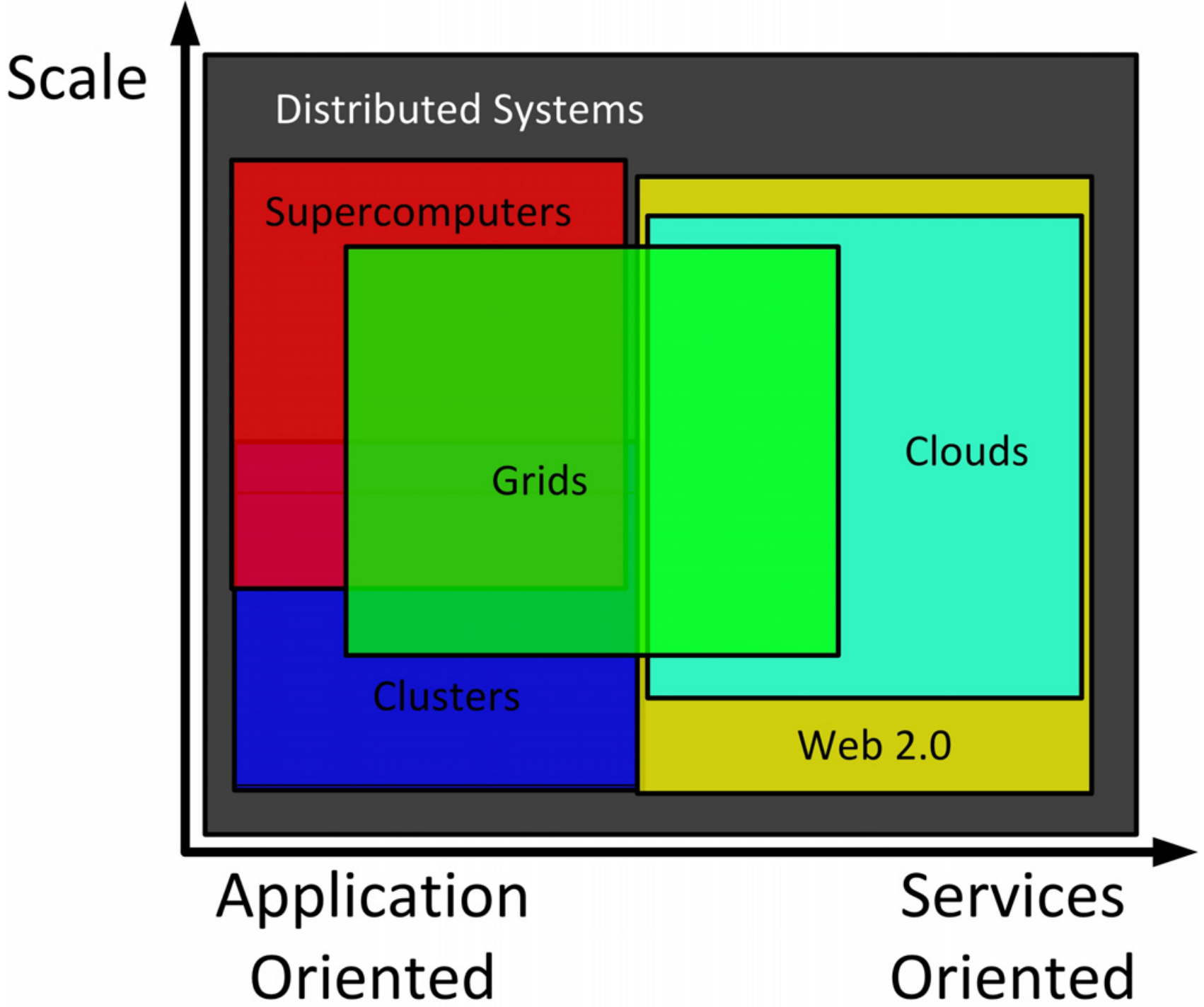
- Virtual Machine (VM)
- Storage Capability
- Memory Capability
- Processor Speed
- Ethernet port
- Availability
- Response Time
- Server Reboot Time
- Service Credit

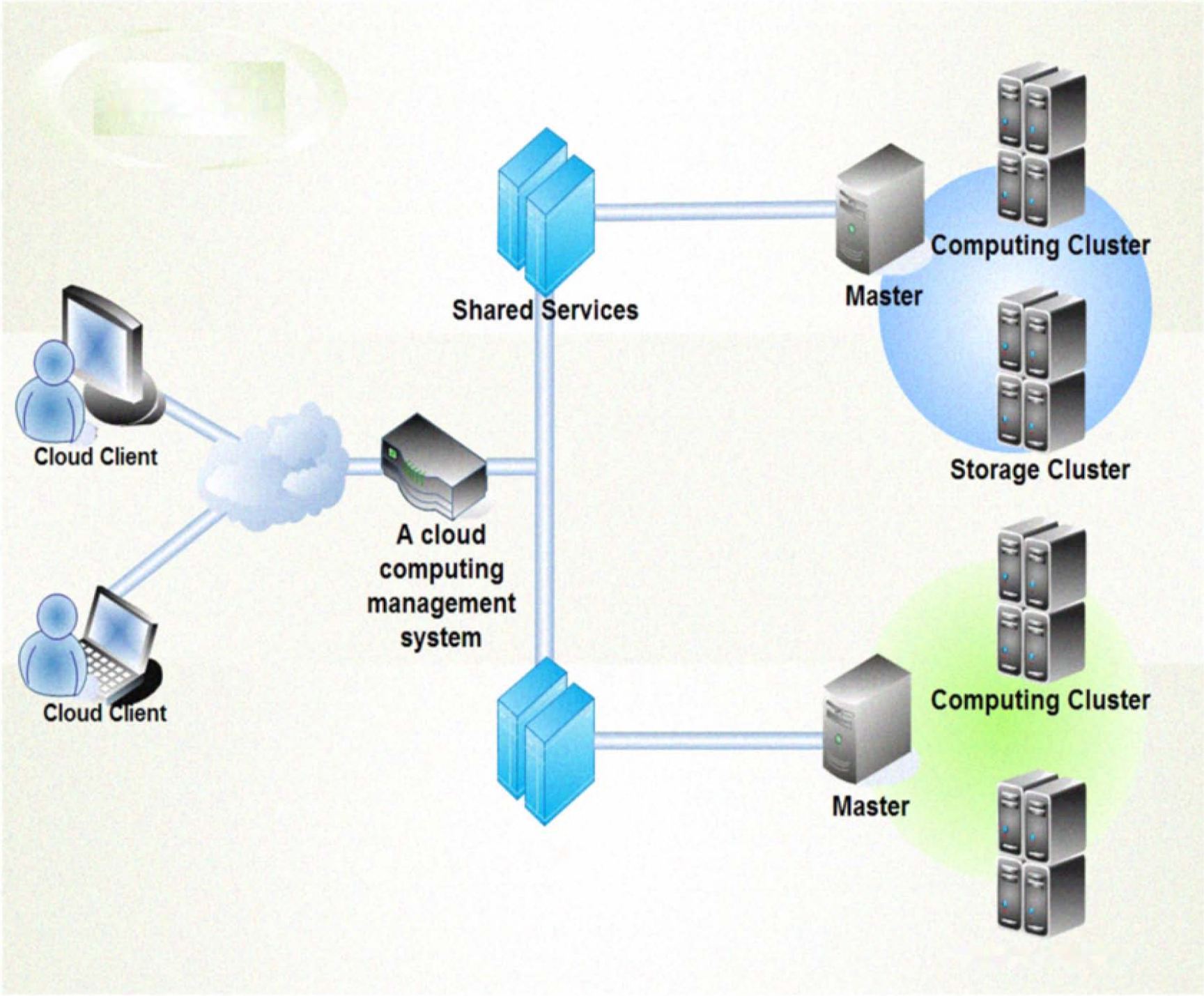
Cloud Computing

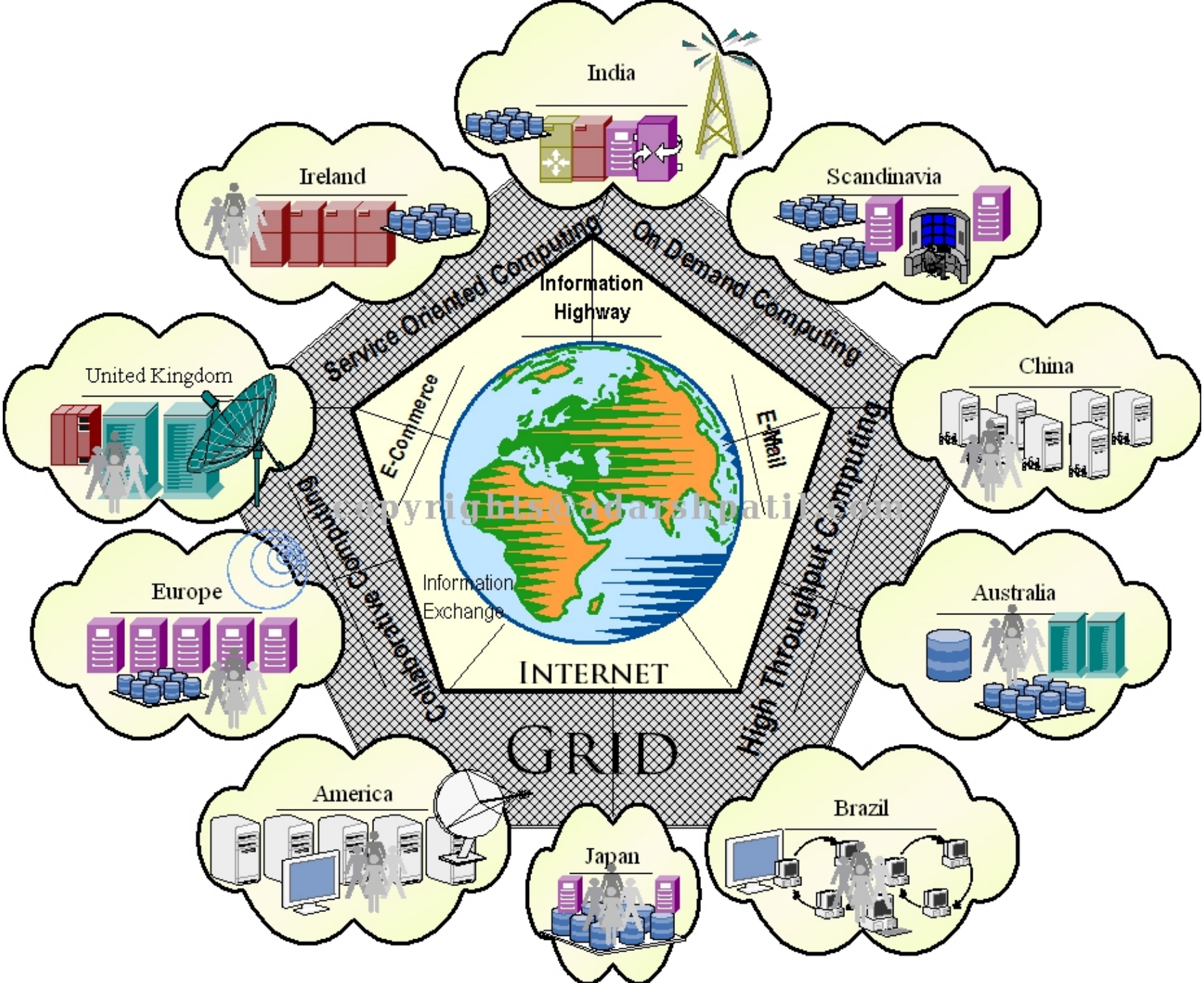
- This cloud model promotes availability and is composed of five essential **characteristics,** three **service models**, and four **deployment models**.

Cloud Computing

- **Essential Characteristics:**
  - On-demand self-service
  - Broad network access
  - Resource pooling
  - Rapid elasticity
  - Measured Service
- **Deployment Models:**
  - Private cloud.
  - Community cloud.
  - Public cloud.
  - Hybrid cloud.
- **Service Models:**
  - Software as a Service (SaaS)
  - Platform as a Service (PaaS)
  - Infrastructure as a Service (IaaS)


Scale
Distributed Systems
Supercomputers
Grids
Clouds
Clusters
Web 2.0
Application Oriented
Services Oriented


Shared Services
Master
Computing Cluster
Storage Cluster
Cloud Client
Cloud Client
A cloud computing management system
Master
Computing Cluster



Grid vs Cloud

	Grid	Cloud
Underlying concept	Utility Computing	Utility Computing
Main benefit	Solve computationally complex problems	Provide a scalable standard environment for network-centric application development, testing and deployment
Resource distribution / allocation	Negotiate and manage resource sharing; schedulers	Simple user <-> provider model; pay-per-use
Domains	Multiple domains	Single domain
Character / history	Non-commercial, publicly funded	Commercial

Grid Computing

- Solving large problems with Parallel computing
- Made mainstream By Global Alliance

Utility Computing

- Offering computing resources as a metered service
- Introduced in late 1990s

SaaS Computing

- Network-based subscriptions to applications
- Gained momentum in 2001

Cloud Computing

- Next-Generation Internet computing
- Next-Generation Data Centers

Cloud Terminology

- Virtualization Technology is becoming matured
- Virtualization enable Compute Clouds
- Compute Clouds create demand for Storage Cloud
- Storage + Compute Cloud create Cloud Infrastructure
- Cloud Infrastructure enables Cloud Platform & Application
- Multiple Cloud types lead to Cloud Aggregators
- Niche requirements enable cloud Extenders

Commercial clouds

amazon web services™ Amazon Elastic Compute Cloud (Amazon EC2) - *Beta*

Appistry

hp

vmware

Joyent

CITRIX®

3tera™

GOGRID beta
A ServePath Company

Q-layer

MOSSO
the hosting cloud

b-hive

rPath®

CYCLECOMPUTING

Microsoft

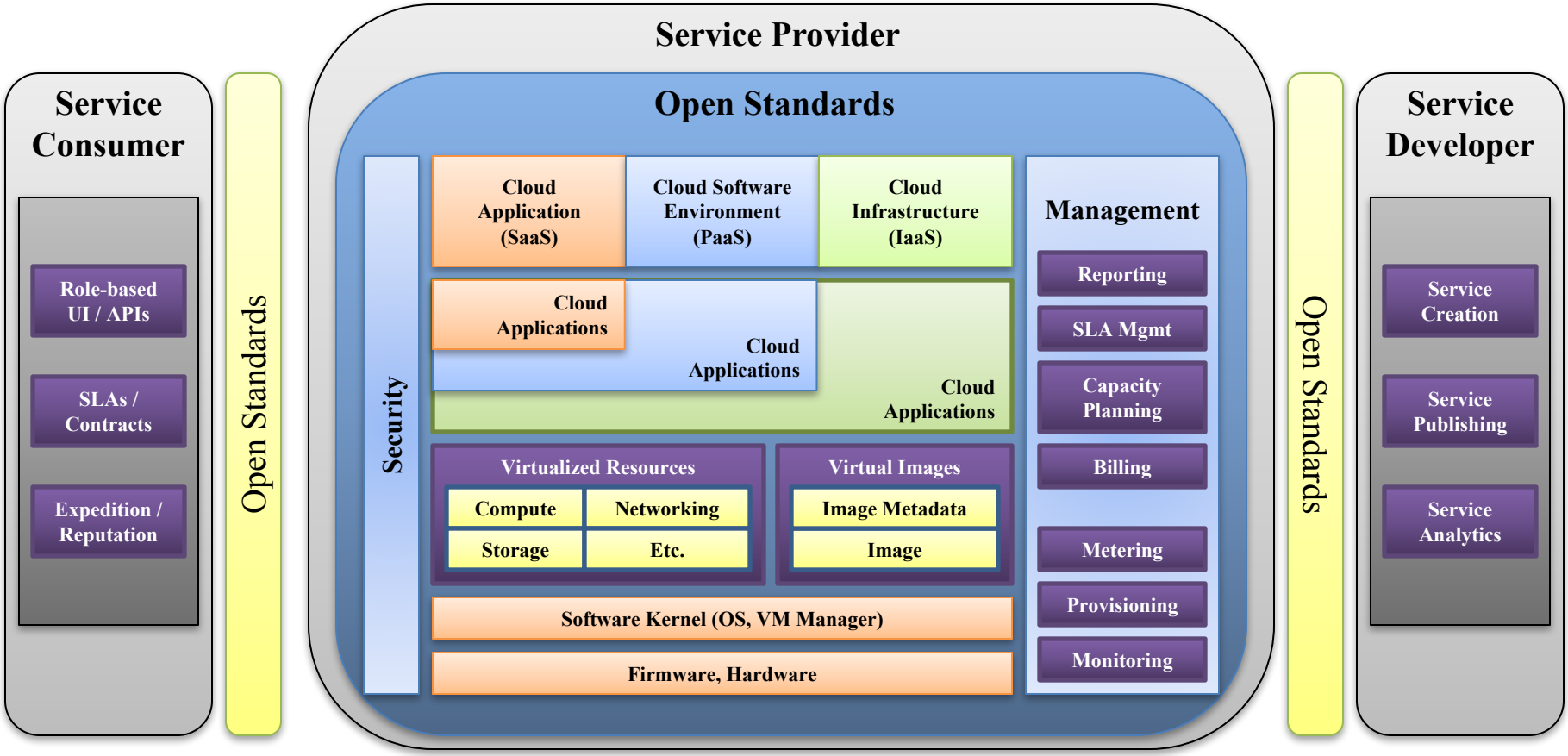
Engine Yard™

VERIO
An NTT Communications Company

Sun microsystems TAP INTO THE POWER OF NETWORK.COM

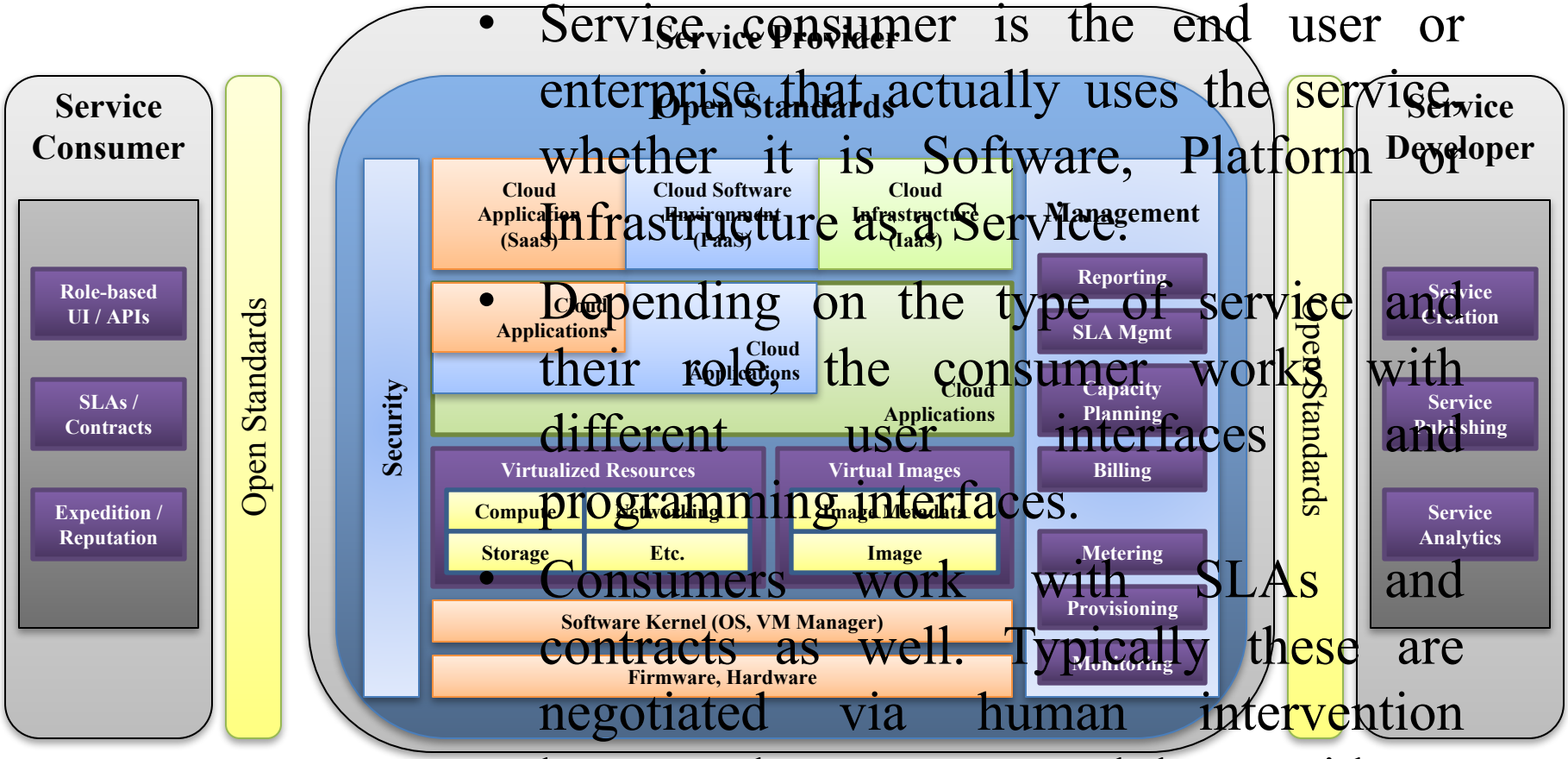
VMOps

Taxonomy for cloud computing



Source: Cloud Computing Use Cases White Paper Version 4.0

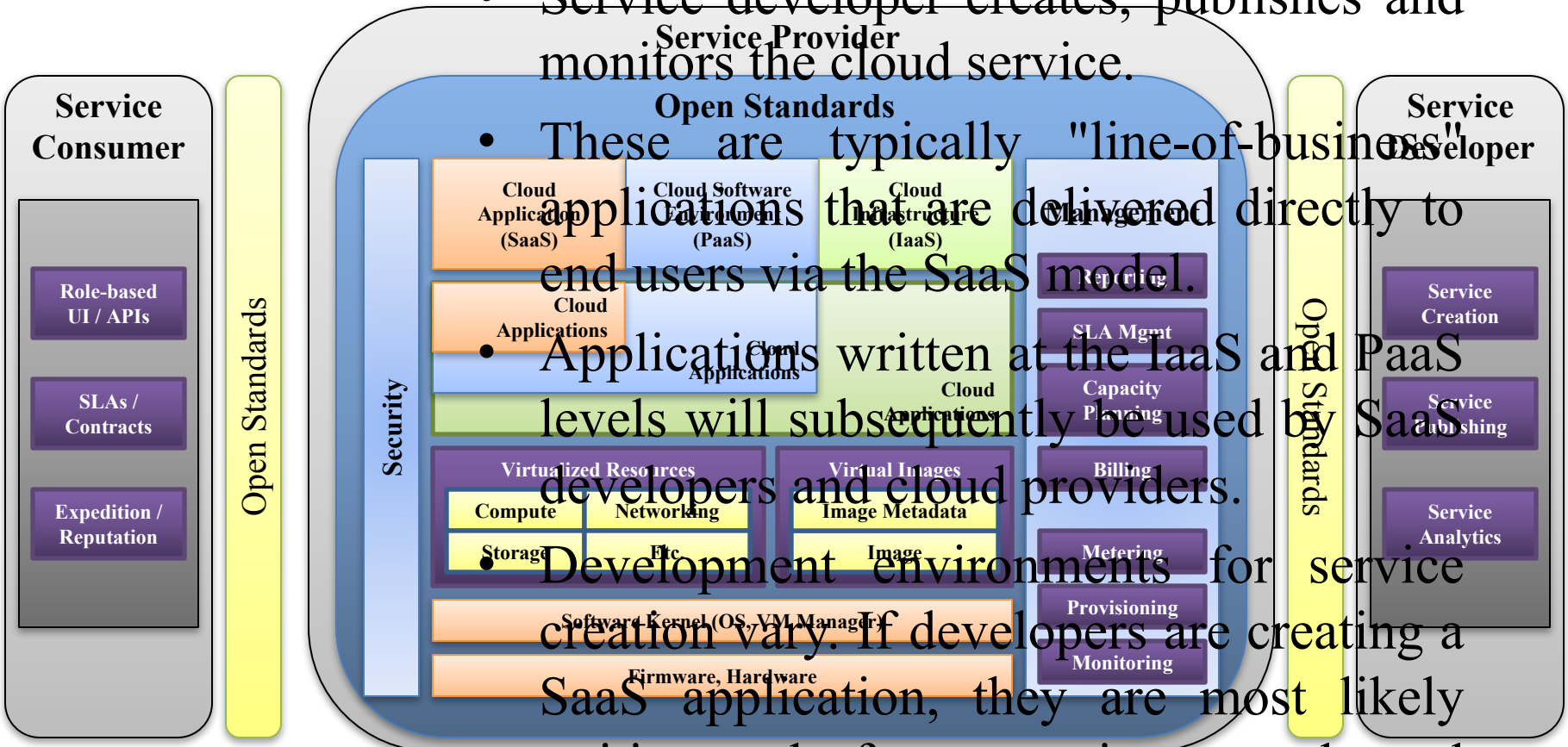
Taxonomy for cloud computing



- Service consumer is the end user or enterprise that actually uses the service, whether it is Software, Platform or Infrastructure as a Service.
- Depending on the type of service and their role, the consumer works with different user interfaces and programming interfaces.
- Consumers work with SLAs and contracts as well. Typically these are negotiated via human intervention between the consumer and the provider.

Source: Cloud Computing Use Cases White Paper Version 4.0

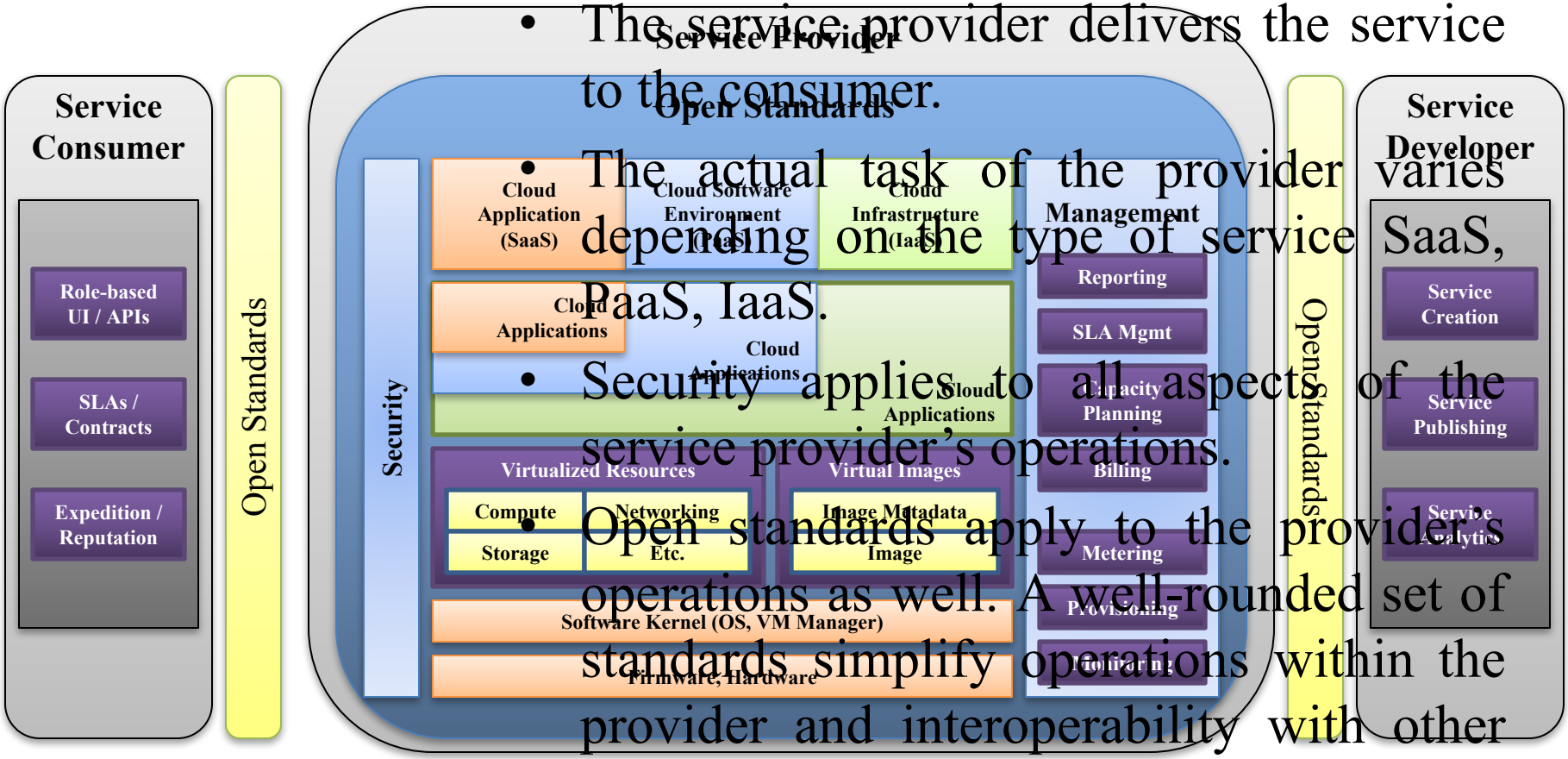
Taxonomy for cloud computing



- Service developer creates, publishes and monitors the cloud service.
- These are typically "line-of-business" applications that are delivered directly to end users via the SaaS model.
- Applications written at the IaaS and PaaS levels will subsequently be used by SaaS developers and cloud providers.
- Development environments for service creation vary. If developers are creating a SaaS application, they are most likely writing code for an environment hosted by a cloud provider.

Source: Cloud Computing Use Cases White Paper Version 4.0

Taxonomy for cloud computing



- The service provider delivers the service to the consumer.
- The actual task of the provider varies depending on the type of service SaaS, PaaS, IaaS.
- Security applies to all aspects of the service provider's operations.
- Open standards apply to the provider's operations as well. A well-rounded set of standards simplify operations within the provider and interoperability with other providers.

Source: Cloud Computing Use Cases White Paper Version 4.0

Foundational Elements of Cloud Computing

Primary Technologies

- Virtualization
- Grid technology
- Service Oriented Architectures
- Distributed Computing
- Broadband Networks
- Browser as a platform
- Free and Open Source Software

Other Technologies

- Autonomic Systems
- Web 3.0
- Web application frameworks
- Service Level Agreements

Types of cloud

1. Public Cloud
   - The cloud infrastructure is made available to the general public or a large industry group and is owned by an organization selling cloud services.
     - pay-per-use manner.
     - Amazon's Elastic Compute Cloud (EC2), Simple Storage Service (S3).
     - Google File System
     - Big Table

2. Private Cloud
   - The cloud infrastructure is operated solely for an organization. It may be managed by the organization or a third party and may exist on premise or off premise.

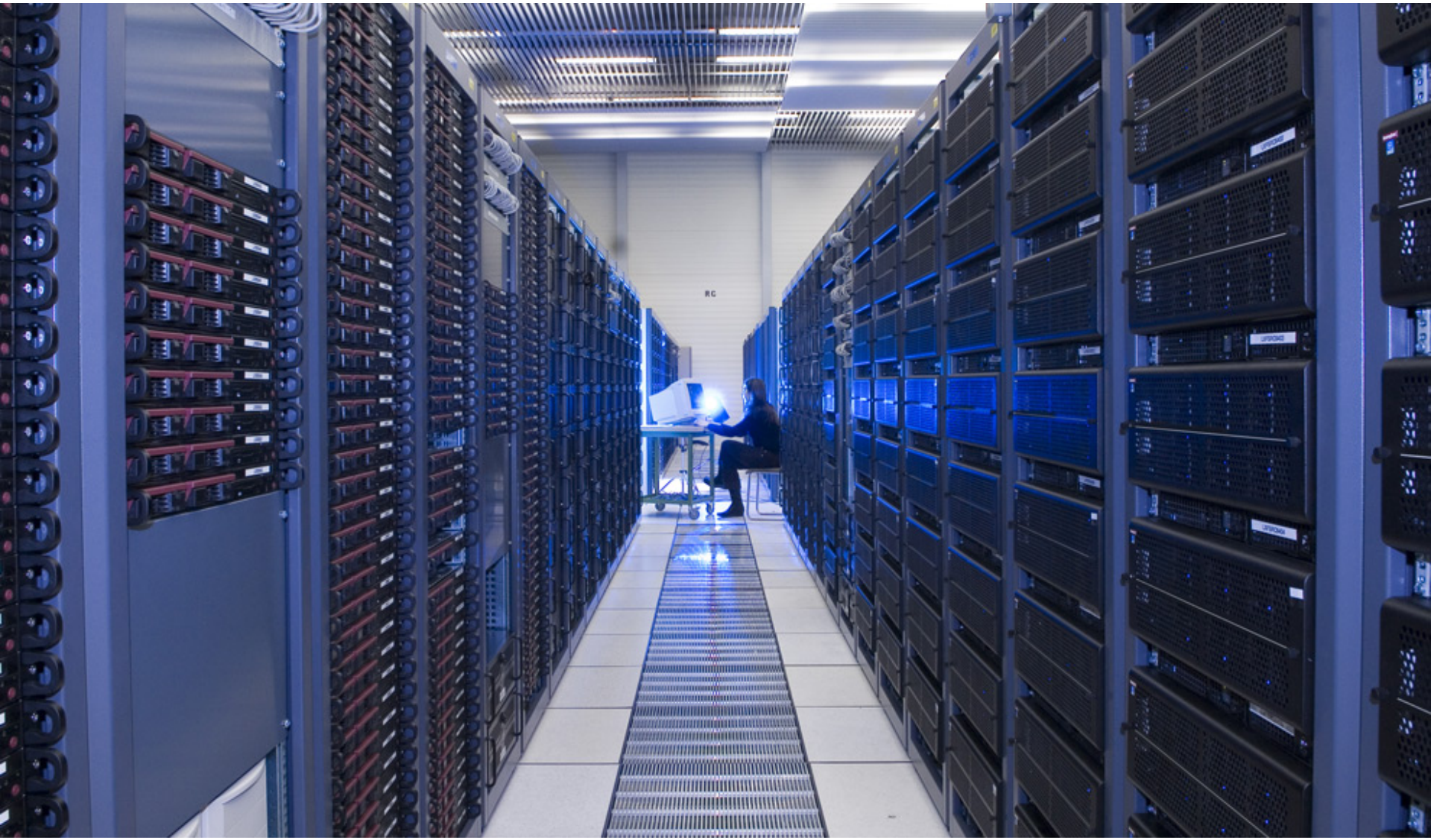
3. Hybrid Cloud
   - Whenever all the resources of a private cloud are allocated and it can not handle new requests because of lack of resources then for these new requests, it can ask assistance from a public cloud.

4. Community Cloud
   - Shared infrastructure for specific community

Application Domains of Cloud

- **Infrastructure as a Service (IaaS)** [3]
  - Infrastructure providers manage a large set of **computing resources** (such as storing and processing capacity).
  - Through virtualization, they are able to **split**, **assign** and **dynamically resize** these resources to build ad-hoc systems as demanded by customers(the service providers).
  - Rent processing, storage, network capacity, and other fundamental computing resources.

Datacenter (real IaaS)



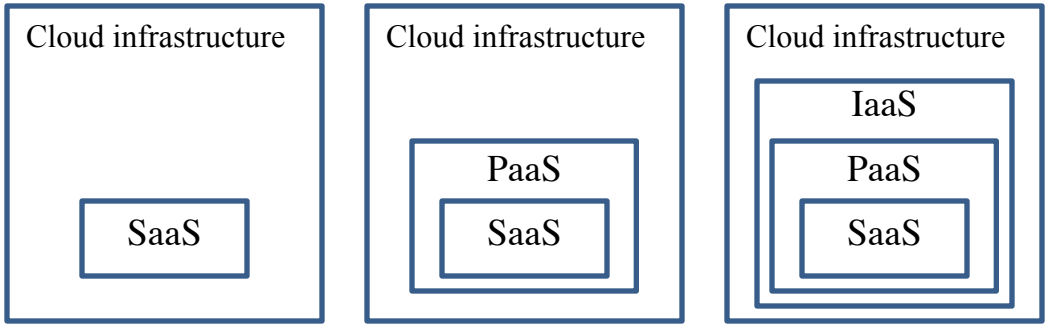
Application Domains of Cloud (Cont)

- **Platform as a Service (PaaS)** [3]
  - Instead of supplying a virtualized infrastructure, cloud systems can provide the **software platform** where applications or services can run.
  - Deploy customer-created applications on a cloud.

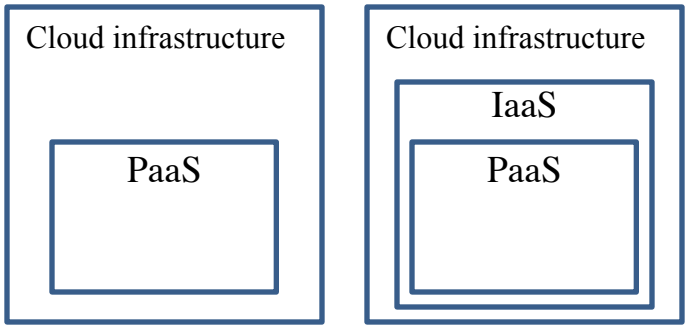
Application Domains of Cloud (Cont)

- **Software as a Service (SaaS)** [3]
  - This is an alternative to **locally run applications**.
  - Use provider's applications over a network .
  - An example of this is the online alternatives of typical office applications such as word processors.

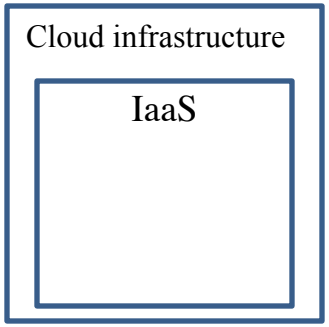
Service model architecture



Software as a Service (SaaS) Architectures

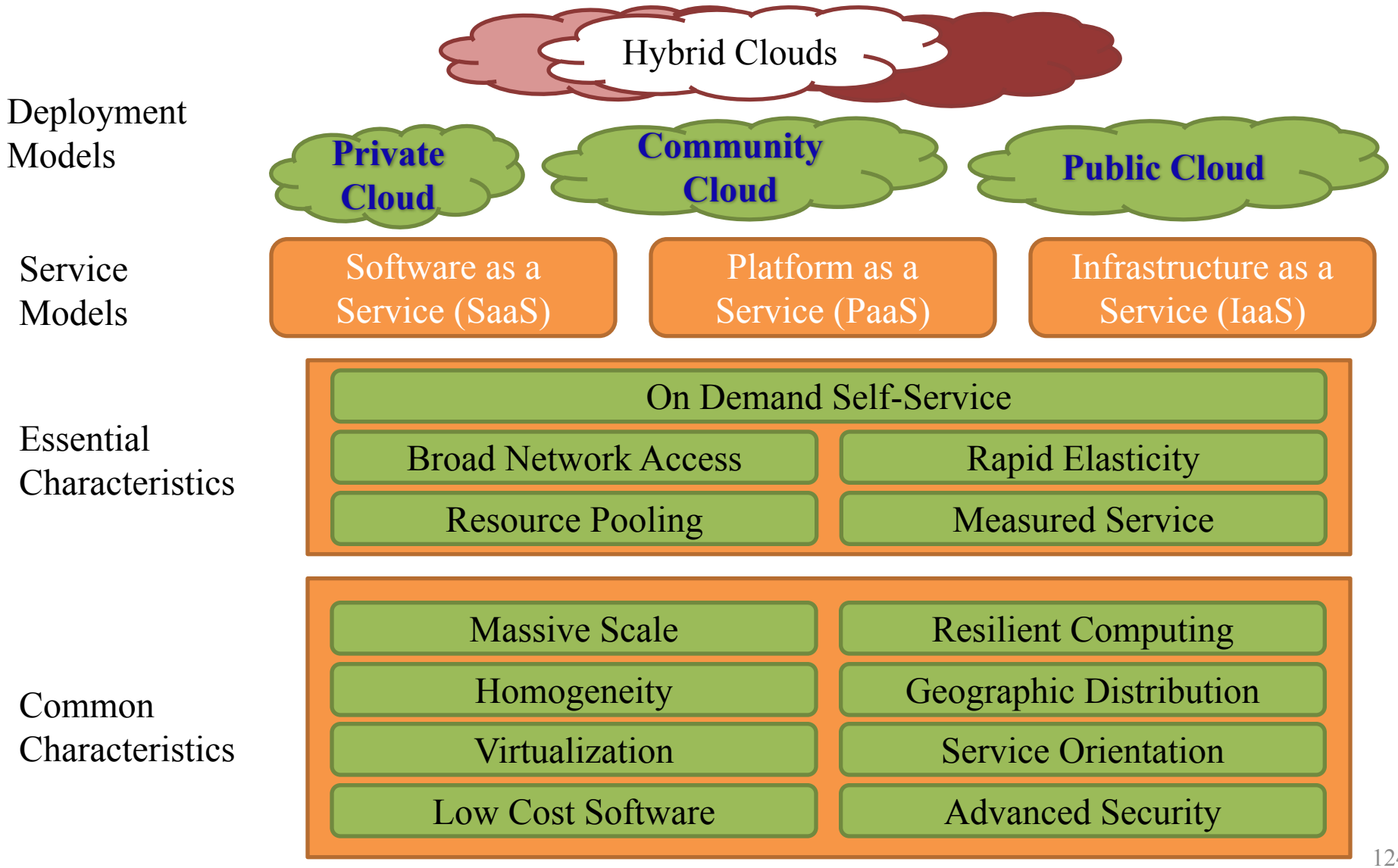


Platform as a Service (PaaS) Architectures

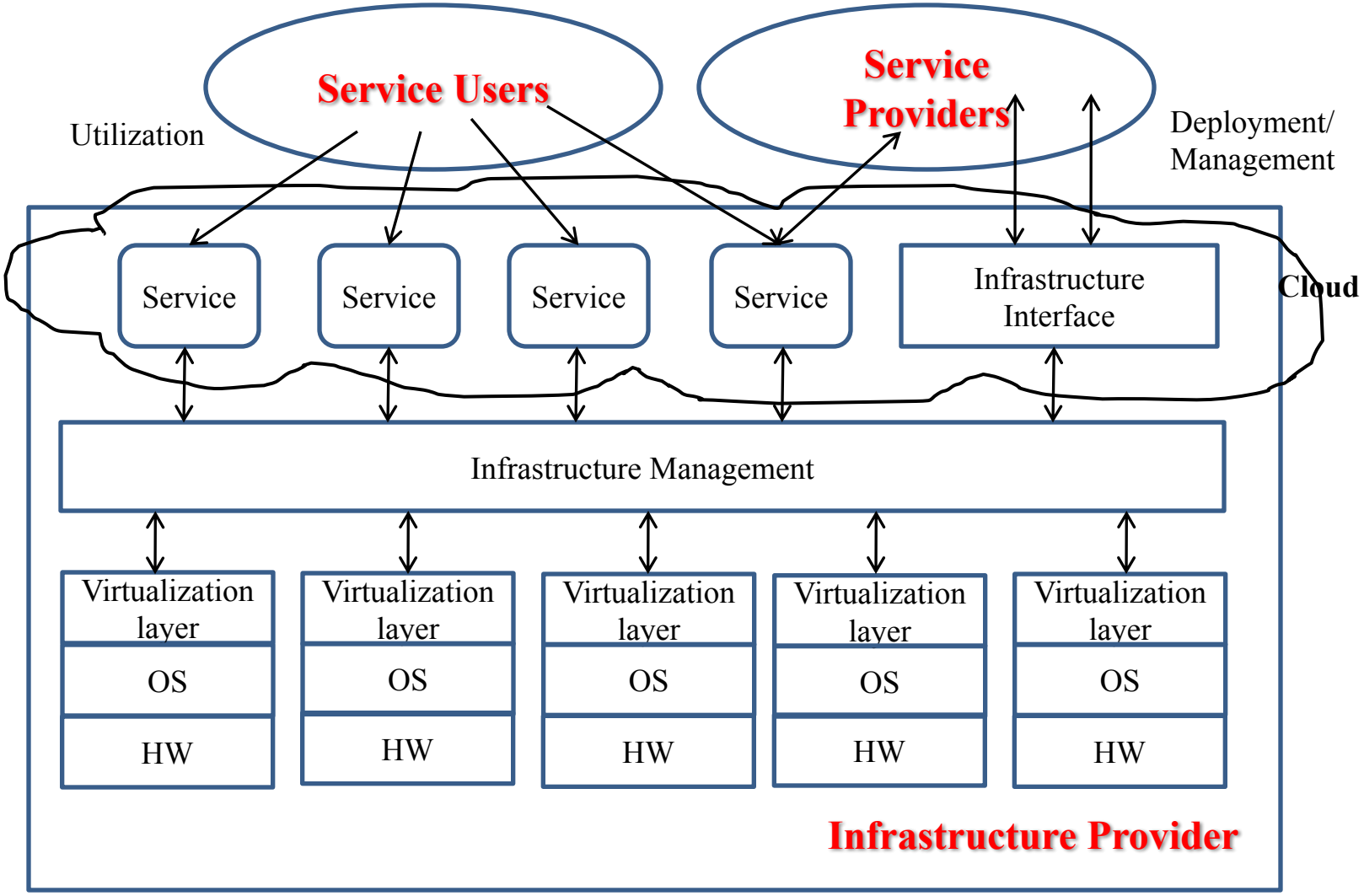


Infrastructure as a Service (IaaS) Architectures

The NIST Cloud Definition Framework



IaaS cloud actors[8]



IaaS Providers (existing)

1. **Amazon's** Elastic Compute CloudAT&T, IBM, and Unisys
2. **AT&T's** AT&T Synaptic Hosting
3. **Gni's** GNi Dedicated Hosting
4. **IBM's** Computing on Demand
5. **Rackspace's** Cloud Servers
6. **Savvis'** Savvis Open Cloud Compute
7. **ServePath's** GoGrid
8. **Skytap's** Skytap Virtual Lab
9. **3Tera's** Applogic Virtual Private Servers; Virtual Private Data Center
10. **Unisys'** Unisys Secure Infrastructure As A Service
11. **Verizon's** Verizon Computing As A Service
12. **Zimory's** Public Cloud Gateway
13. **…**

Comparing 12 IaaS providers[4]

Sr. No	Company/ Service Name	Hypervisor Platforms	OS Support	Minimum Service Contract	Service-Level Agreement Guarantee	Server Pricing
1	Amazon.com Elastic Compute Cloud	Xen	Windows Server, Red Hat, OpenSolaris, Fedora, OpenSUSE, Debian, Ubuntu, Gentoo	None	99.95%	Standard: Starts at 10 cents per hour for Linux, 12.5 cents per hour for Windows
2	AT&T AT&T Synaptic Hosting	Vmware	Windows, Red Hat	Annual	99.70%	Declined to state
3	GNi GNi Dedicated Hosting	VMware, Xen, Microsoft Virtual Server	Windows, Red Hat, CentOS, Debian, Gentoo, Ubuntu	Monthly	100%	Declined to state
4	IBM Computing on Demand	VMware, Xen	Windows Server, Red Hat, CentOS, SUSE, AIX	Annual	Depends on data center location	$5,700 annual membership, plus per-CPU pricing
5	Rackspace Cloud Servers	Xen	Red Hat, Fedora, CentOS, Debian, Ubuntu, Arch, Gentoo	None	100%	1.5 cents per hour for 256 MB RAM, 10 GB disk space
6	Savvis Savvis Open Cloud Compute	VMware	Windows Server, Red Hat, Sun Solaris 10 and x86	Monthly	99.9%; additional application SLAs up to 99.99% are available	$499 per month for a single core, 4 GB RAM and 32 GB disk space

Comparing 12 IaaS providers (cont)[4]

Sr. No	Company/ Service Name	Hypervisor Platforms	OS Support	Minimum Service Contract	Service-Level Agreement Guarantee	Server Pricing
7	ServePath GoGrid	Xen	Windows Server, CentOS,Red Hat	Monthly	99.9%; additional application SLAs up to 99.99% are available	$499 per month for a single core, 4 GB RAM and 32 GB disk space
8	Skytap Skytap Virtual Lab	VMware, Xen	Windows Server, Solaris,	None	100% uptime; for every hour of downtime, customer gets 100 hours free	19 cents per GB RAM per hour
9	3Tera Applogic Virtual Private Servers; Virtual Private Data Center	Xen	Windows, Solaris, Red Hat, SUSE, Debian, CentOS	Monthly	99.999% for virtual private data center	Starts at $500 per month
10	Unisys Unisys Secure IaaS	VMware	Windows Server, Red Hat	One year	99.90%	Declined to state
11	Verizon Verizon Computing As A Service	VMware	Windows Server, Red Hat	Monthly	100% if Verizon manages servers	$250 per month plus daily use
12	Zimory Public Cloud Gateway	Vmware, Xen	Windows Server, Red Hat, Fedora, Debian, Ubuntu	None	Three tiers: 99.8%, 99.95%, 99.99%	Gold level: 0.20 euro per hour for up to two CPUs

IaaS Providers (create your own cloud)

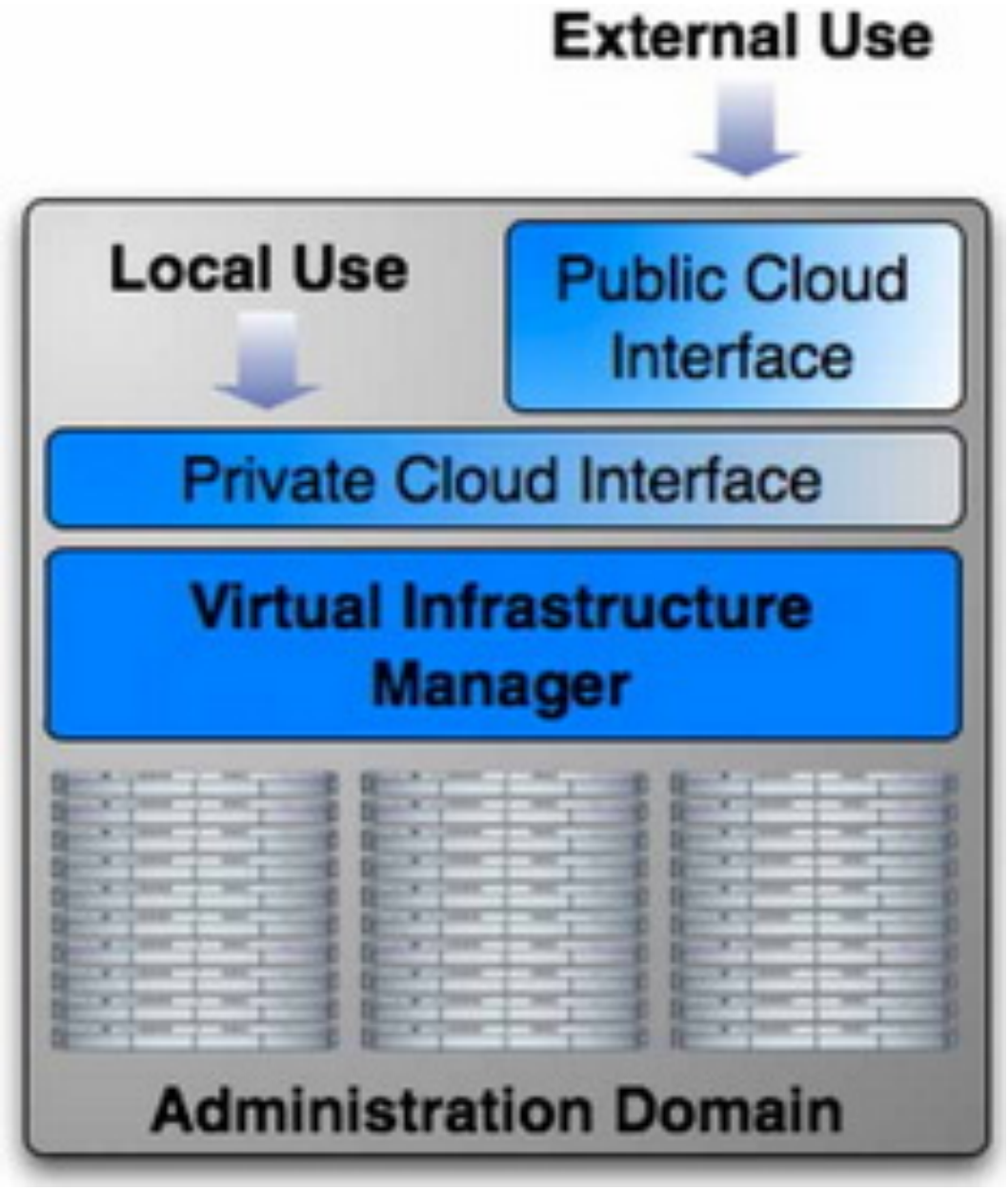
- OpenNebula[5] cloud toolkit
- EUCALYPTUS[6] (Elastic Utility Computing Architecture Linking Your Programs To Useful Systems) cloud toolkit
- Nimbus cloud toolkit

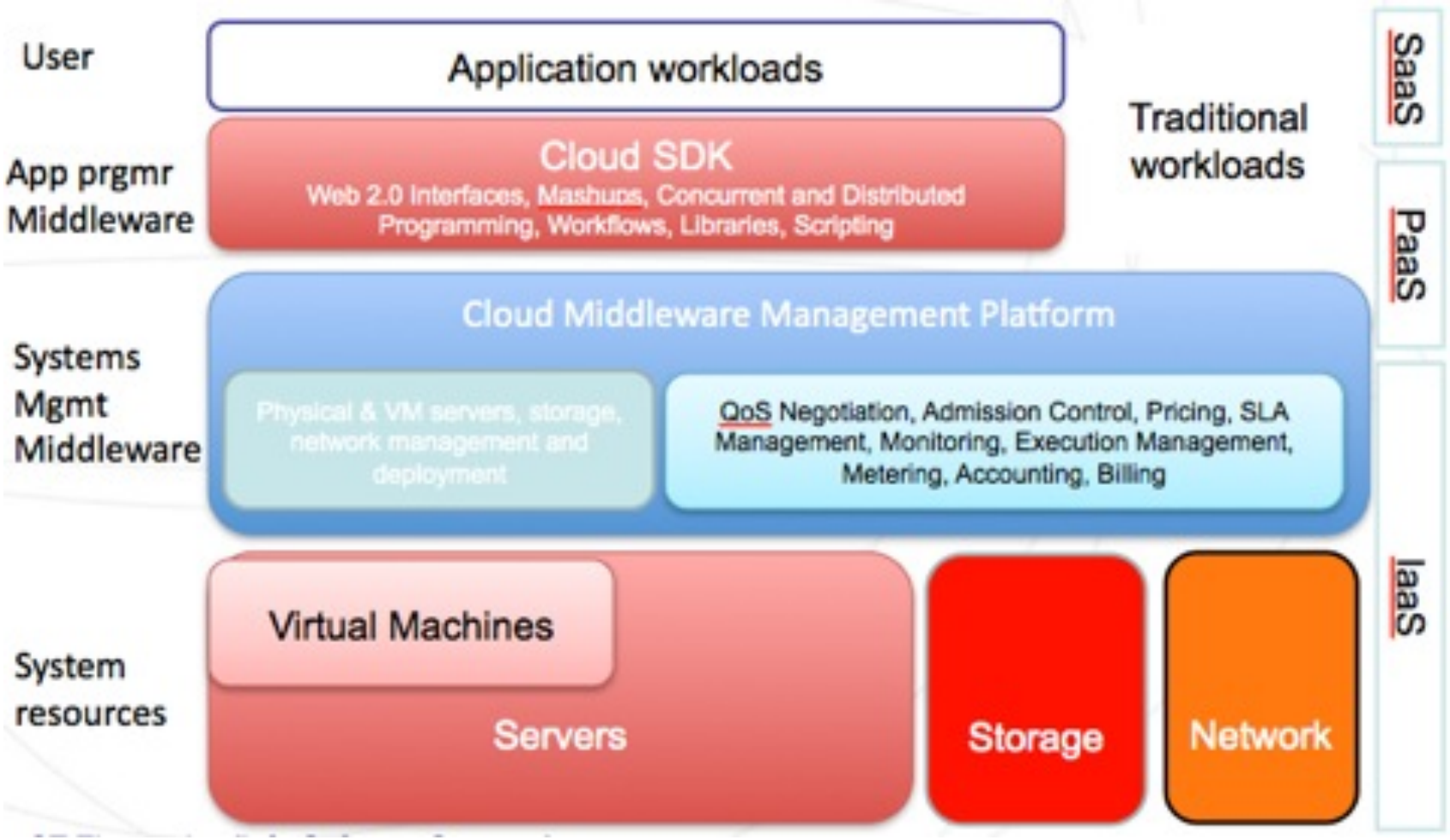
Comparison of tools[7]

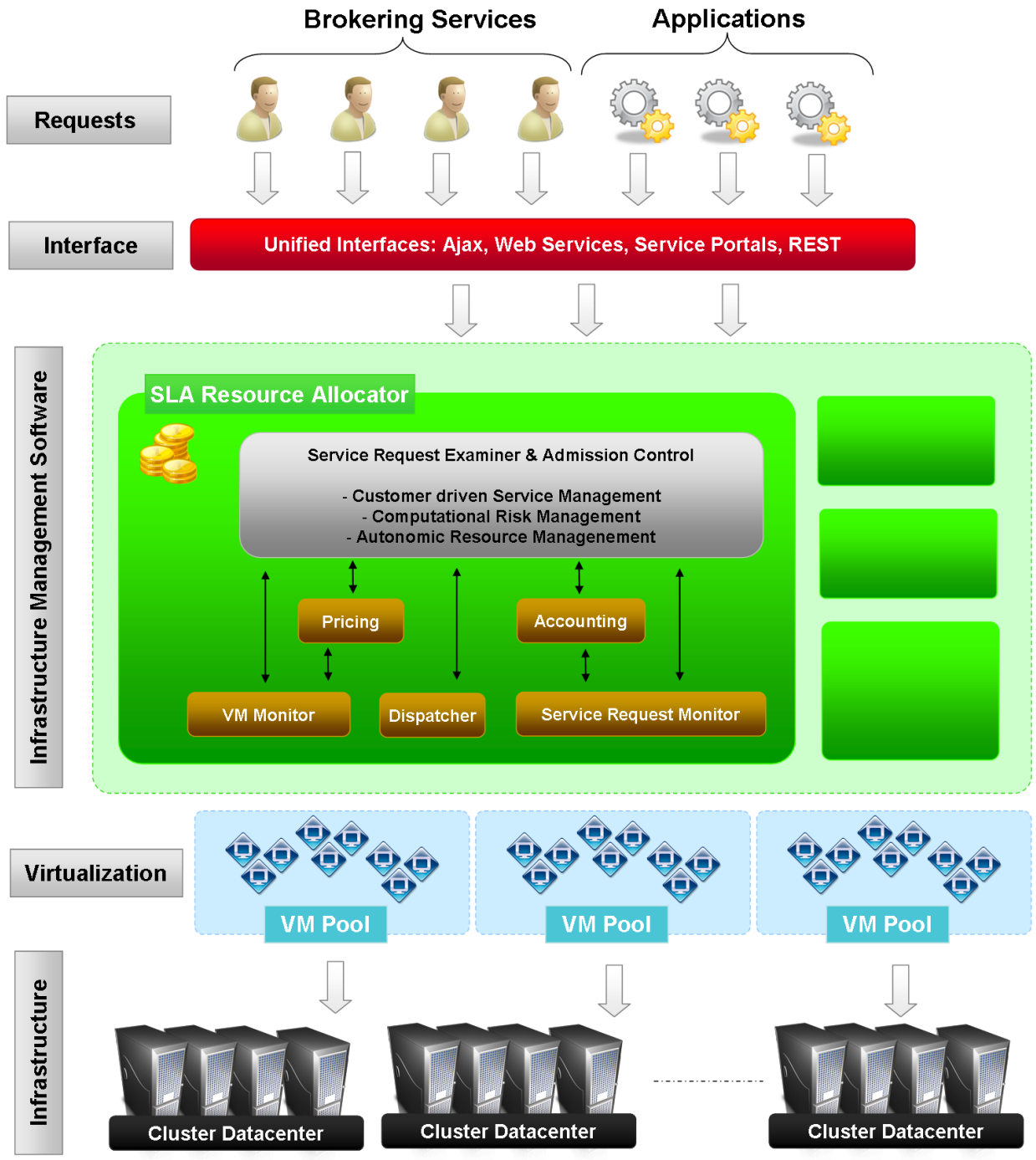
Tool	Provisioning model	Default placement policies	Configurable placement policies	Support for hybrid cloud	Remote interfaces
Amazon EC2	Best-effort	Proprietary	Proprietary	No	EC2 Web services AP
VMware vSphere	Immediate	Initial placement on CPU load and dynamic placement to balance average CPU or memory load and consolidate servers	No	Only when both the local and external cloud use vSphere	vCloud API
Platform Orchestrator	Immediate	nitial placement on CPU oad and migration policies based on policy thresholds on CPU utilization level	No	No	No
Nimbus	Immediate	Static-greedy and round-robin resource selection	No	Includes an “EC2 back end” that can forward requests to EC2, but local and remote resources must be managed separately	EC2 Web services API and Nimbus Web Services Resource Framework
Eucalyptus	Immediate	Static-greedy and round-robin resource selection	No	No	EC2 Web services API
oVirt	Immediate	Manual mode	No	No	No

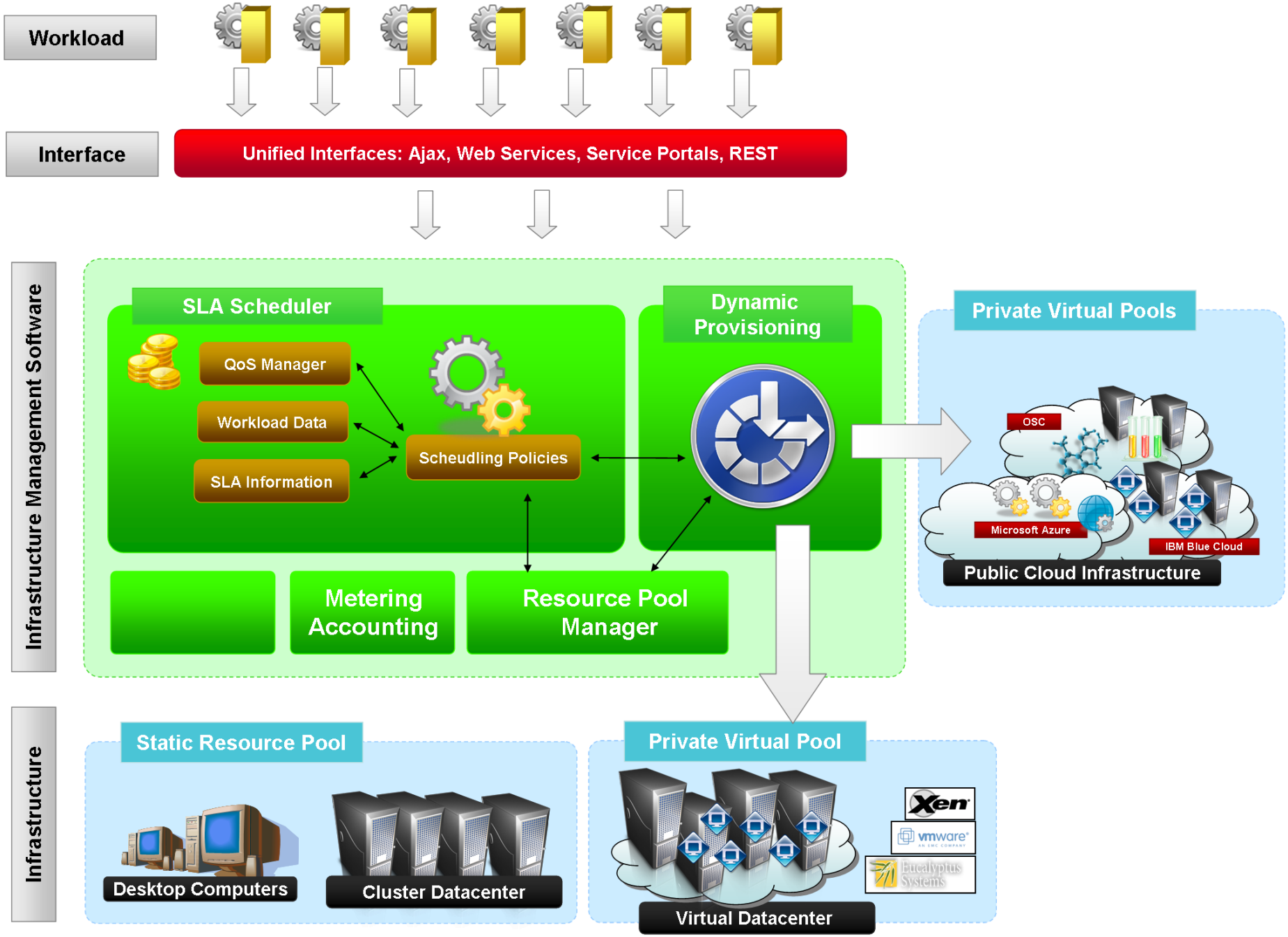
Comparison of tools[7] (cont)

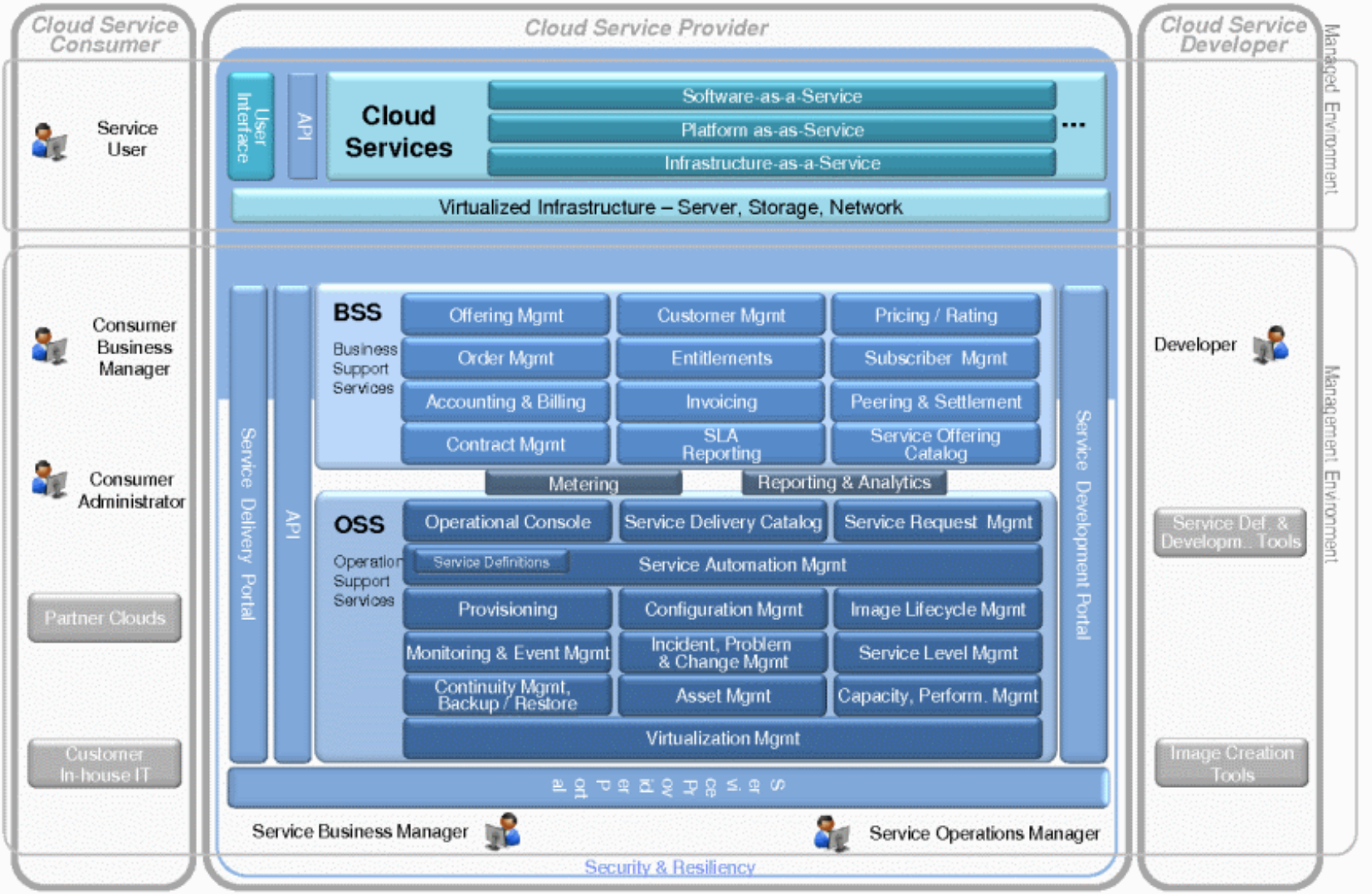
Tool	Provisioning model	Default placement policies	Configurable placement policies	Support for hybrid cloud	Remote interfaces
OpenNebula 1.2	Best-effort	Initial placement based on requirement/rank policies to prioritize those resources more suitable for the VM using dynamic information and dynamic placement to consolidate servers	Support for any static/dynamic placement policy	Driver-based architecture allows interfacing with multiple external clouds; supports EC2-compatible clouds and ElasticHosts	No
OpenNebula 1.2/ Haizea	Immediate, best-effort, and advance reservation (AR)	Dynamic placement to implement AR leases	VM placement strategies supporting queues and priorities	Driver-based architecture allows interfacing with multiple external clouds; supports EC2-compatible clouds and ElasticHosts	No
OpenNebula 1.2/ Reservoir	Immediate and best-effort	Load balancing and power-saving policies	Support for policy-driven probabilistic admission control and dynamic placement optimization to satisfy site-level management policies	Driver-based architecture allows interfacing with multiple external clouds; supports EC2-compatible clouds and ElasticHosts	Reservoir VEE (virtual execution environment) manager interface


External Use
Local Use
Public Cloud Interface
Private Cloud Interface
Virtual Infrastructure Manager
Administration Domain

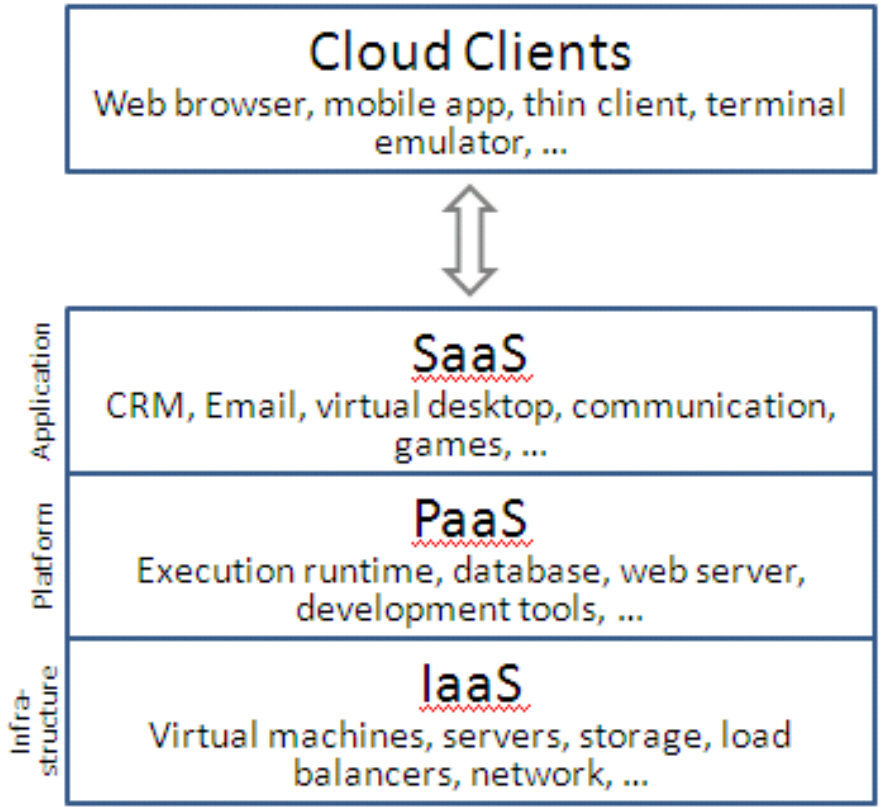

User
Application workloads
App prgmr
Middleware
Cloud SDK
Web 2.0 Interfaces, Mashups, Concurrent and Distributed Programming, Workflows, Libraries, Scripting
Traditional workloads
SaaS
PaaS
Systems
Mgmt
Middleware
Cloud Middleware Management Platform
Physical & VM servers, storage, network management and deployment
QoS Negotiation, Admission Control, Pricing, SLA Management, Monitoring, Execution Management, Metering, Accounting, Billing
IaaS
System
resources
Virtual Machines
Servers
Storage
Network


Brokering Services
Applications
Requests
Interface
Unified Interfaces: Ajax, Web Services, Service Portals, REST
Infrastructure Management Software
SLA Resource Allocator
Service Request Examiner & Admission Control
- Customer driven Service Management
- Computational Risk Management
- Autonomic Resource Managenement
Pricing
Accounting
VM Monitor
Dispatcher
Service Request Monitor
Virtualization
VM Pool
VM Pool
VM Pool
Infrastructure
Cluster Datacenter
Cluster Datacenter
Cluster Datacenter


Workload
Interface
Unified Interfaces: Ajax, Web Services, Service Portals, REST
Infrastructure Management Software
SLA Scheduler
QoS Manager
Workload Data
SLA Information
Scheduling Policies
Dynamic Provisioning
Private Virtual Pools
OSC
Microsoft Azure
IBM Blue Cloud
Public Cloud Infrastructure
Metering Accounting
Resource Pool Manager
Infrastructure
Static Resource Pool
Desktop Computers
Cluster Datacenter
Private Virtual Pool
Xen
vmware
Eucalyptus Systems
Virtual Datacenter



Layers of Cloud computing



PaaS

- In the PaaS model, cloud providers deliver a computing platform typically including operating system, database, and web server.
- PaaS allows developers not to worry about software installation and cloud infrastructure management and allows for rapid application development.
- The application developer need not be worried about the underlying OS.

Cloud Foundry: a PaaS implementation

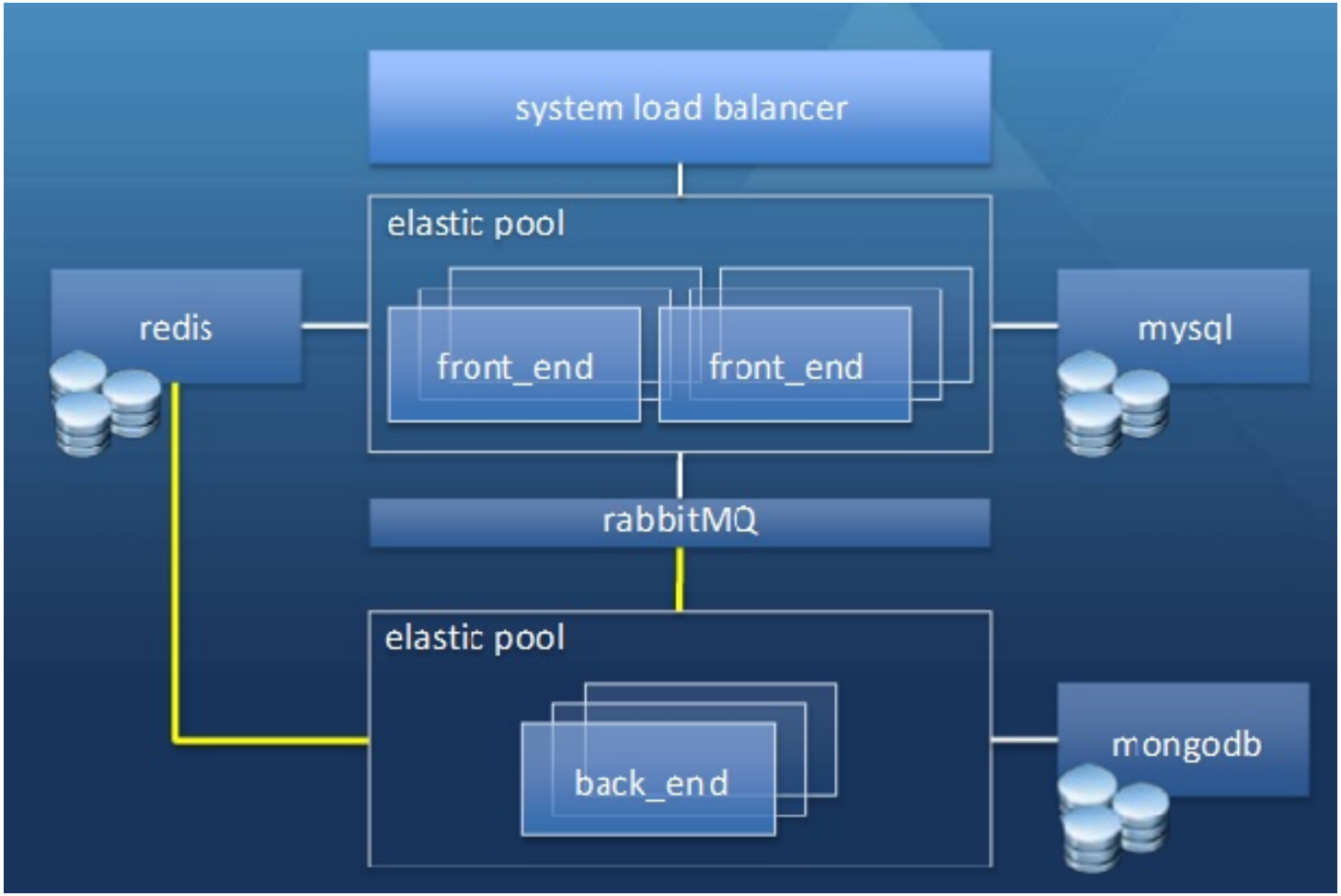
- Cloud Foundry is an open source cloud computing platform as a service (PaaS) software developed by VMware.

- It allows rapid deployment of application written in many languages including
  - Spring Java, Ruby on Rails and many others.
  - It also provides services such as PostgreSql, Mysql, MongoDB and many more.

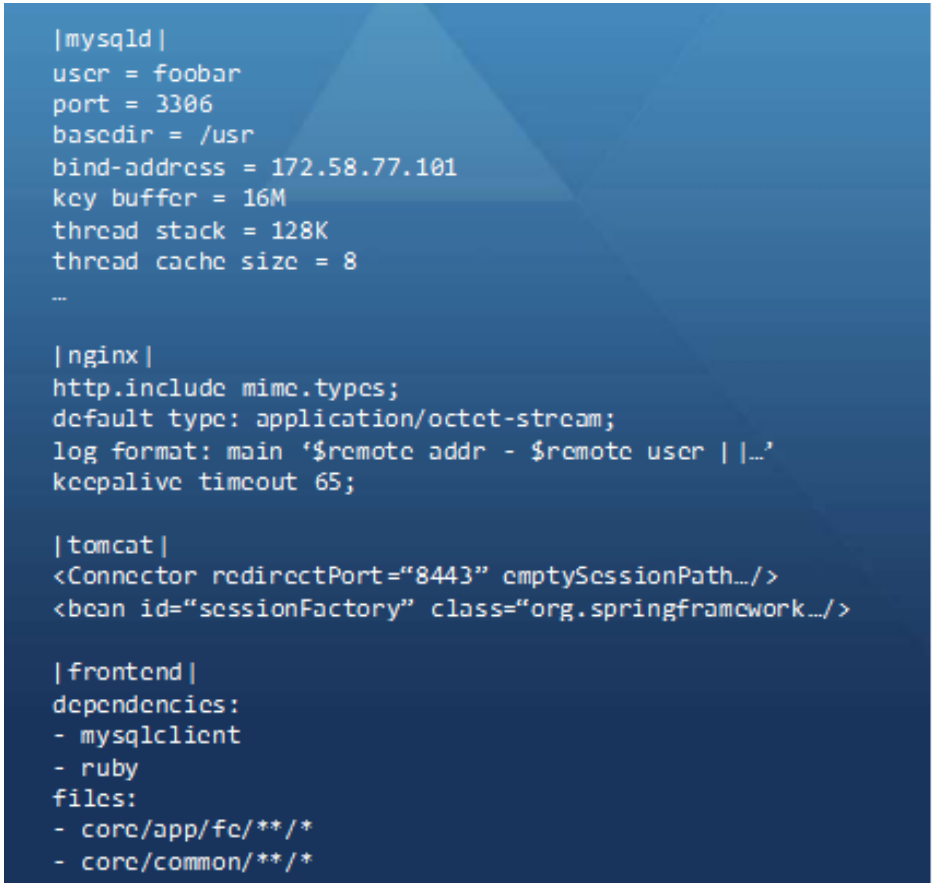
Why cloud foundry

- Open source
- Portability without changing code
- Public and private cloud
- Scope for rapid improvement

Architecture of cloudfoundry



Visible benefits of using cloudfoundry



```
|mysqld|
user = foobar
port = 3306
basedir = /usr
bind-address = 172.58.77.101
key buffer = 16M
thread stack = 128K
thread cache size = 8
…

|nginx|
http.include mime.types;
default type: application/octet-stream;
log format: main '$remote addr - $remote user ||…'
keepalive timeout 65;

|tomcat|
<Connector redirectPort="8443" emptySessionPath…/>
<bean id="sessionFactory" class="org.springframework…/>

|frontend|
dependencies:
- mysqlclient
- ruby
files:
- core/app/fe/**/*
- core/common/**/*
```

Deploying an application the usual way

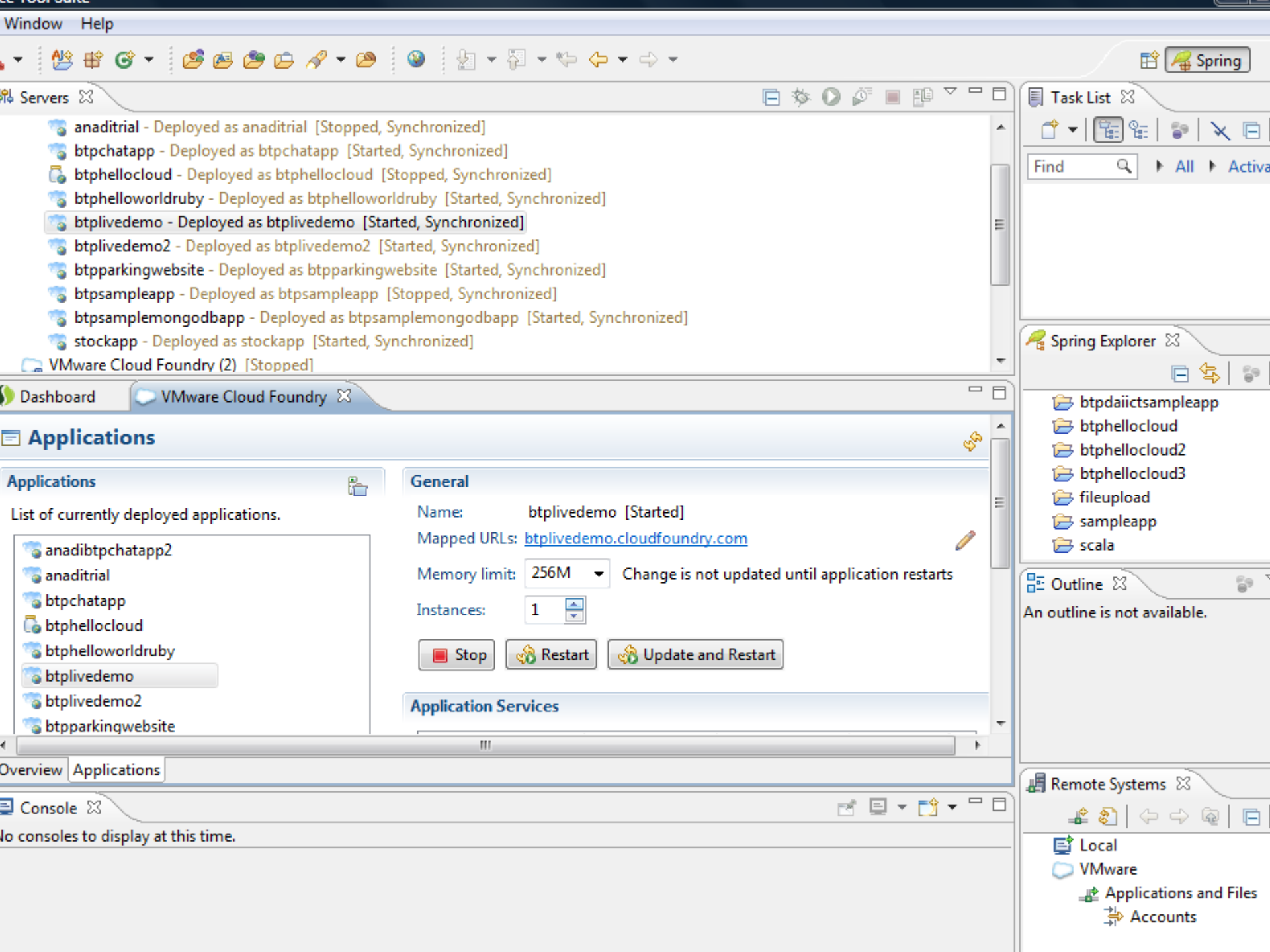
```
# to target and login to cloud foundry
vmc target http://api.cloudfoundry.com
vmc login

# to create and boot the app for the first time
vmc push myapp -instances 2 -mem 64M -path ../code

# to create the database and bind it to the app
vmc create-service mysql -name mydb -bind myapp

# update live app with new code
vmc update myapp -path ../code
```

Deploying similar code in cloud foundry

Tools to deploy code on cloud foundry

- SpringSource tool
- Vmc – a command line utility

- Install ruby
- Install vmc using command ‘gem install vmc’
- Install apache maven2


Window
Help
Spring
Servers
anaditrial - Deployed as anaditrial [Stopped, Synchronized]
btpchatapp - Deployed as btpchatapp [Started, Synchronized]
btphellocloud - Deployed as btphellocloud [Stopped, Synchronized]
btphelloworldruby - Deployed as btphelloworldruby [Started, Synchronized]
btplivedemo - Deployed as btplivedemo [Started, Synchronized]
btplivedemo2 - Deployed as btplivedemo2 [Started, Synchronized]
btpparkingwebsite - Deployed as btpparkingwebsite [Started, Synchronized]
btpsampleapp - Deployed as btpsampleapp [Stopped, Synchronized]
btpsamplemongodbapp - Deployed as btpsamplemongodbapp [Started, Synchronized]
stockapp - Deployed as stockapp [Started, Synchronized]
VMware Cloud Foundry (2) [Stopped]
Task List
Find
All
Activa
Spring Explorer
btpdaiictsampleapp
btphellocloud
btphellocloud2
btphellocloud3
fileupload
sampleapp
scala
Outline
An outline is not available.
Dashboard
VMware Cloud Foundry
Applications
Applications
List of currently deployed applications.
anadibtpchatapp2
anaditrial
btpchatapp
btphellocloud
btphelloworldruby
btplivedemo
btplivedemo2
btpparkingwebsite
General
Name: btplivedemo [Started]
Mapped URLs: btplivedemo.cloudfoundry.com
Memory limit: 256M
Change is not updated until application restarts
Instances: 1
Stop
Restart
Update and Restart
Application Services
Overview
Applications
Console
No consoles to display at this time.
Remote Systems
Local
VMware
Applications and Files
Accounts

Micro Cloud Foundry

- Allows setup of a complete instance of Cloud foundry on a local computer
- Hardware requirements:

✓ 64 bit compatible processors.

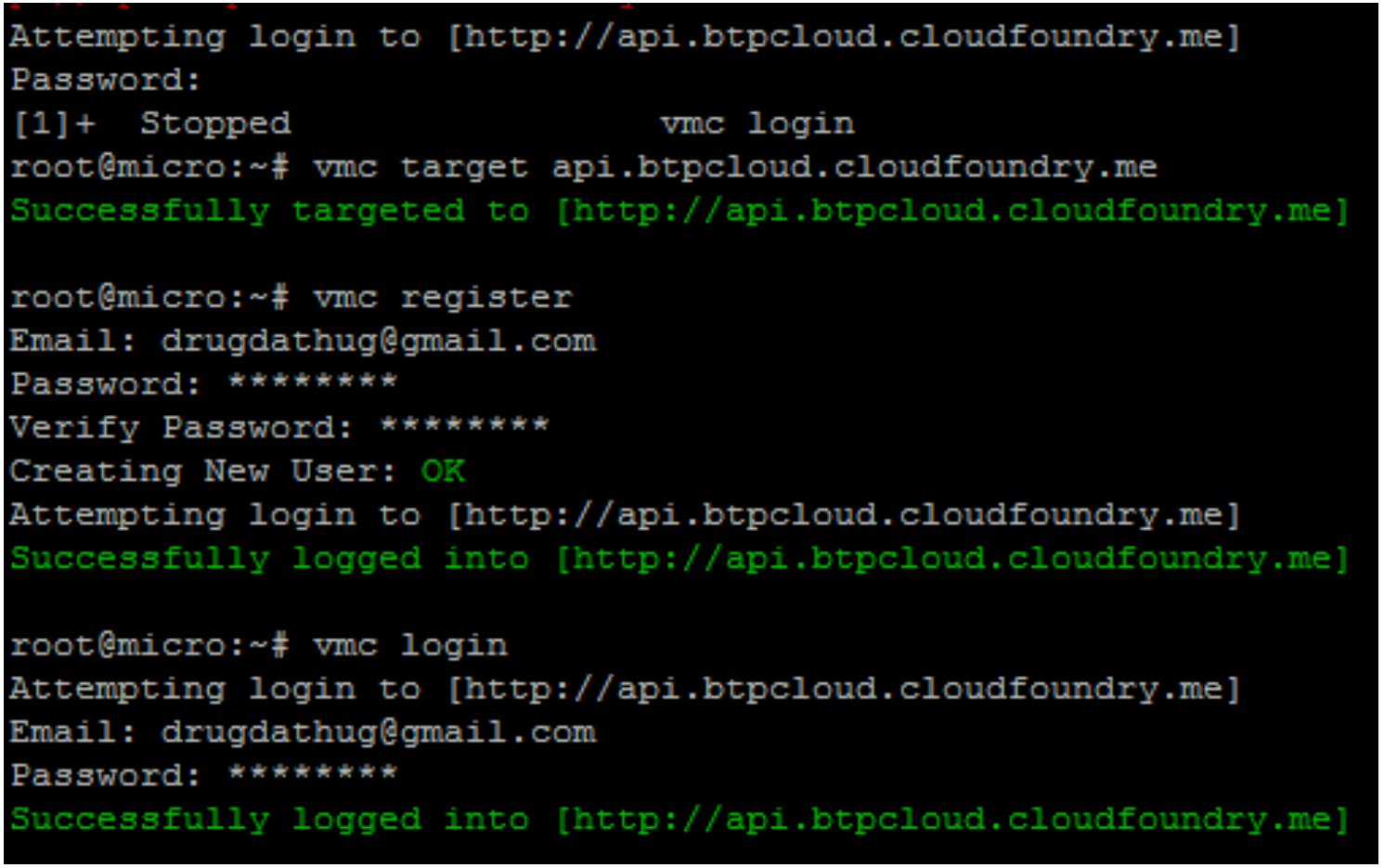
✓ 64 bit OS.

✓ VT technology must be enabled.

- Domain tokens

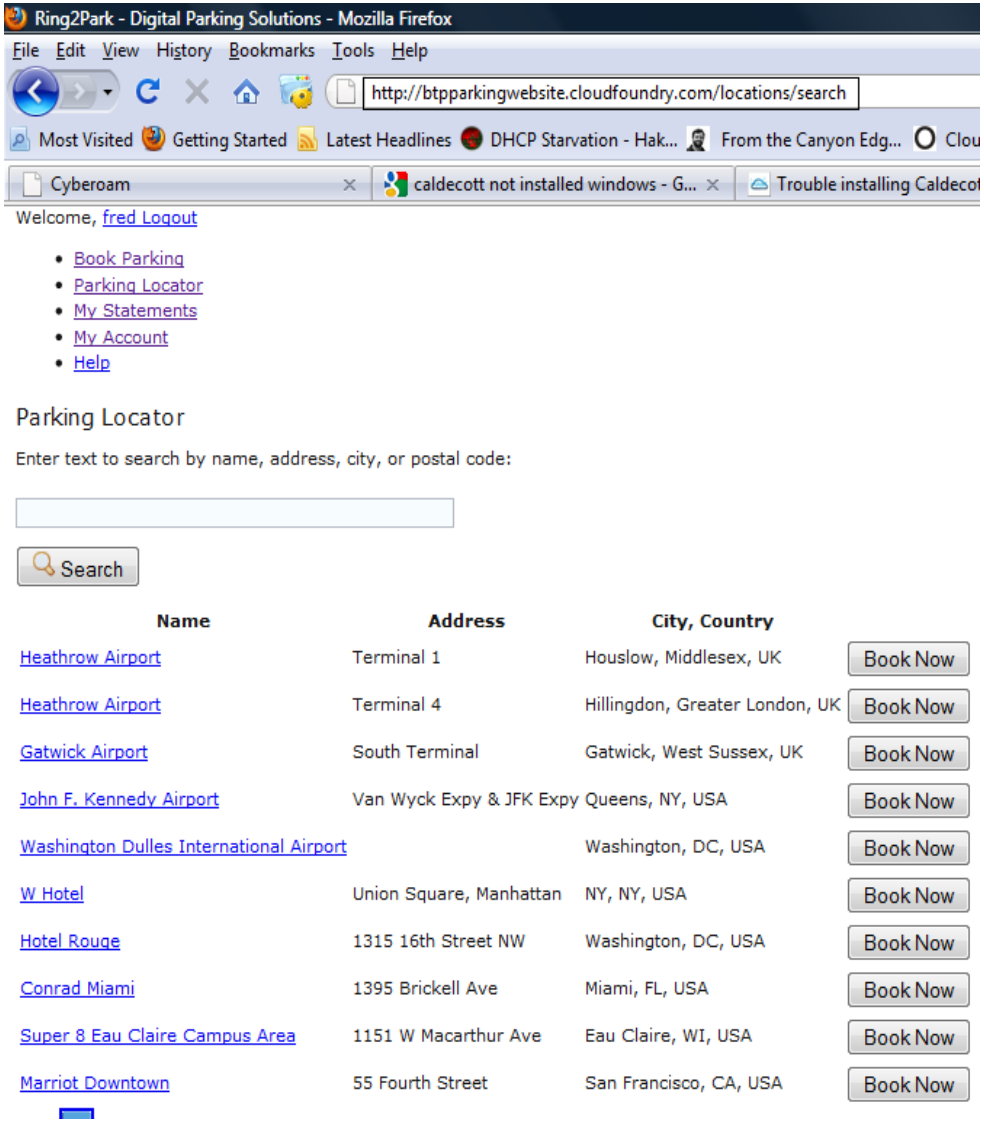
Installing Micro cloud foundry

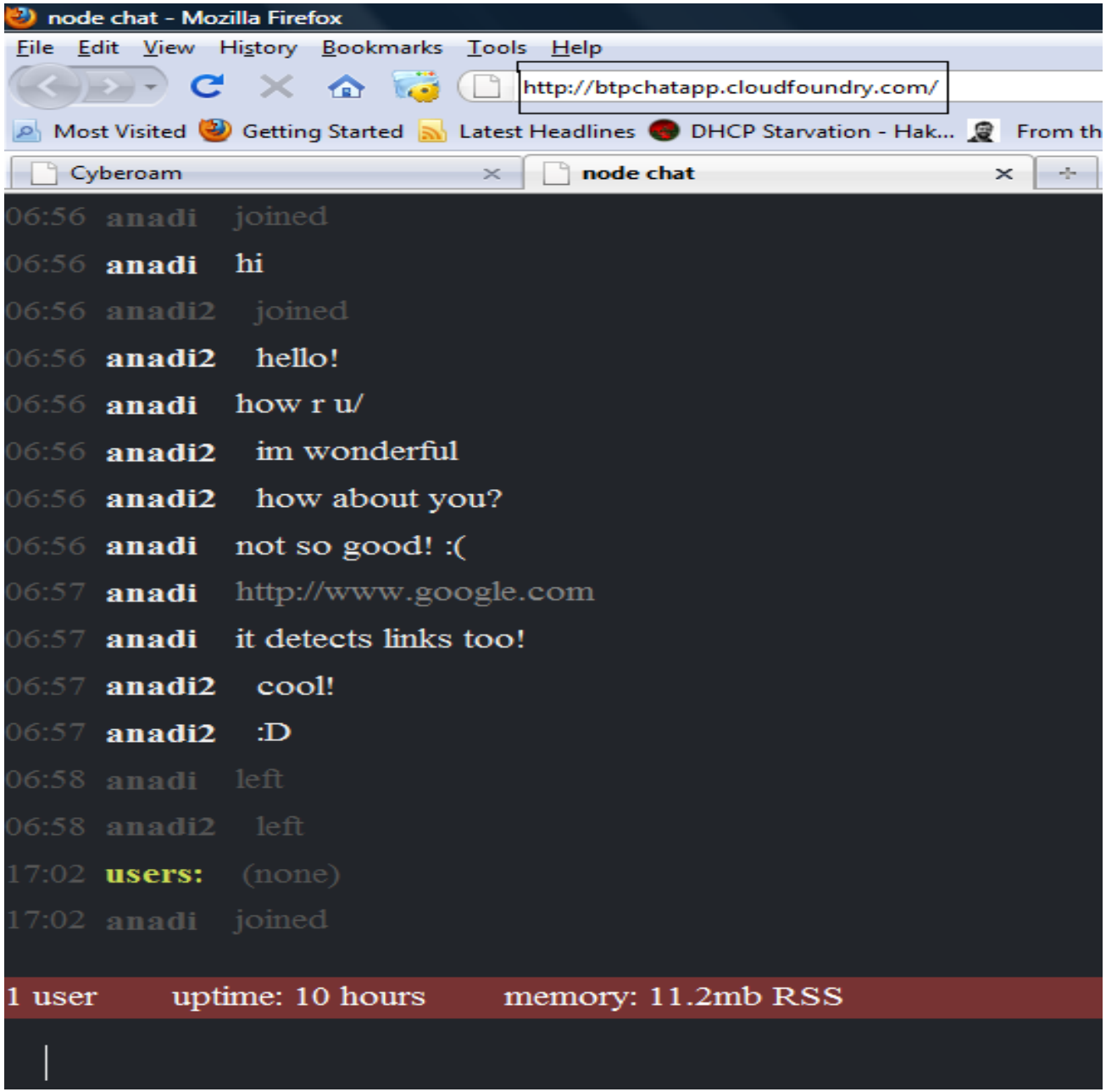
- Install vm player
- Download the micro cloud foundry virtual image. It is preconfigured with cloud settings and cloud foundry api.
- Enter settings such as network settings, username password and domain tokens.

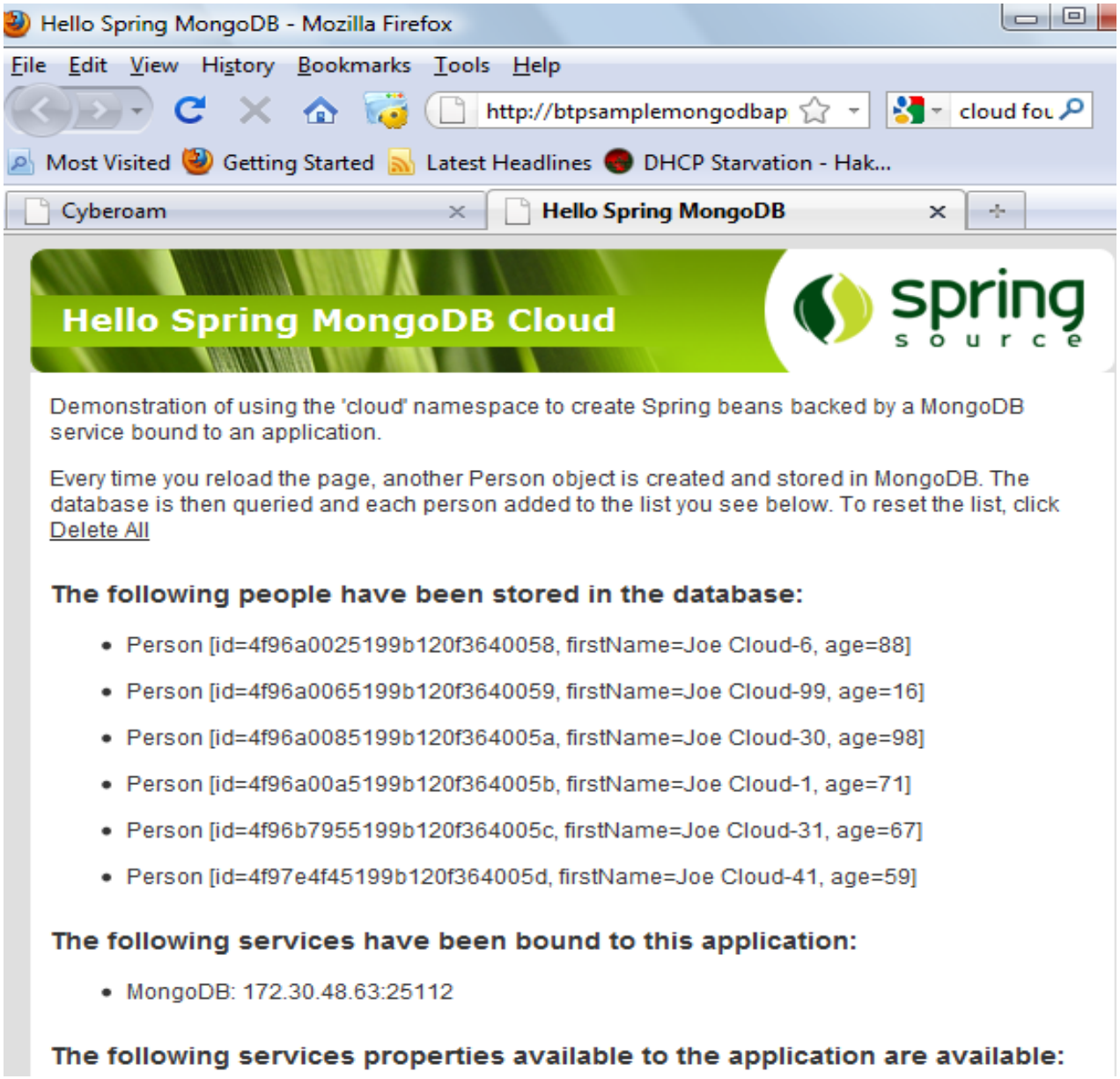


Screenshot of MFC installed in our lab

Sample applications




node chat - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://btpchatapp.cloudfoundry.com/
Most Visited Getting Started Latest Headlines DHCP Starvation - Hak... From th
Cyberoam
node chat
06:56 anadi joined
06:56 anadi hi
06:56 anadi2 joined
06:56 anadi2 hello!
06:56 anadi how r u/
06:56 anadi2 im wonderful
06:56 anadi2 how about you?
06:56 anadi not so good! :(
06:57 anadi http://www.google.com
06:57 anadi it detects links too!
06:57 anadi2 cool!
06:57 anadi2 :D
06:58 anadi left
06:58 anadi2 left
17:02 users: (none)
17:02 anadi joined
1 user uptime: 10 hours memory: 11.2mb RSS

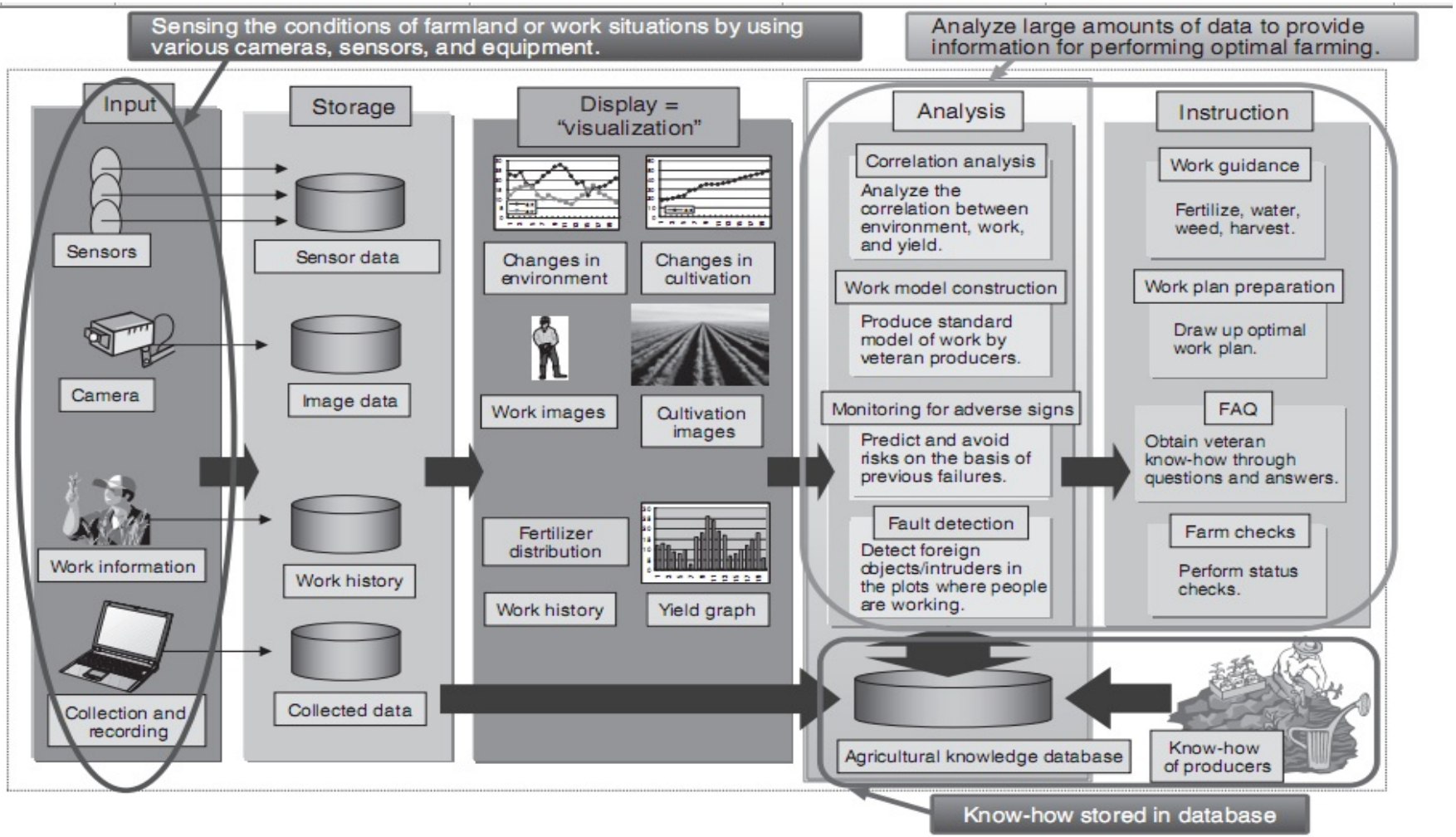

Hello Spring MongoDB - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://btpsamplemongodbap
cloud fou
Most Visited
Getting Started
Latest Headlines
DHCP Starvation - Hak...
Cyberoam
Hello Spring MongoDB
Hello Spring MongoDB Cloud
spring source
Demonstration of using the 'cloud' namespace to create Spring beans backed by a MongoDB service bound to an application.
Every time you reload the page, another Person object is created and stored in MongoDB. The database is then queried and each person added to the list you see below. To reset the list, click Delete All
The following people have been stored in the database:
Person [id=4f96a0025199b120f3640058, firstName=Joe Cloud-6, age=88]
Person [id=4f96a0065199b120f3640059, firstName=Joe Cloud-99, age=16]
Person [id=4f96a0085199b120f364005a, firstName=Joe Cloud-30, age=98]
Person [id=4f96a00a5199b120f364005b, firstName=Joe Cloud-1, age=71]
Person [id=4f96b7955199b120f364005c, firstName=Joe Cloud-31, age=67]
Person [id=4f97e4f45199b120f364005d, firstName=Joe Cloud-41, age=59]
The following services have been bound to this application:
MongoDB: 172.30.48.63:25112
The following services properties available to the application are available:

Conclusion

- Application developers should not be worried about underlying hardware, platform and resource requirements.
- Fortunately Cloud Foundry developed by VMware addresses many such issues.
- There is still a lack of detailed documentation and troubleshooting guide.
- The author found that the SpringSource Tool suite is unstable and not very user friendly and the best way are to use vmc, the command line mode.
- Otherwise the services provided by cloud foundry such as the infrastructure hosted on cloudfoundry.com is top notch and this initiative will definitely benefit the IT community.

Applications of Cloud Computing

Application of Cloud Computing to Agriculture and Prospects in Other Fields



FAQ: Answers to frequently asked questions

Application of Cloud Computing to Agriculture and Prospects in Other Fields (Cont.)

- Figure describes agricultural work with IT through the flow sequence (input → data storage → visualization → analysis → instruction).
- Web applications and mobile phone applications to build prototypes of the four functions listed below.

1. Sales/planting (production) planning:
   - Sales to customers and planting (production) planning for cultivated land can be performed together.
2. Operational planning/results management:
   - Progress management and operational checks can be performed on the basis of preplanned on-site work and automatic collection of results.

Application of Cloud Computing to Agriculture and Prospects in Other Fields (Cont.)

3. Patrolling support:
   - Reports and instructions can be easily and reliably issued by sharing on-site photographs and comments among all administrators and workers.
4. Cultivated land data management:
   - Management of all sorts of data relating to cultivated land, including location, land rights, area, soil, and land characteristics can be integrated

- Above four functions uses two data management technologies:

Application of Cloud Computing to Agriculture and Prospects in Other Fields (Cont.)

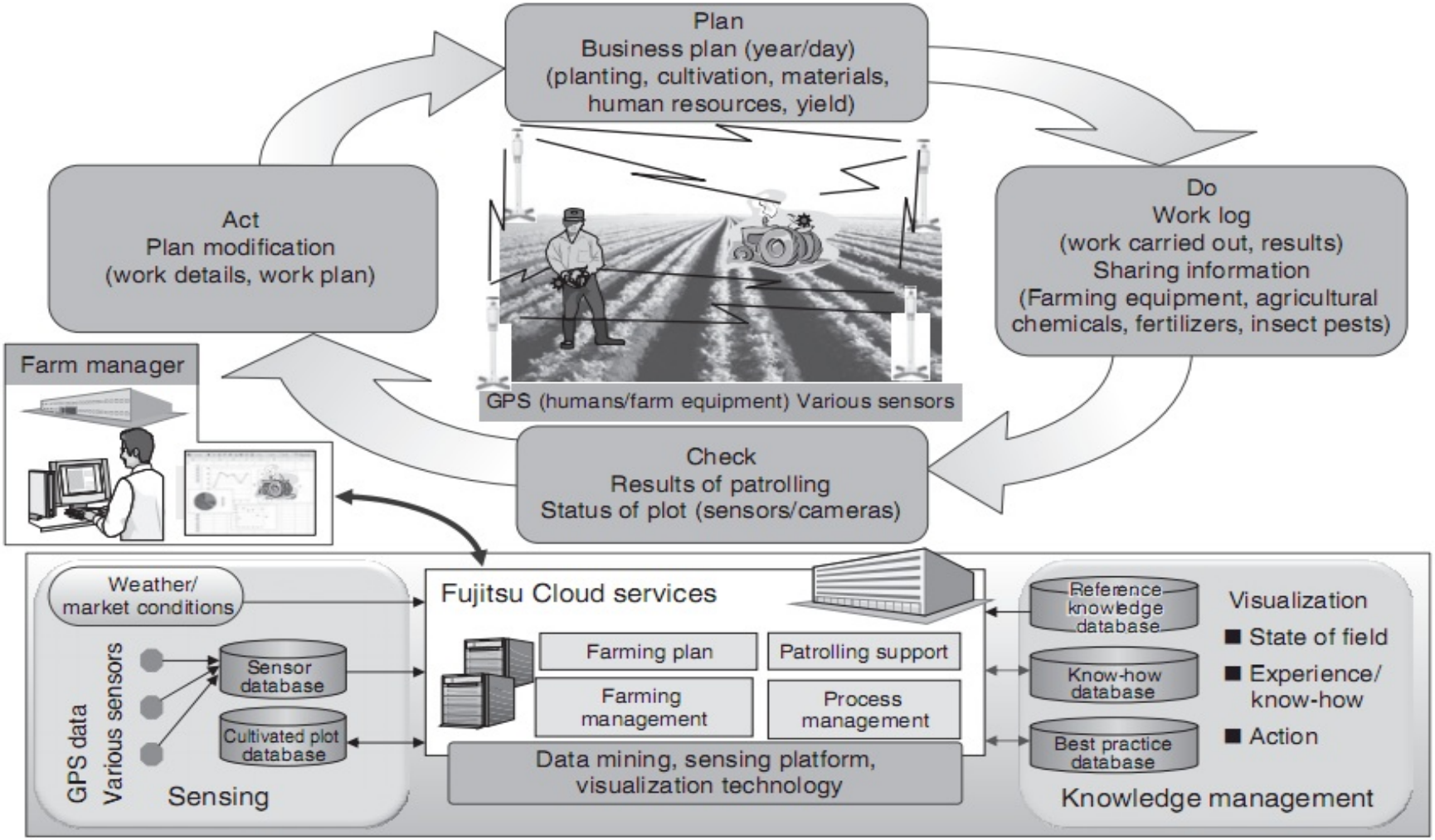
1. Data storage

- Position and time information from mobile phones with GPS functions
- Weather/soil sensor data
- Image and audio data obtained by mobile phones (with digital camera and audio recorder applications)
- Noteworthy data extracted from the results of routine work
- Materials management data obtained using mobile phones with barcode reading functions

2. Data analysis

- Registering and updating virtual models
- Data mining

Application of Cloud Computing to Agriculture and Prospects in Other Fields (Cont.)



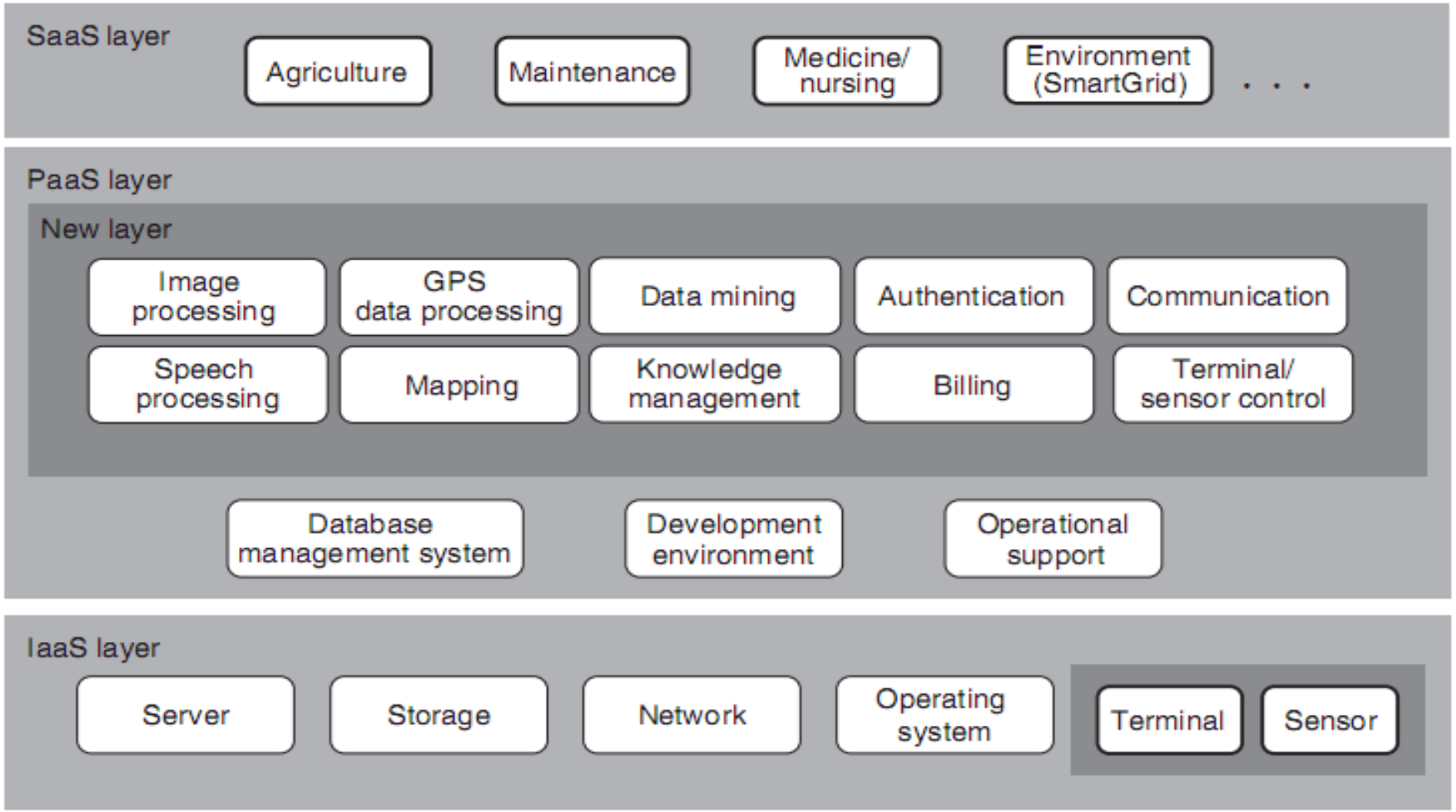
Application of Cloud Computing to Agriculture and Prospects in Other Fields (Cont.)

- Plan: Draw up production and operation plans.
- Do: Gather work results (this involves performing the actual work on-site, though IT support cannot be provided for this).
- Check: Perform progress management and patrol the cultivated plots.
- Act: Make any necessary modifications to the plans.
- Routinely collected data includes weather and soil data, GPS data, image data, worker observations, and data related to cultivated plots of land.
- The quantity is 5–10 megabytes per case per day.

Application of Cloud Computing to Agriculture and Prospects in Other Fields (Cont.)

- Agricultural data has to be stored for 10–30 years according to a report by the National Agriculture and Food Research Organization5) and
- between half a million and one million cases are targeted in the development of business, the total amount of data exceeds 100 petabytes (less than the medical field where the data storage requirements of personal health records are expected to reach 2 petabytes per patient).
- To obtain advice and recommendations by analyzing this stored data, analysis engines such as data miners are operated on this large amount of data in the Cloud.

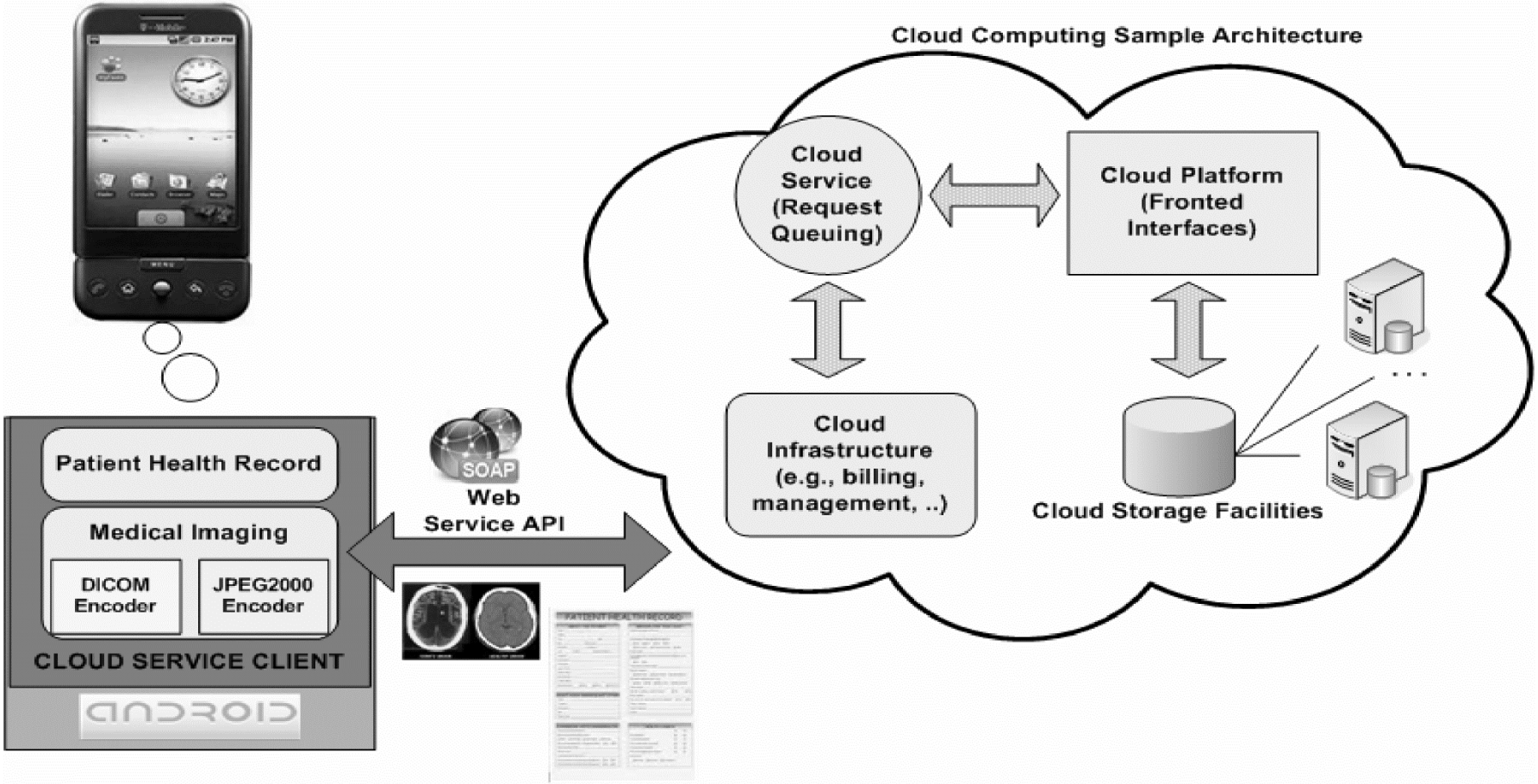
Application of Cloud Computing to Agriculture and Prospects in Other Fields (Cont.)



Application of Cloud Computing to Agriculture and Prospects in Other Fields (Cont.)

- Considering just one agricultural service application, the required functions include
  - basic authentication and billing functions that can be shared with other fields such as GPS data processing and mapping systems.
- Other functions that should be shared with other fields are too numerous to mention, but include
  - image/speech processing and data mining.
- These functions can probably be used not only in agriculture but also in any business where work needs to be done on the spot, such as
  - medicine/nursing and maintenance work.

Mobile Healthcare Information Management utilizing Cloud Computing and Android OS



Mobile Healthcare Information Management utilizing Cloud Computing and Android OS (Cont.)

- The Android OS platform is adaptable to larger and traditional smart phone layouts and supports a variety of connectivity technologies (CDMA, EV-DO, UMTS, Bluetooth, and Wi-Fi).
- Amazon Simple Storage Service is used as a cloud service.
- The Cloud Service client running on Android OS consists of several modules.
- The Patient Health Record application acquires and displays patient records stored into the cloud.
- The Medical Imaging module is responsible for displaying medical images on the device.
- It decodes images in DICOM, JPEG2000 format displaying both image and heard information data.

Mobile Healthcare Information Management utilizing Cloud Computing and Android OS (Cont.)

- The communication with the Cloud is performed through an implementation of Web Services REST API that is supported natively by Android.
- In order to provide the user with data querying functionality, medical records and related data (images and biosignals) are stored into a SQLite file. SQLite is the database platform supported by Android.
- The file resides into a specific location at the Cloud and is retrieved on the device every time user needs to query data.

Mobile Healthcare Information Management utilizing Cloud Computing and Android OS (Cont.)

TRANSMISSION TIME OF MEDICAL IMAGES USING AMAZON S3 CLOUD SERVICE AND DIFFERENT NETWORK TYPES

Image Type (encoding)	File Size (MB)	Time (secs)	
		3G Network	*WLAN Network*
OT (24-bit JPEG2000 Lossless Color)	6.8	42.532	7.894
CT (Uncompressed)	0.528	4.023	2.382
CT (JPEG2000)	0.102	1.223	0.892
MR (JPEG Lossless)	0.721	9.738	3.894
PET (JPEG2000 Lossy)	0.037	0.923	0.793
Ultrasound (sequence of 10 images, JPEG2000 Lossless)	0.487	3.892	3.251

Categories of Cloud APIs

- Category 1 – Ordinary Programming:
  - The usual application programming interfaces in C#, PHP, Java, etc. There is nothing cloud-specific in this category.
- Category 2 – Deployment:
  - Programming interfaces to deploy applications on the cloud.
  - In addition to any cloud-specific packaging technologies, this includes traditional packaging mechanisms such as .Net assemblies and EAR/WAR files.
- Category 3 – Cloud Services:
  - Programming interfaces that work with cloud Services.
  - These APIs are divided into subcategories for cloud storage services, cloud databases, cloud message queues, and other cloud services.

Categories of Cloud APIs (Cont.)

- Category 4 – Image and Infrastructure Management:
  - Programming interfaces to manage virtual machine images and infrastructure details. APIs for images support uploading, deploying starting, stopping, restarting, and deleting images. Infrastructure management APIs control details such as firewalls, node management, network management and load balancing.
- Category 5 – Internal Interfaces:
  - Programming interfaces for the internal interfaces between the different parts of a cloud infrastructure.
  - These are the APIs developer would have to use if developer wanted to change vendors for the storage layer in cloud architecture.

Application development life cycle for cloud

- Development life cycle for application developed in clouds is not too different than traditional SDLC.
- Importance and duration of some of the stages is significantly different, e.g.
  - For a business application that can be completely modeled on salesforge.com architecture, requirement gathering becomes most important activity.
- Once, requirement is gathered and business logic is decided upon, the application can be configured without any code been written. This is possible in SAAS.
- For IAAS and PAAS, application is required to be coded and uploaded to cloud.

Application development life cycle for cloud (Cont.)

- During application development for PAAS, supported languages and architecture of PAAS cloud is required to be considered.
- Some of the IDEs have introduced the notion of developing on the local machine but deploying to the cloud from within the development environment, such as
  - g-Eclipse project, Aptana Cloud Connect and MS Visual Studio.

References

1. “Web Services: Concepts, Architectures and Applications”, Gustavo Alonso, Fabio Casati, Harumi Kuno, Vijay Machiraju, Springer Verlag
2. "Distributed Computing: Principles and Applications" by M L Liu, Pearson Education
3. http://www.sun.com
4. Ian Foster, Globus Toolkit 4, http://www.globus.org
5. The Anatomy of the Grid Enabling Scalable Virtual Organizations, by Ian Foster, Carl Kesselman, Steven Tuecke, www.globus.org
6. “Grid Computing”, Joshy Joseph and Craig Fellenstein, Pearson Education, ISBN: 81-297-0527-3
7. `Grid Computing: A Practical Guide to technology and Applications’, Ahmar Abbas, Firewall Media
8. Open Grid Services Infrastructure (OGSI) by S. Tuecke, ANL, K. Czajkowski, I. Foster, ANL, http://www.ggf.org/
9. A. Michael, F. Armando, G. Rean, D. Anthony, K. Randy, K. Andy, L. Gunho, P. David, R. Ariel, and S. Ion, others, “Above the clouds: A berkeley view of cloud computing,” *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28*, 2009.
10. R.L. Grossman, “A Quick Introduction to Clouds,” *Relation*, vol. 10, 2008, p. 9822.
11. P. Mell and T. Grance, “The nist definition of cloud computing,” *National Institute of Standards and Technology*, 2009.

References (cont.)

12. Service-Oriented Architecture: Concepts, Technology, and Design" By Thomas Erl, Publisher: Prentice Hall PTR
13. Web Services: Concepts, Architectures and Applications, Alonso, Springer
14. Dr. Srinivas Padmanabhuni, SOA Centre of Excellence, SETLabs, Infosys Technologies Ltd, Bangalore
15. Telecom Research Innovation Centre (TRIC), IBM Research Lab, New Delhi
16. "What 12 Cloud Computing Vendors Deliver." [Online]. Available at: http://www.informationweek.com/1240/240ID2_vendorchart.jhtml [Accessed March 11, 2011].
17. B. Sotomayor, R.S. Montero, I.M. Llorente, I. Foster, and F. de Informatica, "Capacity leasing in cloud systems using the opennebula engine," *Cloud Computing and Applications*, vol. 2008.
18. S. Liu and Y. Liang, "Eucalyptus: a web service-enabled e-infrastructure," *Proceedings of the 2007 conference of the center for advanced studies on Collaborative research*, 2007, pp. 1-11.
19. B. Sotomayor, R.S. Montero, I.M. Llorente, and I. Foster, others, "Virtual infrastructure management in private and hybrid clouds," IEEE Internet Computing, vol. 13, Sep. 2009, p. 14–22.
20. L.M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: towards a cloud definition," ACM SIGCOMM Computer Communication Review, vol. 39, 2008, p. 50–55.
21. Cesare Pautasso, SOAP Vs. REST Bringing the Web back into Web Services, IBM Research Lab
22. M. Hori et. el., Applications of Cloud Computing, to Agriculture and Prospects in Other Fields, Fujitsu Science and Technology Journal, Vol 46, No. 6, pp 446-454, Oct 2010
23. Charalampos Doukas, Mobile Healthcare Information Management utilizing Cloud Computing and Android OS, 32nd IEEE EMBS, Sept. 2010
24. Semantics Utilized for Process Management within and between EnterpRises: Strategic Objectives, Concept, and Results (SUPER), IST Project Number : FP6-026850, Funded by the European Commission, IST, Unit E2 Knowledge Management and Content Creation, http://www.ip-super.org/
25. Patel, Mehul, Chaudhary, Sanjay, Bhise, Minal, "Agro-Produce Marketing Using Social Network", First International Workshop on Social Media Engagement (SoME 2011) co-located with 20th International World Wide Web Conference,2011
26. Moens, Marie-Francine, " Information Extraction: Algorithms and Prospects in a Retrieval Context", Springer, 2010
27. V. Sorathia, Z. Laliwala and S. Chaudhary, "Towards Agricultural Marketing Reforms: Web Services Orchestration Approach", IEEE International Conference on Services Computing (SCC 2005), vol – I, pp 260-267, 2005.
28. Z. Laliwala, V. Sorathia, S. Chaudhary and P. Jain, "Integrating Stateful Services in Workflow", 13th Asia Pacific Software Engineering Conference (APSEC06), ISSN: 1530-1362, ISBN: 0-7695-2685-3, pp. 131-138