



# Process Centric Business Agility and Collaboration

---

Umesh Bellur

Associate Professor

Department of Computer Science,

IIT Bombay



# Outline

---

- The changing face of business
  - Impact on existing business automation technologies
- A process centric and service oriented view of the new @business
- Next generation technology for the @business
- Getting from here to there



# Outline

---

- ***The changing face of business***
  - ***Impact on existing business automation technologies***
- A process centric and service oriented view of the new @business
- Next generation technology for the @business
- Getting from here to there



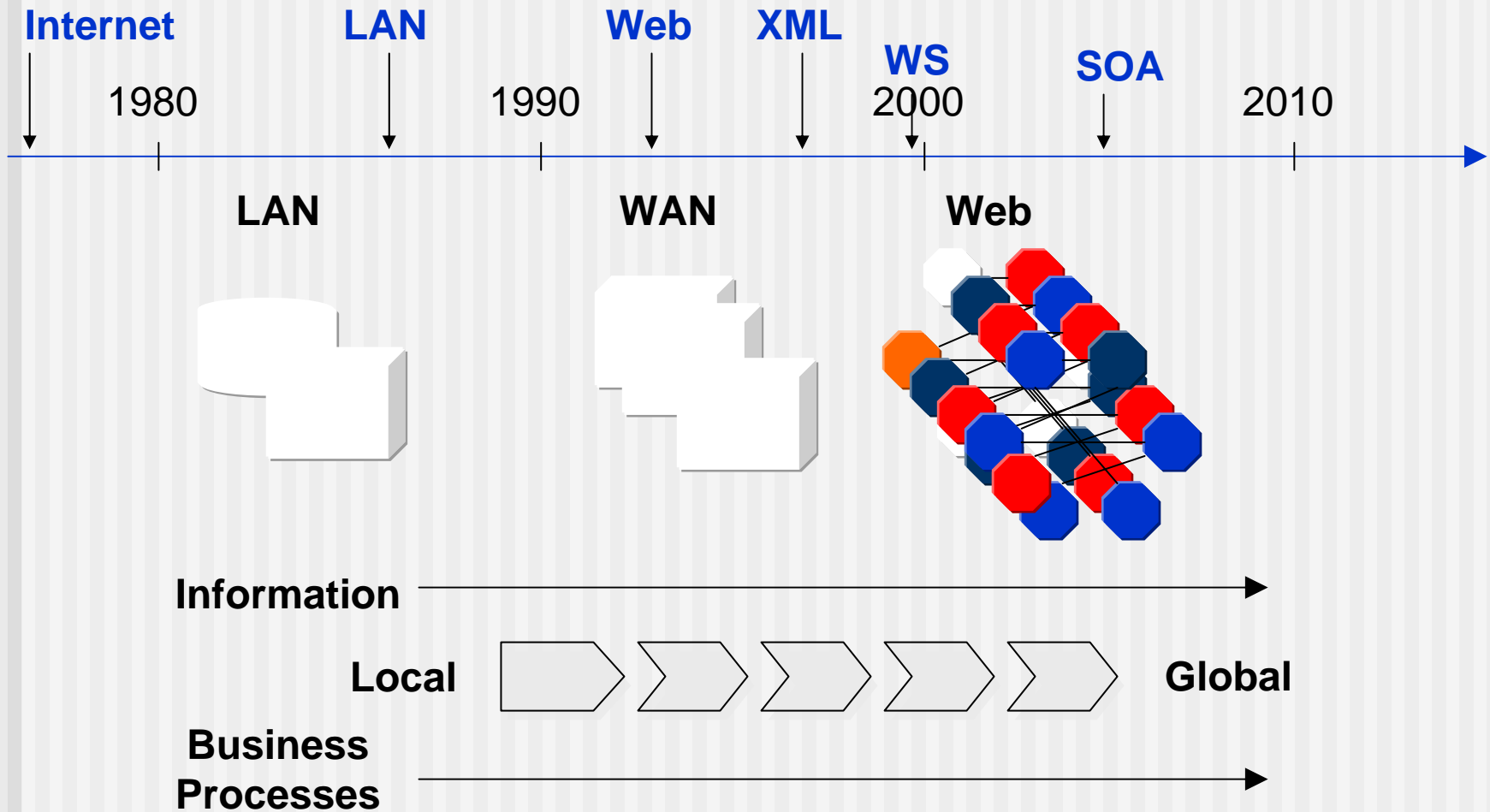
# What triggers change?

---

- Introduction of new technology in a parallel area – e.g., communications networking
- Changing business needs
- Difficulties with the existing paradigm to keep pace with changes



# On demand connectivity – driving globalization





# The business driver – agility!

- **Agility** is strategic but what is it?
  - degree of change over time – number of changes/unit time
  - ability to adapt to change from an IT perspective
  - relative value of change for the business
  
- **BPMI Survey results:**
  - Only 11 percent of executives say they're able to keep up with business demand to change technology-enabled processes.
  - Worse, 36 percent report that their company's IT departments are having either "significant difficulties" (27 percent) or "can't keep up at all" (9 percent)."
    - CIO Magazine



# Change Inertia

---

Agility is seriously hampered by

- ***Lack of Reuse***
- ***Integrating*** rather than ***coordinating/composing***
  - Analyst driven process definitions rather than glue code for integration!



# What's broken?

---

- Existing technology paradigms forces **tight coupling** which is undesirable.
- **Not adaptive** in nature
  - Static hardwired nature of applications
  - Business processes not externalized
- **Difficulties in reuse** and sharing of data and programs cause redundancy, wasted effort, and integrity violations



# What's broken?

---

- **Closed**: typically, use a vendor's proprietary software, and cannot cooperate with other systems
  
- The **level of IT abstraction** is inadequate to describe the offerings



# Hurrah for the internet! But.... the “Seven Inescapables”

---

1. The internet is inherently unreliable
2. Latency in the internet is real
3. Bandwidth on the internet is limited
4. The internet has a constantly fluctuating topology
5. The internet is insecure
6. The internet has multiple administrators
7. The internet is heterogeneous

***Has today's technology accounted for this?***



# The 3 Point Conjecture

---

1. Businesses have become more global **and** collaborative in nature => dependence on the internet
2. Growing competition and technology advances have led to a desperate desire to be agile
3. Business automation built with existing technologies are unable to keep pace with this transformation.



# Outline

---

- The changing face of business
  - Impact on existing business automation technologies
- ***A process centric and service oriented view of the new @business***
- Next generation technology for the @business
- Getting from here to there



# An introspection on today's society

---

Today's world is almost exclusively

1. **Service oriented**

- Transportation
- Telecommunication
- Retail
- Healthcare
- Financial services
- ...

2. **Process Centric** – behind each service is a business process delivering value.



# Characteristics of services

Well defined, easy-to-use, somewhat **standardized, coarse grained interface**

**Self-contained** with no user visible dependencies to other services

(almost) **Always available** but idle until requests come

**"Provision-able"** to different QoS requirements and different variants of a core functionality.

**Readily usable**, no "integration" required

Value can be created by **combining existing services**.

Quantifiable **quality of service**

- Do not compete on "What" but "How well"
- Performance/Quality
- Cost etc.



# Intents and Offers

---

Consumer express “intent”

Providers define “offers”

Sometimes a mediator will:

- Find the best offer matching an intent
- Advertise an offer in different ways such that it matches different intent

Request / Response is just a very particular case of an Intent / Offer protocol



# Multi dimensional Directories

---

Our society could not be “service oriented” without the “Yellow Pages”

Directories and addressing mechanisms are at the center of our service oriented society

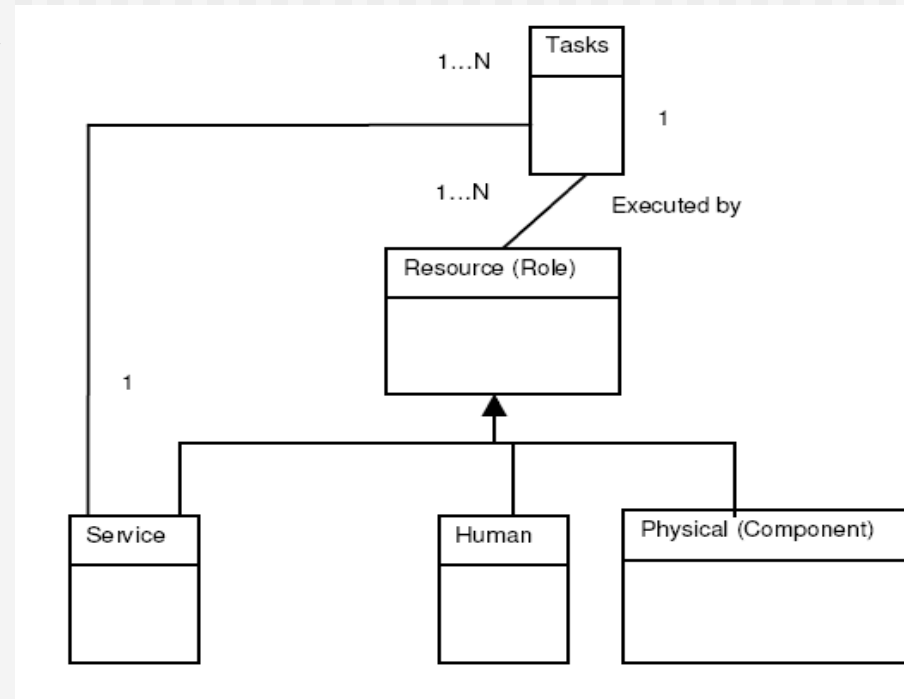
Imagine

- Being able to reach a service just by using longitude and latitude coordinates as an addressing mechanism?
- Only being able to use a service if you can remember its location, phone or fax number?



# Conceptual Model of a Service

- Service realization is a **workflow** for granularity of control
- Points of inter-task handoff are potential “variability” points.
  - Programmer dictates which points are variable
- Shows a set of dependencies on other services or components





# Drawing inspiration

---

**Question: *Can we model and build systems that reflect this service orientation and process centricity?***



# Outline

---

- The changing face of business
  - Impact on existing business automation technologies
- A process centric and service oriented view for the new @business
- ***Next generation technology for the @business***
- Getting from here to there



# Service Oriented Architecture (SOA)

---

A service is an abstracted, logical view of actual programs, databases, business processes, etc., defined in terms of:

- what it does,
- how it can be contacted and invoked
- an SLA it can offer,
- the cost of invocation and
- resources it may need.



# SOA - 2

- Message orientation: service formally defined in terms of the messages exchanged between provider/requester agents,
  - The internal structure agent (implementation language, process structure, database structure) are deliberately abstracted away in the SOA
  - Benefit: legacy systems, software component, applications can be “wrapped” in message handling code to be hosted as service
- Description orientation: described by *machine-process able metadata*.
- Granularity: Services tend to be relatively large with coarse grained and complex messages.
- Platform neutral: Messages are sent in a platform-neutral, standardized format delivered through the interfaces.



# Thinking Differently

---

- An application is NOT a single system running on a single device and bounded by a single organization
- Messages and Services
  - As opposed **Remote Method calls** and **Objects**
- Asynchronous **with callbacks** as opposed to synchronous request reply



# Loose coupling with Dynamic binding – A scenario

---

- Describe the semantics of a client's need
- Discover a provider of this need or compose multiple services dynamically
- Dynamically understand the protocol of messages to be exchanged with the provider
- Negotiate terms of use of a Service
- Move from one provider to another seamlessly



# The Fabric - key elements

---

- Description
- Discovery and Selection
- Engagement
- Collaboration

*But with an emphasis on semantics!*



# Summarizing SOA – Defining Characteristics

---

- Simplicity of interaction.
  - No notion of inheritance, polymorphism, call stack, references etc.
- No lifecycle control.
  - Service provider manages instances / allocations internally to suit its needs.
- Pass self-contained documents.
  - A tree structure (e.g., XML) is well suited for this.



# Outline

---

- The changing face of business.
  - Impact on existing business automation technologies
- A process centric and service oriented view of the new @business
- Next generation technology for the @business
- ***Getting from here to there***



# The value proposition

---

- Implement process centric SOA for two major reasons.
  - First is the ability to save development dollars through reuse of services.
  - Second is the ability *to change* the IT infrastructure faster to adapt to changing needs of the business, or agility.



# “Adoption” Dos and Don'ts

---

## **Don't boil the ocean!**

- Focus on processes that cross departments, lines of business and enterprise boundaries.
- Document & ***optimize*** business process you can identify

## **Both top down and bottom up approaches have to be used together**

- Identify services relating to the chosen BP one domain at a time – reusing existing assets.
- Categorization can help deal with complexity



# "Adoption" Dos and Don'ts

---

## **Embrace organizational change**

- 2 Pizza teams along with end to end ownership
- Process governance is important

## **Change your mindset**

- Move away from the current application centric view that leads to stovepipes and instead focus their organizations on business processes and services.

## **Don't let vendors drive your thinking!**



# "Adoption" Dos and Don'ts

---

## **Be flexible on standards**

- Put standards in the proper perspective – they are indispensable for flexibility when you cross enterprise boundaries – *but not if you don't.*

## **Don't over commit – not everything need be service oriented.**

- High transaction rate environments with hardly any complexity in BP may not be worth the while to move over.
- Do POCs before you commit to moving over.



# The shift in thinking

- ❑ Function oriented
- ❑ Build to last
- ❑ Prolonged development cycles

- ❑ Coordination oriented
- ❑ Build to change
- ❑ Incrementally built and deployed – extreme eXtreme



- ❑ Application silos
- ❑ Tightly coupled
- ❑ Call oriented
- ❑ Known implementation

- ❑ Composed solutions
- ❑ Loosely coupled
- ❑ Message oriented
- ❑ Abstraction



# Conclusion

---

## SOA is NOT the Same Old Architecture

- SOA specifically address the needs of going beyond enterprise boundaries and using the internet as a collaborative medium
- The concepts do offer a higher level of abstraction and looser coupling than remotely accessible objects.



# Questions?

---

Umesh Bellur

[umesh.bellur@iitb.ac.in](mailto:umesh.bellur@iitb.ac.in)



# Distributed Component Architectures

---

- Main driver: transparency to developer
  - Remote code looks like local code
- Characterized by:
  - objects maintaining complex internal state required to support their methods
  - fine grained or "chatty" interaction between an object and a program using it
  - shared implementation type systems and interface hierarchy between object and program using it



# Fallacies of DOA

---

- The distributed object approach ignores:
  - Latency (network, marshalling, applications)
  - Disconnected or intermittently connected networks
  - Lack of shared memory access (pointers, references)
  - Partial failure and concurrency
  - Independent variability between systems (coupling)



# DOA - Quotes

---

“The first law of distributed objects: Don’t distribute your objects”  
-- Martin Fowler

“Objects that interact in a distributed system need to be dealt with in ways that are intrinsically different from objects that interact in a single address space.”  
-- Waldo et al, 1994

“95% transparent is not good enough. In fact, it is worse because it deceives developers.”  
-- Werner Vogels



# Description

---

The description should be unambiguous, formal representations of

- A service's functionality
- A service's nonfunctional attributes
- A user's needs and preferences



# Discovery and Selection

---

- Semantic matchmaking
- Economic selection
- Reputation and recommendation
- Federated architectures
  - No single repository – need a DNS like resolution mechanism.
- Accommodating quality of service
- Trust



# Engagement

---

- Architecture: P2P, messaging and intelligent routing
- Transactions: replications, recovery
- Negotiation
  - QoS
  - Economics
- Orchestration
  - Workflow involving services



# Collaboration

---

- Reasoning
- Composition
  - Negotiation
    - Protocols, interaction patterns
- Contracts, monitoring, and compliance