Limitations of SOA
Service Oriented Architecture

- Architecture
- Service
- Enterprise ready SOA stack
  - Services
  - Connectors
  - Containers

- What are people saying?
- How much SOA

- Its all in the semantics
Architecture

1471-2000:

“Architecture is the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution”.

- This is a good start, what does this mean for SOA?
In SOA solutions there are *service providers*—elements offering services to be used by others and *service users*—elements that invoke services provided by others. These categories are not mutually exclusive. A service provider may use other services, and a service user may provide a service interface.

We define SOA as an **architectural style where systems consist of service users and service providers**.

- An architectural style defines a vocabulary of component and connector types, and constraints on how they can be combined [Shaw 1996].

- For SOA, the basic component types are service user and service provider. Auxiliary component types, such as the enterprise service bus (ESB) and the directory of services, can be used. SOA connector types include synchronous and asynchronous calls using SOAP, bare http, and messaging infrastructure.

- Many properties can be assigned to these component and connector types, but they are usually specific to each implementation technology.
### Capabilities Enterprise Ready = SOA + ESB

#### Service
A service is a network-aware unit of independent deployment, which can deliver a set of capabilities to a managed SLA.
**Capabilities:** service proxies, interfaces, orchestration, 
**Requires:** standard notations and a meta-model for data.

#### Connector*
A connector defines a dynamic view on the overall architecture by defining a picture of run-time entities and potential interactions.
**Capabilities:** intelligent messaging (reliable messaging, content-based routing, transformation, protocol switching, protocol adaptors...), business event visibility (event capture, feed to dashboards, link to network events...).

#### Container
A container monitors and administers the dynamic integration of services at run-time.
**Capabilities:** such as failover and load balancing, application-level security functions (authentication, authorisation, encryption / decryption, data integrity / non-repudiation, digital signatures...), audit-logging and registry functionality, centralised management and distributed enforcement of application policies, logging, audit, registry services etc.

---

Many alternatives here – choice will impact overall quality attributes of the SOA architecture:
- **WSDL, BPEL, Directories (lookup)**
- **RMI/IIOP – EJB etc**

Enterprise Service Bus vs. point to point, Service Fabric technology – inifinflow

---

*Adapted from SEI definition

The Cloud is in here somewhere
**Definition**:
HelloService

**Message**:
1. sayHelloRequest : firstName parameter
2. sayHelloResponse: greeting return value

**Port Type**: sayHello operation that consists of a request and response service.

**Binding**: Direction to use the SOAP HTTP transport protocol.

**Service**: Service available at http://www.examples.com/SayHello/.

**Port**: Associates the binding with the URI http://www.examples.com/SayHello/ where the running service can be accessed.

WSDL Describes the syntax of a web service but NOT its semantics.
What have we lost with SOAP/XML that we have with RMI/IIOP ???

RPC

Document

RMI/IIOP

public interface BankAccount extends java.rmi.Remote
{
    public void deposit(float amount) throws java.rmi.RemoteException;
    public void withdraw(float amount) throws OverdrawnException,
        java.rmi.RemoteException;
    public float getBalance() throws java.rmi.RemoteException;
}
Why an Enterprise Service Bus ??

How to get effective re-use from SOA and ESB, re-use achieved through canonical form embedded in ESB architecture
"That's my biggest worry about all this discussion about SOA is that I'm hearing plenty about the 'S' (there's quite little of 'O' going on) but the 'A' seems to vary from person to person, and that makes me worry that we're going to end up with a world where interoperations is left for the reader rather than a world where the interoperability is built in from the ground up."

(Simon Phipps, Sun)

"Services are an important but insufficient mechanism for the construction of systems".

(Grady Booch, IBM Fellow)

"The semantics issue (the meaning of parameters and so forth) has to be solved (domain modeling). This is key in any business-to-business (B2B) and dynamic invocation scenario".

("Elements of Service-Oriented Analysis and Design", IBM, 02 Jun 04)

"Information Management, which includes both data and content management, is an essential building block for SOA".

"Ultimately, without a solid and robust information management environment, SOA is limited and presents fewer opportunities for end-to-end business integration and transformation".

("Information management in Service-Oriented Architecture, Part 1: Discover the role of information management in SOA", IBM)
An Ideal World

"Where is the 'A' in SOA? With the emphasis on turning every piece of software in the enterprise into a collection of services, are we in danger of replacing spaghetti code of old with spaghetti architecture of innumerable services with innumerable interdependencies?"
(David Chappell, Sonic Software)

“A software process architecture is key to understanding and interpreting the needs of just enough process”
(‘Just Enough Process’ for Web Services Delivery, Gartner, October 2002)

“Building an enterprise SOA requires an architecture for the configuration, hosting, and management of integration components as services, an invocation framework for the routing of messages between services, and flexibility of control over protocols, quality of service (QoS) settings for message delivery, and security. The ESB provides that architecture. It puts the “A” in SOA”.

“A key part of the architecture is a distributed service container model that can be used to host and manage application assets and integration components as event-driven services that all share a federated namespace for locating and accessing each other”.
(David Chappell, Sonic Software)

“SOAs don't address the challenge of ensuring that business data is delivered in a consistent and semantically correct manner. In fact, they exacerbate the problem of maintaining consistent definitions for business entities such as customer, product, or invoice”
(“SOA’s Achilles’ Heel, Larry Allston, Vice President of products for Pantero Corporation).
How much SOA and how far

Loose IT coupling and strong business coupling – really desirable ??
Service Granularity

How granular to make a service
getFieldOne(); getFieldTwo()
vs. GetDataElement()

Representational State Transfer
- URI, HTTP
- GET (list), PUT(update), POST(create), DELETE (delete)
- Note REST is an architecture NOT a protocol.
Issues with service and distributed architectures

- **Latency.**
  - There is a speed differential between a local (in the same process space) call and a remote network call. This differential must be taken into account when designing systems that will not suffer from potential performance problems. This issue has not yet been rendered obsolete by faster networks and hardware and remains a key design issue to be solved when designing a distributed system. See J2EE design patterns Fast Lane Reader and Page-by-Page Iterator.

- **Memory Access.**
  - This essentially revolves around accesses to resources via pointers. Addresses for a local space are not valid for a remote space. We can either provide a language mechanism that hides the local/remote nature of a resource or we treat it programmatically i.e. the programmer is aware of the difference.
  - If we provide a language mechanism for abstracting out the differences then ALL pointer usage must follow the distributed rationale. The language must not be allowed to provide address-space relative pointers otherwise we are back to the local/remote arguments. This tends to exclude languages such as C/C++ etc. from being able to support such a language extension.
Partial Failure.
- This has been described as the central reality of distributed computing see [Waldo]. The issues boil down to the inability of some agent to adequately identify what components have failed and reliably inform other components of that failure. There is no global state that can be examined to determine the nature of the failure, be that network or remote processor. To be able to ensure a consistent state is maintained in the presence of failures distributed applications must deal with indeterminacy, that is components must be designed to react correctly to the various possible partial failures that can occur.

Concurrency.
- Objects involved in providing services across a network to multiple clients must be able to handle concurrent method invocations. Thus if we strive for the one model view all objects in our application must have adequate concurrency semantics. There are subtle differences between multi-threaded applications where the threading model can to some extent be controlled and a true distributed object where the threading model is unknown.
The Semantics problem

Tools – the world of services, the picture depicts a hammer. But what makes it a hammer?

Context and Environment – we possess an understanding of hammering – that is how to use a hammer.

Context and Environment underpins our view of services through the notion of suitability.

However, our view of services underlies Context and Environment through the notion of intelligibility.

Reflex Hammer – used in medicine!!