



# The next inflection point

Adam Bosworth,  
Chief Architect, SVP  
BEA-Crossgain



## Agenda

---

- Waves of Change
- What this one will look like
- What will be required
- Challenge



## **Anatomy of a wave**

---

- Enabling Technology
- Disruptive Innovation(s)
- Standards and languages
- Platform (usually only one!)
- Applications Framework(s)

# Waves

---



- Mainframes, 60's, COBOL/FORTRAN, VM, Automate mission critical systems, Time-sharing
- Minis, 70's, C, Unix, Automate big departmental systems, reduce cost of MIS and other automation
- PC's, 80's, Pascal, DOS, independence from MIS departments
- LAN's Take I, 85/95 on, C++, Share resources in enterprise
- LAN's TAKE II, 92-95, TCP/IP any app can talk to any app in the enterprise or any data in the enterprise, Asynch, Tightly coupled
- Internet Take I, 95, Ecmascript, HTTP/HTML, IMAP/POP, anyone can connect to anyone in the world, anyone can connect to any application in the world, Synch, Loosely coupled
- Internet Take II, Mobile, now, XL/XQuery?, WSDL/XML, any application can connect to any application/data in the world, Asynchronous, Loosely coupled



## Internet Take II

---

- What the Net added is the ability to connect to anything or anyone.
  - Talking to people means you can talk to anyone
  - Talking to apps means you can talk to anything
- This dramatically raises the marginal value of communications and that, in turn, drives standards
- App to App is the next big win
- Mobile computing
  - Affordable by 10 times the people
  - Has a micro-billing infrastructure in place
  - Will be the disruptive innovation for PC's in my expectation
  - SMS volumes alone are already huge
  - Asia is leading the way here (3G, DoCoMo, JPhone)



## Benefits

---

- Reuse of Information
  - Weather, Stock Prices, Items for sale
- Scalable access to information and Services
  - Fed Ex, Travel, Schedules
- Automatic execution of manual processes
  - Car Rentals, Purchasing, Meeting Coordination

# Considerations for Architecture

---



- Apps are built by different groups of people in different places and at different times
- They are deployed and altered on different schedules
- Often they are legacy apps that either have scheduled downtime or constraints on throughput
- App's aren't people. They aren't as smart

# Why Web Services is the foundation

---

- Coarse Grained Communication
- Loose Coupling
- Asynchrony
- Reliable Messaging
- Performance
- RAS

# Coarse Grained Communications

---



- Can't overload the network
- Can't overload the applications and servers
- Can't mandate the object model
- Can't expect to hang onto references
- Can't expect to mandate a platform
- In short, App's are not objects
- Answer is a formal model for requesting and delivering coarse grained sets of data using XML



## Loose Coupling

---

- Can't understand the other applications implementation
- Can't break if it changes
- Can't know its underlying data model implementation
- Can't assume its platform or language
- Can't know how to query it
- In short, App's are not databases or Interfaces
- Today's answer is a formal model for describing the XML messages being sent and requested and, to the extent that they are stateful, the legal sequences in which this may occur



## Asynchrony

---

- Can't assume that the other application is running
- Can't assume that it can execute the service or deliver the information instantly
  - Load constraints
  - Process Constraints
- Can't even assume that the connections are reliable
- In short, App's aren't functions or methods
- Answer is to enable asynchronous communication and coordinated conversations

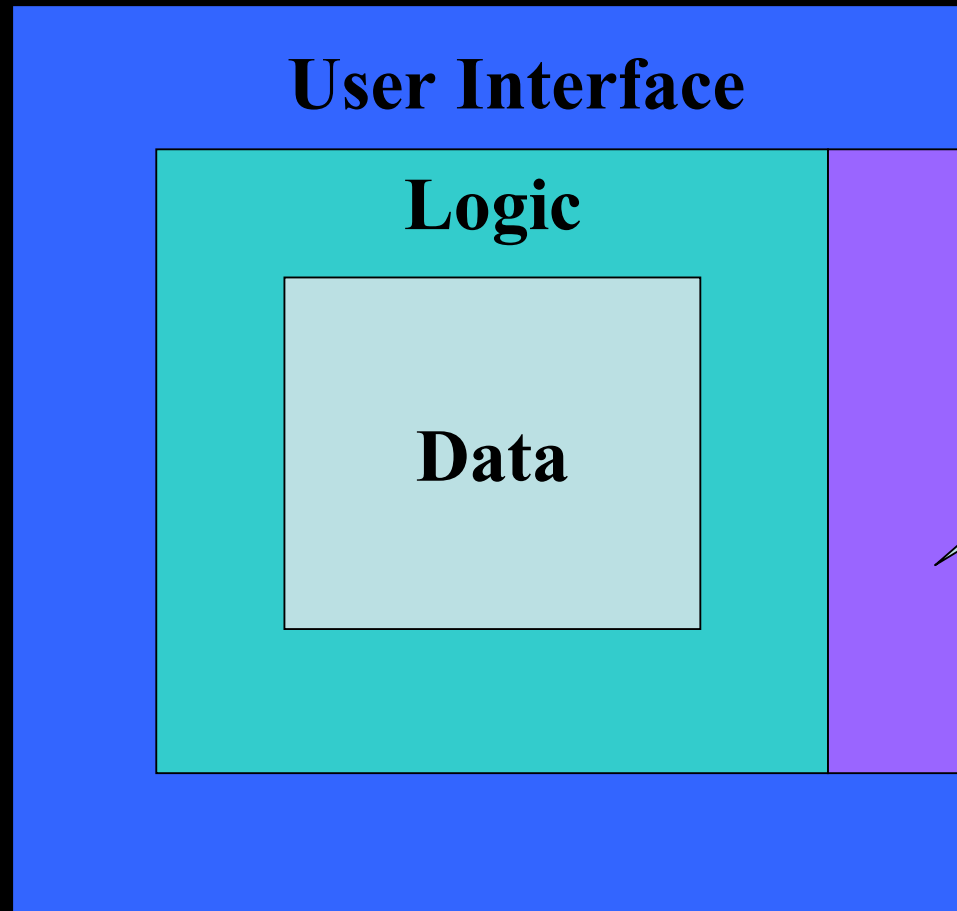


## Performance and RAS

---

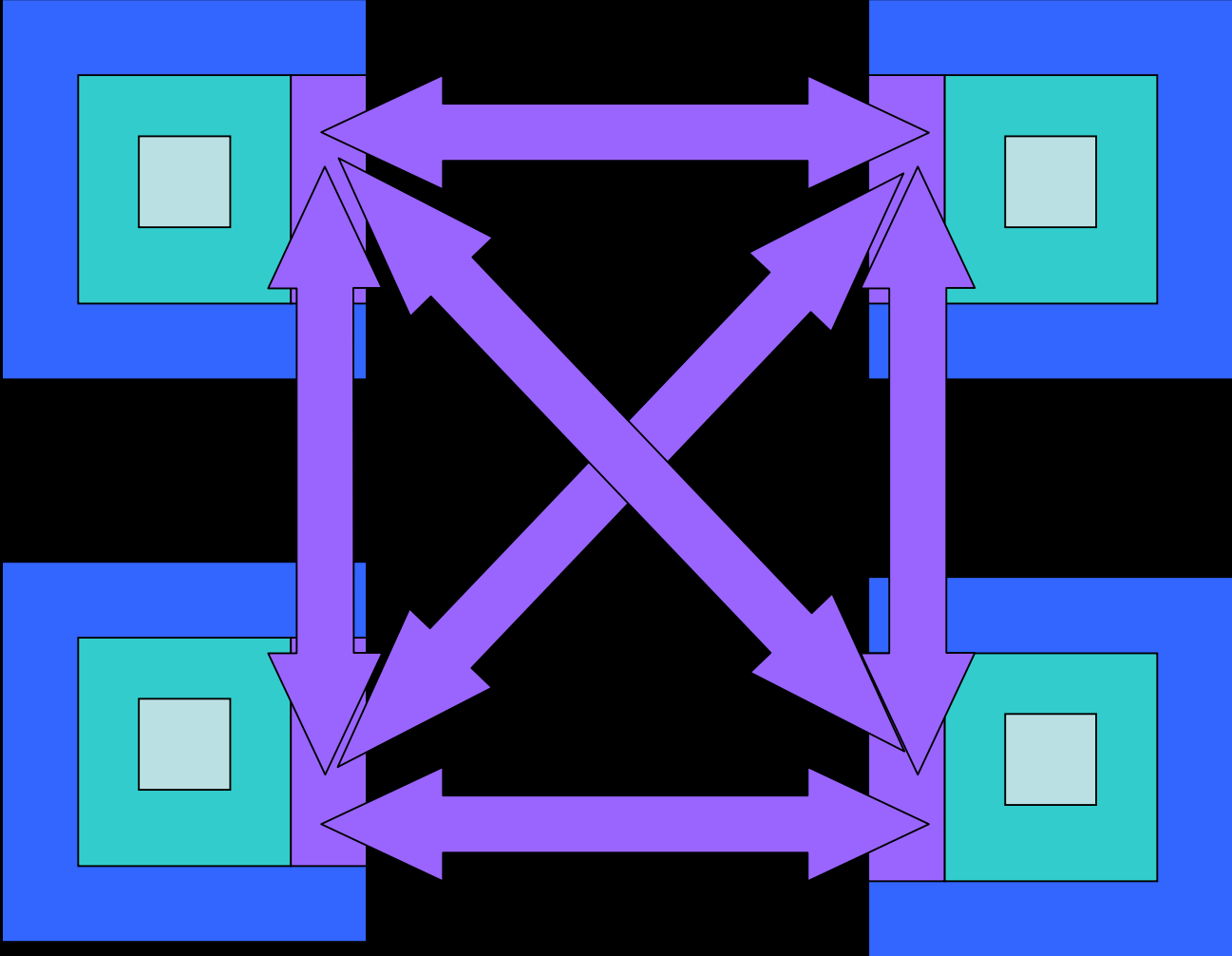
- Moore's Law and Clusters, but
- Latency
- Management Costs
- Exchanges and Hubs
- So, Systems need to handle reliable scalable communications, sometimes with very low latency

# What are Web Services

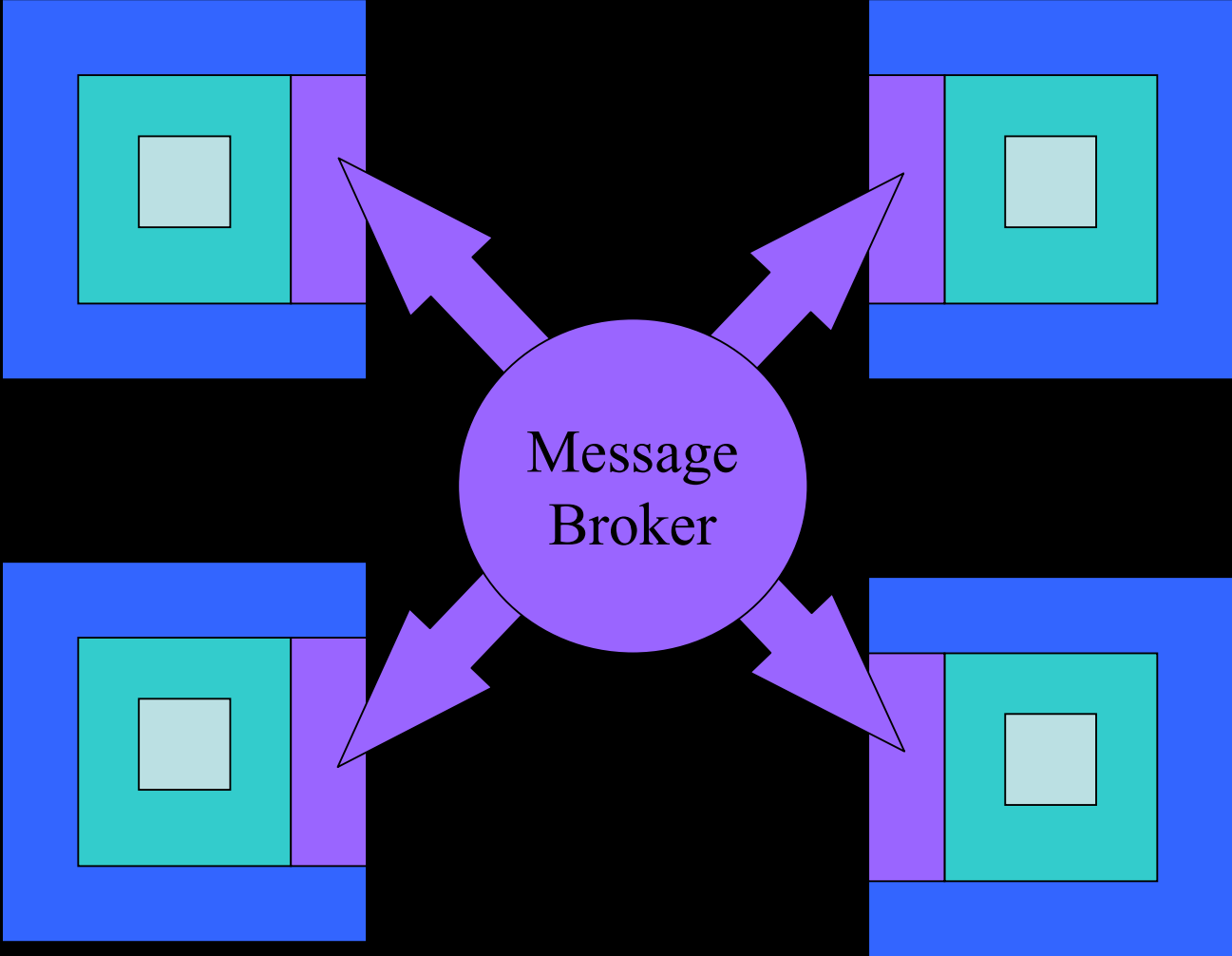


Apps  
Talking 2  
Apps

# App's talking to App's



# App's talking to App's

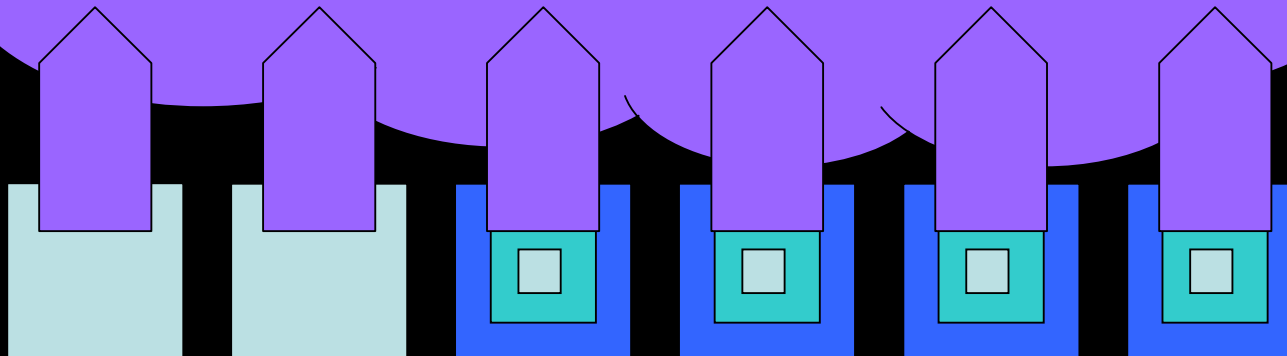




**App's talking to App's**

# Web Services

Message/Integration  
Broker  
(Workflow)



DB

Queue

ERP

J2EE

Legacy

W/S enabled app



**App's talking to App's**

**Message Broker**





## Queues

---

- Massively Scalable
- Basic Building block for messages
- XML Messages
- Tokenize on the way in
- Filter by content
- Age messages



## Message Broker

---

- Massively scalable and potentially very fast
  - Can provide ordered delivery when required
- XML state and context
- Listens to incoming messages
- Decides who needs to know
- Manages persisting/retrieving the state
- Runs your workflows for you – Speaks workflow
- Manages the long-running transactions/undo logic
- Like all application servers:
  - Highly Available
  - Fully Transacted
  - Completely and dynamically Scalable

## Workflow

---



- Massively scalable
- XML aware
- Can save/reload state between messages
- Can force message ordering when required
- Deterministic model for declarative logic
- Pi-Calculus and Lambda-Calculus
- Error handling and recovery
- Long-Running Transactions and Undo logic



## Language

---

- XML is a native type
- Seamless integration with predicates and queries
- Procedurally extensible
- Easy to learn/teach
- Native support for Workflow primitives
  - Wait for messages with known patterns
  - Conversational state
  - Parallel Fork and Join
  - Iteration over XML elements

# XML Database

---



- Extremely fast
  - Must serve up entire messages, pieces of messages, and persisted state at a rate of 1000's/second
- Intelligent XPATH and pattern recognition
- A great deal of “single-user” transient data
- Should have optimized retrieval for known filters
- This isn't the Relational DB as XML solution
- This is true storage/retrieval of XML messages

# Standards

---



## Key Solutions

## How are we doing

**XML**

Coarse grained communication, Digital Signature, Encryption, Types, Tokenization

**SOAP**

Loosely coupled integration, Message Correlation, Return Directions, Cumulative Latency, Reliable Messaging, Query Type

**WSDL**

Wire Formats/contracts as the unit of agreement, Reliable Messaging, Compensating Txns and Exceptions, Encryptions, Acceptable Latency, Security

Conversational Contract

**UDDI**

Discovery, Lease

**Workflow**

ebXML, XML Query, BPML?, XLANG?, WSPL?, WSFL?, BEPL?

- Coarse Grained Communication
  - XML can be big
    - Efficient Parsing of large documents today:
    - Coming – Total support for XSD, “Token-Streams”
- Loosely Coupled
  - Maps and EcmaScript
    - Rich procedurally extended language for mapping between Java classes and XML messages
- Workflow
  - Full support for long-running asynchronous conversations and persisted state between messages



# Programming Languages

---

- Coarse Grained Communications
  - XML <-> Java
  - XSD isn't Classes
  - Exploit the self defining capabilities of XML
  - XML <-> XML
- Loose Coupling
  - Seamless extension for declarative mapping between XML and code.
- Workflow
  - Built in support for Asynchrony in the language
  - Built in support for Workflow in the language



# Applications Framework

---

- Coarse Grained Communication
  - XML Processing (Tokenizing, Security, Reliable)
  - XML Pipelining, Encrypting, Filtering, Reading
  - QOS Enforcement
- Loose Coupling
  - XML Marshaling (XML <-> Java)
  - (Public Contract vs Private Implementation)
- Workflow
  - Web Service queues
  - Trading Partner Management
  - Coordination and Workflow
  - Efficient State Management between messages

# RAS

---



- Web Services need to support a framework that is
  - Reliable,
  - Scalable,
  - Invoked through standard protocols,
  - Triggered by asynchronous message driven events
- We see J2EE as a great starting point



## Not what I expected

---

- App to App
  - Data Models
    - Queries
    - Standard Schemas and Ontology's
    - Meta Data
    - <http://xml.coverpages.org/bosworthXML98.html>
  - Functional
    - Conversations, Messages, Humans
    - <http://www.xmlmag.com/upload/free/features/xml/2001/06jun01/ab0103/ab0103.asp>



## Conclusion

---

- You have a lot to do
- Basic theory for:
  - XML/Web Service Aware queues
  - Message Broker that:
    - Listens to the Queues
    - Dispatches to the right workflows/destinations
    - Runs the workflow language
    - Interacts with an XML store
  - Extend the Programming languages for XML, QOS, Asynchrony, and Workflow
- **So do we**
  - Extend the Applications Framework
  - Build the Industrial versions of all this