

Lecture #11: 8th March 2004

Web Services Critical Issues

Suresh Chande



Web Services Critical Issues in a Nut Shell

- Loosely Coupling : RPC-oriented → Document oriented Services Communications
- Synchronous → Asynchronous Messaging oriented.
- Fault handling
- Protocol Performance
- Proximity based Service Discovery
- Peer 2 Peer Services
- Web Services Inter-operability
- Role of Standards bodies
- Tools and IDE's
- IPR



Loose Coupling: RPC - 2 - Document oriented

- RPC World of Web Services
 - This has been the initial developments in Web Services and hence has a good set of tools and solutions to simplify the RPC style of web Services development
 - Depicts an API call on a remote Object/component typically still looking at a granular API centric view to Web Services
 - A Natural means of integration of existing Enterprise Applications as it demands less requirements for exposing/integrating existing RPC styled native services/systems
 - Developer is exposed to a minimum to web services protocols, as the tools can isolate the details by generating most of the communication protocols.
 - Suitable for small scale services which are granular and limited in functionality.



Loose Coupling: RPC - 2 - Document oriented

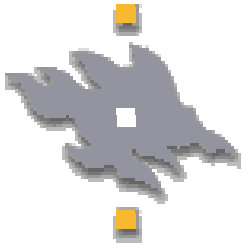
- RPC World of Web Services
 - Typically based on the request-Response Protocols as is the typical RPC based technologies(RMI, DCOM, etc). This is not the most likely scenario in the Web based enterprise applications
 - Tightly bound to the native platform as the interfaces in Web Services are very close the actually interfaces. This could lead to a ripple effect if any small changes in the deployed services in the native platform, it would reflect to the complete set of clients who are using this service
 - Not the most suitable for Services oriented interfaces with coarse grained interfaces



Loose Coupling: RPC - 2 - Document oriented

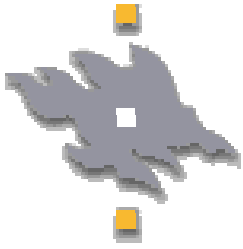
Document Oriented / Loosely coupled Web Services

- Techniques to develop loosely couple services [Web Services. org]
 1. Abstract platform independent messages
 2. Coarse-grained, self-describing and self-contained messages.
 3. Well-defined interfaces
 4. Extensible Versionable interfaces
 5. Stateless messaging
 6. Asynchronous exchange patterns where possible and appropriate
- Challenges to be addressed:
 1. No well supported solutions/tools towards document oriented /loosely coupled
 2. Convenience APIs or tools and solutions are missing in the space of Document oriented service
 3. Developers today require to develop a huge set of solutions\wrappers around legacy systems in house to support document oriented services.
 4. Document transformation and routing would
 5. Process oriented views to are just being planned and developed currently



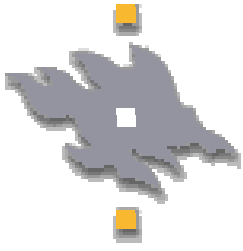
Synchronous 2 Asynchronous

- Web Services are most commonly been developed today utilise synchronous mode of communication baically due to the fact that the underlying transportation is based on HTTP which is a synchronous request-response protocol.
- Web Services are though meant to be utilised for business processes, EAI, whose basic mode of interaction are long running and asynchronous in nature.
- This requires a new set of specifications which can bind web services to non Synchronous communication protocols.
- The most commonly thought out infrastructure is based on :
 - MessageQueues (Web Sphere Message Queue, MSMQ)
 - JMS
 - ESB



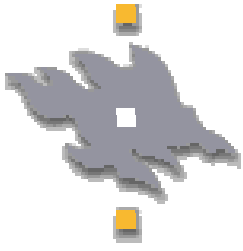
Synchronous 2 Asynchronous

- There are very few supporting tools and solutions to enable such a asynchronous
- Several existing proposal web services technologies should play a major role to simplify and realize the asynchronous web services communication as a common commodity
- Few most critical and related technologies :
 - Addressing
 - Routing
 - Message Co-orelations
 - Reliable Messaging
 - Callback mechanisms (for example publish-subscribe and subsequent notifications)
 - Process based operations
 - Bus Architectures



Fault handling

- Fault handling in the Web Services infrastructures lacks a standard conventions
- The Web Services provides place holders for carrying faults, but then the missing portion in the standard conventions of how this faults are handled :
 - How do you map the Web Services level faults to deployment environments ?
 - How well do you overly WS faults onto the underlying infrastructures (HTTP/SMTP/Web)
 - What are standard patterns of fault handling and propogations?
 - Should Web Services fault handling contain details of the implementation faults or just the service interface contract level alone.



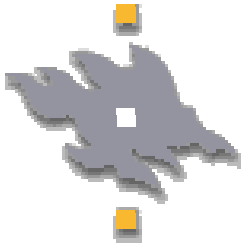
Protocol Performance

- Web Services do not claim to have high performance protocols for over the wire/ specifically wireless transmissions
- Should the XML based protocols have alternative bindings / binary encoded ?
- Should these bindings/ protocol enhancements been standardized ?
 - What are the means of :
 - Advertise the enhanced protocols
 - Negotiating alternative enhancements to protocols
- Should the enhanced protocols only be applied between well agreed and understood Partners, Gateways, External
- What are the alternatives means to improve protocol performance :
 - Caching ?
 - Encoding - ASN.1, binary formats
 - XML binary optimized Packagin
 - SOAP Message Transmission Optimization



Proximity based Web Services

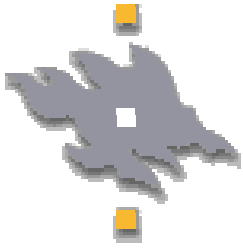
- As mobile enters into application of Web Services, the local/proximity based web services will become important.
- How well will the existing webservices technologies will address :
 - Local Connectivity
 - Discovery of local Services
 - UDDI being location sensitive / Alternative approaches based on local access points ?
 - How will bluetooth, WLAN based access can discover local services
- How will Proximity based application of Web Services extend the Services Oriented Architectures ?



Peer 2 Peer Web Services

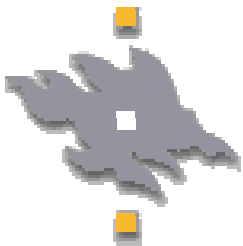
- Traditional Peer 2 Peer Services:
- Peer to Peer based application have been typically applied to consumer domain for end user media applications
- The protocols have been XML based and have been standardized to some extent (JAXTA)
- The application of the Web Services protocols could allow greater interoperability and also introduce new usage scenarios
- Traditional Web Services
- The application of Web Services has been typically been in the area of Business 2 Business.

- As the Web Services start to be emerge into a device to device environments the value of Peer 2 Peer Web Services become greater.
- The Challenges will be in :
 - Negotiation of the protocols
 - Discovery of Peer Services
 - Invocation and communication
 - Session / Context Maintainance



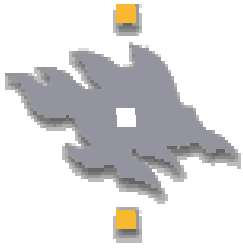
Web Services Inter-operability

- Why are we talking of Inter-operability in Web Services, Aren't the Web Services protocols the basis for Inter-operability ?
- Inter-operability can be maintained at the different layers in web services architecture
 - Transportation
 - Messaging Frameworks
 - Conventions and processing Model
 - Platform binding : Objects/Components ,Data and Legacy Integration
 - Application level
 - Development environment tools



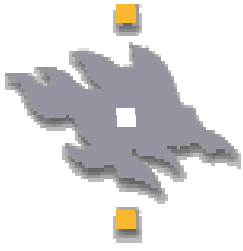
Web Services Inter-operability [Contd..]

- **Over the wire they are inter-operable with the basic set of commonly agreed specifications !!**
- How about at the deployment levels ?
- This is where inter-operability needs to be addressed :
 - Mapping Datatypes
 - Homogenous environments
 - Single Solution provider (Either J2EE or .NET)
 - Standards based
 - Consistency
 - Versioning (Backward and forward compatibility)
 - Addressability
 - Security



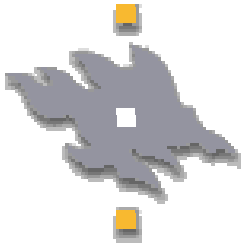
Role of Standard bodies

- Industry has several standard/ standard like bodies addressing technologies related to Web Services : W3C, OASIS, WS-I, OMA, Liberty, OMG, several smaller industry coalitions
- A clear and well declared roles and relationship between these standards will solve all the confusions from the standardizations front.
- A clear well agreed roadmap and co-relations between the different standards organization will make the web services a better predictable technology. This will avoid all the below at a very early stage:
 - Inter-operability issues
 - Overlapping efforts
 - Alternative solutions
 - Conflicting interests



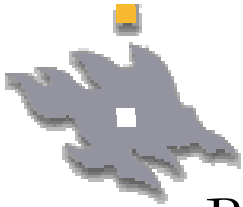
Tools and IDE's

- The most common means of developing Web Services will be the ones that is generated by tools and IDEs taking as input the existing solution interfaces developed using traditional application environments (Python, Perl, J2ee, .Net, etc..)
- Currently support for only the basic Web services specification exists.
- The tools today:
 - Generate WSDL documents from service interfaces available from the native platforms
 - Facilitate client development by generating the native platform binding wrappers.
 - Do not support the more advanced web services specifications
 - Bottom-up automised specification generation



Tools and IDE's

- Tool support beyond the basic core web service technologies is very crucial and the most challenging and complex
- The inter-operability to be maintained between tools and IDEs require mature standards.
- Tool and IDE challenges:
 - Support in binding legacy to the various web service specifications beyond the basic
 - Reuse of already defined interfaces not supported
 - Compliance to namespaces
 - Top-down design principles and applications of patterns
 - Web Services modelling tools are totally missing / poorly addressed.
 - Debugging and monitoring tools features



IPRs

- Big IT vendors and solutions providers are interested in owning and promoting their own solutions as they are well protected by their IPR portfolio's.
- The Industry coalitions bring forward to the standardization organization a well baked solutions for standardizations. The ingredients into the baked solutions contains bindings to their owned IPRs.
- Most big players claim to make them available on a royalty free on a conditional basis
- Standardization along with the IT vendors should provide means to be compliant with a standard proposed and being totally free from any IPRs , this will ensure wider and inter-operable Web services technologies
- There could be parts which are protected by IPRs, but they should be the optional features.
- Open Source should take a lead in the space of Web Service solutions and implementations to solve the IPR issues