



**MuleSource**

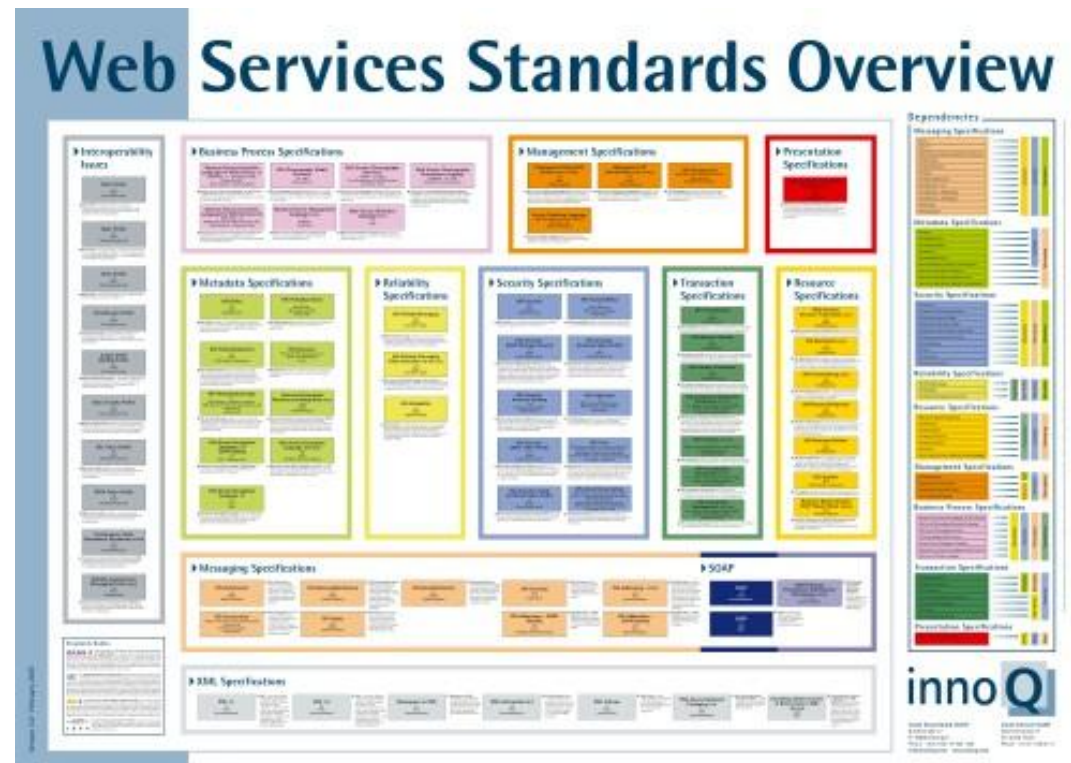
the open source choice for SOA infrastructure

# Implementing Messaging Patterns Using Web Services with Mule

Daniel Feist, Principal Software Engineer

Dan Diephouse, Principal Software Engineer

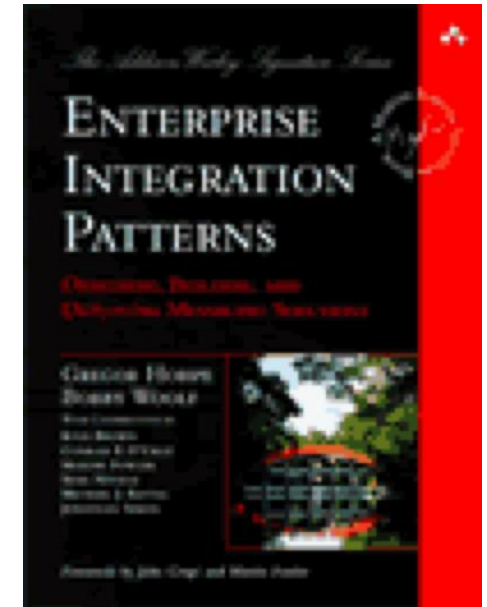
- ▶ Complex! (SOAP is not Simple)
- ▶ Myriad of, sometimes competing, WS-\* standards.
- ▶ Often RPC style
- ▶ Hard to extend easily



<http://www.infoq.com/news/2007/03/innoq-ws-standards-poster>

- ▶ The Boss says so ..
  - .. maybe because a WS-centric product was purchased
  - .. or WS is part of a “prescribed” architecture
- ▶ The Interface is important
  - External partners and providers
  - Cross departmental
- ▶ Interoperability
  - Cross Platform
- ▶ Ease of client generation

- ▶ Loosely coupled services
  - Flexible topologies
  - Ability to change and scale more easily
- ▶ Choice of transports
- ▶ Transparent transformation
- ▶ Interoperable with different platforms
- ▶ Message Filtering and Routing





**MuleSource**

the open source choice for SOA infrastructure

# **Web Services & Messaging**

Web-Services

+

Mule ESB

Well defined Web-Service interfaces

Decoupled implementation

Flexible internal architecture

Messaging patterns

# The Business

- ▶ Sell products
  - Home-wares, Electronics, Clothes etc.
  - Multiple channels
    - Online Store
    - Partners
    - Stores
  
- ▶ Very young dynamic company: lots of change

## Requirements

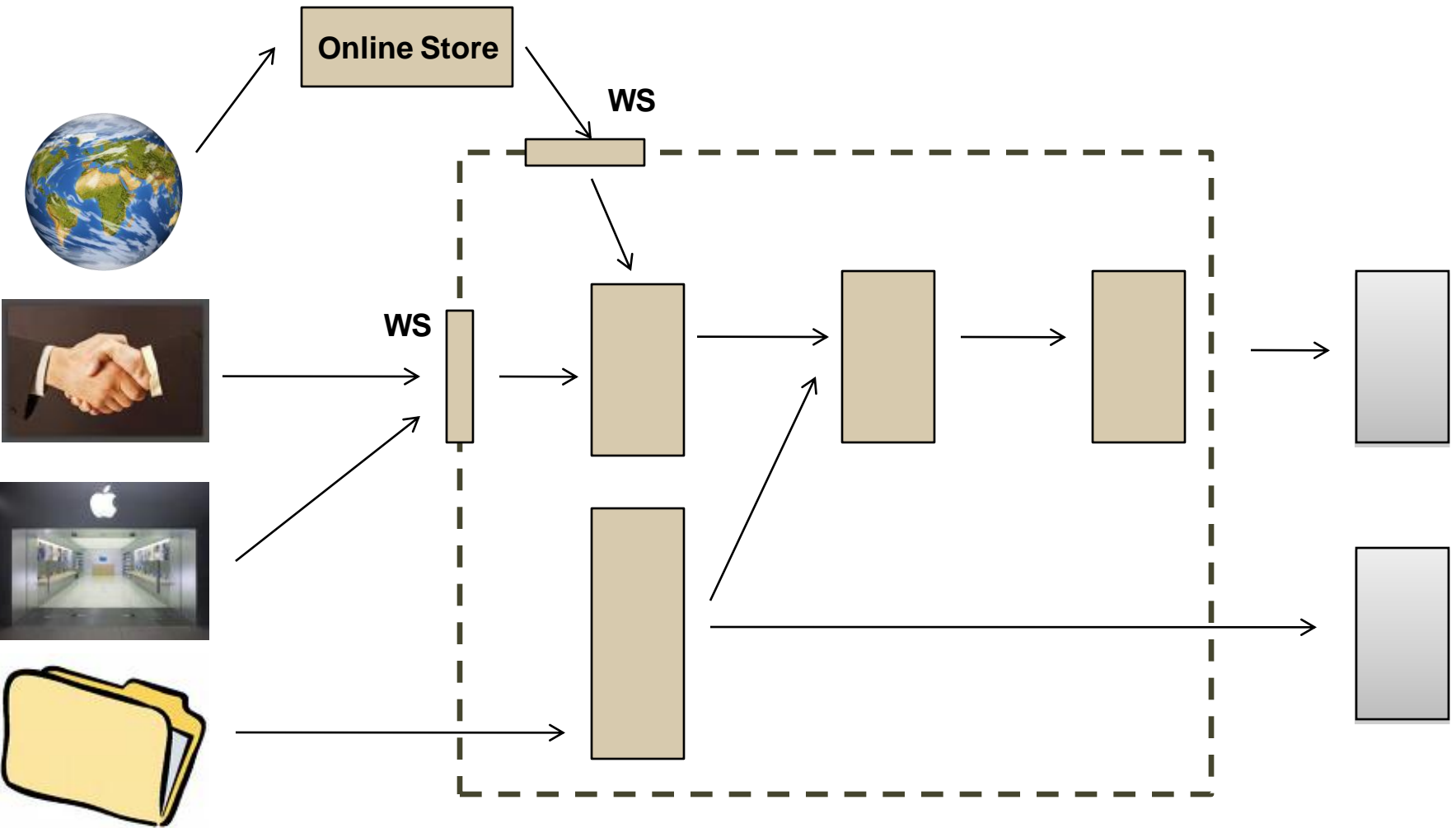
- ▶ Well-defined and stable Web-Service interfaces.
- ▶ Services may be:
  - Implemented in multiple technologies
  - In-house or provided by partners
- ▶ Ability to:
  - Scale operations
  - Change/migrate service implementations
  - Easily add new product lines and providers
- ▶ Performance

# 'Nile' Architecture

## Channels

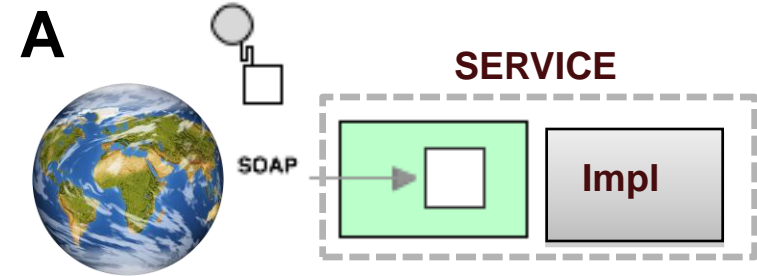
## Nile SOA

## Providers

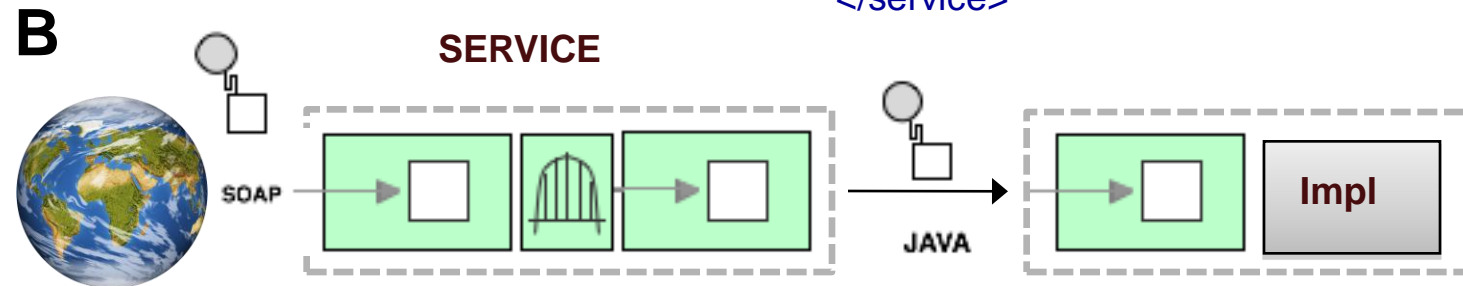


- ▶ Generate WSDL using introspection
  - **Problem:** Interface and implementation are coupled
  - **Solution:** Generate WSDL from Java interface (Java Only!)
- ▶ Use RPC Style
  - **Problem:** Tight coupling
  - **Solution:** Use 'Document' style. (RPC is now disallowed in WS-I basic profile)
- ▶ Use WS-<sup>\*</sup>
  - **Problem:** Complexity!
  - **Solution:** Avoid wherever possible 😊

- ▶ Web-Service frameworks typically do all of the following:
  - Processing of SOAP Header/Body/Faults etc.
  - Implementation of WS-\* features.
  - Data Binding
  - Service Invocation
  
- ▶ Simple WS stack integration with ESB does all of this inside ESB when using WS endpoint
  - Service interface or implementation is required
  - Binding has to be somehow turned off if XML message is required

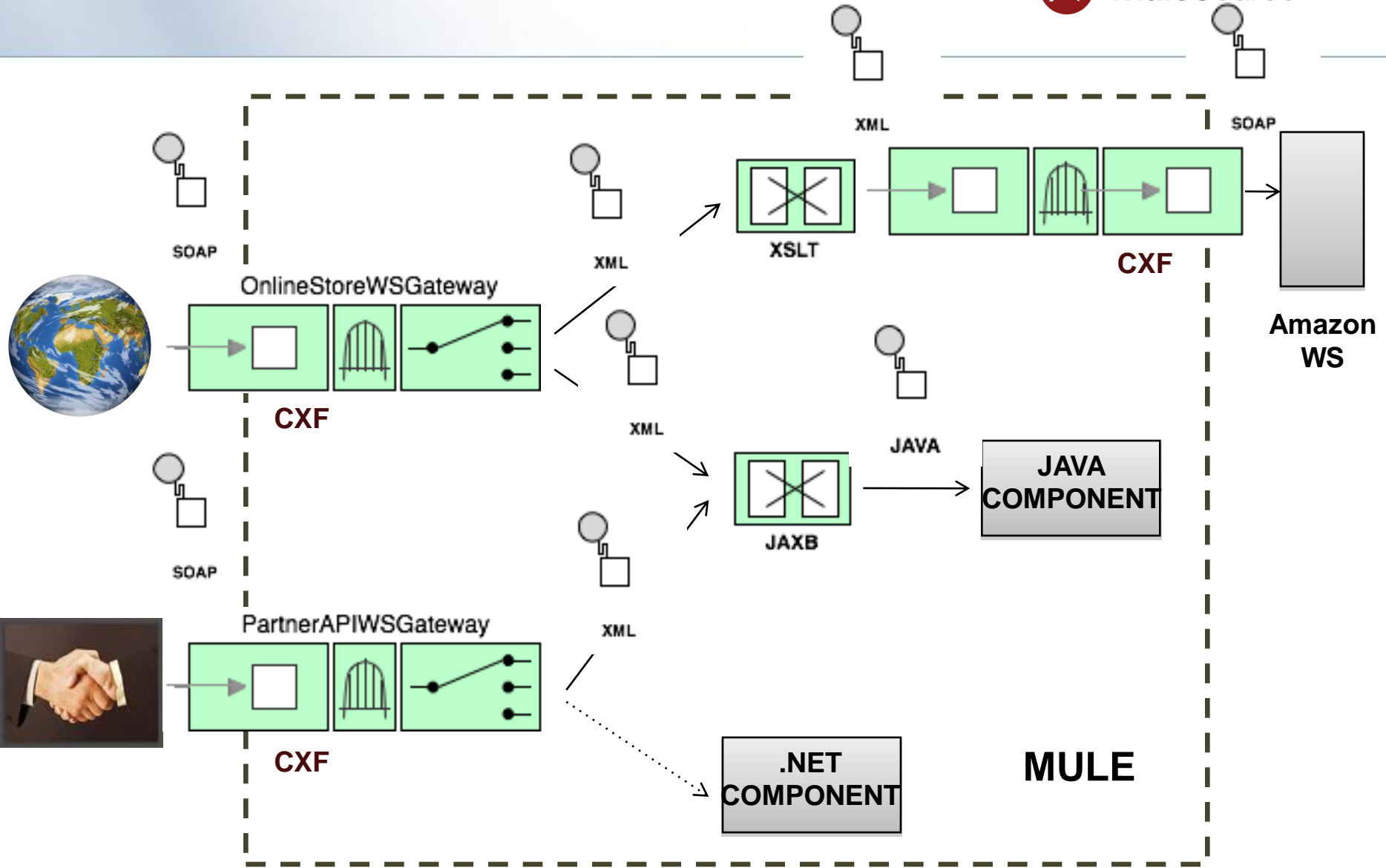


```
<service name="OnlineStoreGateway">  
  <inbound>  
    <cxfinbound-endpoint  
      address="http://host:port/path"/>  
    </inbound>  
    <component class="com.nile.StoreService"/>  
  </service>
```



```
<service name="OnlineStoreGateway">  
  <inbound>  
    <cxfinbound-endpoint address="http://host:port/path"  
      serviceClass="com.nile.StoreServiceInterface">  
    </inbound>  
    <component class="come.nile.StoreService"/>  
  <outbound>  
    ...  
  </outbound>  
</service>
```

# 'Nile' Mule Architecture



- ▶ Document/Literal Style WS
  - Ideally use a canonical data model
  
- ▶ CXF “proxy” mode
  - Turn off Data-Binding and Invocation
  - XML payload
  
- ▶ Content Based Routing
  - Route based on message content
  - Map Services, Operations or SOAPAction to outbound endpoints

- ▶ Forward WS Request
  - XML payload
  - Any transport
  - Any type of implementation (internal/external, Java/.Net, XML/Object)
  
- ▶ Catch all strategy to return a SOAPFault when no route
  
- ▶ Transform (bind) only when needed
  - Performance (Filter/route using XML, avoid transforming twice)
  - Interoperability (Not everything is JAVA, maybe want to proxy request elsewhere)

```
<service name="OnlineStoreGateway">
  <inbound>
    <cxf:inbound-endpoint address="http://localhost:8777/services/onlinestore"
      wsdlLocation="onlinestore/NileOnlineStore.wsdl"
      proxy="true" synchronous="true"/>
  </inbound>
  <outbound>
    <filtering-router>
      <outbound-endpoint address="..."/>
      <filter ref="..."/>
    </filtering-router>
    <custom-catch-all-strategy class="..."/>
  </outbound>
</service>
```

CXF Proxy



WSDL (Contract-first)



No component





**MuleSource**

the open source choice for SOA infrastructure

**Q & A**