Web services conversations: why do they matter and what they mean to you

Fabio Casati
Senior Scientist, HP Labs- Palo Alto

© 2004 Hewlett-Packard Development Company, L.P.
The information contained herein is subject to change without notice
Outline

- Web services basics
- Protocols
- Composition
- Opinions
- Research issues
Basics
B2B Integration before Web services

- Customer
- Supplier
- Warehouse

Diagram:
- Internal procurement requests
- Internal infrastructure
- Web server
- Internal infrastructure
- Web server

Copyright Springer-Verlag, 2004
Centralized approach
Distributed approach

customer

message broker
XYZ

customer's adapters

internal infrastructure

supplier

message broker
XYZ

supplier's adapters

internal infrastructure

copyright Springer-Verlag, 2004
Distributed approach (cont.)

- customer
- warehouse
- another party (XYZ)
- yet another party (ABC)

Middleware for:
- supplier-customer interaction
- supplier-warehouse interaction
- supplier-XYZ interaction
- supplier-ABC interaction

Supplier's adapters

Internal infrastructure

Copyright Springer-Verlag, 2004
Essence of Web services approach

- services
- standards
- decentralized protocols
Benefits

- Reduced heterogeneity (also useful for EAI)
- Enable interactions among peers

integrating application (contains the composition logic)

Web service-enabled broker

SmartQuotation  DBMS applications  SmartForecasting  sendmail application  XYZ

copyright Springer-Verlag, 2004
Costs

- Web service interface
- Access to internal systems
- Web services middleware
- Service interface
- Integration logic
- Conventional middleware
- Other tiers
- Service provider
- Clients from other companies
Basic Web services standards

- Basics: description and interaction
  - SOAP, WSDL
- Discovery/binding
  - UDDI
- Protocol
  - BPEL, WSCI, WSDL, WS-CDL
- Business processes
  - BPEL
- Management: WSMF, WS-R
- Extensions: WS-xyz
  - security
  - addressing
  - routing
  - coordination
  - transaction
- All of the above: ebXML

- More or less formal consortia
- Unprecedented speed
  - at least it used to be that way...
- Continuous evolution
“Historic” Web services standards

Customer

SOAP-based middleware

SOAP messages (invoke)

WSDL interfaces

Supplier

SOAP-based middleware

SOAP messages (publish)

UDDI registry

SOAP-based middleware

SOAP messages (bind)
Development and runtime support

- WSDL of service provider
- WSDL compiler (client side)
- WSDL compiler (server side)

Service requestor:
- Application object (client)
- Stub
- SOAP-based middleware

Service provider:
- Application object (service provider)
- Skeleton
- SOAP-based middleware

SOAP messages
Evolution or Revolution?

- Technology is evolutionary
  - revolutionary aspects (dynamic service selection, Semantic Web services) are only on paper for now

- Standardization and adoption pace, impact are revolutionary
A successful technology

• Very rapid adoption, even for mission critical applications and for B2B
  • although EAI still #1 application
• Relatively unaffected by the downturn

• Barriers that prevented an even faster adoption:
  • lack of skills
  • lack of vertical standards
  • immature security standards
Protocols and Conversations
Conversations and protocols

- complements interface definition
- key differences wrt conventional middleware
- opportunity for more middleware, simpler development
- protocols, choreography, conversation, orchestration, ...

Copyright Springer-Verlag, 2004
Multi-party conversations

1: requestQuote
2: orderGoods
3: checkShipAvailable
4: confirmOrder
5: makePayment
6: orderShipment
7: getShipmentDetail
8: confirmShipment
9: confirmShipment
Why business protocols

- Key opportunity and need enabled by Web services
  - loosely coupled, need more specs
  - conversational by nature
  - more middleware, more development support
Why protocols? matchmaking

- lots of interesting problems wrt **syntactic** compatibility
- in principle, it comes for free! All done by the middleware
Why protocols? development support

- generate skeletons
- global to role-specific

**top down**

- Protocol specs
- protocol compiler
- service skeleton
  
  **add biz logic**

- service implementation

**bottom up**

- Protocol specs
- protocol generator

- service implementation
Why protocols? middleware: routing, validation, logging

service provider

object for $P_1$

object for $P_2$

object for $P_3$

object for $P_4$

object for $P_5$

conversation controller

logs

protocol defs (BPEL)

service requestor

service requestor

service requestor

service requestor

service requestor

copyright Springer-Verlag, 2004
Why protocols? Management support

- Conversations are what matters from an external perspective
  - Conversation quality is what customers care about
- Business SLA, metrics, associated with conversations
- Metrics are ad hoc, not predefined
Different facets of protocol models

- Perspective
- Formalism
- Expressive power
Perspective

- Global (cooperative) vs service-centric (power play)
Formalism

- Sequence diagrams
- State machines/ statecharts
- Process-based models
Sequence diagrams

1: requestQuote
2: orderGoods
3: checkShipAvailable
4: confirmOrder
5: makePayment
6: orderShipment
7: getShipmentDetail
8: confirmShipment
9: confirmShipment
Statecharts

- good for service-centric
- they emphasize state (good for monitoring), reactive behavior
Process-based

- Service-centric
- not a workflow
- easy to go from protocol to implementation
- BPEL

```
receive requestQuote
reply requestQuote
receive orderGoods
invoke checkShipAvailable

send cancelOrder
send confirmOrder
receive makePayment
send orderShipment
receive confirmShipment
```
Expressive power

- Ordering not enough
- Implicit (timed) activations
- Data, conditions, state management
- Correlation
- Transactions
- Modularization, exception handling
Layers of protocols

<table>
<thead>
<tr>
<th>Verticals (e.g., RosettaNet)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Business Protocols (BPEL, WSCI, WS-CDL,..)</strong></td>
</tr>
<tr>
<td>Security, reliability, transactions,..</td>
</tr>
<tr>
<td>(ws-security, ws-rm, ws-coord, ws-transactions)</td>
</tr>
<tr>
<td>Encoding (SOAP)</td>
</tr>
<tr>
<td>Transport (http, email,..)</td>
</tr>
</tbody>
</table>
Composition
Service Composition

- develop services by composing other services
- implementation technology
- dream: develop by drag and drop (not new)
Failures of the past (and of the present)

- Workflow technology is the predecessor of service composition
- Failed to keep its promises
  - high license costs
  - complex software
  - heterogeneity

- But service composition is different (or, the business and IT environment is different)
  - standardization (components and composers)
  - maturity
  - reduced costs (small layer on top of other middleware)
Composition and conversation

conversational process
(not like workflows)
Again: development support

- generate composition skeletons (comes for free with BPEL)
- graphical tools
Composition middleware

Runtime support

Management and optimization support

Company A

(composite) Web service

Web services middleware

service composition support
(modeling and execution)

other tiers

logs

Company B

Web service

Web service

Company C

Web service

Web service

Company D

Web service

Company A

invoke checkShipAvailable
receive requestQuote
reply requestQuote
receive orderGoods
send confirmOrder
receive makePayment
send orderShipment
receive confirmShipment
send cancelOrder

copyright Springer-Verlag, 2004
Lots of standards here as well

- BPEL, BPM L
- WF standards (WFMC)

- BPEL appears to be the clear “winner”
Issues and non-issues

• Standardization
  • in the components
  • in the composer (less critical)
• Security and privacy
• Verification of compatibility w/ invoked services
• Process discovery – reduce modeling costs
• Development tools
• Monitoring and analysis tools

• Non issues (for now)
  • automatic goal-directed process definition, automatic planning
  • dynamic composition
  • generic dynamic binding
Opinions
Proliferation of standards

- Everybody is sickened by the number of specs
- Many of them are really needed...
- ... but there are lots of unnecessary duplicates
- Some are missing
- Proliferation dictated by middleware greed (and commercial interests)

- (more later)
BPEL is good for you

• not perceived as a protocol language, but it is
  • although mostly role-specific
• Key benefit: makes it easy to go from external specs (behavior) to internal specs (implementation)
• No need to learn many languages
• Rich
Languages are irrelevant

- tools matter
- users don’t want to read XML, they want to drag and drop
  - need tools to simplify development and runtime, that’s all users care about
- development of languages should be driven by what tools can do with it

- and that’s (also) why big SW companies drive “standards”
Semantics, and the Semantic Web

- Also irrelevant (for practical applications), at least for a few years

- Why?
  - dynamic binding is not an issue
  - static binding? barely
  - too complex

- Will be applicable and potentially very useful when
  - sophistication in WS reaches the point where static or even dynamic binding is used
  - there are tools that are easy to use

- Need to start from simple things, add semantics to the middleware
Research issues
Protocol compatibility

- enables better development support tools
Protocol compatibility (cont.)

- Substitutability
- Equivalence
  - absolute
  - relative to a client
- Pinpoint non-compatible parts
- Compatibility by layer
Protocols and composition

customer

local service offered by the supplier

warehouse

supplier

receive

orderGoods

invoke

checkLocalStock

inStock=false

invoke

checkShipAvailable

shippingAvail=false

send cancelOrder

inStock=true

shippingAvail=true

send confirmOrder

receive

orderGoods

invoke

checkLocalStock

inStock=false

invoke

checkShipAvailable

shippingAvail=false

send cancelOrder

inStock=true

shippingAvail=true

send confirmOrder
Static analysis

• Which properties are significant in a protocol?
  • termination? deadlocks?
  • relationship with interfaces

• Research in the early stages
Discovery

- extract protocol definition from WS definition, data analysis
- once extracted, enables better description, tool support

Conversation logs

Specs

Conversation logs
Runtime support

- Conversation controllers. What else?
- Interaction among layers

<table>
<thead>
<tr>
<th>controllers</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>web services</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

- Verticals (e.g., RosettaNet)
- Business Protocols
- Security, reliability, transactions,..
- Encoding (SOAP)
- Transport
State: who controls it?

- Verticals (e.g., RosettaNet)
- Business Protocols
- Security, reliability, transactions,
- Encoding (SOAP)
- Transport

State info at both ends

Controllers

Web services