Service Composition

- [ACKM04] Ch 8
Coordination vs Composition

• Coordination is *public*
  • client is aware of coordination protocol and participates in it
  • like a participant in 2PC

• Composition is *internal*
  • external client unaware it is happening
Hierarchy of Composite Services

- Supply chain
  - Inventory planning
    - ...
    - ...
  - Accounting
  - Procurement
    - Approval
  - Supplier
  - Another supplier
Most Common Approach

Customer

supply chain

inventory planning

accounting

procurement

supplier

approval

another supplier

Implement “root” of composite service in conventional high-level language
The internal application implements the composition logic, by invoking Web services as needed. No support is provided by the Web services middleware in this case.
Web Services Composition Middleware

- Composition Model and Language
  - *Composition Schema* specifies business logic
- Development Environment
  - GUI tools for development and deployment
- Run Time Environment
  - A *composition engine*
Web Services Composition Middleware

- Service composition model and language (usually characterized by a graphical and a textual representation)
- The run-time environment executes the Web service business logic by invoking other services (through SOAP and HTTP modules)

Web service composition middleware

- Development environment
- Run-time environment (composition engine)
- Schema definitions
- Composite service execution data
- Other Web services middleware (e.g., SOAP engine and conversation controller)

- Services offered by other providers:
  - Supplier
  - Warehouse
  - Accounting

- Schema designer
- A service provider
Web Services Composition Middleware

The composite service is directly implemented at the middleware level, by the composition engine.
Contrasting the two schemes ...

- The basic difference: with WS Composition middleware the lower tiers are not involved in composition ...

CS530 S05
Contrast - Composition vs Coordination

The procurement business protocol executed among Web services

1. requestQuote
2. orderGoods
3. confirmOrder
4. makePayment

If the supplier is implemented by means of composition technologies, then its business logic is defined by a composition schema and its execution is driven by a composition engine.

Depending on the implementation of the (composite) service, the supplier may contact other Web services. The customer is unaware of these interactions, that may occur according to other protocols.

Another Web service, possibly offered by another company
Yet another Web service
Composition Models

• Dimensions of Composition Model:
  • Component
    • (Web Service)
  • Orchestration
    • activity diagram, Petri net, pi-calculus, ...
  • Data
    • specification and exchange
Composition Models - II

- Service Selection
  - static or dynamic
- Transactions
  - transactional semantics of composition
- Exception Handling
  - abortion and recovery
Component Model - How General?

- WSDL
  - restrict applicability
  - easy implementation
- XML messaging
  - more general
  - harder integration
Orchestration Models

- Illustration:
  - Customer places order with supplier
  - Supplier implemented using composition
Orchestration Models

- Activity Diagram ...

```
orderGoods

supplier

receive orderGoods

invoke checkLocalStock

inStock=false

invoke checkShipAvailable

shippingAvail=false

send cancelOrder

inStock=true

shippingAvail=true

send confirmOrder

warehouse

checkShipAvailable

local service offered by the supplier

checkLocalStock

cancelOrder

confirmOrder

customer

orderGoods
```
Orchestration Models

- Activity Hierarchy ... *(deprecated)*

```
process order
  sequence
  receive orderGoods
  invoke checkLocalStock
  discriminate based on local search
    inStock=false
      search external supplier
        sequence
        invoke checkShipAvailable
      send confirmOrder
    inStock=true
      discriminate based on external search
        choice
        shippingAvail=true
          send confirmOrder
        shippingAvail=false
          send cancelOrder
```
Orchestration Models

• Statecharts

[start on new order request]

[started]

/searching for products locally

/local search complete(inStock) [inStock=false]/start "invoke checkShipAvailable"

/local search complete(inStock) [inStock=true]/start "send confirmOrder"

/external search complete(shippingAvail) [shippingAvail =false]/start "send cancelOrder"

/external search complete(shippingAvail) [shippingAvail =true]/start "send confirmOrder"

/order canceled

/order completed

[condition]/action
Orchestration Models

- Petri Nets *(yuck)*

START (upon invocation of orderGoods operation)

invoke checkLocalStock

LOCAL SYSTEM ACCESSED

Do nothing inStock=true

EXTERNAL SUPPLIER ACCESSED

invoke checkShipAvailable
inStock=false

Do nothing shippingAvail=false

send cancelOrder

COMPLETE (CANCEL)

shippingAvail=true

send confirmOrder

COMPLETE (CONFIRM)

READY TO SEND CONFIRMATION
Orchestration Models

- Pi-calculus

\[ A ::= B . C \text{ (sequential)} \]
\[ \text{or} \quad B|C \text{ (parallel)} \]
\[ \text{or} \quad B + C \text{ (choice)} \]
\[ \text{or} \quad [\text{var}=\text{value}] B \text{ (condition)} \]

\[ A = \text{receiveOrderGoods} . \text{invokeCheckLocalStock} \]
\[ B = [\text{shippingAvail}=\text{false}]\text{sendCancelOrder} \]
\[ + [\text{shippingAvail}=\text{true}]\text{sendConfirmOrder} \]
\[ C = \text{invokeCheckShipAvail} . B \]

\[ \text{Procurement} = A . ( ([\text{inStock}=\text{false}]C) \]
\[ + ([\text{inStock}=\text{true}]\text{sendConfirmOrder}) ) \]
Orchestration Models

- Event-Condition-Action rules
- Analogous to triggers, active databases

```plaintext
ON receive orderGoods
IF true
THEN invoke checkLocalStock;

ON complete(checkLocalStock)
IF (inStock==true)
THEN send confirmOrder;

ON complete(checkLocalStock)
IF (inStock==false)
THEN invoke checkShipAvailable;

ON complete(checkShipAvailable)
IF (shippingAvail ==true)
THEN send confirmOrder;

ON complete(checkShipAvailable)
IF (shippingAvail ==true)
THEN send cancelOrder;
```
Data

• Data Types
  • Control flow data (simple)
  • Application data
    • “black box” - URLs (simple, opaque)
    • typed data / XML (inefficient, blobs)
Data Transfer

Two classes:

<table>
<thead>
<tr>
<th>custName</th>
<th>“Homer Smith”</th>
</tr>
</thead>
<tbody>
<tr>
<td>partNum</td>
<td>123-456-98765</td>
</tr>
<tr>
<td>quantity</td>
<td>3</td>
</tr>
<tr>
<td>etc</td>
<td></td>
</tr>
</tbody>
</table>

Blackboard

Blackboard is more “intuitive”
... but ...
Data Flow makes control dependencies and data races explicit

Data Flow
Service Selection

- Static binding
  - service at fixed URL
- Dynamic binding by reference
  - service URL in a process variable
- Dynamic binding by lookup
  - look up service URL in directory
- Dynamic operation selection
  - like procedure valued variable
Dynamic Binding by Lookup

Newly added node that accesses a (statically specified) UDDI registry to determine which warehouse should be contacted for the product being ordered. The result is stored into the warehouse process variable. Note that in practice several invocations of the UDDI API may be needed to get the desired information.

Variables:
- warehouse: URI
- inStock, shippingAvail: bool
- customer: String
...

Subsequent nodes can use the reference approach and determine the URI based on the value of the warehouse variable.
Transactions

- ACID transactions
- Saga-like (with compensation)
Sagas

• Web Services trend: service responsible for
  • transaction
  • compensation
• (but note WS-Transaction compensates all-at-once rather than in reverse sequence)
• a significant change to the model!
Exceptions

• Flow-based
  • test return values and write code ...

• Try-catch-throw
  • e.g. Java

• Rule-based
  • ECA
• Already said: composition is private, coordination is public ...

• So: must design composed services that can participate in coordination protocols
Procurement Coordination Protocol

- customer
  - requestQuote (to supplier)
  - orderGoods (to supplier)
  - makePayment (to supplier)
  - confirmShipment (to warehouse)

- supplier
  - checkShipAvailable (to warehouse)
  - confirmOrder (to customer)
  - makePayment (to customer)
  - orderShipment (to warehouse)
  - cancelOrder (to customer)

- warehouse
  - warehouse confirms
  - warehouse cancels
  - getShipmentDetails (to customer)
  - confirmShipment (to supplier)
  - confirmShipment (to supplier)

Customer, Supplier, and Warehouse interactions.
Supplier View

- customer
  - requestQuote (to supplier)
  - orderGoods (to supplier)
  - makePayment (to supplier)

- supplier
  - checkShipAvailable (to warehouse)
  - confirmOrder (to customer)
  - cancelOrder (to customer)
  - orderShipment (to warehouse)

- warehouse
  - confirms
  - cancels
  - confirmShipment (to supplier)
Translate Operations to Activities

- Request/Reply from R -> Receive/Reply
- Send from R -> Receive
- Send to R -> Send
- Request/Reply to R -> Invoke
Result of Translation

- receive requestQuote
- reply requestQuote
- receive orderGoods
- invoke checkShipAvailable
- send cancelOrder
- send confirmOrder
- receive makePayment
- send orderShipment
- receive confirmShipment
From Protocol to Executable Process

receive requestQuote
invoke lookupQuote
reply requestQuote
receive orderGoods
invoke checkShipAvailable

shippingAvail=false
send cancelOrder

shippingAvail=true
send confirmOrder
receive makePayment
invoke collectPayment
send orderShipment
receive confirmShipment
Where is Conversation Control?

- Unfortunately, two places ...
(Recall) WS Conversation Controller

- B: conversation compliant with a business protocol
- H: conversation compliant with an horizontal protocol
(Recall) Composition Middleware:

- Service composition model and language (usually characterized by a graphical and a textual representation)
- The run-time environment executes the Web service business logic by invoking other services (through SOAP and HTTP modules)

Web service composition middleware:
- Development environment
- Run-time environment (composition engine)
- Schema definitions
- Composite service execution data

Other Web services middleware (e.g., SOAP engine and conversation controller)

Services offered by other providers:
- Supplier
- Warehouse
- Accounting

A service provider

Schema designer
Put Them Together ...

- Composition Engine is required to do routing without an explicit context ID ...
- e.g. Application’s order ID

(messages related to protocols implemented by basic Web services (or anyway services implemented by means of conventional programming languages)

(instance of a composition schema)

the engine has to match messages with instances, just like the conversation controller has to match messages with objects

(messages related to protocols implemented through service composition technologies)
Better ...

service provider

object (Web service implementation)

Instance of a composition schema

composition engine

conversation controller

service requestor

B: conversation compliant with a business protocol

H: conversation compliant with a horizontal protocol
BPEL

- Business Process Execution Language
- WS Specification *du jour* for Composition
Scope of BPEL

Abstract and/or executable process
orchestration,
variables and data transfers,
exception handling,
correlation information (for instance routing)

Variables:
warehouse: URI
inStock, shippingAvail: bool
customer: String
...

roles

port types

customer

warehouse

local service offered by supplier

supplier

receive orderGoods

invoke checkLocalStock

invoke checkShipAvailable

invoke cancelOrder

invoke confirmOrder
BPEL Orchestration Activities

• Basic Activities
  • invoke / receive / reply
  • assign / wait

• Structured Activities
  • sequence
  • switch / pick
  • flow
  • while (!)
Example

processOrder
sequence
receive orderGoods
invoke checkLocalStock
chooseLocal
switch
inStock=false
searchExternal
sequence
invoke checkShipAvailable
inStock=true
invoke confirmOrder
chooseExternal
switch
shippingAvail=true
invoke confirmOrder
shippingAvail=false
invoke cancelOrder
BPEL Data Transfer

- Blackboard approach
- Data types are WSDL message types
- Assignment
  - input / output parameters
  - explicit using assign activities

<table>
<thead>
<tr>
<th>custName</th>
<th>“Homer Smith”</th>
</tr>
</thead>
<tbody>
<tr>
<td>partNum</td>
<td>123-456-98765</td>
</tr>
<tr>
<td>quantity</td>
<td>3</td>
</tr>
<tr>
<td>etc</td>
<td></td>
</tr>
</tbody>
</table>
BPEL Service Selection

- Partner Link - type and definition

**partner link type**

orderLT

**port type**

supplierPT

**customer**

**warehouse**

**supplier**

**local service offered by the supplier**

**partner link definition**: Further qualifies the interactions occurring through a partner link type. Its definition refers to a partner link type and specifies the role played by the composite service as well as the one played by the other partner.

```
<partnerLink name="customerP"
  partnerLinkType="orderLT"
  myRole="supplier"
  partnerRole="customer">
</partner>
```
BPEL Scopes, Exceptions, Transactions

- Nested Scopes
- Exception Handling
- Transaction
- Compensation
Scopes Example - Exceptions

processOrder
  sequence

receive orderGoods

invoke checkLocalStock

chooseLocal
  switch
    inStock=false
    searchExternal
      sequence
      invoke checkShipAvailable
      invoke confirmOrder
      chooseLocal
      switch
        inStock=true
        invoke confirmOrder
        invoke cancelOrder

invoke checkLocalStock

scope of the searchExternal activity
due to the behavior of the default handler, implicitly associated with each activity, a fault F occurring in activity send confirmOrder would propagate up until activity searchExternal, where the handler resides

includes fault handler for fault F
Routing

- Define Correlation Sets
  - set of data items used as routing key

orderID can be used for correlating the two messages

message checkAvailability
  - orderID
  - requestedDeliveryDate
  - deliveryLocation
  - ...

message availability
  - orderID
  - shippingAvail