Service Interaction Patterns

Alistair Barros
SAP Research Centre, Brisbane, Australia

Marlon Dumas and Arthur ter Hofstede
Queensland University of Technology, Australia

Problem statement

- Languages for capturing service interactions:
  - Implementation: consensus around BPEL, WS-CDL?
  - Modelling: space still open (UML AD/SD, BPMN)
- These languages focus on:
  - Bilateral interactions between services or roles
  - Strict (wait-for-all) synchronization
  - Parties collaborate, as opposed to compete
  - Requestor-respondent-requestor scenarios

Do these characteristic always hold?

Problem statement (cont.)

- Many scenarios have been gathered, but not always used in the development of these languages, e.g.
  - BPEL and WS-CDL requirements task forces
  - B2B protocol standards, e.g. RosettaNet, xCBL
  - ebXML, ebBP
- Many of these scenarios break the above assumptions

Proposal: service interaction patterns

- Collect scenarios
- Identify classification dimensions and categories
- Consolidate scenarios into patterns
- Evaluate capabilities & limitations of existing languages
- Use patterns as basis for language design

Classification

Dimensions:

- Max number of parties involved in the interaction → bilateral vs. multilateral
- Max. number of exchanges between two parties → single-transmission vs. multi-transmission
- Is receiver of the response the sender of the request? → round-trip vs. routed

Classification (cont)

Four categories of patterns:

- ***Basic bilateral***: single-transmission bilateral interaction patterns
- ***Basic multilateral***: single-transmission multilateral interaction patterns
- ***Multi-transmission*** interaction patterns
- ***Routed*** interaction patterns

Patterns overview

- Basic bilateral:
  1. Send
  2. Receive
  3. Send/receive
- Basic multi-lateral:
  4. Racing messages
  5. One-to-many send
  6. One-from-many receive
  7. One-from-many send/receive
- Multi-transmission:
  8. Contingent sending
  9. Multi-responses
  10. Transactional multi-cast notification
- Routed:
  11. Request with referral
  12. Relayed request
  13. Dynamic routing

One-to-many send/receive

Description

- Requests are sent to several parties
- Responses required in given timeframe –not all parties may respond and some responses may arrive outside timeframe
- Success of interaction depends on set of responses gathered

Example

Outsourced validation tasks – e.g. insurance companies redundantly outsource aspects of claims search to private/public search brokers

Issues/choices

- Set of parties not necessarily known in advance
- Several such one-to-many interactions may occur concurrently, and responses should not get mixed
- Individual send actions may result in faults

One-to-many send/receive

View animations

One-to-many send/receive
BPEL pseudo-code

```
while (more partners)
    set partner link to the address of next partner
    send message to next partner (invoke without output);
begin scope
    onAlarm timeLimit : throw timeoutFault
    catch timeoutFault : do nothing (empty)
    while (not stopCondition)
          receive one response;
    end while
end scope
switch (successCondition)
    ...
```

Issue: it's tricky to insert correlation sets in this solution

Patten 8: Contingent requests

Description:

Party X makes a request to party Y. If X does not receive a reply within a timeframe, X sends a request to another party Z, and so on.

Example:

A travel agency allows contingent reservations of flights. Customers nominate the preference of flight carriers. In order of preference, reservations are sought in short-timeframes. If a reservation is secured, the interaction ends.

Issues/design choices:

- After contingency request is issued, response to original request may arrive
- Multiple responses may arrive at the same time

Contingent requests

View animations

Contingent requests
BPEL pseudo-code

```
while (no response && more partners)
    begin scope
        onAlarm timeLimit : throw timeoutFault
        catch timeoutFault : do nothing (empty)
        invoke next partner;
        response := true
    end scope
end while
```

Issue: *if first partner responds before second one, its response is ignored*

Contingent requests
BPEL pseudo-code (improved solution)

```
while (no response && more partners)
    begin scope
        onMessage X :
            begin scope
                onAlarm timeLimit : throw timeoutFault
                catch timeoutFault : do nothing (empty)
                invoke next partner;
                response := true;
            end scope
        send a message X to myself
    end scope
end while
```

Conclusion of BPEL evaluation

- Basic bilateral patterns supported.
- Basic multi-lateral patterns: complete and direct support not available.
- Multi-transmission patterns: solutions possible but no direct support. Transactional multicast out of scope of BPEL.
- Routed interaction patterns: solutions possible but no direct support. Dynamic routing out of the scope of BPEL.

Completed and future work

- ✓ Documentation of the 13 patterns
- ✓ Solutions in BPEL (where possible), see: http://www.serviceinteraction.com
- ✓ Formalisation (using ASMs) – ICFEM'2005
- Derive a (meta-)model of service interactions
- Define languages for service interactions modelling