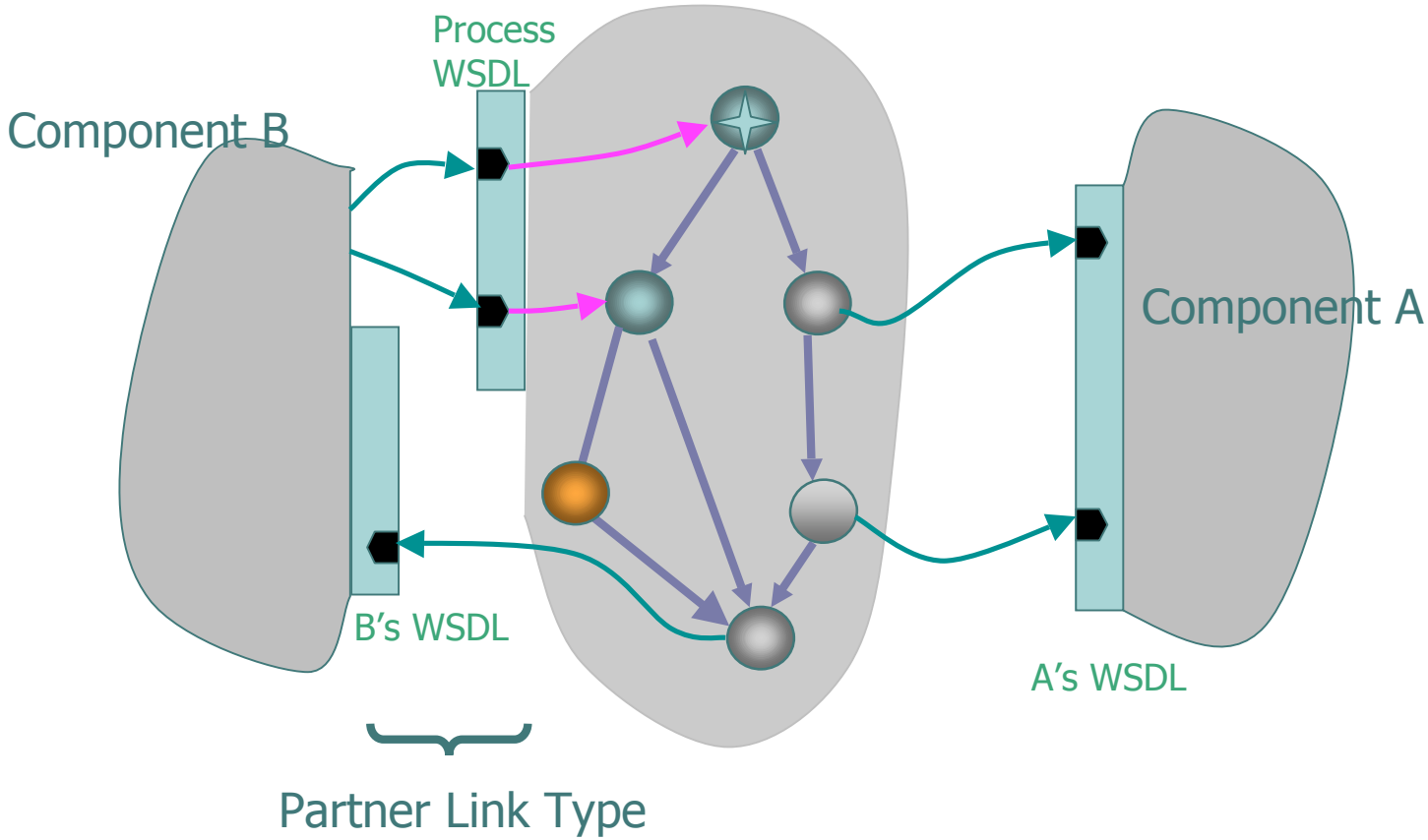
From RosettaNet PIP Documents To BPEL Processes: *A Three Level Approach for Multi–Party Business Processes*

Rania Khalaf (rkhalaf@us.ibm.com)
IBM T.J. Watson Research Center
(Presented by Francisco Curbera, curbera@us.ibm.com)

Motivation

- Provide standardized process definitions for a particular domain.
- At the same, lower barrier to entry for small and medium businesses.
- What is the problem?
  - RosettaNet has Centralized Design but Decentralized Execution
  - Highly specialized technical expertise are highest at the center (RosettaNet itself) and must not be required of enacting partners (possibly small and medium businesses with no BPEL skills).
  - The hardest steps must therefore be done by RosettaNet, minimizing choices without compromising flexibility at steps done by enacting parties.
- Encode and exploit process similarities into templates and provide a streamlined mechanism to generate executable processes.
- Simplify to enacting parties the main difficulties of abstract process:
  - compliance of executables, and
  - partner process compatibility.

BPEL Composition of Web Services



BPEL: Abstract and Executable Processes

- BPEL provides one approach for both:
  - Executable processes
    - Contain the partner's business logic behind an external protocol
  - Abstract processes:
    - Partial process definitions. Examples:
      - Define the publicly visible behavior of some or all of the services an executable process offers
      - Define a process "template" embodying domain-specific best practices

RosettaNet

RosettaNet: consortium of companies in the Electronics industry that define an open e-business environment.

The RosettaNet Implementation Framework (RNIF) defines part of the interaction:

- Generic Business **message structure**
- Steps required for transmitting the message between trading partners
- Message **packaging** and unpackaging
- **Transmission** protocols
- **Error** handling
- **Validation** of some content

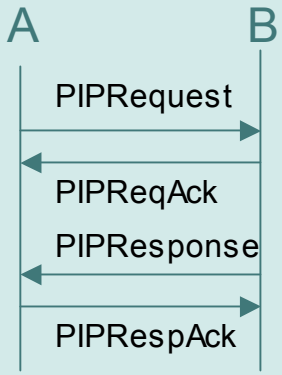
Individual PIP Specifications define Complete Business Level Requirements that sit on top of an RNIF compliant system

Partner Interface Processes (PIPs) define:

- Roles involved in the interaction (re; buyer and seller)
- Structure, content, and ordering of the business messages
- Timing constraints
- Security requirements
- Non-repudiation requirements

RosettaNet PIPs

- The pattern of message exchanges is reused for different PIPs with certain points of variability.
- Challenge in properly splitting RNIF/PIP info into:
  - business level processing (BPEL) from
  - infrastructure level processing (WS-Security, WS-RM, etc)
- Focus on one such pattern: Asynchronous Two-Action PIPs.
- Consists of two main business messages:
  - PIPRequest and PIPResponse, each has its corresponding business level acknowledgment
  - Possible dictionary-level content validation
  - Time-outs and corresponding fault notification.



Asynch 2-action PIP (one possible trace of a successful run, no faults)

Proposal:
A 3-Step Approach to PIPs

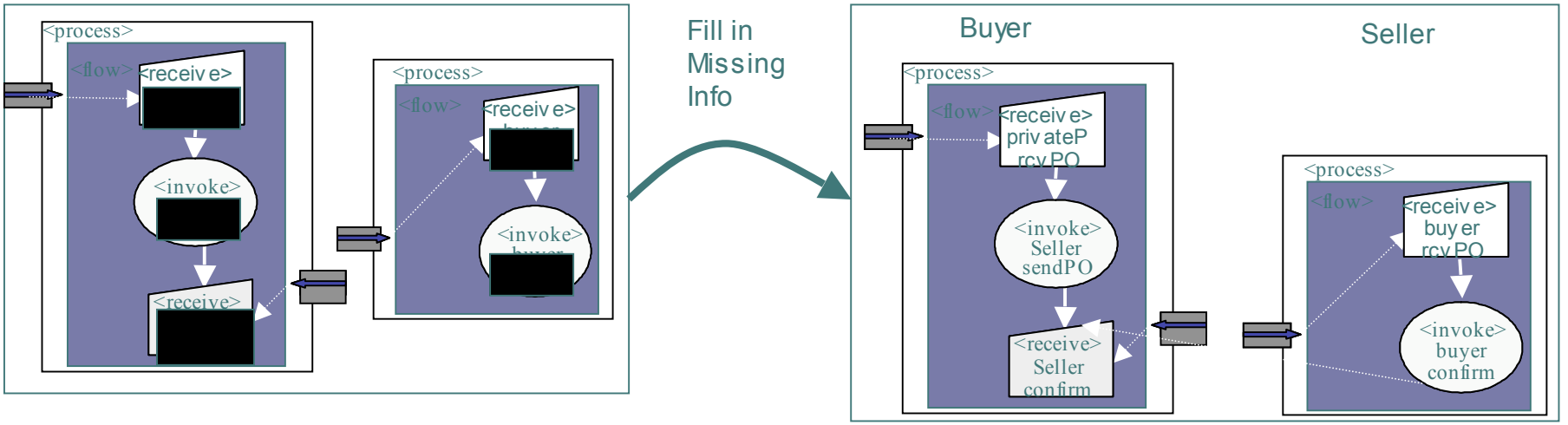
Templating→Specialization→Implementation

- **Templating**: Creating BPEL abstract "templates" that capture the message exchange and behavioral pattern of each party (one template for several PIPs).
- **Specialization**: Fill templates to create full valid *abstract BPEL* processes (provides a BPEL the definition of a particular PIP).
- **Implementation:** Use simple completion rules to create executable artifacts from the abstract processes (many executables of a particular PIP).
  - Results in a per party implementation with *minimal effort and low adoption barrier*.

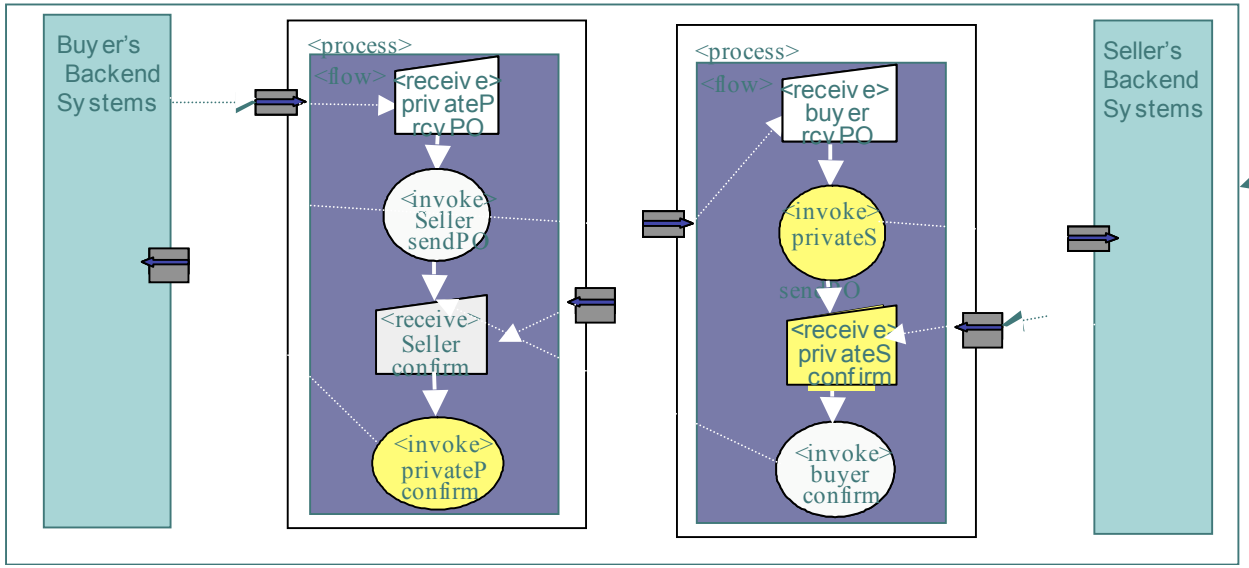
An intuitive (v. simplified) look:

1. Templates

2.Abstract Processes



3.Executable Processes (optional)



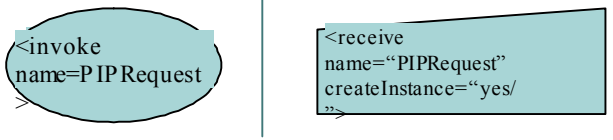
1 Templates

- Template: pseudo-abstract BPEL process that captures a behavioral pattern, with placeholders for completion.
- Templating **placeholders**: by omission of XML artifacts.
- For decision points, fork on the value of ‘opaque’ variables.
- BPEL points of variability for Asynch 2 action PIPs:
  - *PartnerLinks: Use names from one’s specific PIP. No ambiguity in template because PIPs are always two-party.*
  - *PortType and Operation names: From* the portType and operation on the WSDL of the specific PIP.
  - *Variable Message Types: From WSDLs of PIP.*
  - *Timer values* on the alarm handlers from PIP definition
  - *Correlation sets*. From the WSDLs of the PIP.

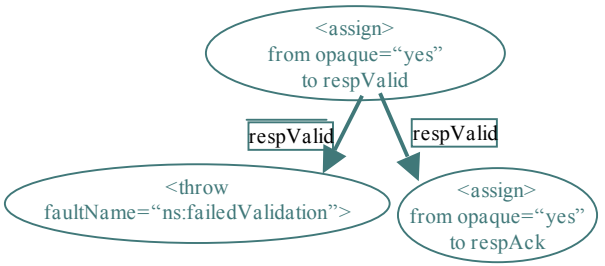
1 Templates, cont'd

Information must be provided on conventions used for the template. Here:

- The name of an 'invoke' on the sender process matches the name of the 'receive' on the receiver process.



- Validation, optional on some PIPs, is shown with activities and variables named *Valid.

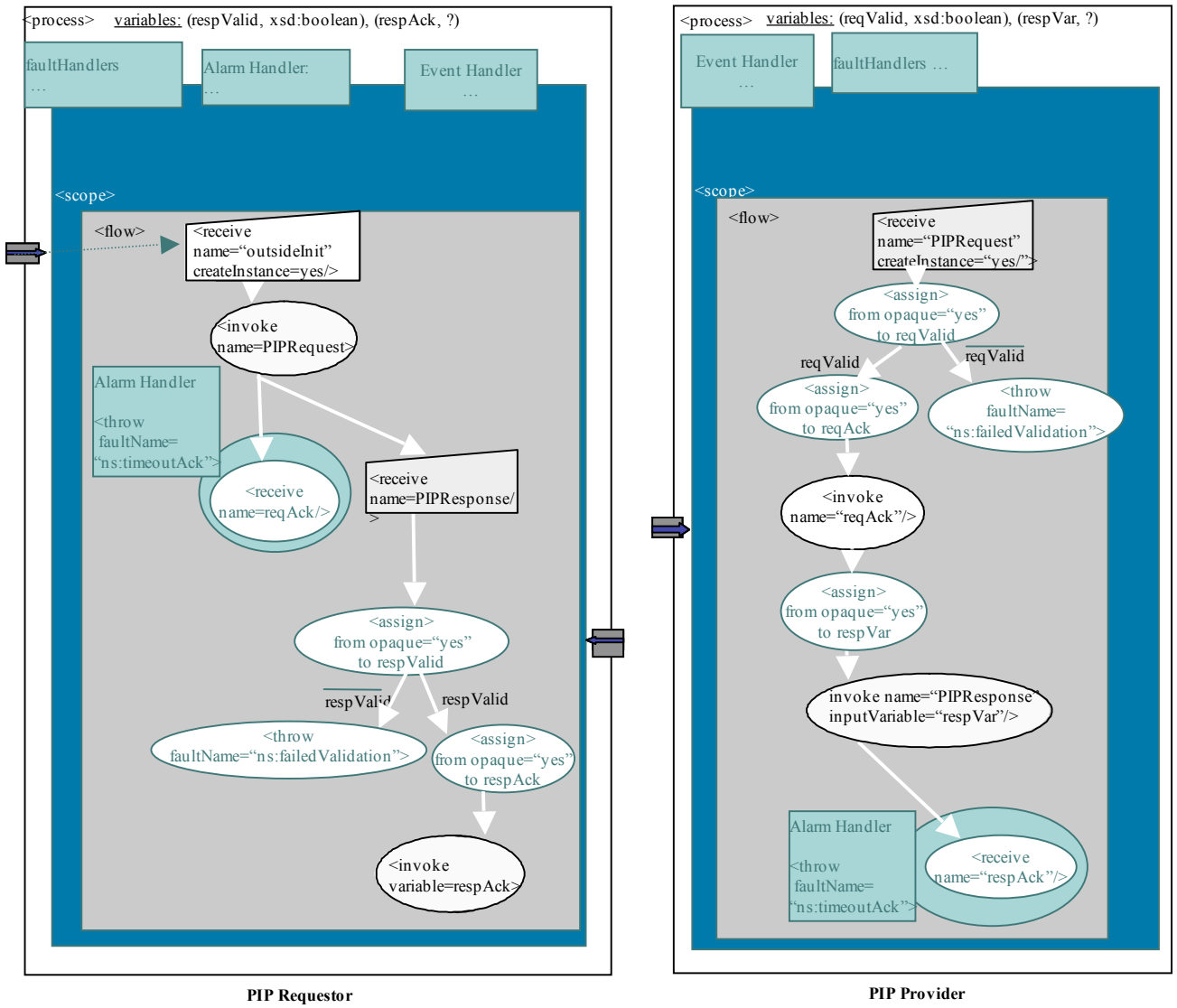


- A reliable message delivery mechanism with settable retry intervals is required by an implementor.

Asynch 2-Action PIP templates

- 2 Parties → 2 BPEL processes.
- Simple send/receive is just invoke/receive
- For throwing errors after a time-out:
  - Use alarm handlers for the timer.
  - If the alarm fires, then throw a fault that send a message to the partner and kicks off the ‘Notification of Error PIP’ (out-of-band error notification).
- Catching errors on each partner:
  - Use a global message handler. Abort the process on reception and provide a completion point for proprietary clean-up.
- Retries: Handled by messaging layer. If maximum retries reached, throw BPEL error as above.

Asynch 2-Action PIPs



2 Specialization

- Creating the Abstract BPEL processes:
  - Preparation: WSDL creation from PIP defintion:
    - XML Schema definitions from PIP DTDs
    - One portType per party containing one-way operations that can accept:
      - The PIP messages
      - An error message from the partner.
    - Additionally: One portType for initiator. One portType for Notification of Error PIP.

2 Specialization cont'd

- To complete the template:
  - Create the partnerLink definitons.
    - For each process, provide the partnerLink name on all interaction activities that interact with the other PIP party.

partnerLinks: sellerPL,0a1PL,initializerPL

partnerLinks: buyerPL,0a1PL

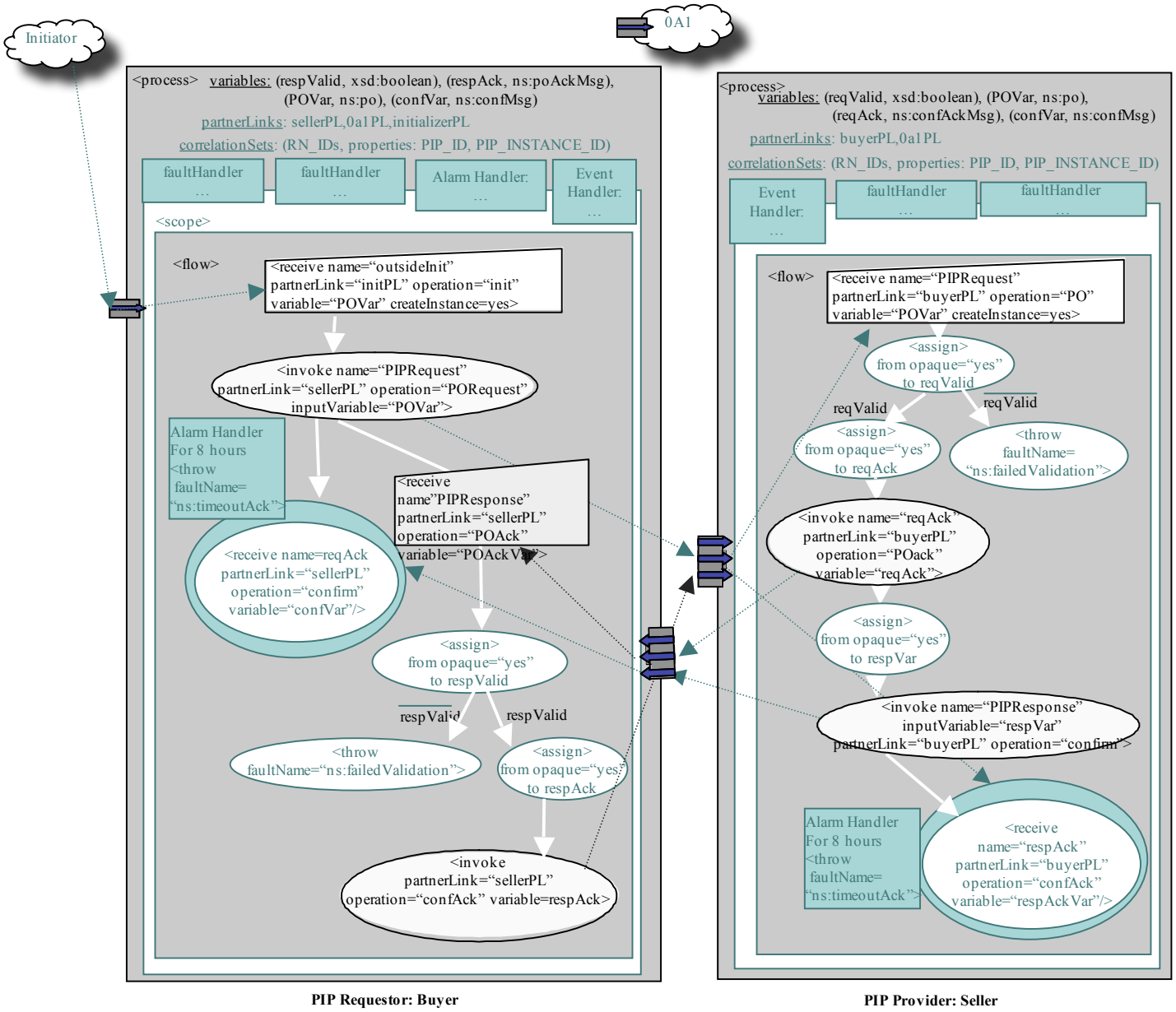
  - Add the operations and variables:
    - Any 2-action PIP has a request message and response message.
    - In the template, use the operation and variable for the PIPRequest message on activities called PIPRequest. Use operation and variable for the PIPResponse message on activities called PIPResponse.

```
<invoke name="PIPRequest"
  partnerLink="sellerPL"
  operation="PORequest"
  inputVariable="POVar">
```

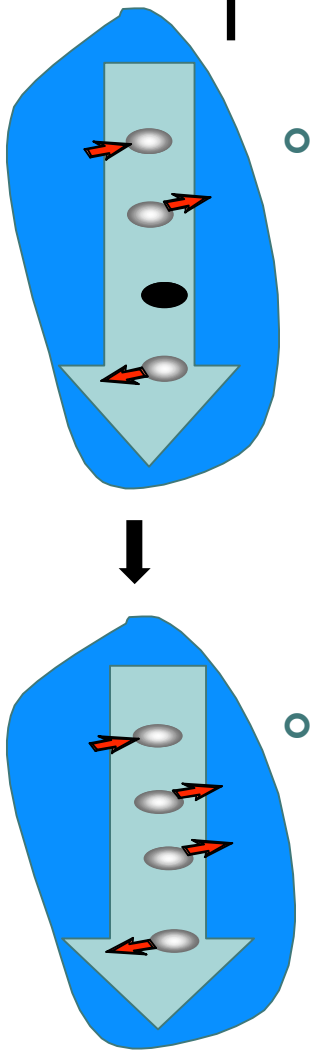
```
<receive name="PIPRequest"
partnerLink="buyerPL"
operation="PO" variable="POVar"
createInstance=yes>
```

  - Add timer values:
    - Global timer from PIP Definition
    - Other timers = retry_count * retry_interval.
  - Correlation:
    - Use the PIP Code and PIP Instance ID (part of RN messages).

Purchase Order PIP Abstract BPEL Processes



3 Implementation



- Follow simple rules to create compliant executables:
  - abstractProcess='no'
  - Add any new partnerlinks, correlation sets, and variables.
  - Replace each Opaque assign activities with a sequence activity containing:
    - Interaction activities, that write to the opaque variable with a new partner (backend systems).
    - One or more assign activities with no opacity.
  - Add assigns in the handlers for setting variable values.
  - Optionally add fault handler activities for notifying backend systems of failures as necessary.
- Note that these rules do not allow addition of any interactions with the existing partner or changes to any of the pre-specified behavior and are thus true to the abstract processes.

Conclusion

- Provides a use case for abstract BPEL processes for domain specific needs.
- One more level of BPEL abstraction: templates.
- Provides a use-case of abstract process definition where only maintaining the interactions or only filling in opaque tokens are not enough to provide a compliant implementation.
- In cases of centralized design but decentralized execution, non-centralized process models such as BPEL provide a more streamlined approach for partner participation
- Difficutly in design decreases as the *required* skill level of the user decreases:
  - Highest at RN (template and AP design); lowest at participant (completion rules)

Notes, references, and acks

- BPEL V2.0 has provided a more generalized concept of opacity that will make templates more usable.
- For a discussion on mapping to Web Services and business vs infrastructure work in a PIP, see:

P. Bunter et al. An approach to moving industry business messagung standards to web services. Online at

http://www-128.ibm.com/developerworks/webservices/library/ws-move2ws.html

- Acknowledgements:
  - Francisco Curbera for discussions and for presenting on my behalf.
  - Paul Bunter and Sreedhar Janaswamy especially. Keeranoor Kumar, Ralph Hertlein, Peter Williams, Shishir Saxena for work on project. Axel Martens and Frank Leymann for advice.