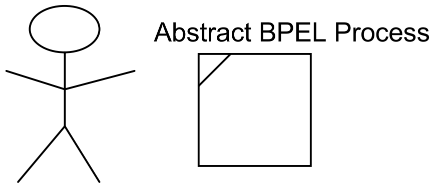
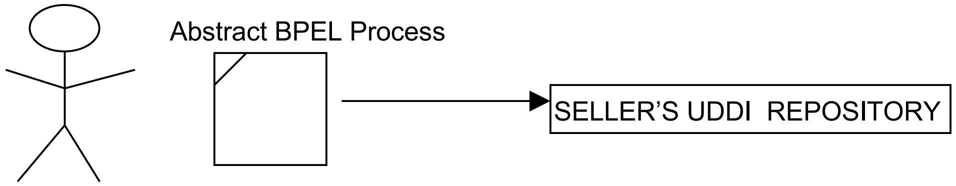
Abstract Process Example: Automated Negotiation

Seller (BPEL4WS + J2EE Shop)

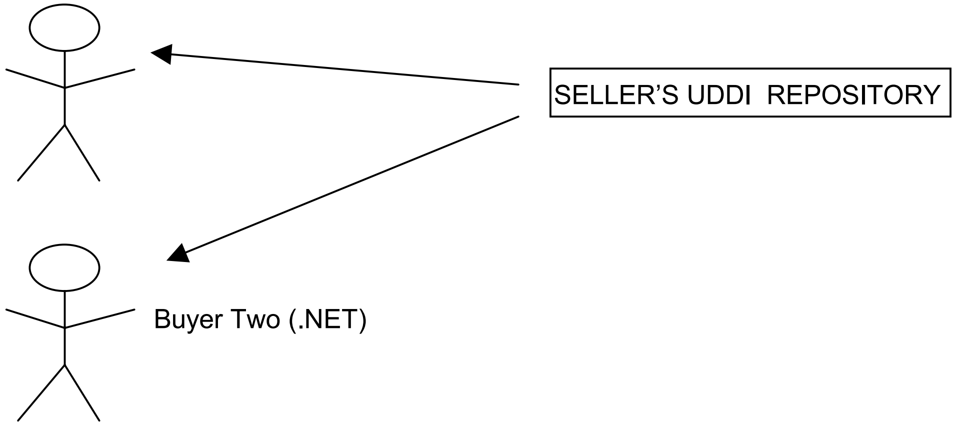


(1) Seller writes abstract BPEL process that embodies trading rules:
(2) Buyer sends offer.
(3) Seller replies with acceptance, counter-offer, or “no deal.”
(4) Buyer replies with acceptance, counter-offer, or “no deal.”
(5) If after three rounds with no acceptance, then it is “no deal.”
If there is an acceptance, then buyer sends order based on acceptance.



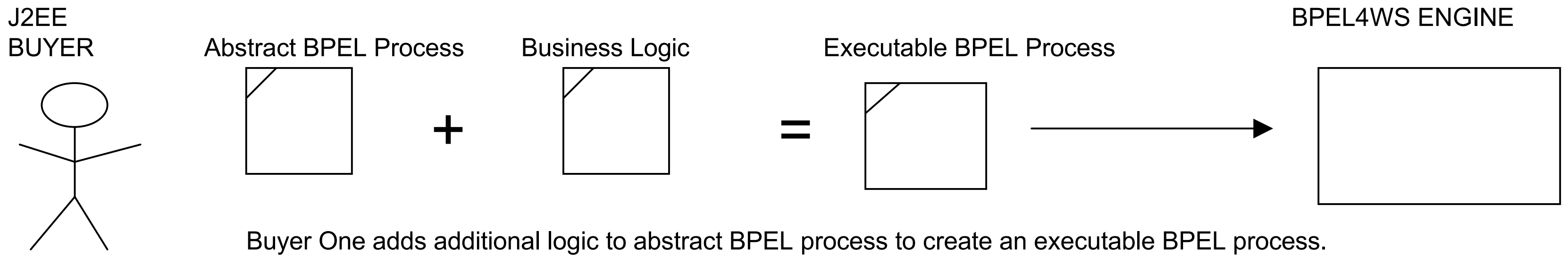
(2) Seller publishes abstract BPEL process in a public place

Buyer One (BPEL4WS + J2EE)

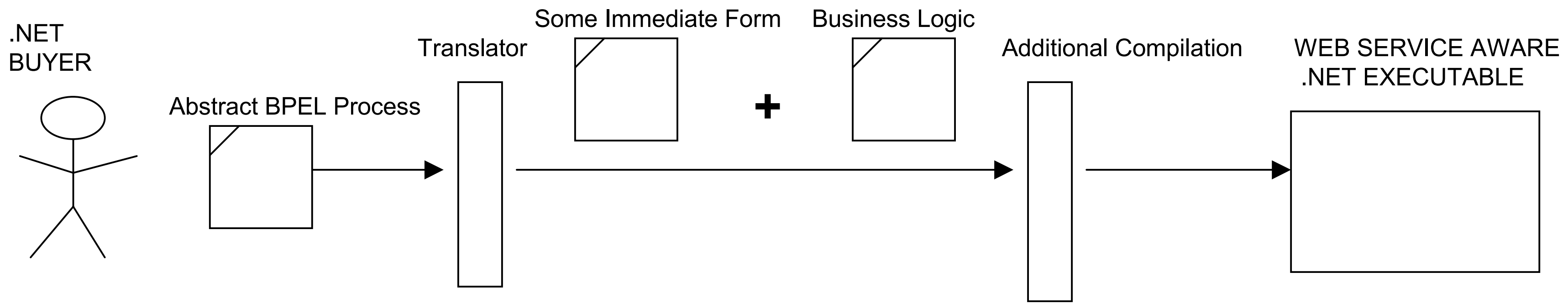


(3) Third party buyers read abstract BPEL process

Abstract Process Example Continued : Automated Negotiation



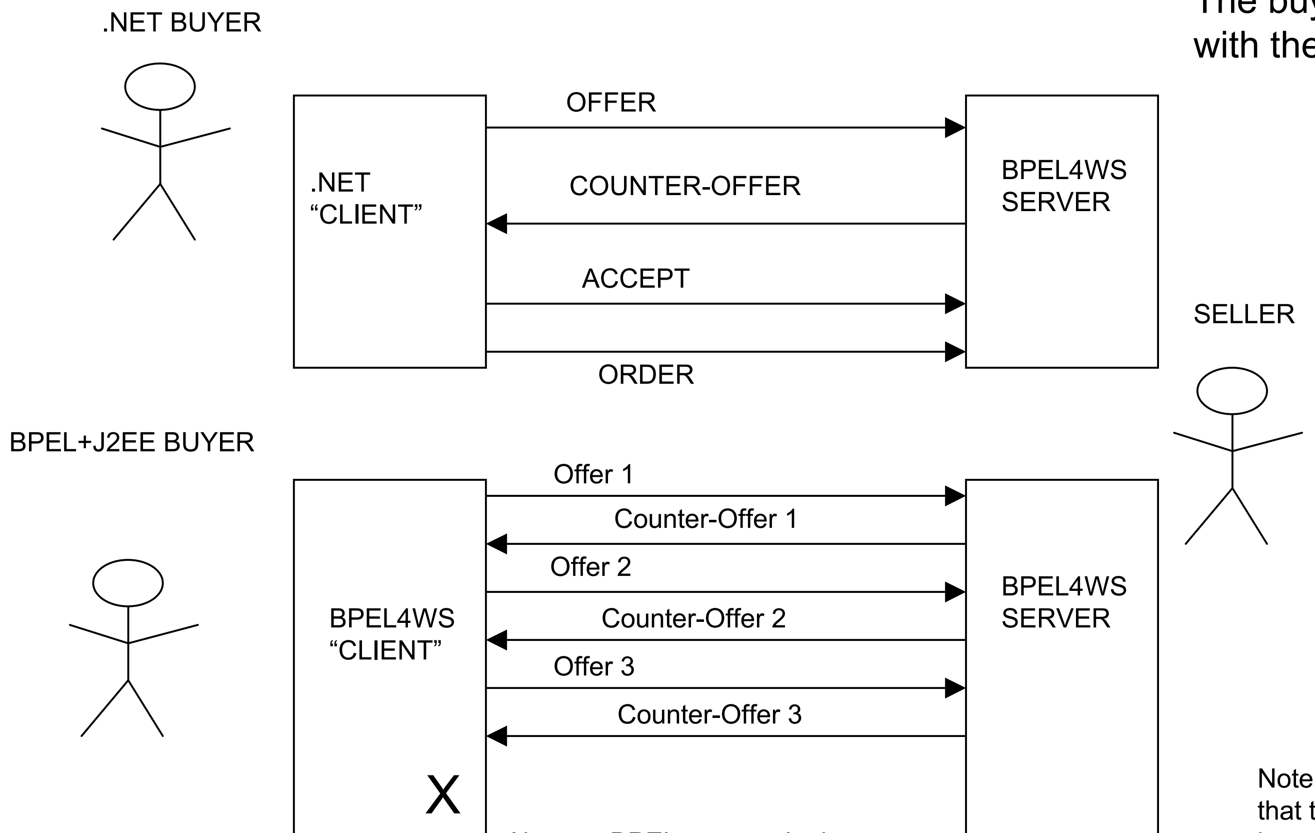
Buyer One adds additional logic to abstract BPEL process to create an executable BPEL process.
Executable BPEL process is ready to be executed by a BPEL4WS engine.



Buyer Two runs abstract BPEL process through a translator to get an intermediate form (i.e., a C# or IronPython skeleton).
Additional business logic is added. The final program is compiled to produce an executable.

Abstract Process Example Continued : Automated Negotiation

The buyers are now ready to interact with the seller.



Abstract BPEL process logic dictates that conversation should end after three unsuccessful rounds

Note: it does not matter what language the client was written in, as long as the implementation adheres to the rules laid out in the abstract process.

Note: An external observer cannot tell how Buyer One's and Buyer Two's executable were implemented: they are effectively black boxes. WSDL/Partner links define interfaces (explained later).

Note: An external observer will notice that the message exchanges are similar but not the same. The changing internal state of the client and server can alter the message exchange sequence (i.e., seller ran out of stock and need to re-order item at a higher cost…..