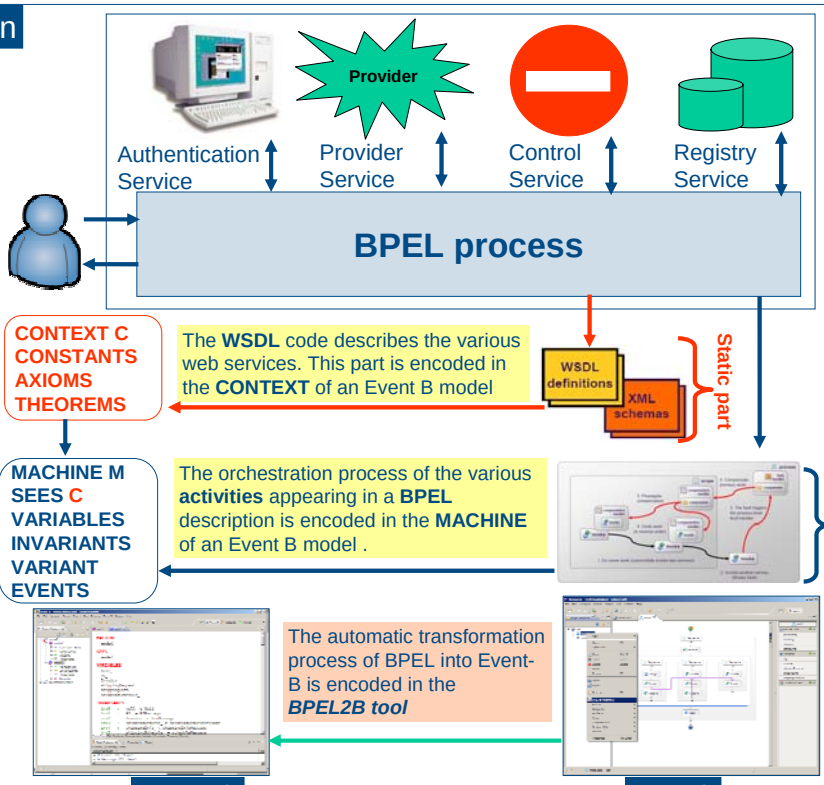
École Nationale Supérieure de Mécanique et d'Aérotechnique / Université de Poitiers
Laboratoire d'Informatique Scientifique et Industrielle (www.lisi.ensma.fr)
ENSMA - Téléport 2-1 Avenue Clément Ader - BP 40109 - 80961 FUTUROSCOPE CHASSENEUIL Cedex - France

Verification and Validation of Web Services Compositions Using the Event-B Method

Formal Semantics for Web Services Composition

One of the key ideas of the ***Service-Oriented Architecture*** (SOA) based on the web service technology is the ability to create **services compositions**.

Our work deals with the ***formal verification of the web services composition. => Formal validation of*** services composition using proof and refinement based techniques (**Event-B)**



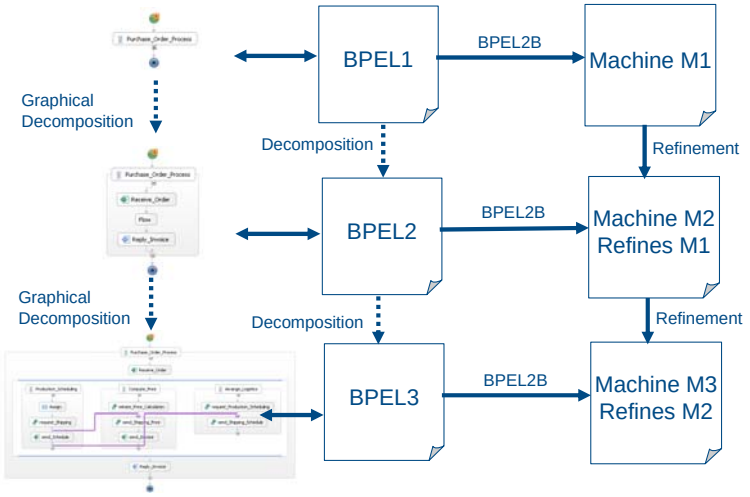
Interest of the Approach

From the expressiveness point of view, the proposed approach covers the *whole* **BPEL** specification. Both dynamic and static parts are formalized in a single and common formal method.
The **WSDL** associated to primitive processes and the processes composition operations are respectively encoded within a **CONTEXT** and a **MACHINE** parts of an **Event-B** model.

From the validation point of view, since the approach is based on theorem proving for establishing the properties, it does not suffer from the state explosion problem faced by model checking.

The Validation Methodology

From a methodological point of view, the validation of services composition is performed according to this scenario **=>** *Each decomposition of a complex activity in BPEL is translated into Event B by refining the event corresponding to this activity. This refinement introduces the decomposition defined in the original BPEL specification. A step by step BPEL description and validation is performed in parallel.*



Refinement allows the developer to express the relevant properties at the refinement level where they are expressible. Then, further refinements will preserve these properties avoiding reproving them again provided that a gluing invariant is produced in the refined model

Event-B code

```
INVARIANTS
inv1 : PO ∈ POMessage
inv2 : Invoice ∈ InvMessage
inv3 : shippingRequest ∈ ShippingRequestMessage
inv4 : shippingInfo ∈ ShippingInfoMessage
inv5 : shippingSchedule ∈ ScheduleMessage
...
THEOREMS
(( sequ1 = 2 ∧ flow2 = 1 ∧ sequ12 = 0) ∨
( sequ1 = 2 ∧ flow2 = 1 ∧ sequ12 = 3) ∨
( sequ1 = 2 ∧ flow2 = 1 ∧ sequ12 = 2 ∧ship_to_invoice = 1) ∨
( sequ1 = 2 ∧ flow2 = 1 ∧ sequ12 = 1 ∧
(∃ xx1 (xx1 ∈ dom(sendInvoice)))
...
EVENTS
// first event of the sequence
Initiate_Price_Calculation =
WHEN
...
grd3 : sequ12 = 3
THEN
act1 : void := initiatePriceCalculation(PO)
act2 : sequ12 := sequ12 - 1
END
// second event of the sequence
send_Shipping_Price =
WHEN
...
grd3 : sequ12 = 2
grd4 : ship_to_invoice = 1
THEN
act1 : void := sendShippingPrice(shippingInfo)
act2 : sequ12 := sequ12 - 1
END
// third event of the sequence
send_Invoice =
ANY
receive
WHERE
grd1 : receive ∈ dom(sendInvoice)
...
grd4 : sequ12 = 1
THEN
act1 : Invoice := receive
act2 : sequ12 := sequ12 - 1
END
// the Compute Price event
Compute_Price =
WHEN
...
grd3 : sequ12 = 0
THEN
...
END
...
VARIANT
sequ12
...
```

BPEL code

```
<bpws:sequence name="Compute_Price">
    <bpws:invoke inputVariable="PO"
        name="initiate_Price_Calculation"
        operation="initiatePriceCalculation"
        partnerLink="invoicing"
        portType="lns:computePricePT">
        <bpws:documentation>
        Initial Price Calculation
        </bpws:documentation>
    </bpws:invoke>
    <bpws:invoke inputVariable="shippingInfo"
        name="send_Shipping_Price"
        operation="sendShippingPrice"
        partnerLink="invoicing"
        portType="lns:computePricePT">
        <bpws:documentation>
        Complete Price Calculation
        </bpws:documentation>
        <bpws:targets>
            <bpws:target linkName="ship-to-invoice"/>
        </bpws:targets>
    </bpws:invoke>
    <bpws:receive name="send_Invoice"
        operation="sendInvoice"
        partnerLink="invoicing"
        portType="lns:invoiceCallbackPT"
        variable="Invoice"/>
</bpws:sequence>
```

Checked Properties

Properties are defined in the **INVARIANTS** and **THEOREMS** clauses. Preconditions and guards are added to define the correct event triggering order

No livelock. A decreasing **VARIANT** (***sequ12***)is introduced.

No deadlock. This property is ensured (in the **THEOREMS** clause) by the disjunction of all events guards.

Typing of messages exchanged between partners during the triggering of the different activities through the invariant of typing expressed in the **INVARIANTS** clause

Verification of properties related to the functionality of the web service using refinement

Participants
•Yamine AIT-AMEUR
•Idir AIT-SADOUNE – Doctorant - MESR