A Transformation and Verification Framework for BPEL

BODEVEIX Jean-Paul, FARES Elie, FILALI Mamoun

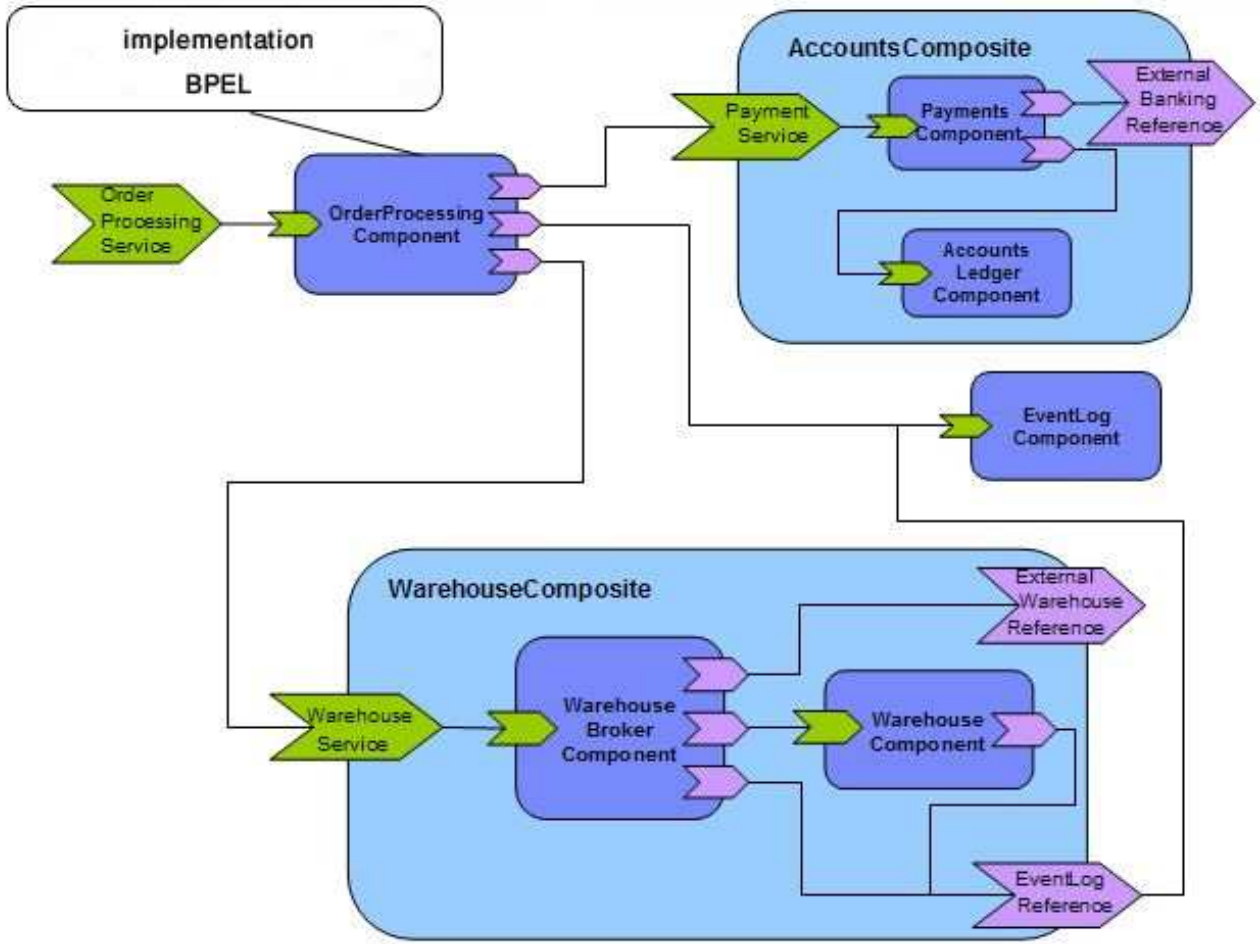
Journée Services, Lille, 10th June 2011

Table of contents

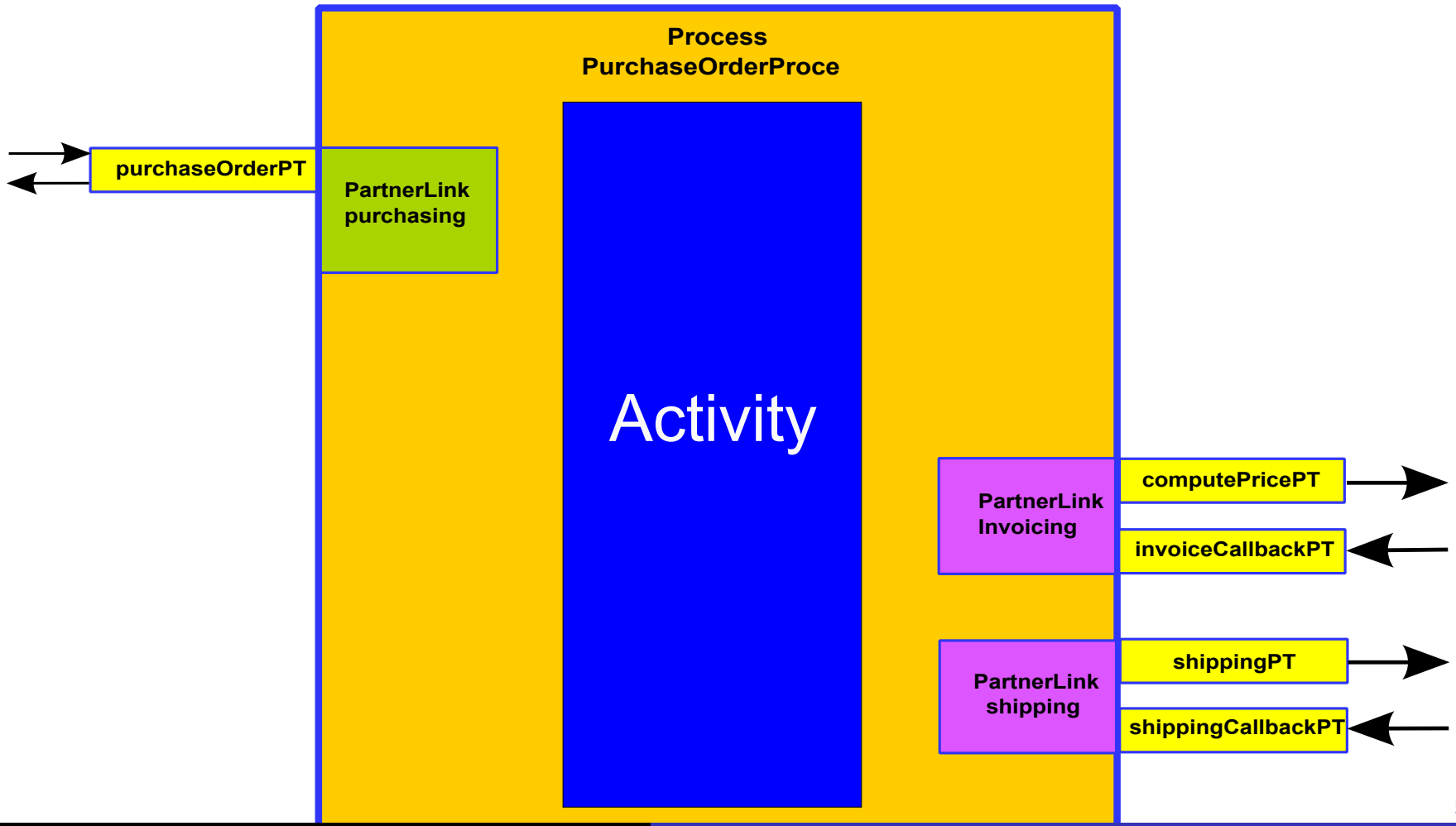
BPEL and SCA assembly

SCA : Service Component Architecture.

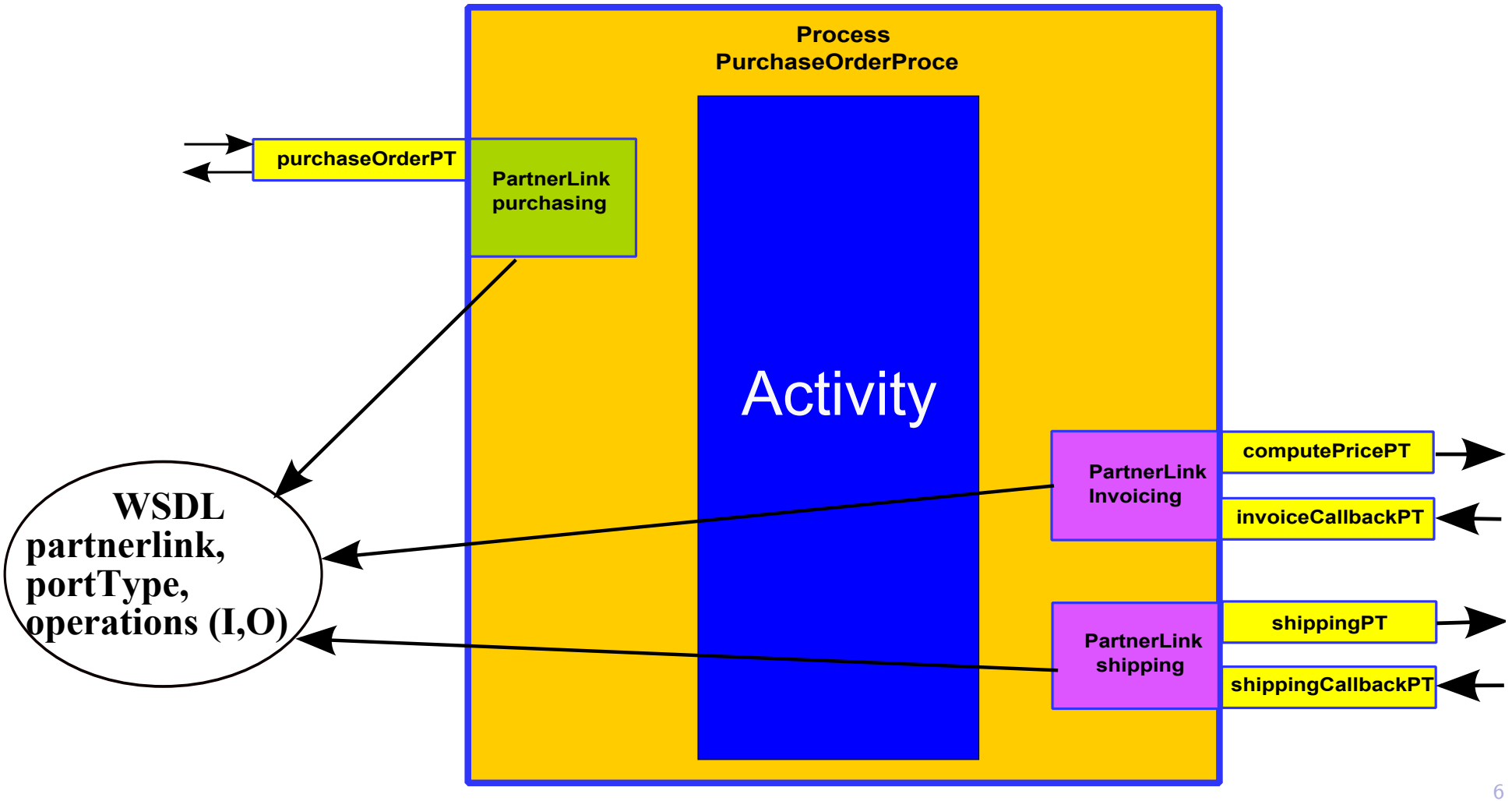
BPEL : Web Services composition language.



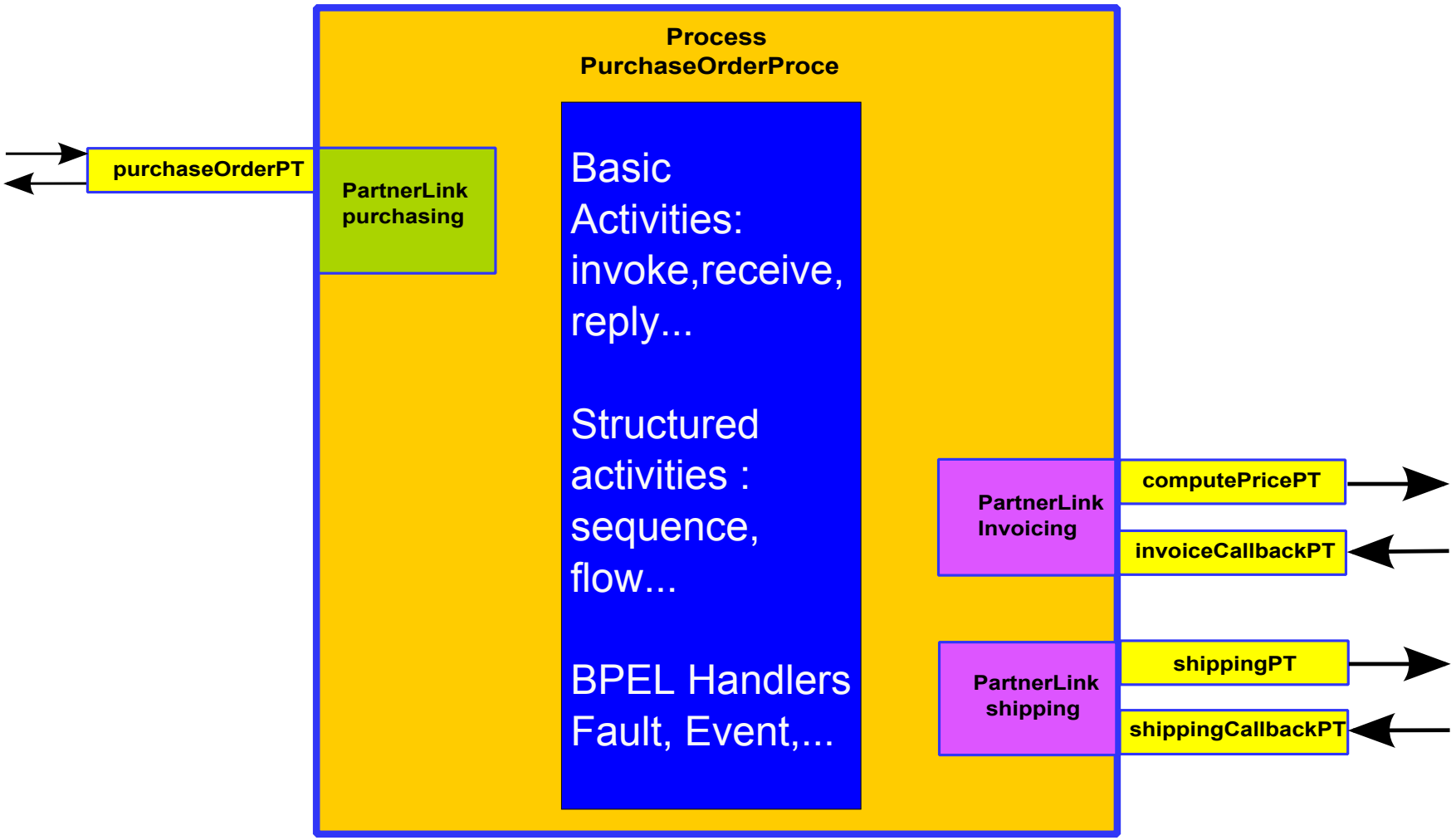
BPEL Process Architecture : PurchaseOrder Example



BPEL Static Part : WSDL



BPEL Dynamic Part : Activities and Handlers



Our Goal

Goal :

1. Verify in advance requirements on the BPEL code.
2. $\rightsquigarrow$ Need for formal semantics.
3. $\rightsquigarrow$ Formal Methods : Model checkers , assistant theorem provers . . .

Example of BPEL requirements :

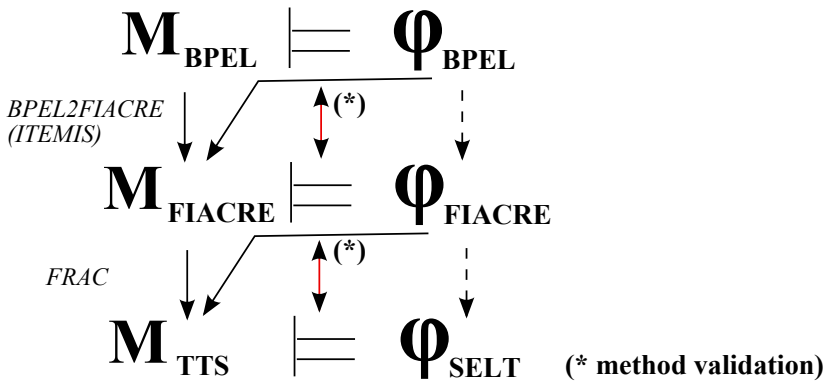
1. Structural Properties : Well-formedness of the BPEL code.
2. Temporal properties : absence of deadlock . . .
3. Timed properties : Response Time of the process . . .

Model checking

- Verify if a model of the system M satisfies a specification φ : $M \models \varphi$.
  1. M : Hardware or software systems.
  2. φ : Specification are logical properties written in logic (temporal).
- Here,
  1. M is the BPEL model,
  2. φ are the BPEL requirements expressed as temporal properties.

How to?

1. Transformation from M_{BPEL} to M_{FIACRE}.
2. Specification of property patterns and properties (φ_{LTL}, φ_{MITL} ...) .
3. Verification of properties using the model checker TINA .



FIACRE

FIACRE : Real Time verification language.

1. **Process :** Sequential behavior.
2. **Component :** Parallel composition of subcomponents.
3. **Data types :** High level data (arrays, queues . . .).
4. **Communication** : Shared variables and timed ports.

FIACRE Process

```
process P [ p: none, q: none ] is
states s0,s1
    init to s0
    from s0 p ; to s1
    from s1 q ; to s0
```

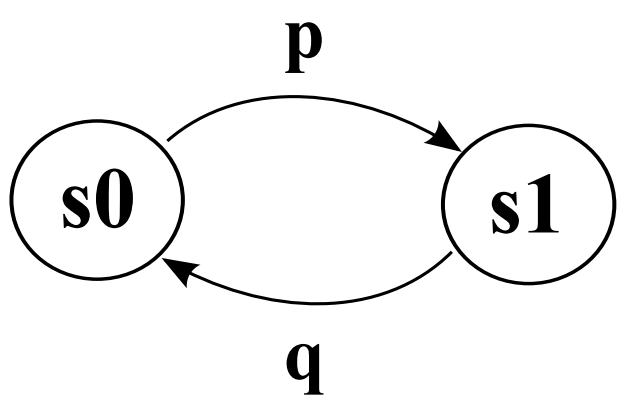


Figure: Processus P

FIACRE Component

Example : Associate temporal constaints.

```
component main is
    port p : none in ]2,4],
         q1 : none,
         q2 : none
    par
         p ⟶ P [p,q1]
      || p ⟶ P [p,q2]
    end
```

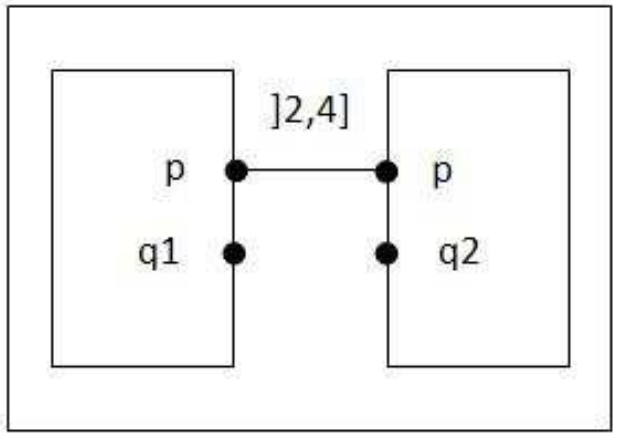
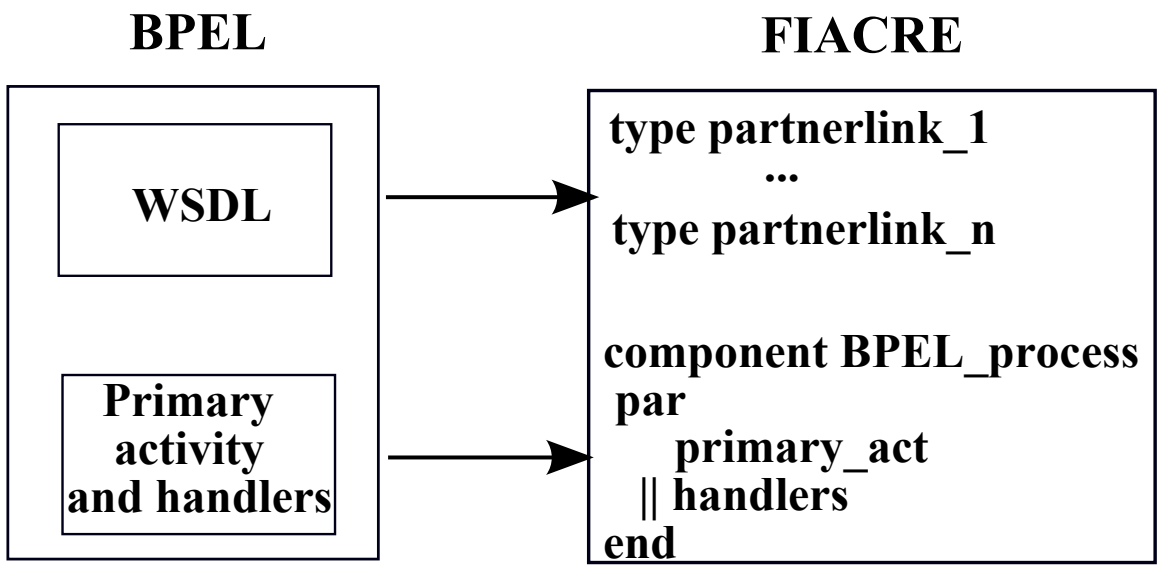


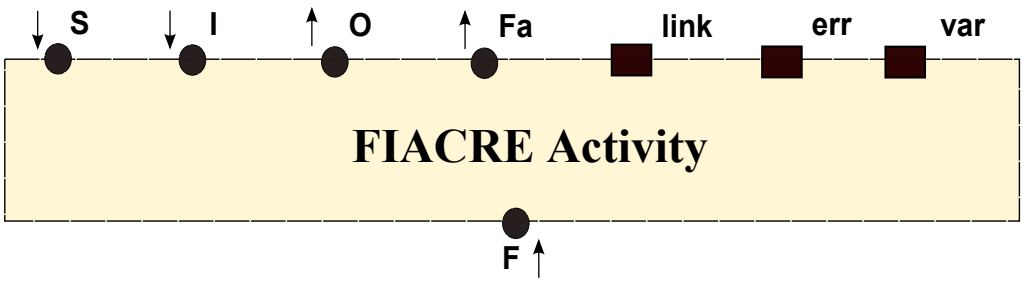
Figure: Composition

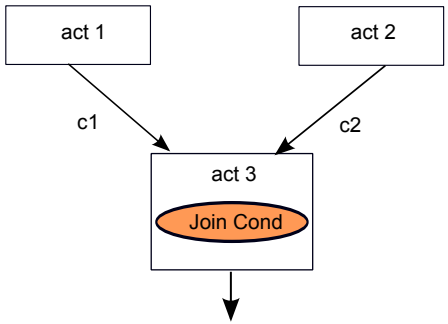
Overview of the transformation



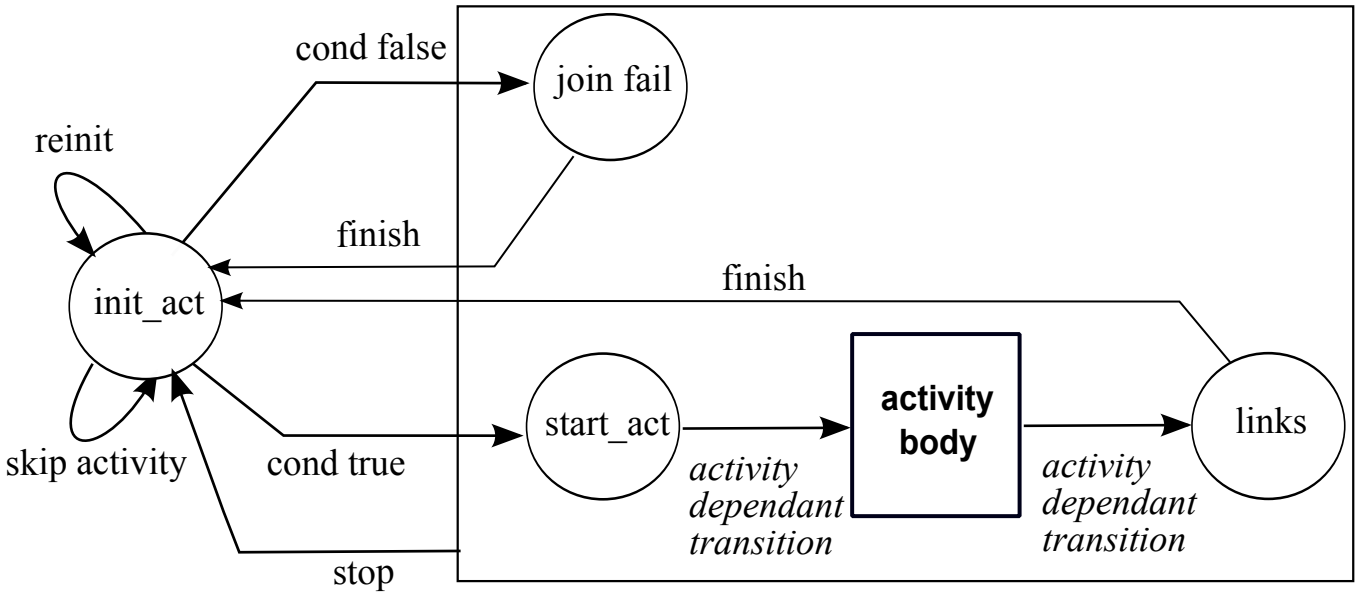
Transformation Principles

- Each BPEL construct corresponds to a FIACRE component.
  - Use of ports (•) and shared variables (■) for synchronization.
- Data abstraction :
  - Boolean are preserved.
  - A constant is created for each other type.

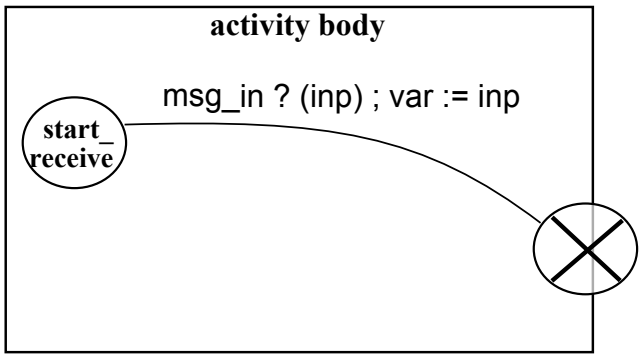




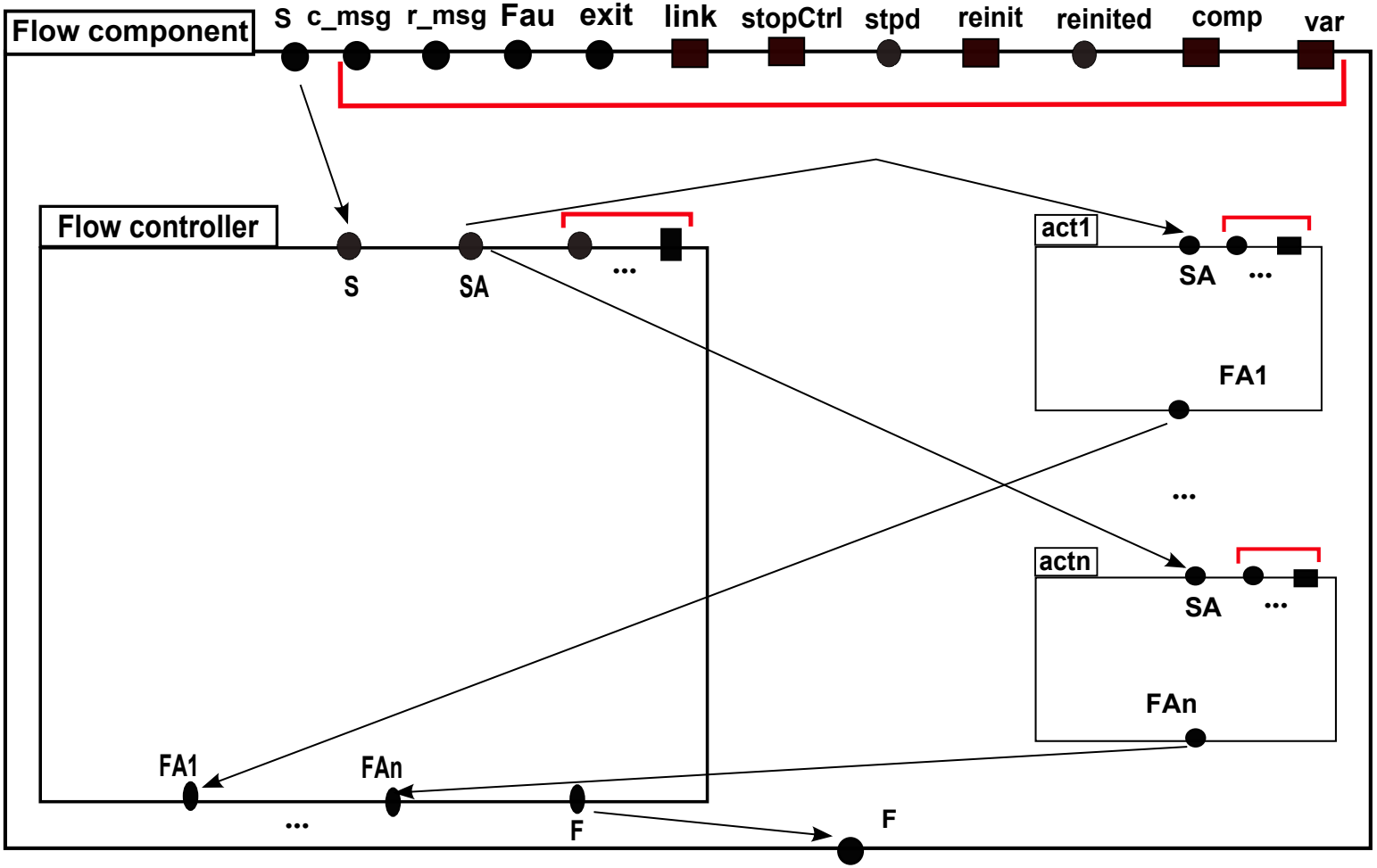
Structure of the generated code (simplified)



Receive Pattern

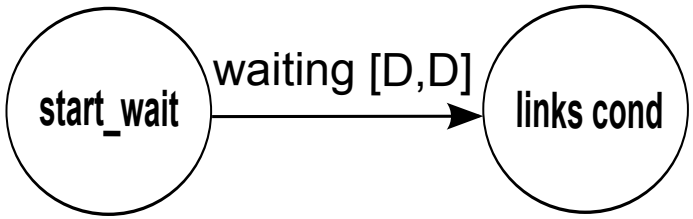


Flow Pattern



Relative and Absolute Time

- Relative Time : For duration D $\rightarrow$ *waiting*$[D, D]$

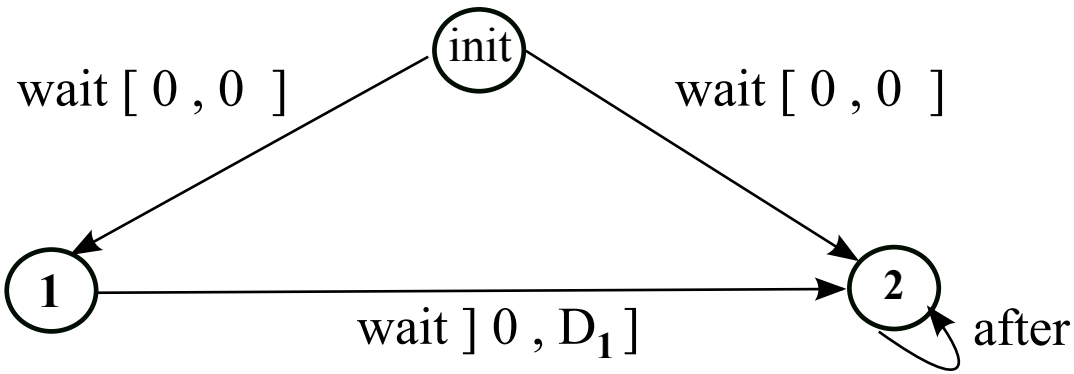


- Absolute Time : Until Deadline
  1. Interesting to verify the model independently of the current date.
  2. Ex : A service is offered *until "June 29, 2011"* but the current date is unknown.

One Absolute Date

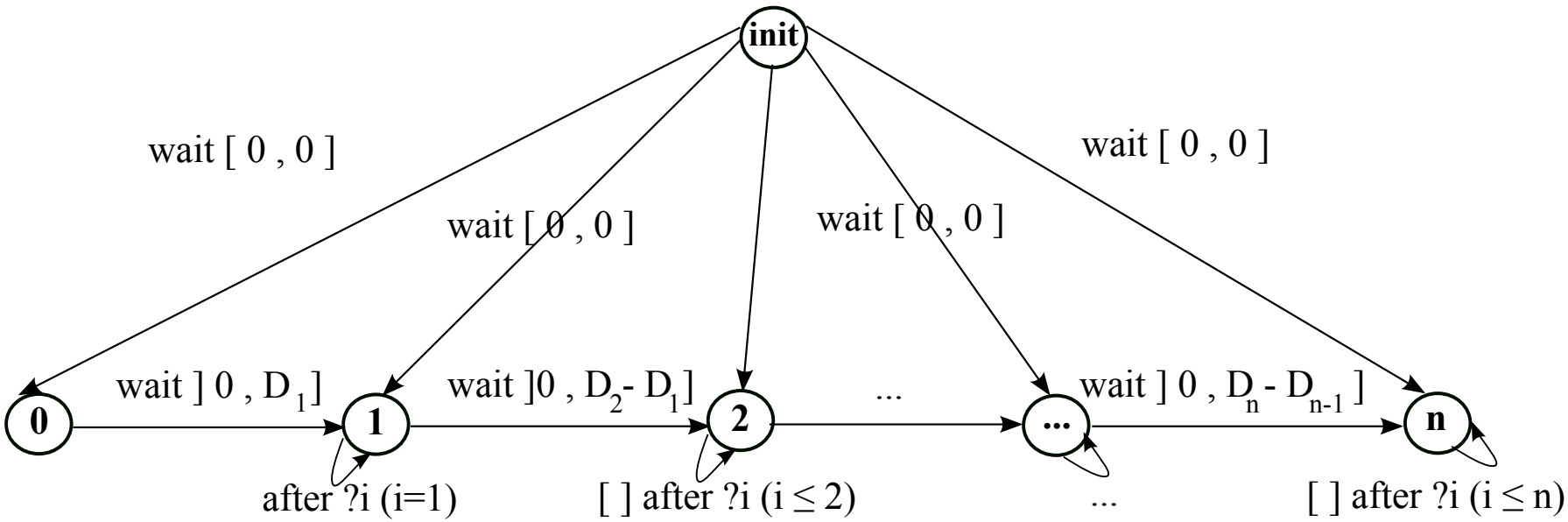
Modeling of Until D_1 :

- starting date could be **any** date $\rightsquigarrow$ Model a non deterministic delay between 0 and D_1.
- Enable a synchronization on `after` once the date is reached.

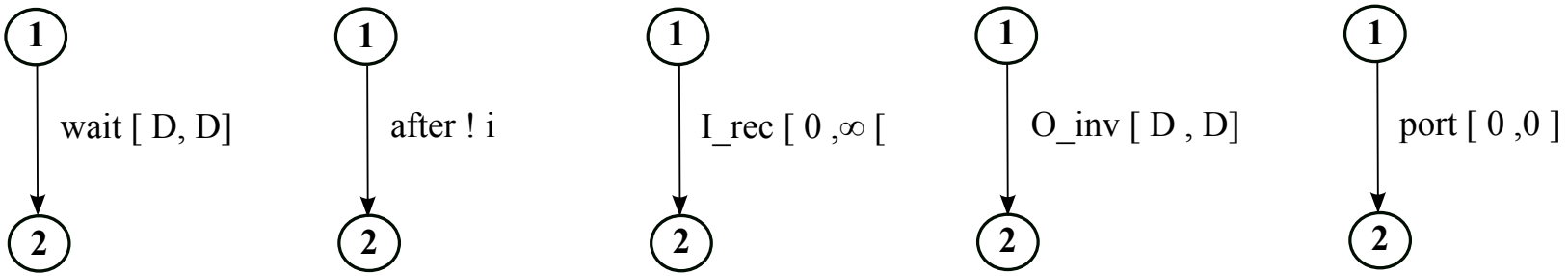


Multiple Absolute Dates

- Sort chronologically the dates of the system $D_1 < \ldots < D_n$.
- Allow a synchronization depending on the reached date.



Other Timed Related Constructs



(a) Wait For D on alarm For D

(b) Wait Until D_i on alarm Until D_i

(c) receive on message

(d) synchronous invoke annotated with delay D

(e) other BPEL constructs Fiacre control events

Supported Properties

1. Structural Properties : express the "well-formedness" of the BPEL code.
   1. A receive is always matched with a reply.
   2. Two concurrent receive cannot wait for the same operation.
2. Temporal properties : LTL properites as the safety, liveness and response properties.
3. Timed properties : MITL properties are not supported by TINA $\rightsquigarrow$ Timed Observers.

Verification : Bounded Response Property

Timed properties : Not supported by TINA $\rightsquigarrow$ Timed Observers.

- $\Box(receive \Rightarrow \Diamond_{\leq T} finish)$:
  - The elapse of time between the initial receive and the end of the process is bounded by a fixed time T.

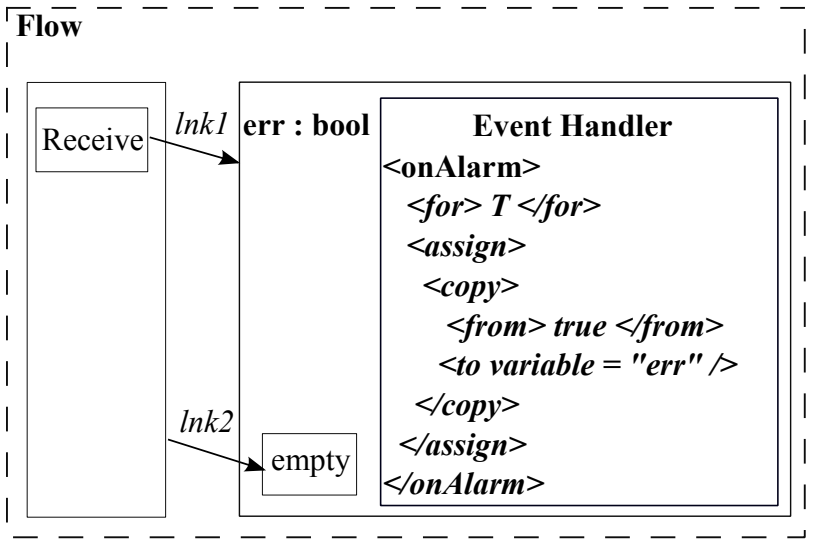
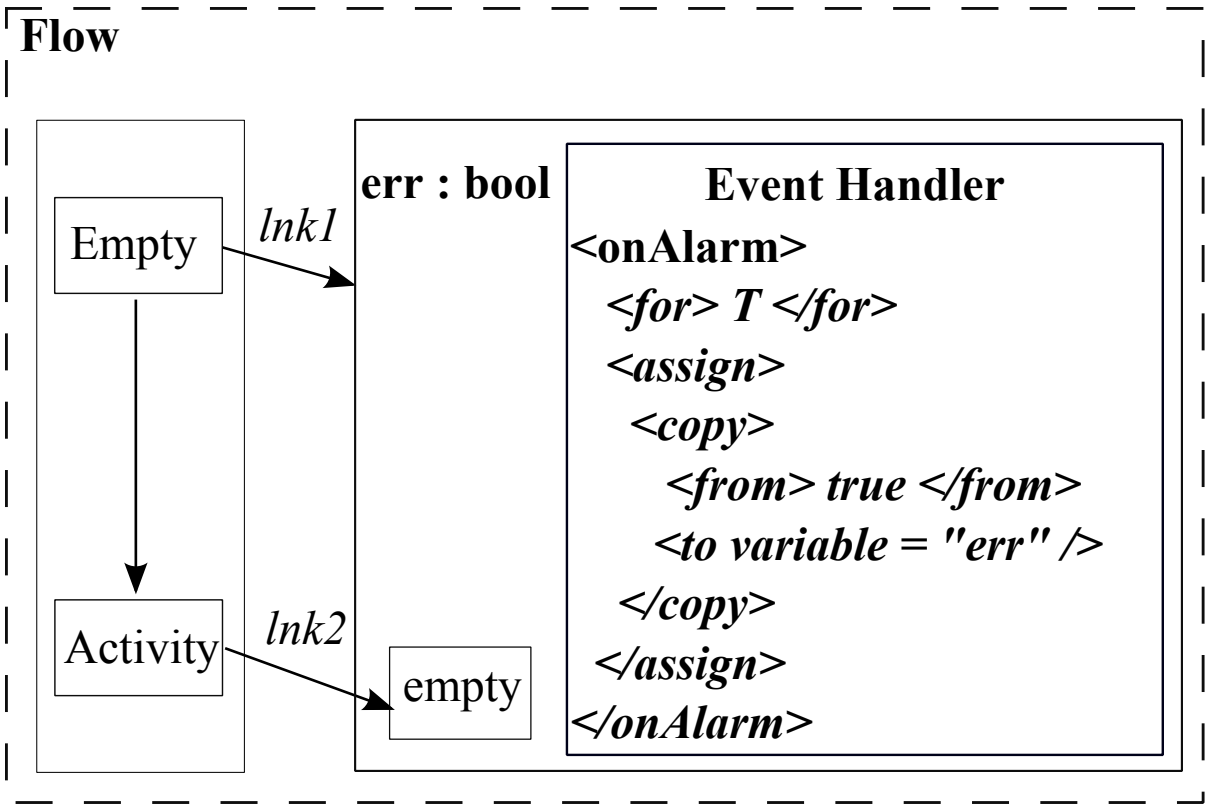


Figure: Timed Observer

Verification : Duration of an activity



Camlp4

- Camlp**4** : **P**re-**P**rocessor-**P**retty-**P**rinter for Objective Caml.
- Introduce new syntactic categories (FIACRE syntax) into OCaml.
- Manipulation of abstract syntax using source code (concrete syntax).
- Use of Camlp4 :

```
let f = <:fiacre< type t_err is union no | exit | other end  ... >>
```

The corresponding generated Ocaml code in Abstract Syntax.
Use of Abstract syntax :

```
let f = [ Fiacre.TypeDecl ("t_err", Fiacre.Union
[ ("no", None) ; ("exit", None) ; ("other", None) ]) ]
```

Example

```
let flow a lc =
<:decls<
 $decl:flow_controller a (List.map (fun c → c.num) lc)$

component $a.name$
    [startflow: none,finishflow: none,c_msg: inputs,r_msg: outputs,f: fault,
stpd :none,reinited : none]
    ( &info: read write t_info,&err_var: read write t_scope_err,
      &rec_var: read write t_var_record,& reinit: read bool) is
  port start:none, {c:lc}. {%finish_ $string_of_int c.num$%}:none
  par * in
      {%flow_controller_ $a.name$ %}
        [startflow, finishflow, start,
           {c:lc}. {%finish_ $string_of_int c.num$ %}, stpd, reinited]
        ( &info,&err_var,& reinit)
  || ||_{c:lc}. $c.name$
        [start,{%finish_ $string_of_int c.num$ %},
         c_msg,r_msg,f,stpd,reinited]
        ( &info,&err_var,&rec_var,& reinit)
  end
```

Results

Results :

- Transformation and verification framework for BPEL 2.0.
- Covering all BPEL 2.0 constructs.
- Translation and analysis of a rich set of timed properties.
- Ease of use, readability and clarity of the proposed patterns thanks to Ocaml/Camlp4.

Future Works :

- Definition of a property pattern language.
- Integrate the abstraction/refinement notion in the transformation.

Questions

Thank you

Questions ?