Formal Development of Choreographic Business Processes

Journée "Méthodes formelles et processus métier", Paris, 16 octobre 2014

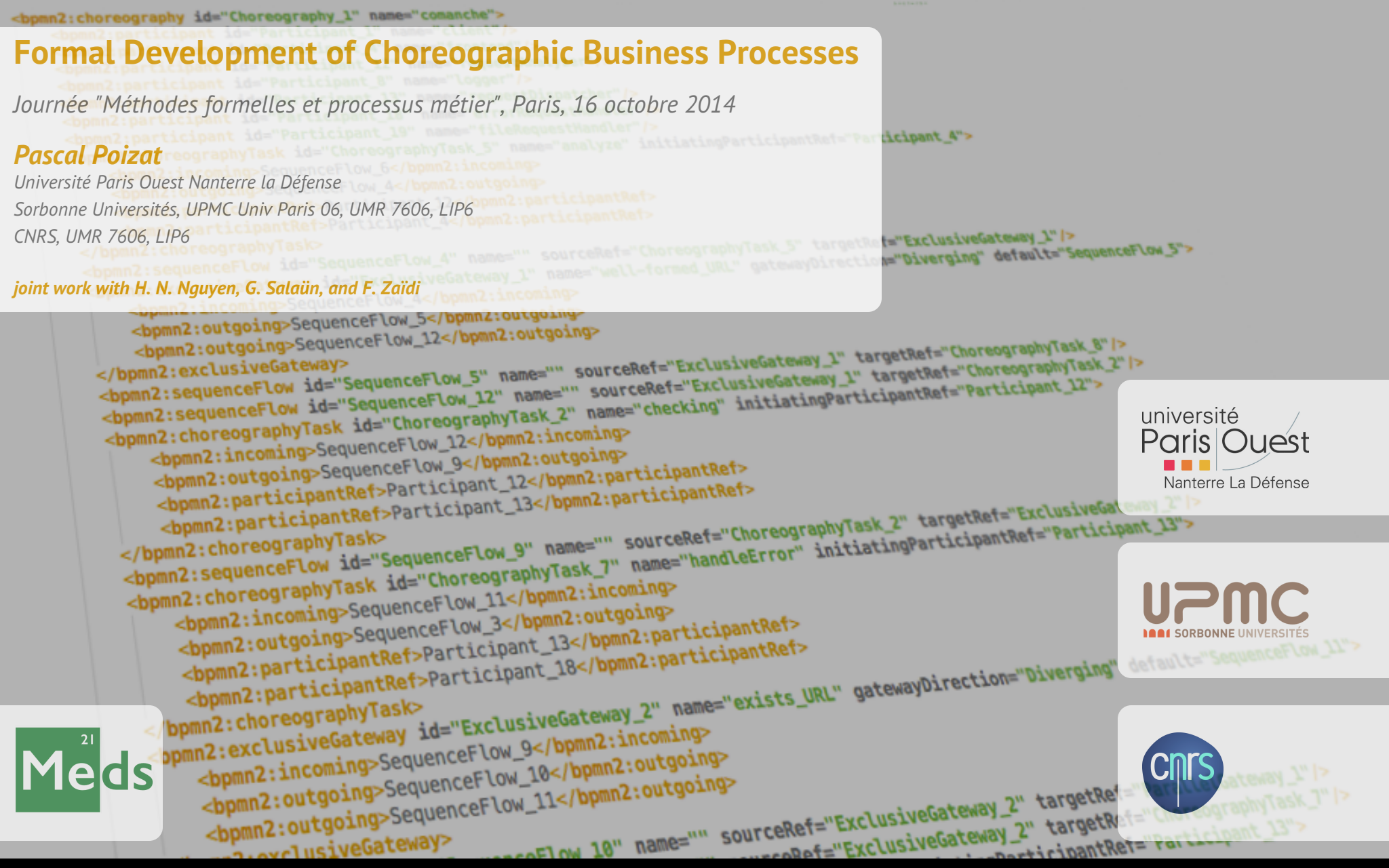
Pascal Poizat

Université Paris Ouest Nanterre la Défense

Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6

CNRS, UMR 7606, LIP6

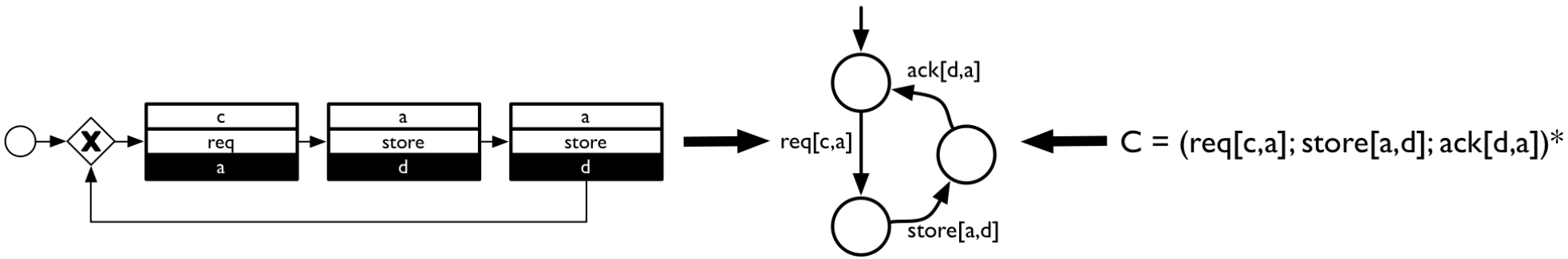
joint work with H. N. Nguyen, G. Salaün, and F. Zaïdi



A Short Comment on (Formal) Models

- In this talk, formal models are Transition Systems (LTS, STG)
  in these models, ! stands for emission and ? stands for reception

- why? mainly because they have a good tool support
  e.g., model-checkers and equivalence-checkers in CADP
  but also because they support numerous model transformations
  e.g., from Business Process languages and notations (WS-BPEL, BPMN)



>Diving into

Choreographies

The Shop-n-Deliver Collaboration

- three participants : buyer, vendor, and shipper
- → A request for goods first takes place.
- → If the goods are available, a shipping request
  → and then a delivery are achieved.
- → Else, a cancellation notification is sent (vendor to buyer).

Local Views over Shop-n-Deliver

- orchestration models
- centered on a specific role
  buyer or vendor or shipper
- the view is local to the role of interest
- captures conversations
  between the role and its partners

- e.g., WS-BPEL code

```
buyer = invoke(vendor, request);
        pick {
          onMessage(vendor, abort):
          onMessage(shipper, deliver):
        }
```

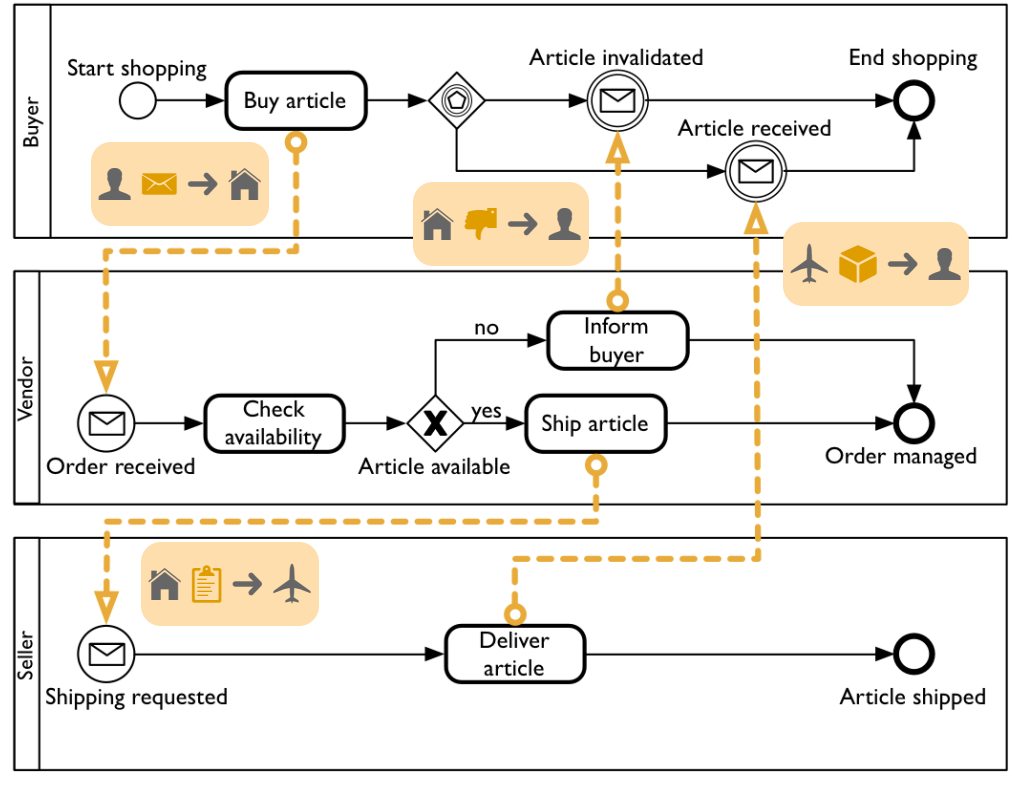
```
vendor = receive(buyer, request);
         if (article is not available) {
           invoke(buyer, abort);
         } else {
           invoke(shipper, ship);
         }
```

```
shipper = receive(vendor, ship);
          invoke(buyer, deliver);
```

Global View over Shop-n-Deliver

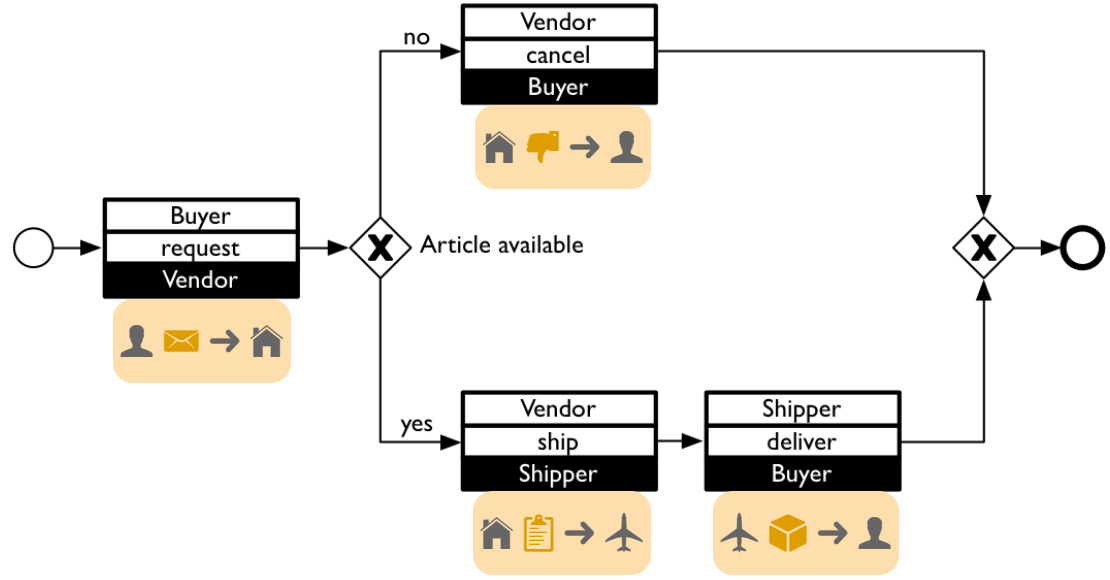
- choreography interconnection model
- takes into account all roles
  buyer and vendor and shipper
- the view is global
  with internal details of the roles
- captures conversations
  between all roles
  interaction is not central

- e.g., BPMN collaboration diagram



Global View over Shop-n-Deliver

- choreography interaction model
- e.g., BPMN choreography diagram
  since BPMN 2.0
- takes into account all roles
  buyer and vendor and shipper
- the view is global
  without internal details of the roles
- captures conversations
  between all roles
  interaction is central



Conclusion on Global vs. Local Views

- global view: interactions matter, good for specification

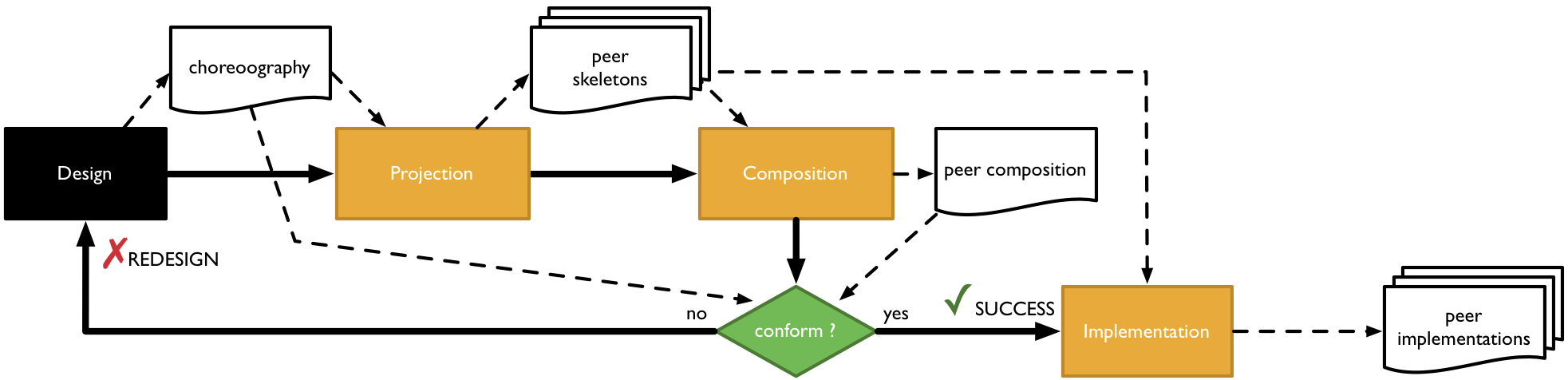
  A request for goods first takes place (buyer to vendor).
  If the goods are available, a shipping request (vendor to shipper) and then a delivery (shipper to buyer) are achieved. Else, a cancellation notification is sent (vendor to buyer).

- local view: peers matter, good for implementation

  The buyer issues requests about goods and is notified about cancellation or gets the goods.
  The vendor checks if the goods are available and sends cancellation notification or requests shipping.
  The shipper ships goods upon request.

↓ Top-Down Development

- A form of generative development
  1. *interaction skeletons are ⚙ generated using projection from the choreography*
  2. *conformance of the skeletons (composition) to the choreography is ⚙ verified*
  3. *interaction skeletons are 👤 implemented*

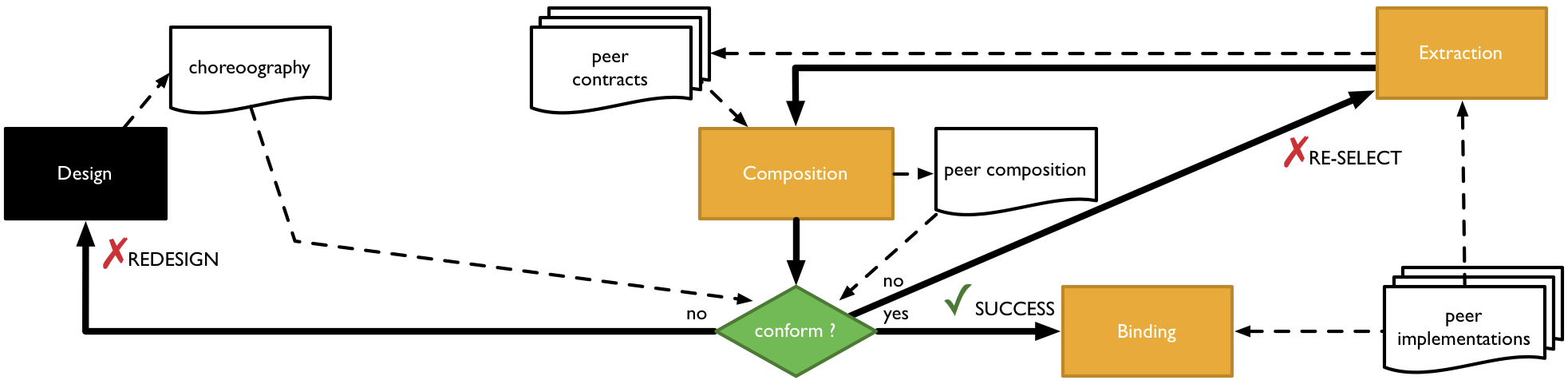


↑ Bottom-Up Development

- A form of development guided by reuse

  1. local contracts are ⚙ extracted from reused (design time) or dynamically bound (runtime) peers

  2. conformance of the contracts (composition) to the choreography is ⚙ verified



>VerChor

[Poizat and Salaün, SAC'12] [Güdemann et al, FASE'13]

VerChor (Python) is available at VerChor Framework. VerChor (Java) is ongoing work on GitHub.

Synchronizability (↓↑)

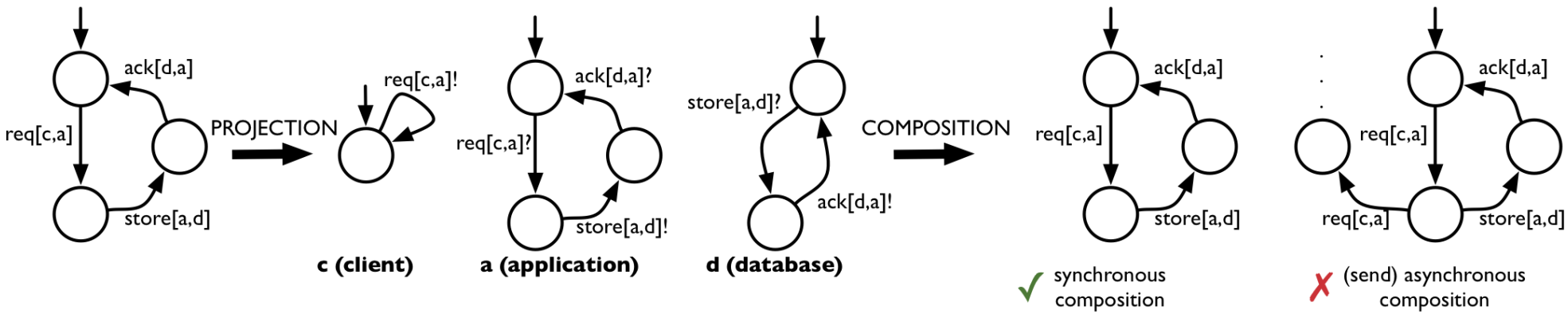
objective: avoid state-space explosion due to asynchronous communication

- synchronizable?(P1, ..., Pn) ⇔ synch(P1, ..., Pn) ≡ asynch(P1, ..., Pn)

  where synch is the synchronous product and asynch is the asynchronous product (buffers)

  takes into account the emission time in the asynchronous product
- synchronizable?(P1, ..., Pn) ⇔ synch(P1, ..., Pn) ≡ asynch-1(P1, ..., Pn)

  [Basu et al, POPL'12], where asynch-1 is the 1-bounded asynchronous product
- synchronizable?(C) ⇔ synchronizable?(C↓1, ..., C↓n)



Realizability (↓) & Conformance (↑)

objective: check if peers can be implemented by (natural) projection or be reused

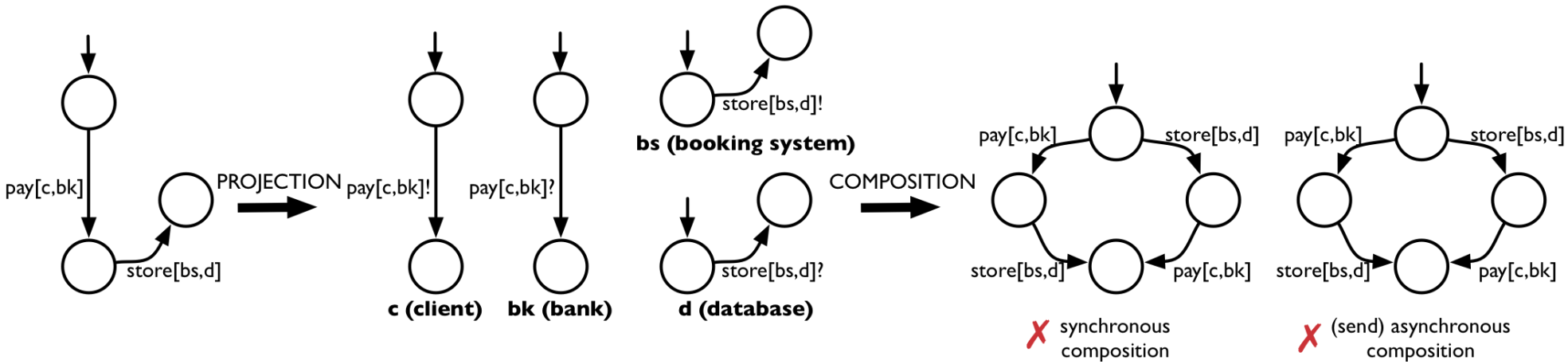
- realizable?(C) ⇔ C ≡ Π(C↓1, ..., C↓n)

  where Π is an operation that yields the semantics of the peers collaboration

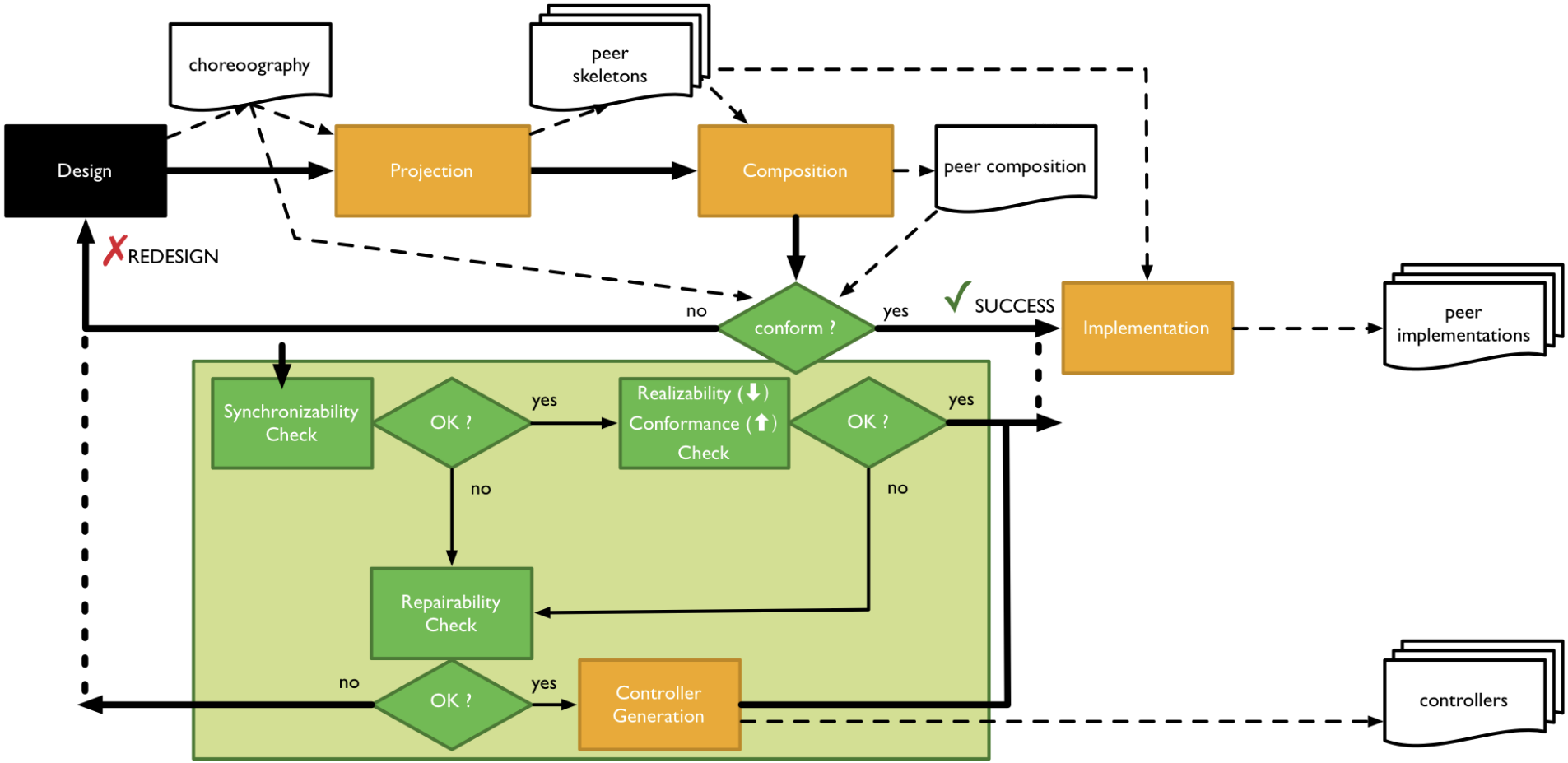
- realizable?(C) ⇔ synchronizable?(C↓1, ..., C↓n) ∧ C ≡ synch(C↓1, ..., C↓n)

  [Basu et al, POPL'12]

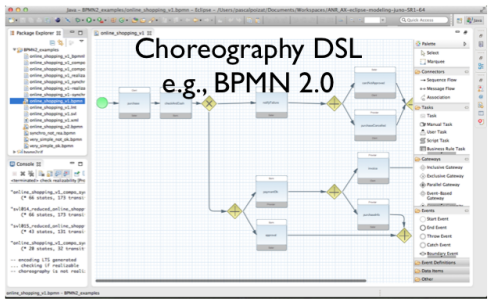
- conform?(C, P1, ..., Pn) ⇔ synchronizable?(P1, ..., Pn) ∧ C ≡ synch(P1, ..., Pn)



Choreography Development in VerChor



From Formal Framework to Formal IDE



Choreography Design Environment

model transformation

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<choreography xmlns="http://convecs.inria.fr">
    <choreoID>online_shopping_v1</choreoID>
    <participants>
        <peer>
            <peerID>Client</peerID>
        </peer>
        <peer>
            <peerID>Seller</peerID>
        </peer>
        <peer>
            <peerID>Bank</peerID>
        </peer>
        <peer>
            <peerID>Provider</peerID>
        </peer>
    </participants>
    <alphabet>
        <message>
            <msgID>ChoreographyTask_10</msgID>
            <sender>Seller</sender>
            <receiver>Client</receiver>
            <messageContent>totalValue</messageContent>
        </message>
```

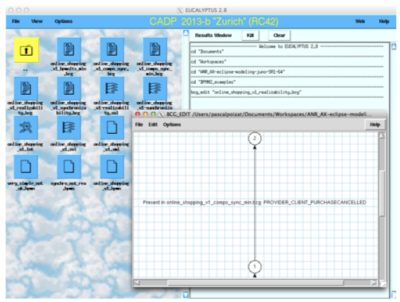
Intermediary Format (CIF)

script generation

```
module online_shopping_v1 with "get" is

% CAESAR_OPEN_OPTIONS="-silent -warning"
% CAESAR_OPTIONS="-more cat"

% DEFAULT_PROCESS_FILE=online_shopping_v1.lnt

"online_shopping_v1_bpmnlts_min.bcg" = safety reduction of tau*.

"online_shopping_v1_compo_sync.bcg" = root leaf branching reduct
par Seller_Client_totalValue,Provider_Seller_purchaseInfo,Seller
 peer_Seller [Seller_Client_totalValue,Provider_Client_purchaseC
||
par Provider_Client_purchaseCancelled,Provider_Client_invoice in
 peer_Client [Seller_Client_totalValue,Provider_Client_purchaseC
||
par Bank_Provider_paymentOk in
 peer_Provider [Seller_Client_totalValue,Provider_Client_purchas
||
```

Formal Models (LNT) +Verification Scripts (SVL)

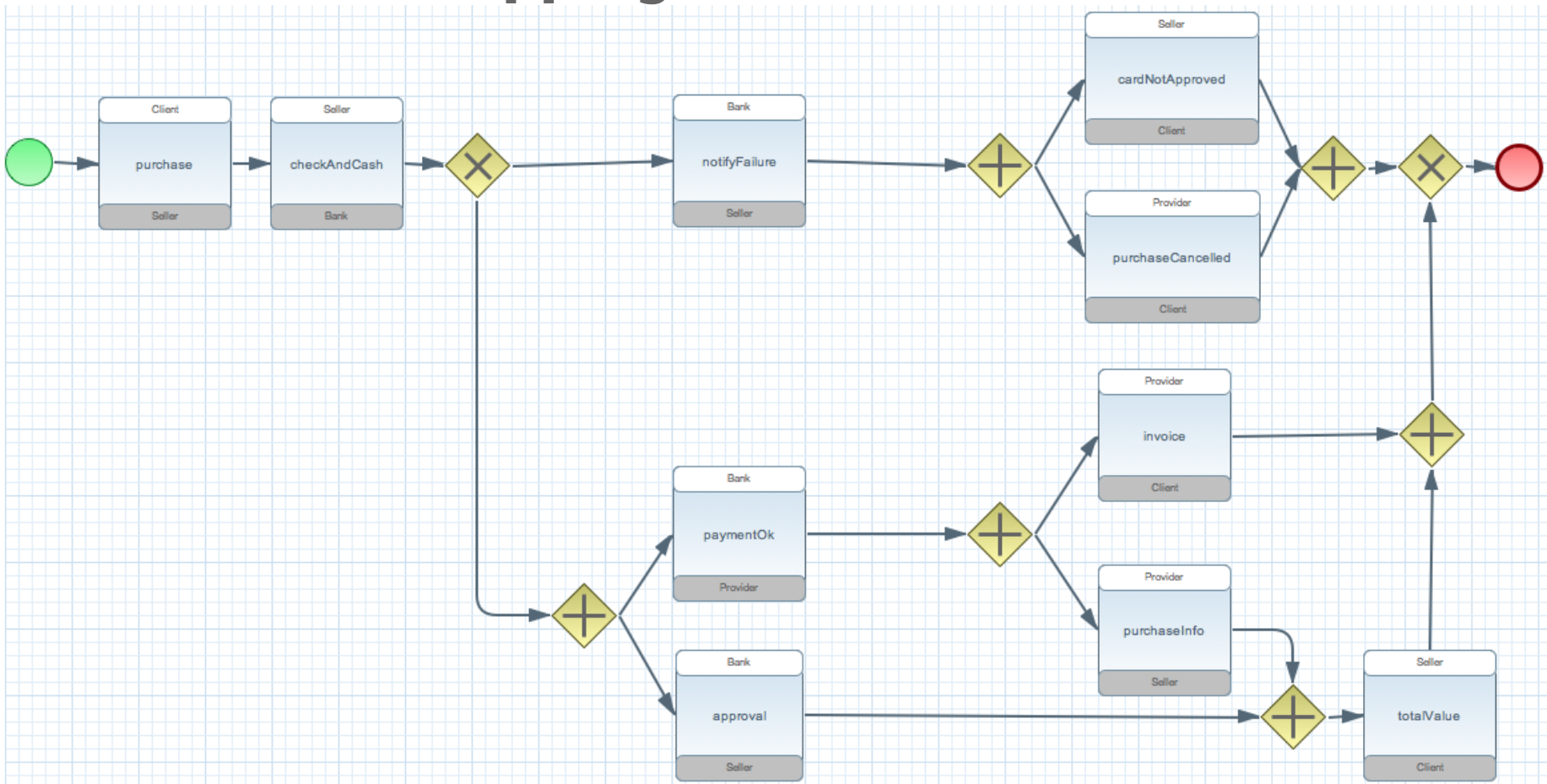
script execution



CADP toolbox

diagnostic

Online Shopping 1.0: BPMN



Choreography Intermediate Format (CIF)

- pivot model for choreography-related model transformation

  front-ends: BPMN Choreography Diagrams, Conversation Protocols, (ongoing: ChorD)

  back-ends: LNT, (ongoing: STG, Petri Nets)

- workflow constructs

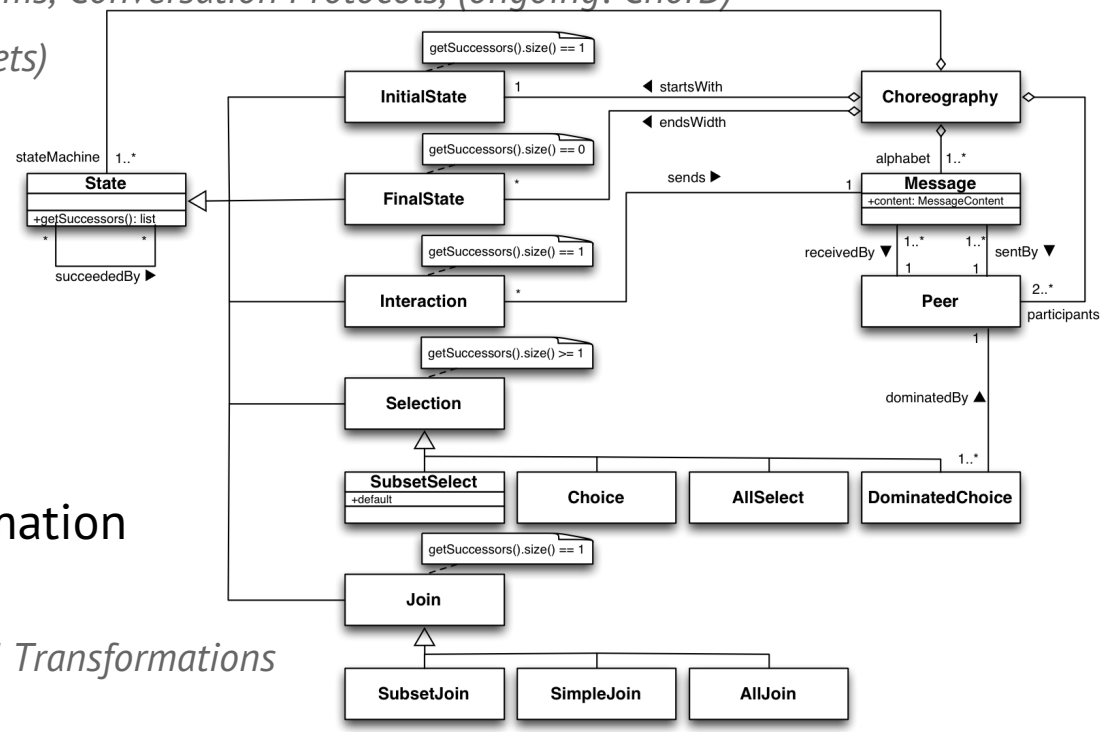
  inspired but simplified wrt. BPMN

  state-machine encoding

- BPMN → CIF model transformation

  plain old Java over EMF meta-models

  part of the framework for Formal Model Transformations

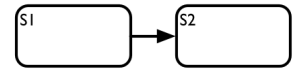


Online Shopping 1.0: CIF

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<choreography xmlns="http://convecs.inria.fr">
    <choreoID>online_shopping_v1</choreoID>
    <participants>
        <peer>
            <peerID>Client</peerID>
        </peer>
        <peer>
            <peerID>Seller</peerID>
        </peer>
        <peer>
            <peerID>Bank</peerID>
        </peer>
        <peer>
            <peerID>Provider</peerID>
        </peer>
    </participants>
    <alphabet>
        <message>
            <msgID>ChoreographyTask_10</msgID>
            <sender>Seller</sender>
            <receiver>Client</receiver>
            <messageContent>totalValue</messageContent>
        </message>
        <!-- 9 other messages -->
    </alphabet>
    <stateMachine>
        <initial>
            <stateID>StartEvent_1</stateID>
            <successors>ChoreographyTask_11</successors>
        </initial>
        <interaction>
            <stateID>ChoreographyTask_10</stateID>
            <successors>ParallelGateway_6</successors>
            <msgID>ChoreographyTask_10</msgID>
        </interaction>
        <!-- 9 other interactions -->
        <allJoin>
            <stateID>ParallelGateway_6</stateID>
            <successors>ExclusiveGateway_2</successors>
        </allJoin>
        <simpleJoin>
            <stateID>ExclusiveGateway_2</stateID>
            <successors>EndEvent_1</successors>
        </simpleJoin>
        <allJoin>
            <stateID>ParallelGateway_2</stateID>
            <successors>ExclusiveGateway_2</successors>
        </allJoin>
        <allSelect>
            <stateID>ParallelGateway_1</stateID>
            <successors>ChoreographyTask_4 ChoreographyTask_5</successors>
        </allSelect>
        <!-- 4 other gateways -->
        <final>
            <stateID>EndEvent_1</stateID>
        </final>
    </stateMachine>
</choreography>
```

CIF → LNT

```
process s1[...]
    s2[...]
end process
```



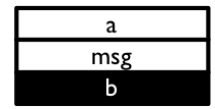
- state machine encoding

  LNT (LOTOS NT) process algebra

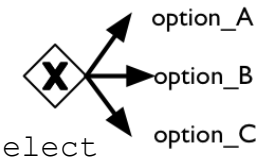
  use of patterns for CIF constructs

  complex for inclusive gateways

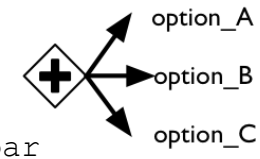
```
process s1[...]
    a_b_msg; s2[...]
end process
```



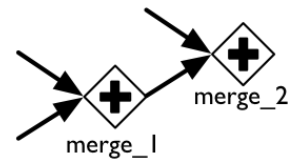

```
process s1[...]
    a_b_msg_REC; s2[...]
end process
```



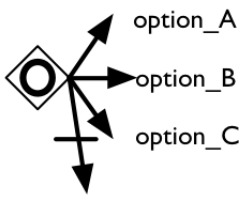

```
select
    option_A[...]
[]  option_B[...]
[]  option_C[...]
end select
```




```
par
    option_A[...]
||  option_B[...]
||  option_C[...]
end par
```




```
hide sync1, sync2 in
par
    sync1, sync2 -> option_A[..., sync1, sync2]
||  sync1, sync2 -> option_B[..., sync1, sync2]
||  sync2        -> option_C[..., sync2]
end par
```




```
select
    option_A[...]||((option_B[...][] null) || (option_C[...][] null))
[]  option_B[...]||((option_A[...][] null) || (option_C[...][] null))
[]  option_C[...]||((option_A[...][] null) || (option_B[...][] null))
[]  default[...]
end select
```

🛠 Verification Scripts (SVL)

- SVL scripts are generated to compute models (LTS)

  choreography model

  peer models if needed (projection using hiding + reduction)

  buffer models

  composition models (synch and asynch-1)

- SVL scripts are generated to check for properties

  synchronizability

  realizability

- using these "Makefile" formal verification scripts, all tasks are automated

  including counter-example extraction in case some property does not yield

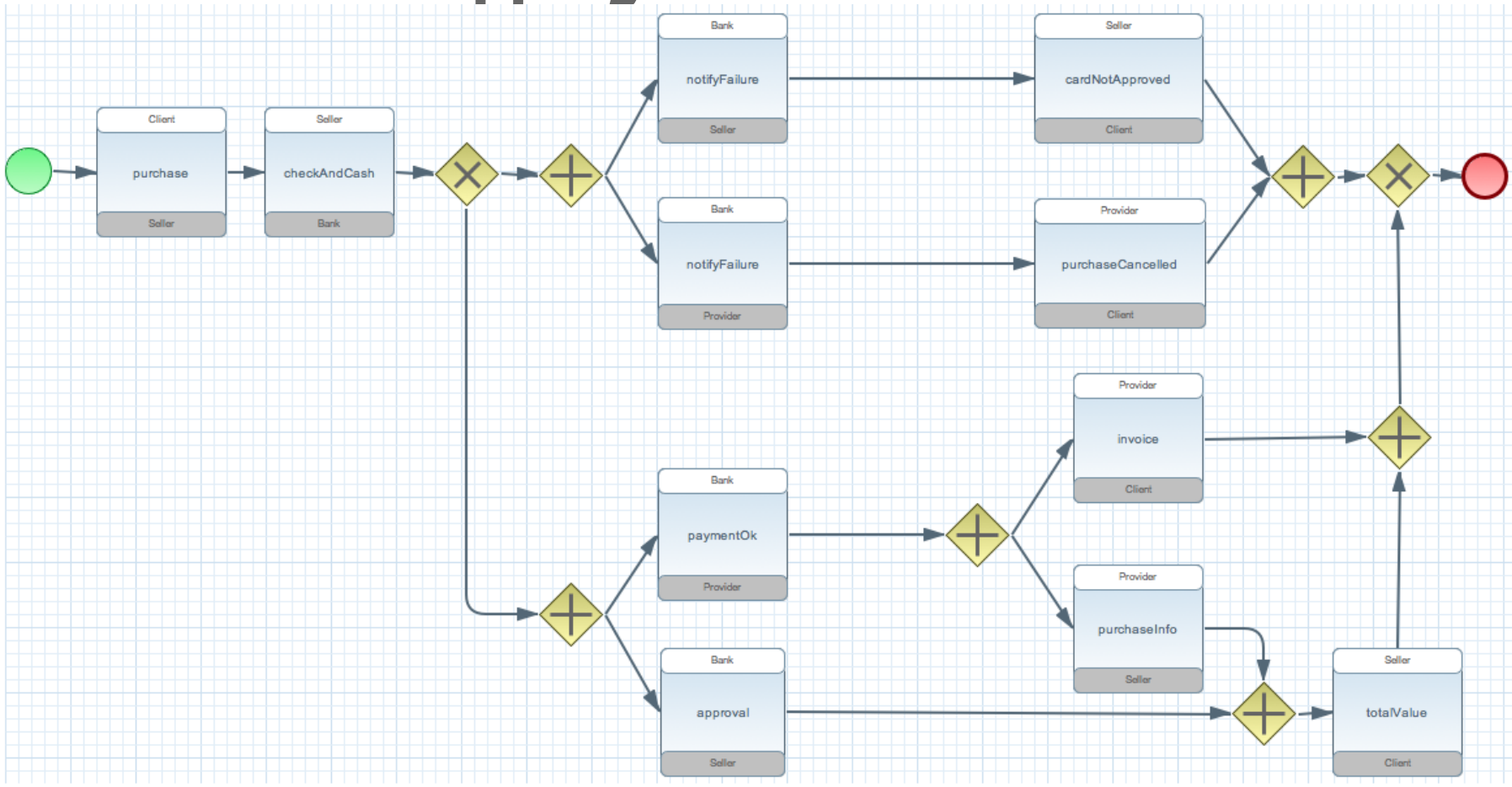
Online Shopping 1.0: LNT and SVL

```
module online_shopping_v1 with "get" is
// definition of types for messages and buffers
...

// definition of choreography workflow

process ChoreographyTask_10 [Seller_Client_totalValue:any,Provider_Client_purchaseCancel
Seller_Client_totalValue; synchro_ParallelGateway_6;null
end process

process ParallelGateway_4 [Seller_Client_totalValue:any,Provider_Client_purchaseCancelle
 par
synchro_ParallelGateway_5 -> ChoreographyTask_9 [Seller_Client_totalValue,Provider_Clier
 ||
synchro_ParallelGateway_6 -> ChoreographyTask
 end par
end process

process ParallelGateway_3 [Seller_Client_toto
 par
synchro_ParallelGateway_6,synchro_ParallelGat
 ||
synchro_ParallelGateway_5 -> ChoreographyTask
 end par
end process

process ParallelGateway_6 [Seller_Client_toto
synchro_ParallelGateway_6; ExclusiveGateway_2
end process

process split_ParallelGateway_3 [Seller_Clier
hide synchro_ParallelGateway_5:any,synchro_Pa
 par
synchro_ParallelGateway_6 -> ParallelGateway_
 ||
synchro_ParallelGateway_6,synchro_ParallelGat
 ||
synchro_ParallelGateway_6,synchro_ParallelGat
 end par
end hide
end process

 (* peers (synchronous communication) *)
```

```
resource is: /Users/pascalpoizat/Documents/Workspaces/ANR_AX-eclipse-modeling-junc
-- current file is: /Users/pascalpoizat/Documents/Workspaces/ANR_AX-eclipse-modeli
... loading choreography
-- choreography loaded
... generating encoding files (LNT, SVL)
-- encoding files generated
... cleaning up workspace
removing old bcg files
removing old diagnostics
-- workspace cleaned up
... generating encoding LTS (BCG) <this can be long>
-- encoding LTS generated
... checking if synchronizable
2
i
Present in online_shopping_v1_acompo_min.bcg: PROVIDER_CLIENT_PURCHASECANCELLED

-- choreography is not synchronizable
```

```
% CAESAR_OPEN_OPTIONS="-silent -warning"
% CAESAR_OPTIONS="-more cat"

% DEFAULT_PROCESS_FILE=online_shopping_v1.lnt

"online_shopping_v1_bpmnlts_min.bcg" = safety reduction of tau*.a reduction of branchi

"online_shopping_v1_compo_sync.bcg" = root leaf branching reduction of
par Seller_Client_totalValue,Provider_Seller_purchaseInfo,Seller_Bank_checkAndCash,Ban
 peer_Seller [Seller_Client_totalValue,Provider_Client_purchaseCancelled,Provider_Sell
||
par Provider_Client_purchaseCancelled,Provider_Client_invoice in
 peer_Client [Seller_Client_totalValue,Provider_Client_purchaseCancelled,Provider_Sell
||
par Bank_Provider_paymentOk in
                                                  Provider_Client_purchaseCancelled,Provider_Se
                                                  ider_Client_purchaseCancelled,Provider_Seller,

                                                  safety reduction of tau*.a reduction of branc
                                                  ler_purchaseInfo,Seller_Bank_checkAndCash,Ban
                                                  Provider_Seller_purchaseInfo_REC,Seller_Bank_
                                                  aseInfo,Bank_Seller_approval,Client_Seller_pu

                                                  vider_Client_invoice in
                                                  Provider_Client_purchaseCancelled_REC,Provide
                                                  lue,Provider_Client_purchaseCancelled,Provide

                                                  lled_REC,Provider_Seller_purchaseInfo_REC,Pro
                                                  entOk,Provider_Client_purchaseCancelled,Provid

                                                  Bank_Provider_paymentOk_REC,Bank_Seller_appro
                                                  h,Seller_Bank_checkAndCash_REC,Bank_Provider_
)
end par
end par
end par;
```

Online Shopping 2.0: BPMN



>SChorA

[Nguyen et al, ICSOC'12] [Nguyen et al, ISSRE'13]
SchorA is available at GitHub

The ChorD Language

- Based on [Qiu et al, WWW'07], extended with support for data

L	semantics
1	skip
α	interaction
L1; L2	sequence
L1 + L2	choice
L1 | L2	parallelism
L1 [> L2	interruption
[ϕ] ▷ L	if-then
[ϕ] * L	while

α	choreography language semantics
o[a,b].x	interaction o from a to b, x is not bound
o[a,b].<x>	interaction o from a to b, x is bound

no need for τs since we have choice.

- shopping = request[buyer,vendor].<x>;
  [x⩽0] * (error[vendor,buyer]; request[buyer,vendor].<x>);
  confirm[vendor,buyer].x

The ChorD Language

- Based on [Qiu et al, WWW'07], extended with support for data

L	semantics
1	skip
α	interaction
L1; L2	sequence
L1 + L2	choice
L1 | L2	parallelism
L1 [> L2	interruption
[φ] ▷ L	if-then
[φ] * L	while

α	role language semantics
o[a,b]?x	reception o in a from b, x is not bound
o[a,b]?<x>	reception of o in a from b, x is bound
o[a,b]!x	emission of o in a to b, x is not bound
o[a,b]!<x>	emission of o in a to b, x is bound
τ	internal action

- vendor = request[buyer,vendor]?<x>;
  [x≤0] * (error[vendor,buyer]!; request[buyer,vendor]?<x>);
  confirm[vendor,buyer]!x

Symbolic Transition Graphs (STG)

- Labelled Transition Systems extended with support for data

  [Hennessy and Lin, TCS:138(2), 1995] with renamings/extensions (STGA, STS, IOLTS, ...) by others

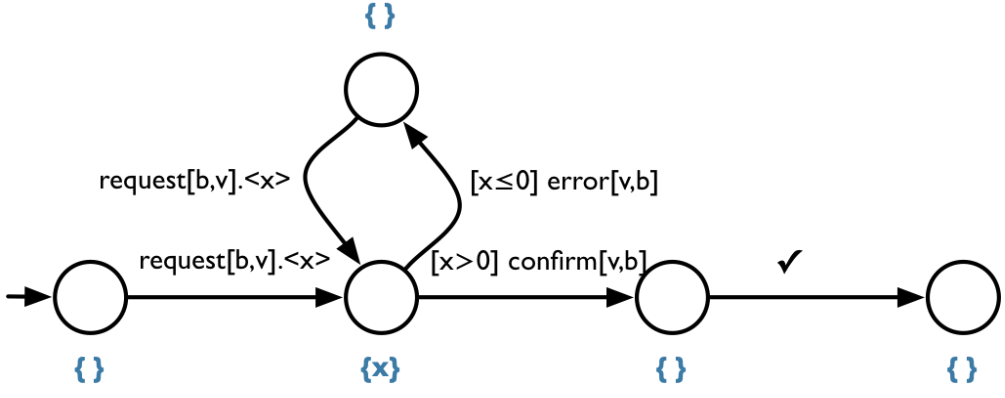
  used for symbolic verification, e.g., WS-BPEL testing in [Bentakouk et al, TESTCOM'07]

- 13 transformation rules from ChorD to STG

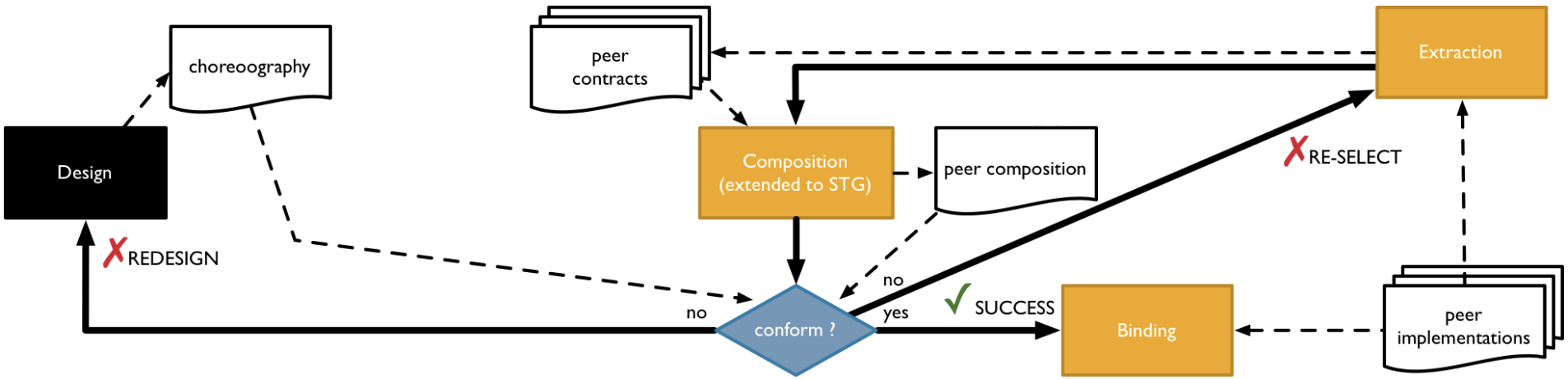
request[buyer,vendor].<x>;

[x≤0] * (error[vendor,buyer]; request[buyer,vendor].<x>);

confirm[vendor,buyer].x



↑ Choreography Development in SChorA



Conformance and Data

- Conformance is a relation between a specification and an implementation

approach	data	support using	loops	assignments	equivalence
[Busi et al, Coordination'06]	yes	closure	no	yes	branching bisimulation
[Li et al, TASE'07]		closure	yes	yes	weak bisimulation
[Kazhamiakin et Pistore, WS-FM'06]		bound domains	yes	no	branching bisimulation

- closure does not work for open choreographies, e.g., o1[a,b].<x>; [x>0] |> o2[b,a].x
  bound domains still yield state space explosion if LTS are used
  possibility to have internal actions in peers, hence in implementations
  ➜ symbolic branching bisimulation
- conformance is not always just true or false
  if x ⩾ 0, Impl = ([x⩾0] ▷ confirm[a,b].x + [x<0] ▷ cancel[a,b].x) is conform to Spec = confirm[a,b].x
  ➜ most general constraint for conformance

Conformance Verification

1. hide (τ) additional interactions in the implementation

   these interactions may have been added by the developper to reach conformance

   they may also result from peers interacting with partners outside the scope of the choreography

2. compute the most general constraint ρ over data exchanged between peers to achieve conformance

   modification of [Li and Chen, TACAS'99], symbolic weak bisimulation → symbolic branching bisimulation

   ρ is obtained as a Predicate Equation System (PES)

3. verify ρ using the Z3 SMT solver

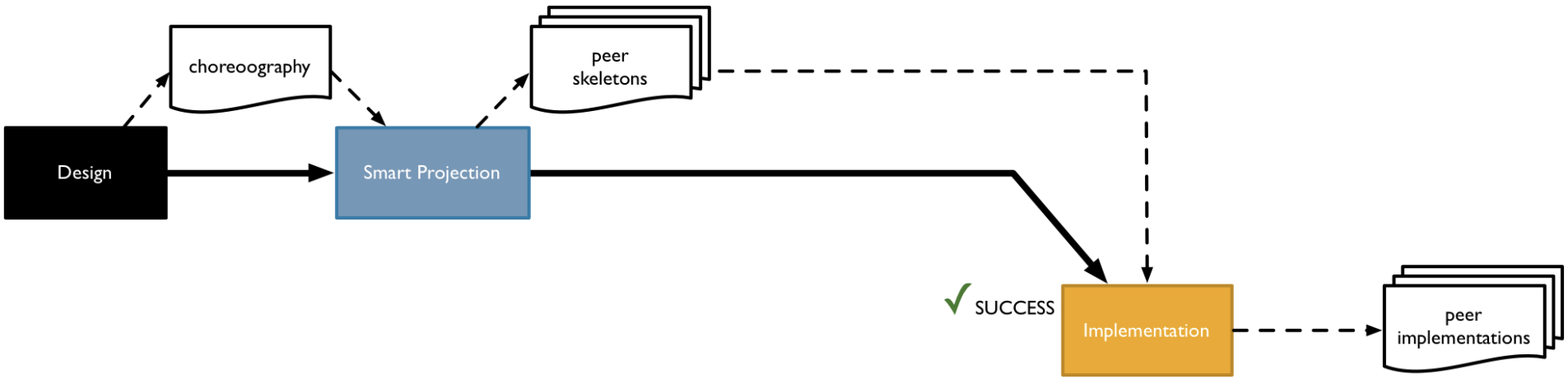
   transformation from PES to the Z3 input language

4. reach verdict (true / false / maybe / inconclusive)

   if [ρ=false]→ unsat then ρ is always true and we have conformance

   else verdict based on [ρ=true]: unsat → false, sat → maybe, and timeout → inconclusive

Choreography Development in SChorA



Realizability and Data

- A choreography is realizable if we can generate conform peers using projection

approach	synchr. communication	asynchr. communication	#causality semantics
[Sun et al, APSEC'10]	yes	no	1

- 4 different causality semantics for m[a,b]; n[c,d]

  [Lanese et al, SEFM'08] synch, send-asynch, receive-asynch, disjoint-asynch

  o1[b,a]; o2[c,a]: ✓ for synch, ✗ for send-asynch (b.o1! < c.o2! not ensured)

  o1[a,b]; o2[a,c]: ✓ for synch, ✗ for receive-asynch (b:o1? < c.o2? not ensured)

  o1[a,b]; o2[c,a]: ✓ for synch, ✗ for disjoint-asynch (b:o1? < c.o2! not ensured)
- full-support for data

  o1[c,a]<x>; [x>0] |> o2[a,b]; [x≤0] |> o3[c,d] is realizable (dead parts)

  o1[a,c].<x>; ([x>0] |> o2[a,b] + [x<0] |> o3[c,d]) is realizable (choice based on known data)
- realizability achieved using a smart projection

  intrusive control well-suited for ↓

🛠 Smart Projection

1. prune unreachable transitions and states

   these are parts of the choreography that are not consistent

2. connect consecutive interactions with different sender/receiver

   [Lanese et al, SEFM'08] e.g., o1[a,b].<x>; o2[c,d].<y> → o1[a,b].<x>; X[a,c]; o2[c,d].<y> (send-asynch)

3. synchronize choice branches

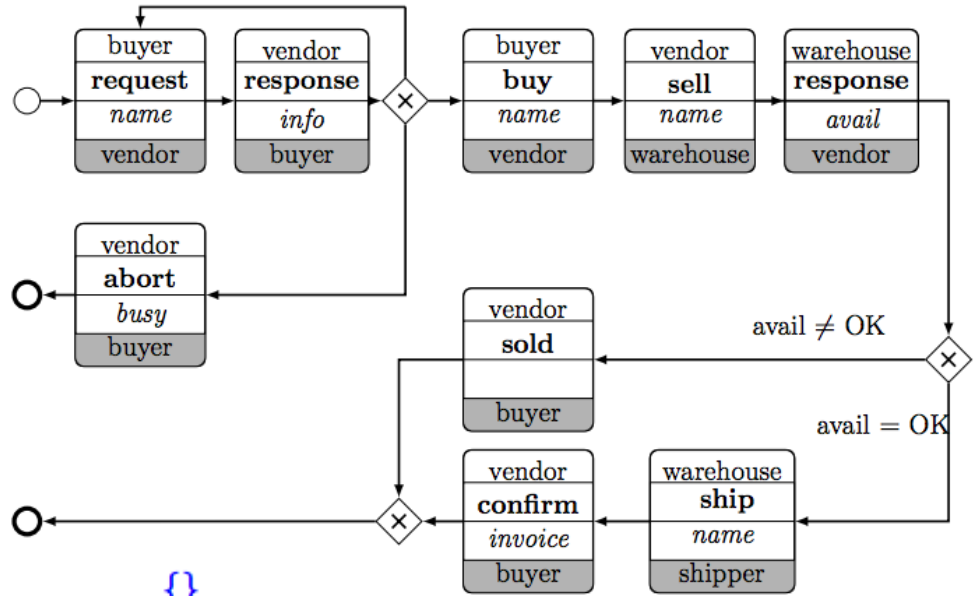
   [Qiu et al, WWW'07] role-based, o1[a,b] + o2[a,c] + o3[d,e] → o1[a,b] + o2[a,c] + X[a,d]; o3[d,e] (send-asynch)

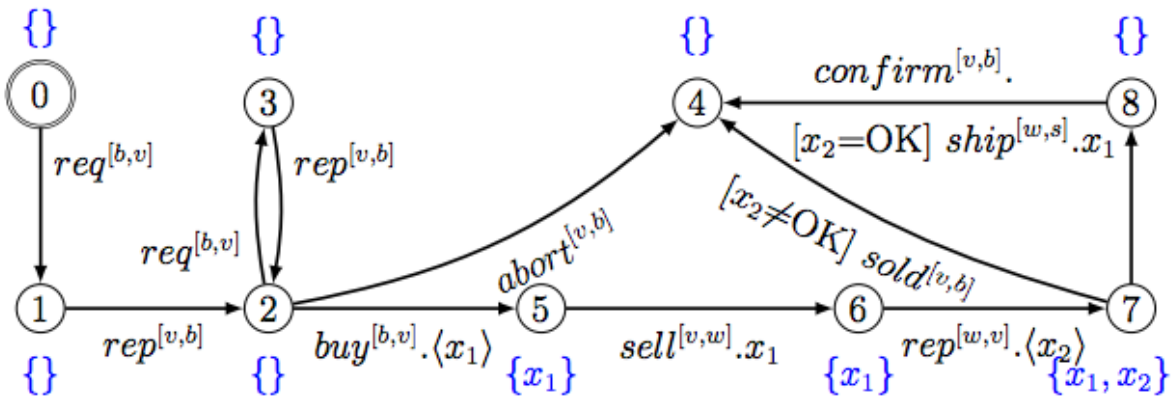
   nothing to do if data-based, o1[a,c].<x>; ([x>0] |> o2[a,b] + [x<0] |> o3[c,d])

4. exchange information relative to data constraints between peers

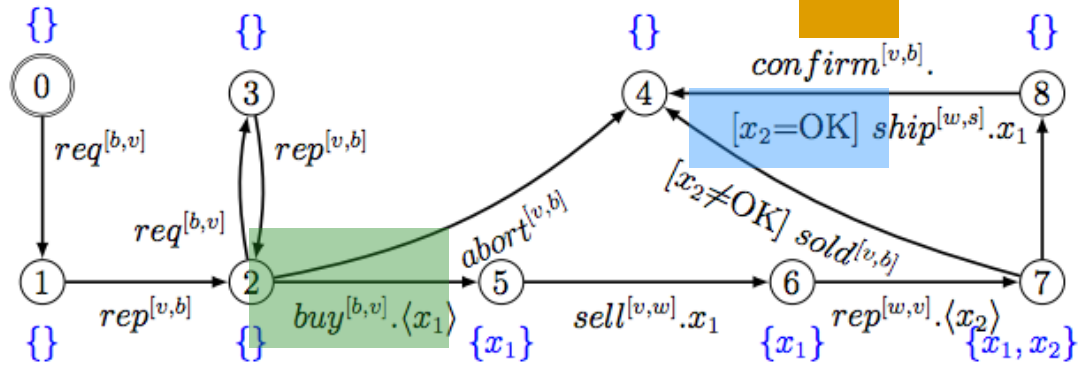
   o1[a,b].<x>; o2[c,d].x → o1[a,b].<x>; X[b,c].x; o2[c,d].x (disjoint-asynch)

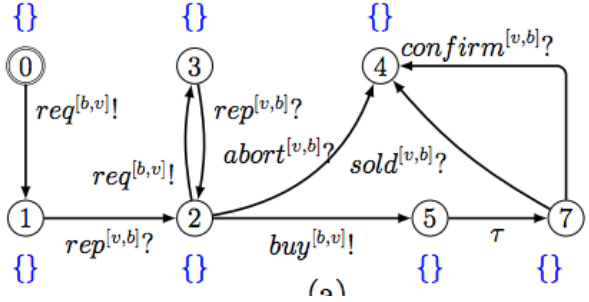
Example

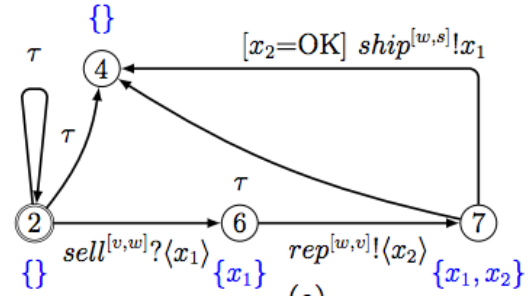

buyer
request
name
vendor
vendor
response
info
buyer
buyer
buy
name
vendor
vendor
sell
name
warehouse
warehouse
response
avail
vendor
vendor
abort
busy
buyer
vendor
sold
buyer
avail ≠ OK
avail = OK
vendor
confirm
invoice
buyer
warehouse
ship
name
shipper

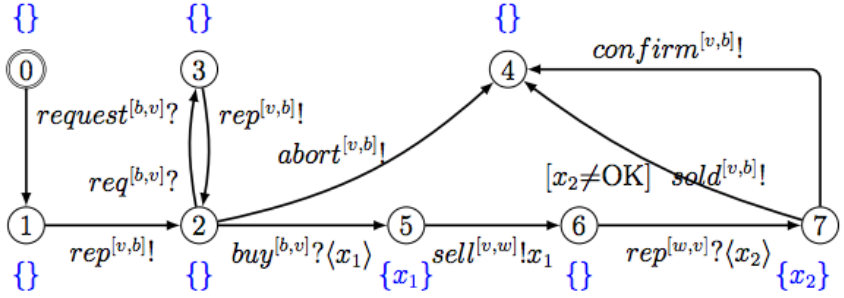

{}
0
req[b,v]
1
{}
req[b,v]
rep[v,b]
3
{}
2
{}
rep[v,b]
buy[b,v].⟨x1⟩
abort[v,b]
4
{}
5
{x1}
sell[v,w].x1
6
{x1}
rep[w,v].⟨x2⟩
7
{x1, x2}
[x2≠OK] sold[v,b]
[x2=OK] ship[w,s].x1
8
{}
confirm[v,b].

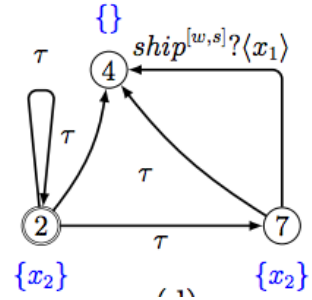
Example


{}
0
req[b,v]
1
rep[v,b]
{}
3
rep[v,b]
req[b,v]
2
buy[b,v].⟨x1⟩
{}
4
abort[v,b]
{}
5
{x1}
sell[v,w].x1
confirm[v,b].
[x2=OK] ship[w,s].x1
8
[x2≠OK] sold[v,b]
6
{x1}
rep[w,v].⟨x2⟩
7
{x1, x2}

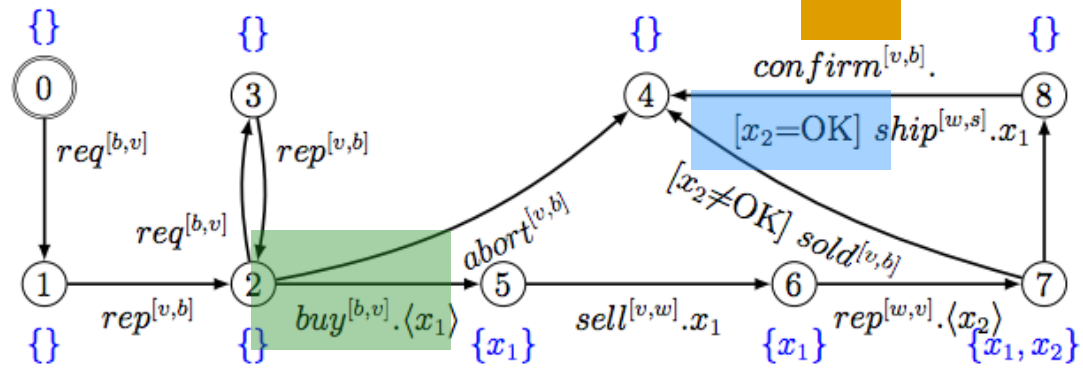

{}
0
req[b,v]!
1
rep[v,b]?
{}
3
rep[v,b]?
req[b,v]!
2
abort[v,b]?
buy[b,v]!
{}
4
confirm[v,b]?
sold[v,b]?
5
τ
7

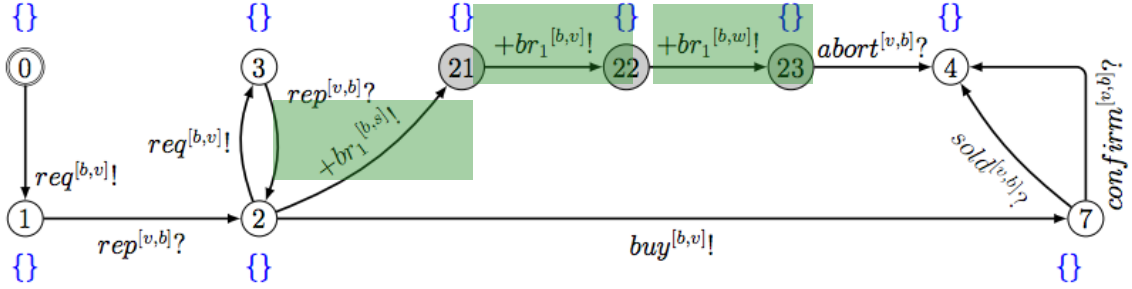

{}
τ
τ
2
sell[v,w]?⟨x1⟩
{x1}
τ
6
4
[x2=OK] ship[w,s]!x1
7
rep[w,v]!⟨x2⟩
{x1, x2}

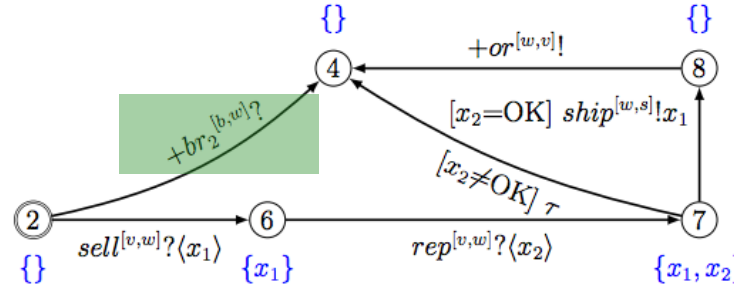

{}
0
request[b,v]?
1
rep[v,b]!
{}
3
rep[v,b]!
req[b,v]?
2
buy[b,v]?⟨x1⟩
{x1}
abort[v,b]!
{}
4
confirm[v,b]!
[x2≠OK] sold[v,b]!
5
sell[v,w]!x1
{}
6
rep[w,v]?⟨x2⟩
7
{x2}

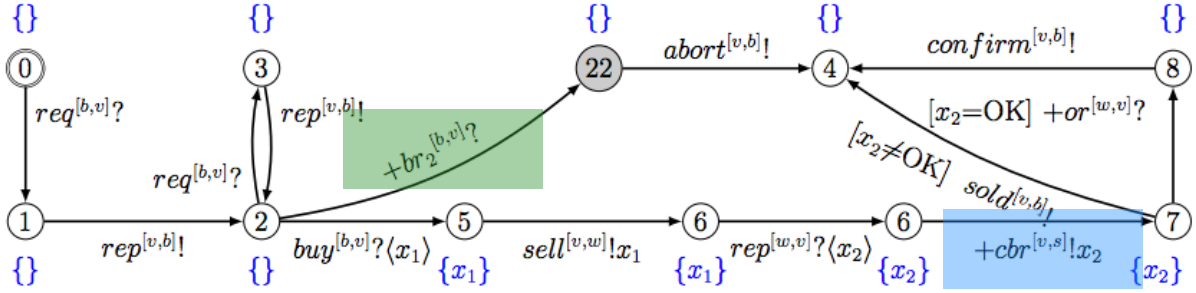

{}
τ
τ
2
{x2}
τ
τ
4
ship[w,s]?⟨x1⟩
7
{x2}

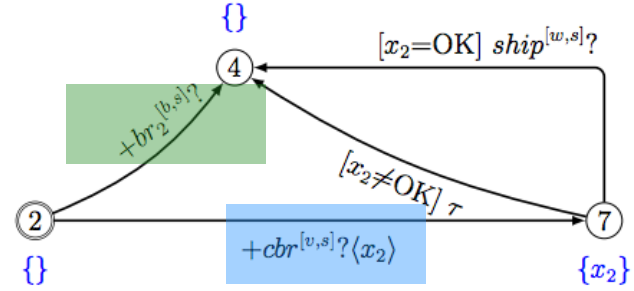
Example


req[b,v]
rep[v,b]
req[b,v]
rep[v,b]
buy[b,v].⟨x1⟩
abort[v,b]
sell[v,w].x1
rep[w,v].⟨x2⟩
[x2≠OK] sold[v,b]
[x2=OK] ship[w,s].x1
confirm[v,b].


req[b,v]!
req[b,v]!
rep[v,b]?
rep[v,b]?
+br1[b,s]!
+br1[b,v]!
+br1[b,w]!
abort[v,b]?
buy[b,v]!
sold[v,b]?
confirm[v,b]?


+br2[b,w]?
sell[v,w]?⟨x1⟩
rep[w,v]?⟨x2⟩
[x2≠OK] τ
[x2=OK] ship[w,s]!x1
+or[w,v]!


req[b,v]?
req[b,v]?
rep[v,b]!
rep[v,b]!
+br2[b,v]?
buy[b,v]?⟨x1⟩
sell[v,w]!x1
rep[w,v]?⟨x2⟩
+cbr[v,s]!x2
[x2≠OK] sold[v,b]!
[x2=OK] +or[w,v]?
abort[v,b]!
confirm[v,b]!


+br2[b,s]?
+cbr[v,s]?⟨x2⟩
[x2≠OK] τ
[x2=OK] ship[w,s]?

Conclusion

- choreographies are central in distributed system development
  well-suited for global specifications with interaction as a 1st class citizen
  support reasoning and rapid development through generative approaches

- formal methods integrate well in choreography-based development
  formal grounding - synchronizability, realizability, conformance, control
  tool development - VerChor and SChorA (both open source)

- going further?
  study new developments on classes of choreographies
  choreographies and dynamic system evolution
  application of the symbolic approach (SChorA) to BPM