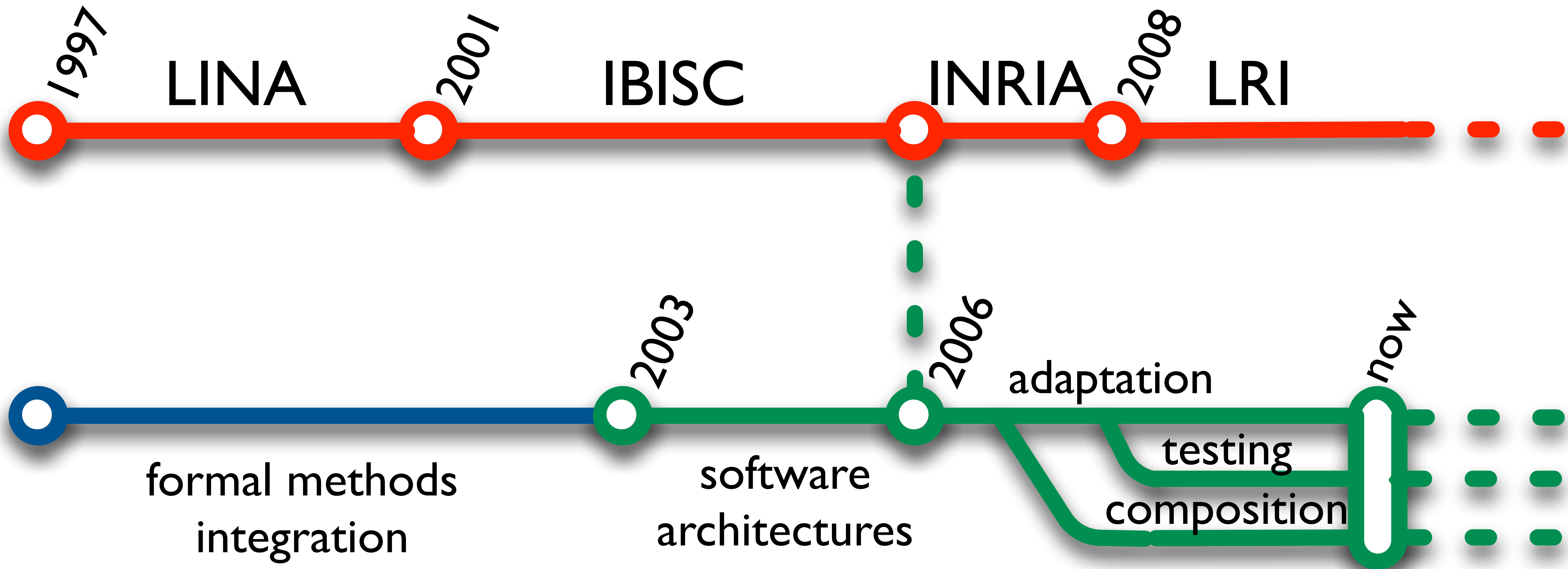
Formal Model-Based Approaches for the Development of Composite Systems

MeFoSyLoMa Seminar
(originally, Habil. thesis defense, Nov. 24th, 2011)

Pascal Poizat
Université d'Evry Val d'Essonne;
LRI CNRS UMR 8623 et Université Paris-Sud 11

Evry, February 17th, 2012

Timeline

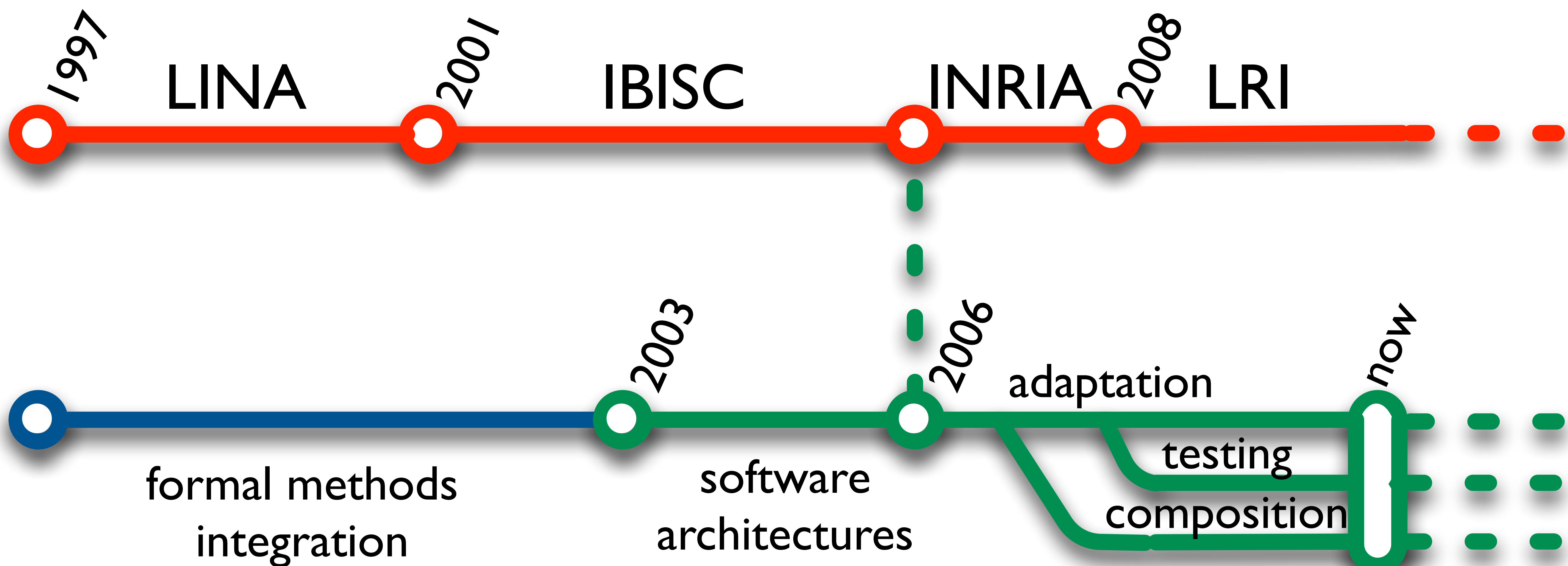


see
IEEE TSE 33(3), 2007

see
J.UCS 12(12), 2006

see
IEEE TSE 34(4), 2008
IEEE TSE under press
TESTCOM'09, SAC'12
ICSOC'08, ISoLa'10
ICWS'10, ICSOC'10

What about MeFoSyLoMa?



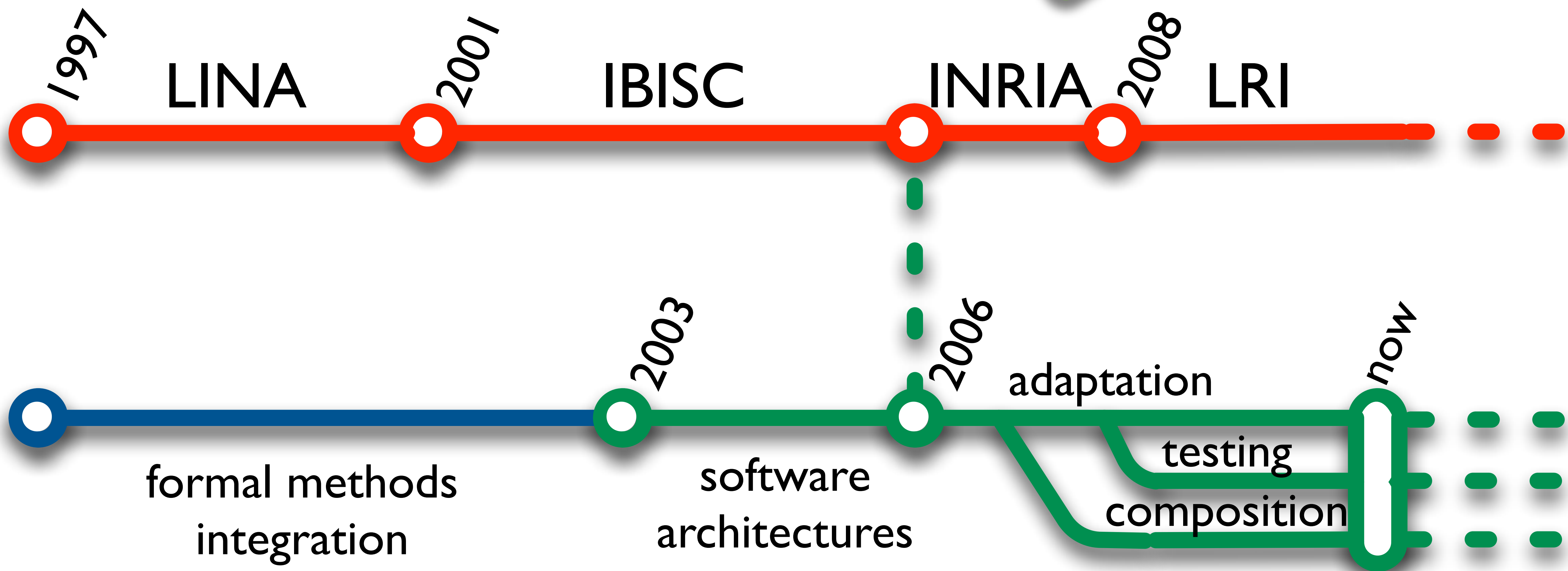
see
IEEE TSE 33(3), 2007

see
J.UCS 12(12), 2006

see
IEEE TSE 34(4), 2008
IEEE TSE under press
TESTCOM'09, SAC'12
ICSOC'08, ISoLa'10
ICWS'10, ICSOC'10

What about MeFoSyLoMa?

Applicative Domain



see
IEEE TSE 33(3), 2007

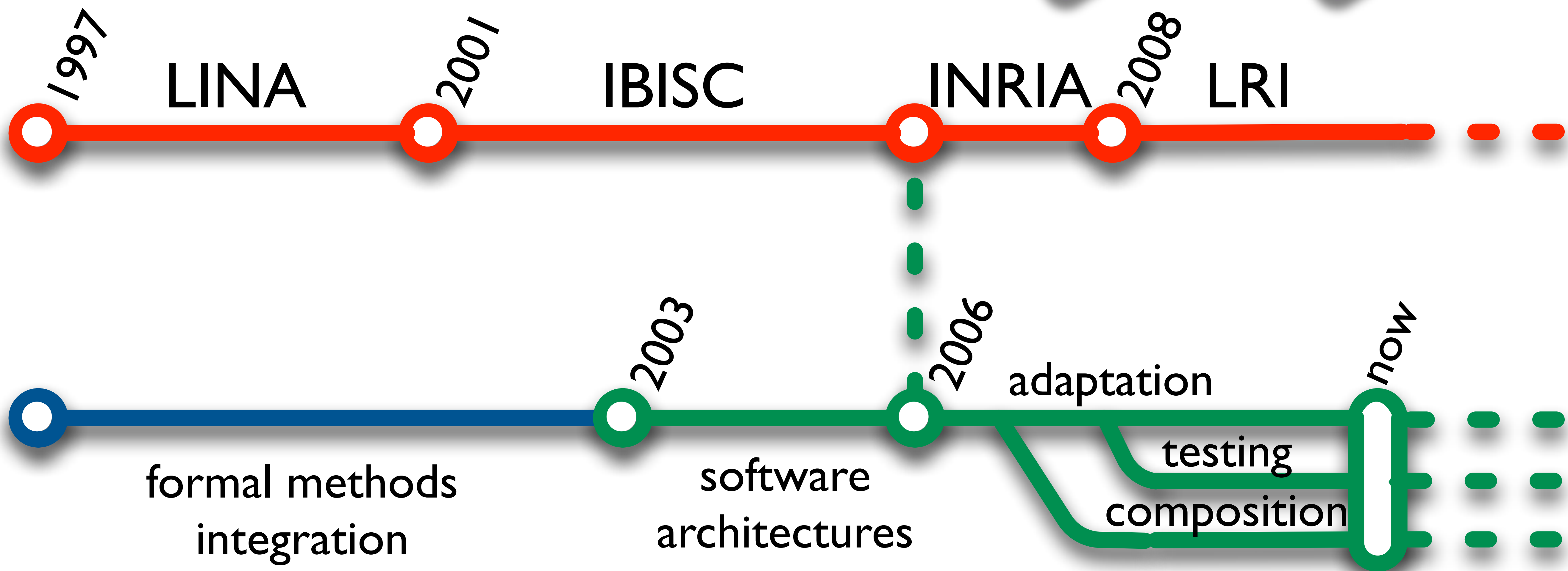
see
J.UCS 12(12), 2006

see
IEEE TSE 34(4), 2008
IEEE TSE under press
TESTCOM'09, SAC'12
ICSOC'08, ISoLa'10
ICWS'10, ICSOC'10

What about MeFoSyLoMa?

Applicative Domain

MeFoSyLoMian Models/Tools



see
IEEE TSE 33(3), 2007

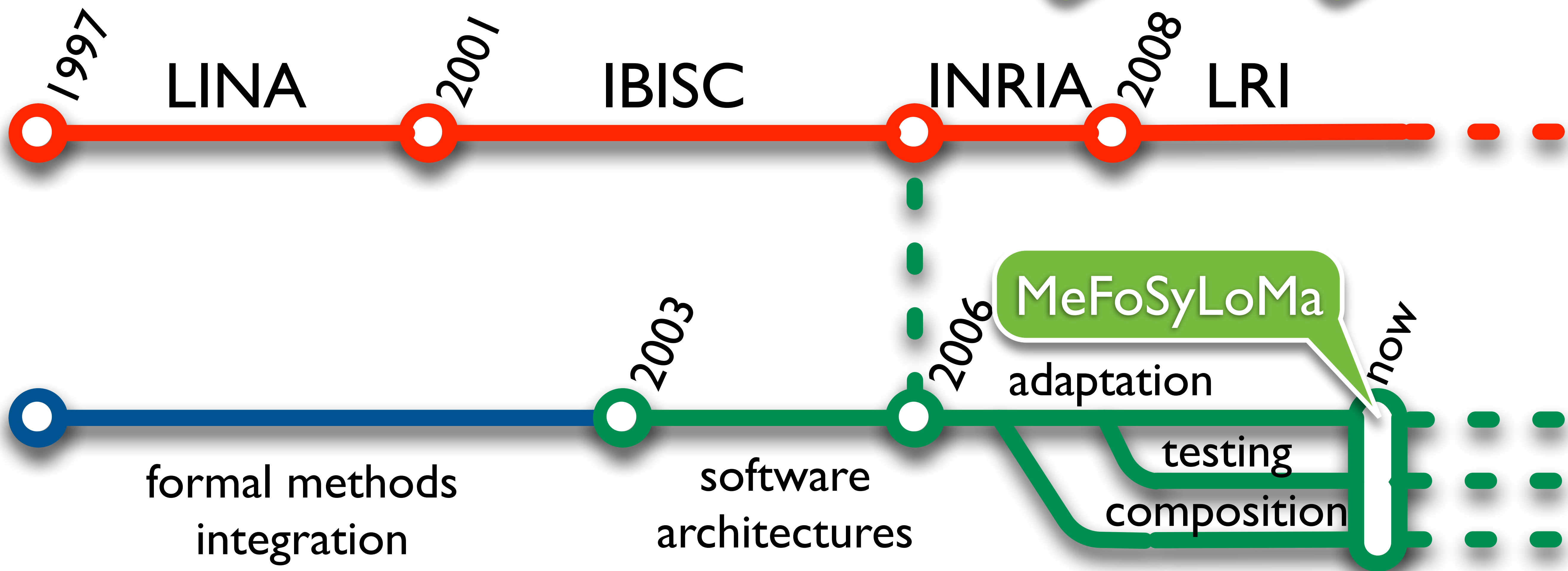
see
J.UCS 12(12), 2006

see
IEEE TSE 34(4), 2008
IEEE TSE under press
TESTCOM'09, SAC'12
ICSOC'08, ISoLa'10
ICWS'10, ICSOC'10

What about MeFoSyLoMa?

Applicative Domain

MeFoSyLoMian Models/Tools



see
IEEE TSE 33(3), 2007

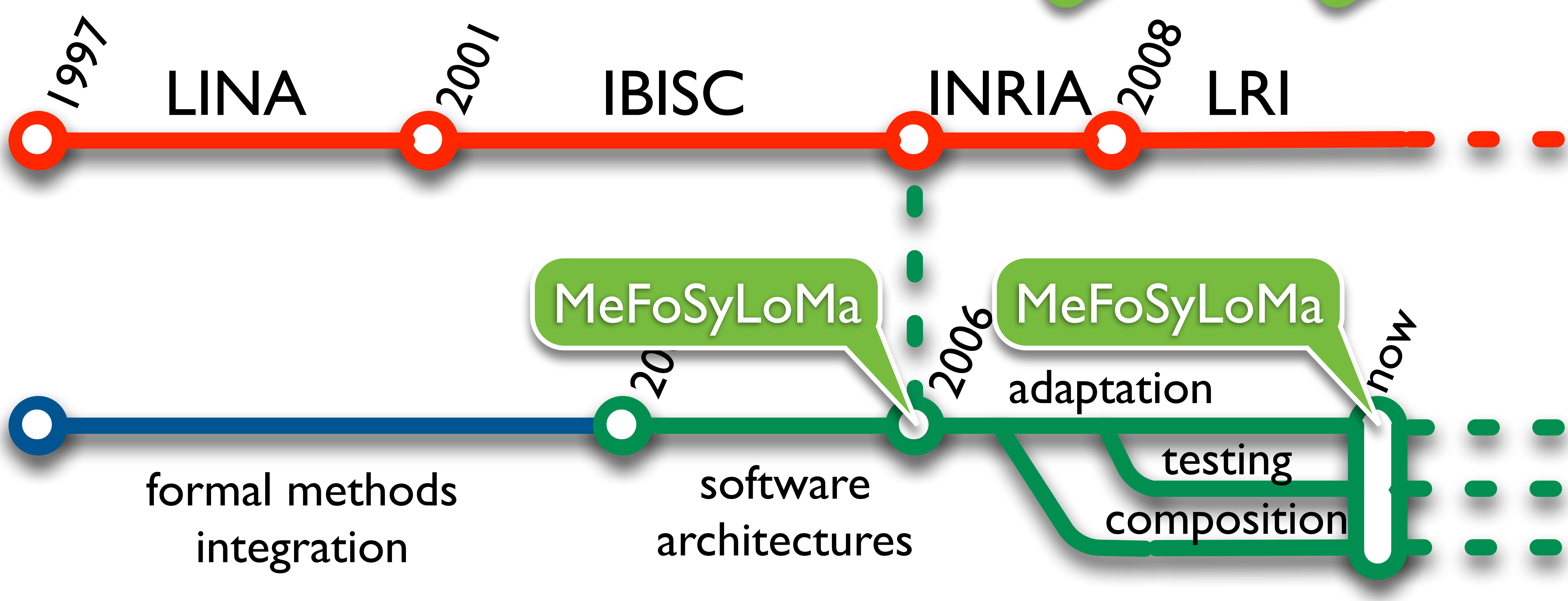
see
J.UCS 12(12), 2006

see
IEEE TSE 34(4), 2008
IEEE TSE under press
TESTCOM'09, SAC'12
ICSOC'08, ISoLa'10
ICWS'10, ICSOC'10

What about MeFoSyLoMa?

Applicative Domain

MeFoSyLoMian Models/Tools

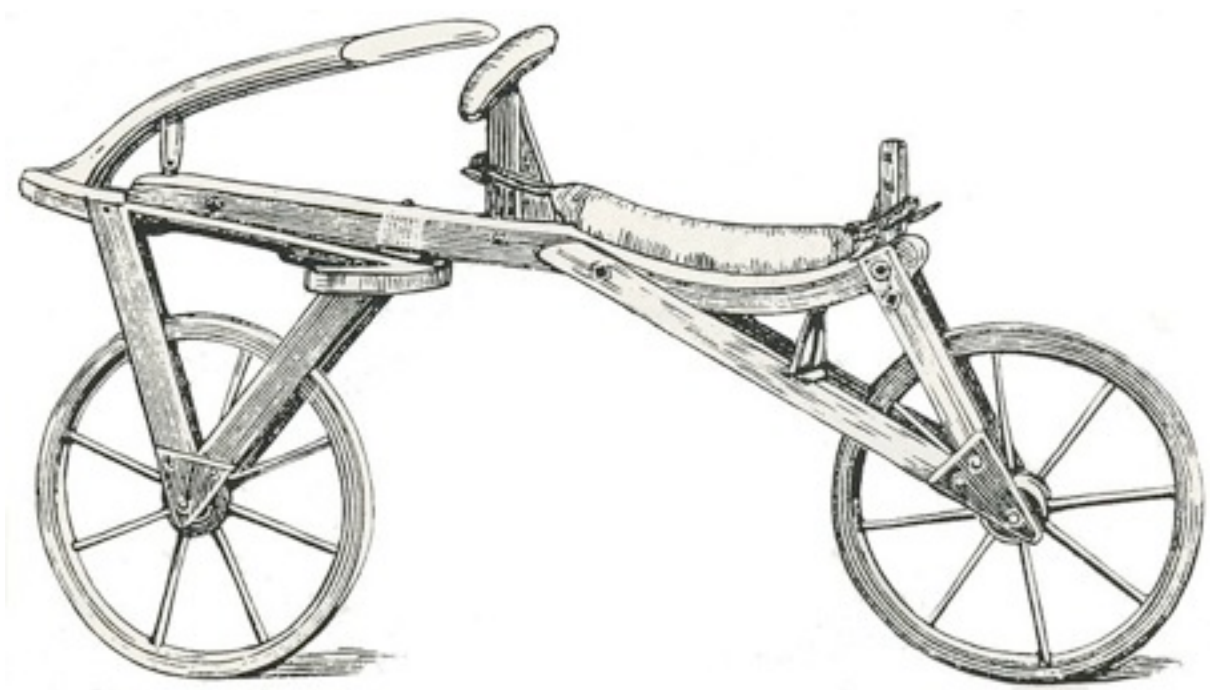


see
IEEE TSE 33(3), 2007

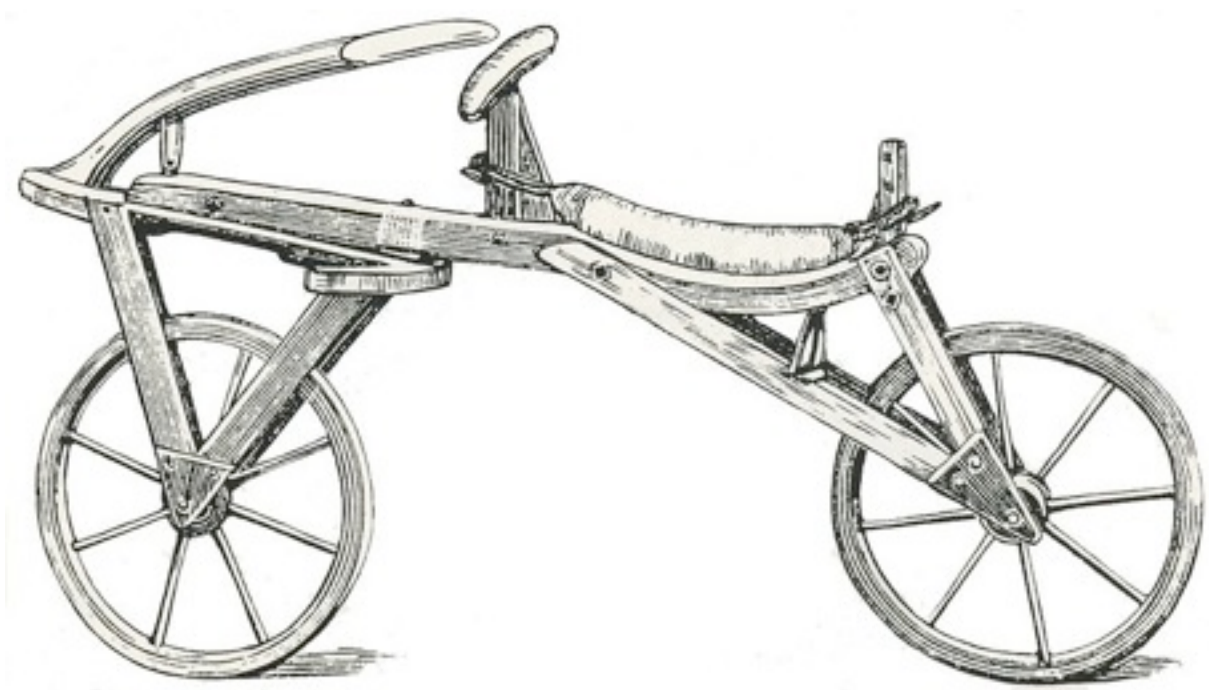
see
J.UCS 12(12), 2006

see
IEEE TSE 34(4), 2008
IEEE TSE under press
TESTCOM'09, SAC'12
ICSOC'08, ISoLa'10
ICWS'10, ICSOC'10

the simplest things



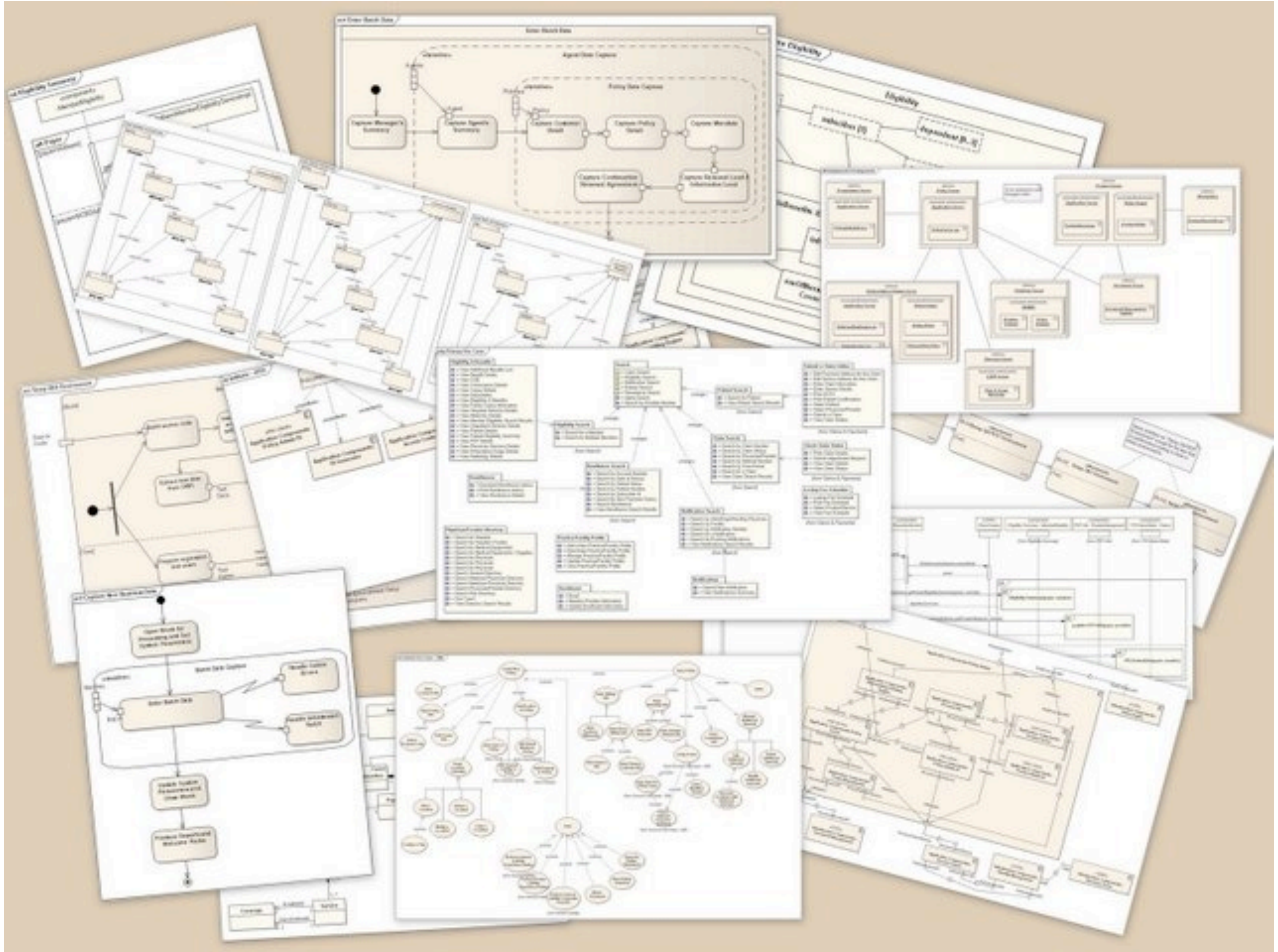
get complex with time



Source: talk by Ph. Merle at GDR GPL 2011

what about software?

- increasing use of:
  - viewpoints
  - distribution
  - interaction



Source: WikiMedia Commons (by Kishorekumar 62)

Structuring

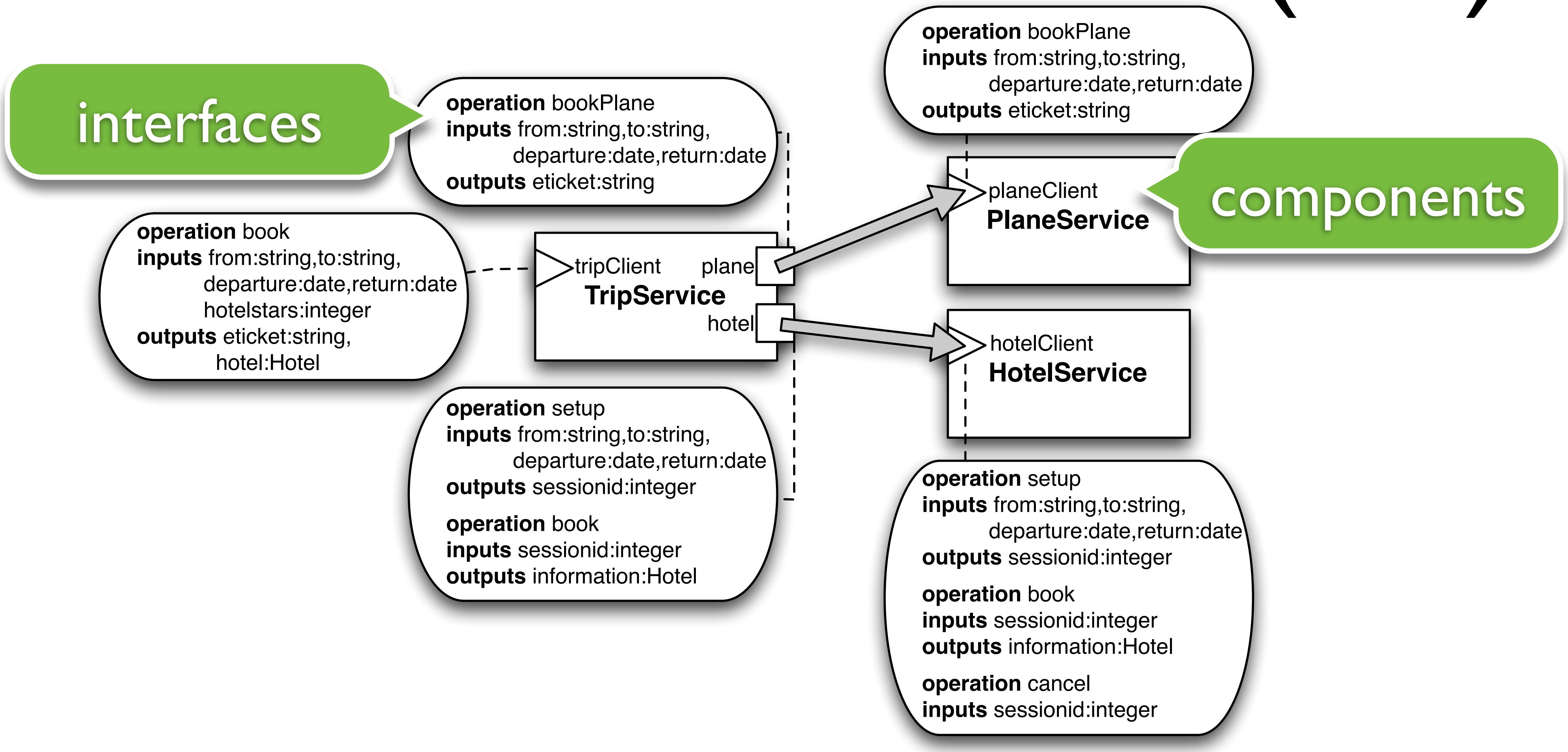
- **Modules**, Object-Orientation
  +: **well-defined provided interfaces**, reusability
  -: **hidden required functionalities**

Structuring

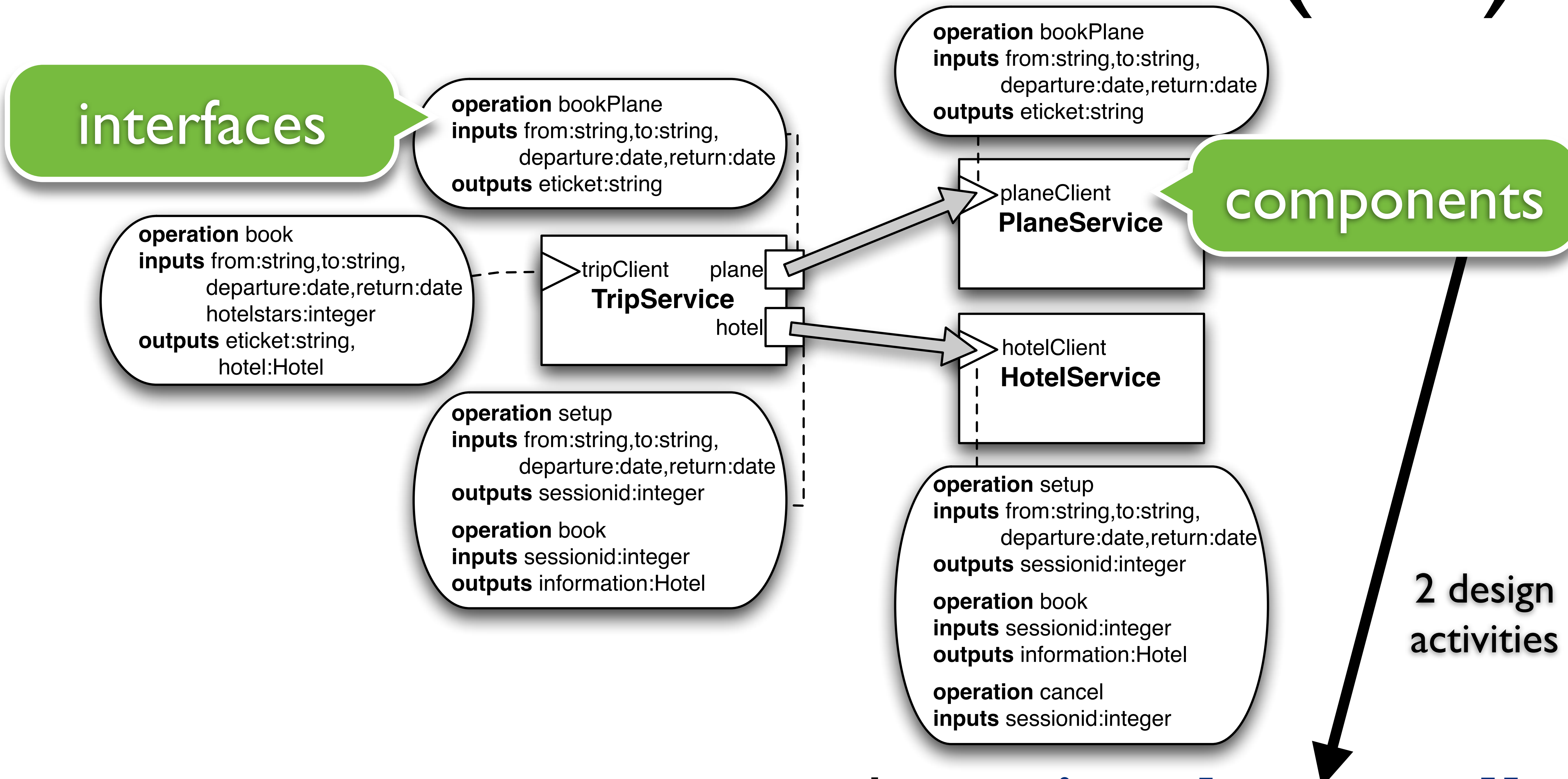
- **Modules**, Object-Orientation
  +: **well-defined provided interfaces**, reusability
  -: **hidden required functionalities**

- **Software Components**, Services
  +: **explicit required functionalities**, dynamic binding
  -: richer interfaces are **harder to use**

Software Architectures (SA)

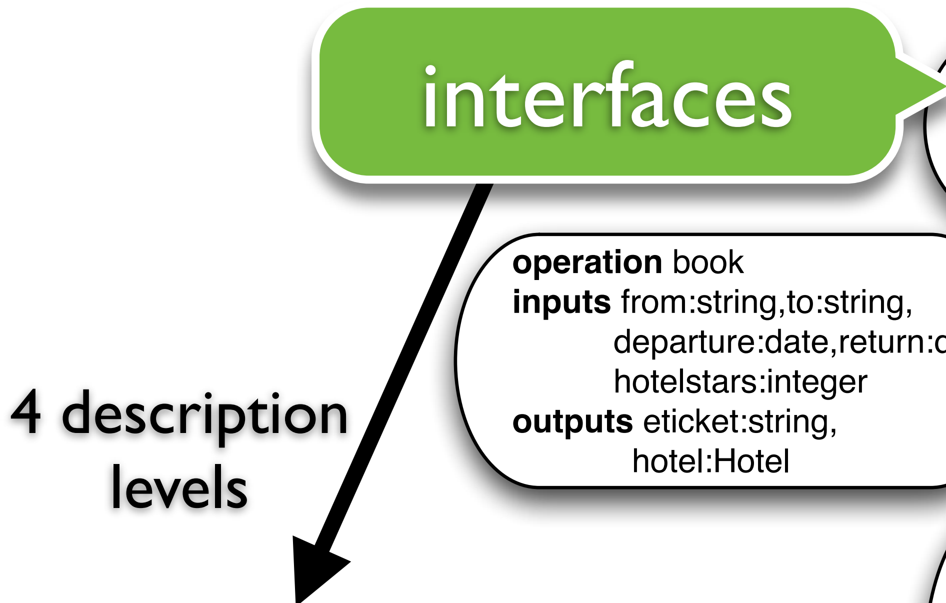


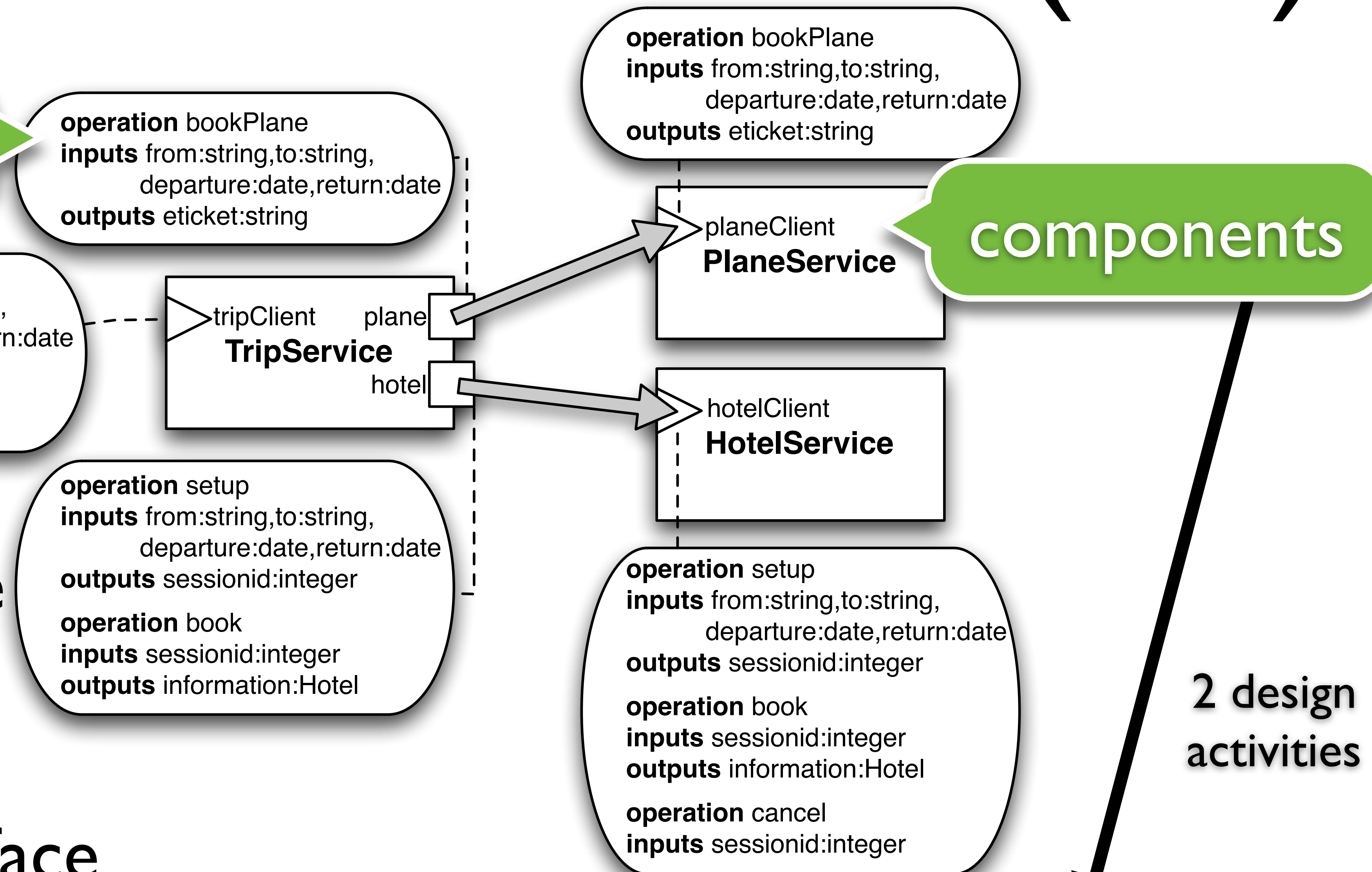
Software Architectures (SA)



- «design **in-the-small**»
  design, implementation, and verification of sub-systems
- «design **in-the-large**»
  structuring of the system as a set of sub-systems

Software Architectures (SA)





- **signature** interface
  provides the access points (ports, operations)
- **behavioural** interface
  defines the usage protocol

additionally,

semantic interface fosters automation

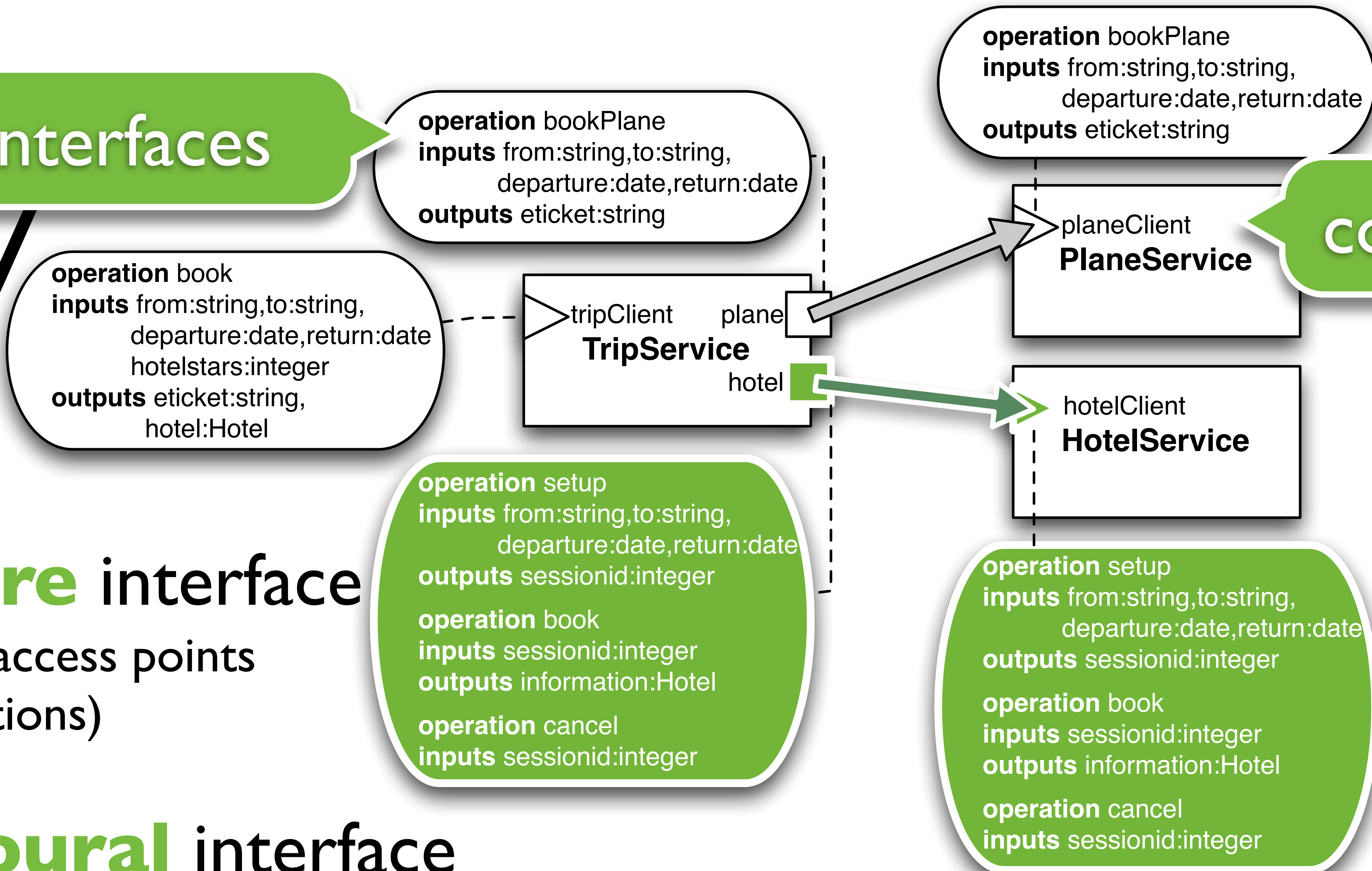
non-functional interface for QoS

- «design **in-the-small**»
  design, implementation, and verification of sub-systems
- «design **in-the-large**»
  structuring of the system as a set of sub-systems

Software Architectures (SA)

interfaces

components



4 description levels

- **signature** interface
  provides the access points (ports, operations)
- **behavioural** interface
  defines the usage protocol

additionally,

semantic interface fosters automation

non-functional interface for QoS

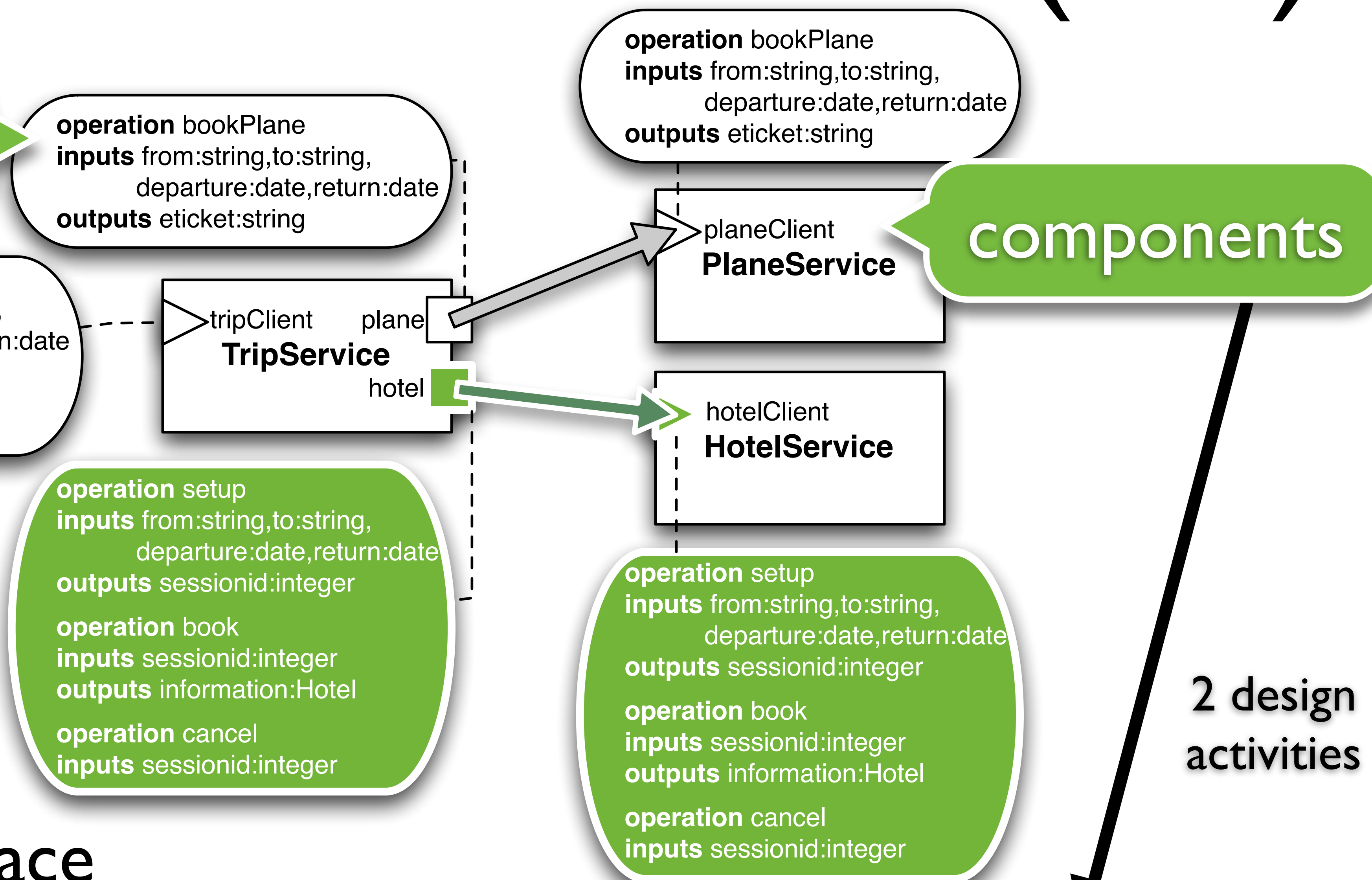
2 design activities

- «design **in-the-small**»
  design, implementation, and verification of sub-systems
- «design **in-the-large**»
  structuring of the system as a set of sub-systems

Software Architectures (SA)

interfaces

4 description levels



- **signature** interface
  provides the access points (ports, operations)

2 design activities

- **behavioural** interface

required by TripService

```
receive(tripClient,book,{from,to,dep,ret});
{hi} := invoke(hotel,setup,{from,to,dep,ret});
{infoh} := invoke(hotel,book,{hi});
{pi} := invoke(plane,bookPlane,{from,to,dep,ret});
if pi.equals(«no plane») { invoke(hotel,cancel,{hi}); }
reply(tripClient,book,...)
```

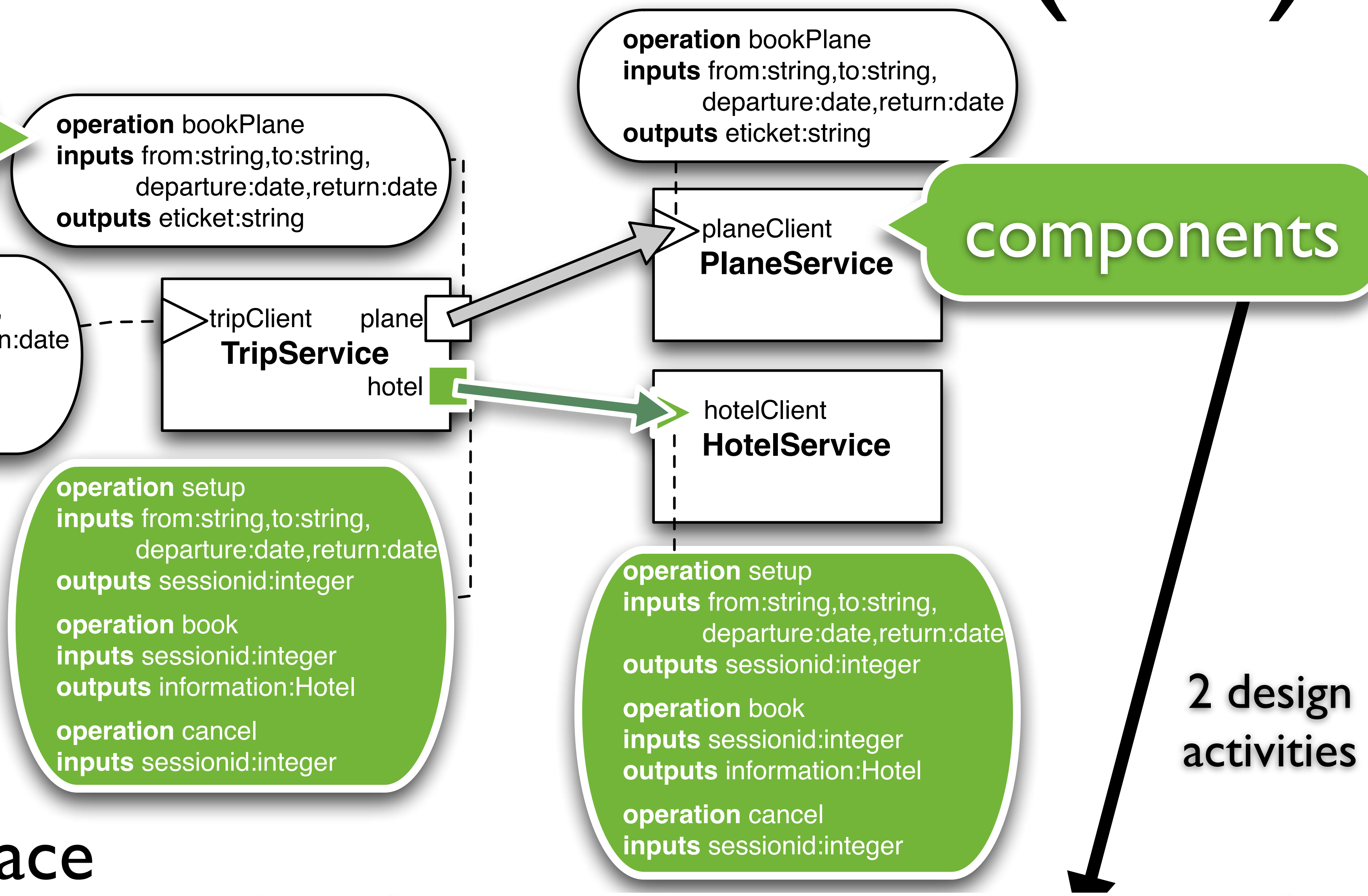
provided by HotelService

```
receive(hotelClient,setup,{from,to,departure,return});
... ; reply(hotelClient,setup,{sessionid});
pick { // «choose between»
onMessage(hotelClient,book,{sessionid}) -> ... ;
onMessage(hotelClient,cancel,{sessionid}) -> ... ; }
```

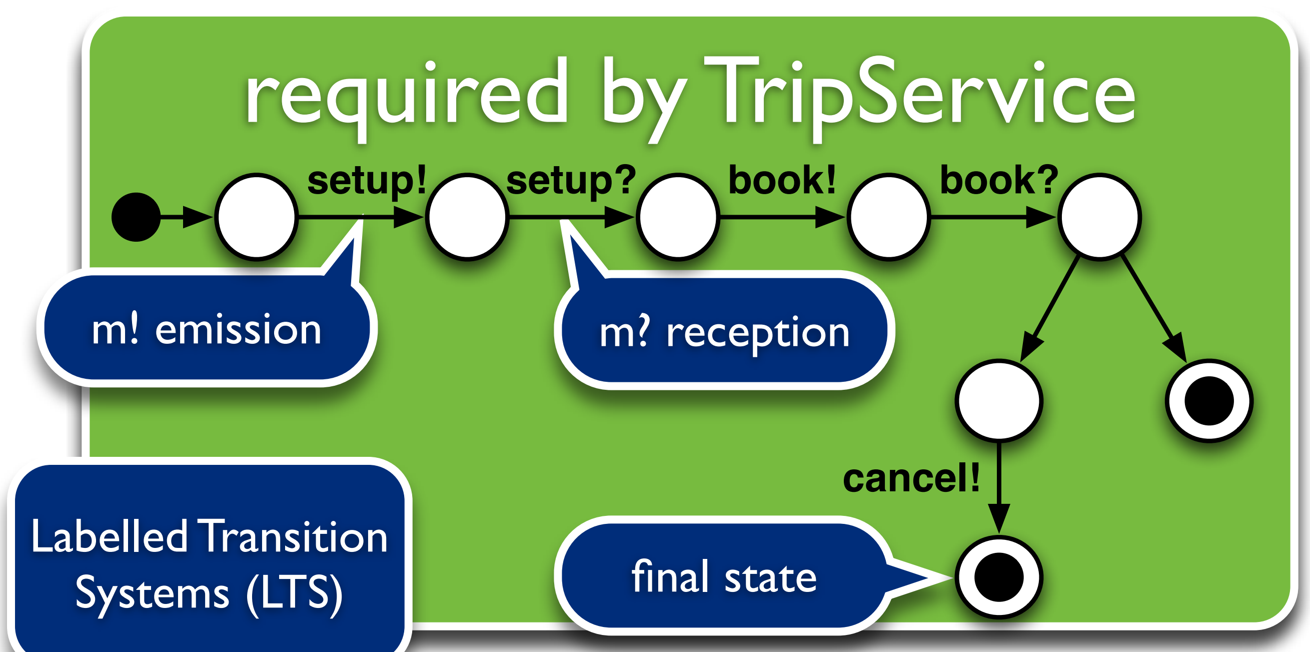
Software Architectures (SA)

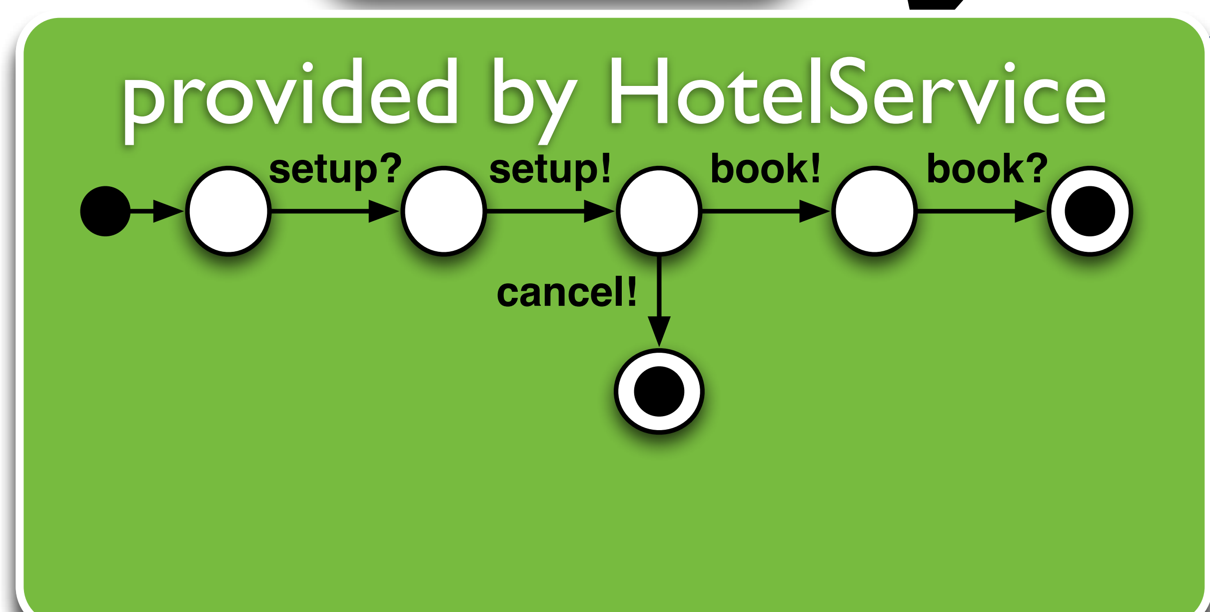
interfaces

4 description levels



- **signature** interface
  provides the access points
  (ports, operations)
- **behavioural** interface





SA *vs.* SOA

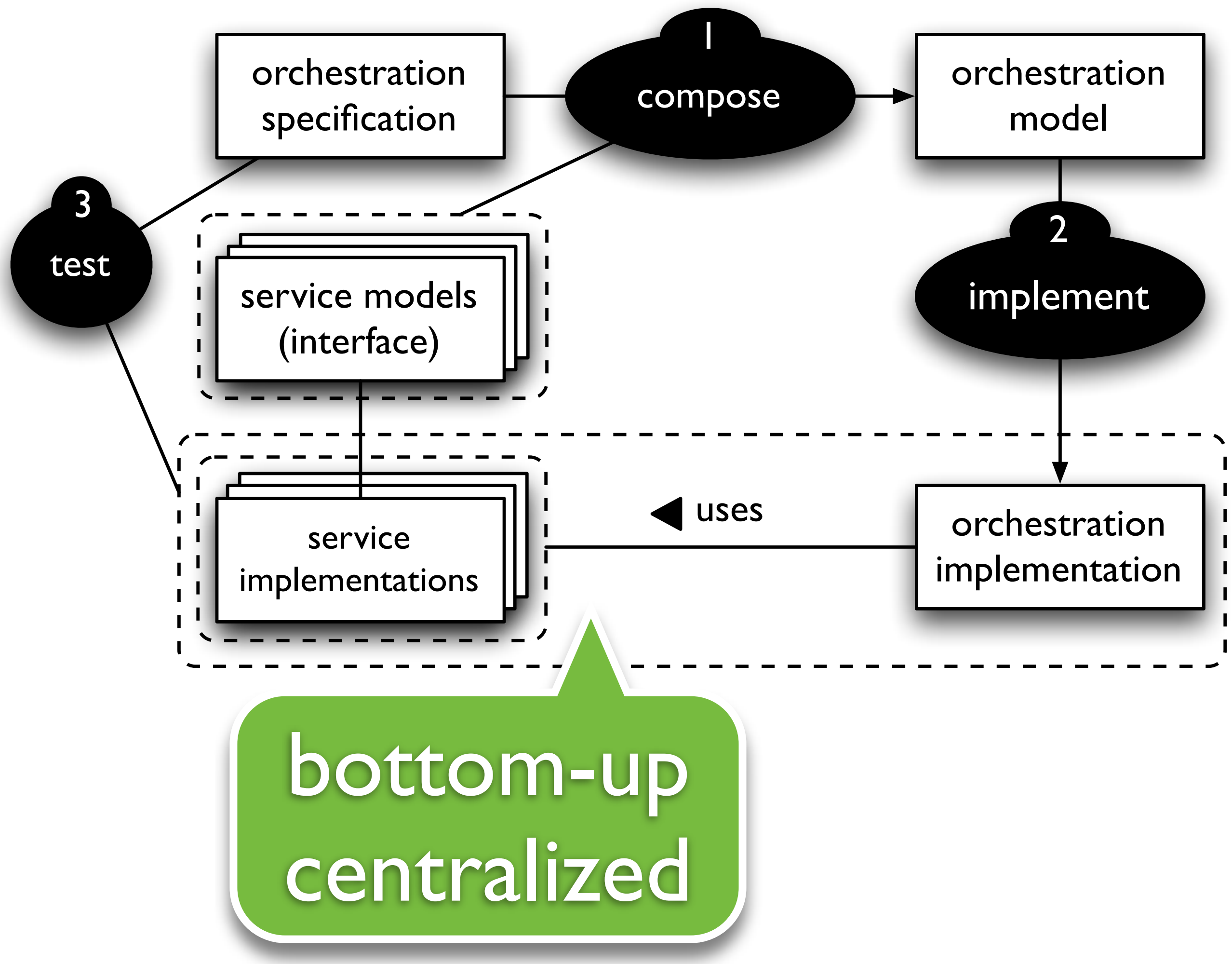
- Service-Oriented Architectures (SOA) are the **modern instance** of Software Architectures

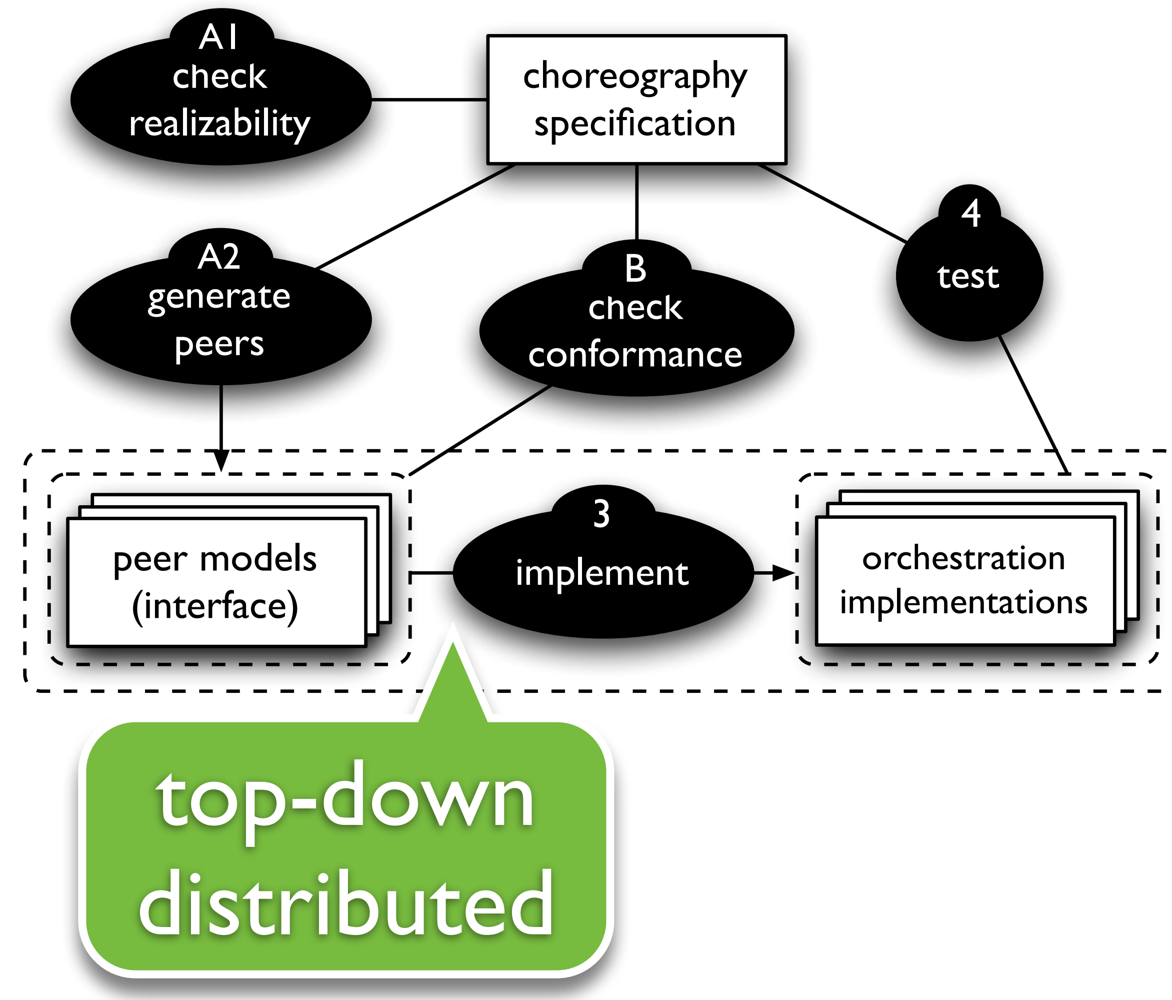
SA *vs.* SOA

- Service-Oriented Architectures (SOA) are the **modern instance** of Software Architectures
- Simple correspondance with SOA major implementation: **Web services**

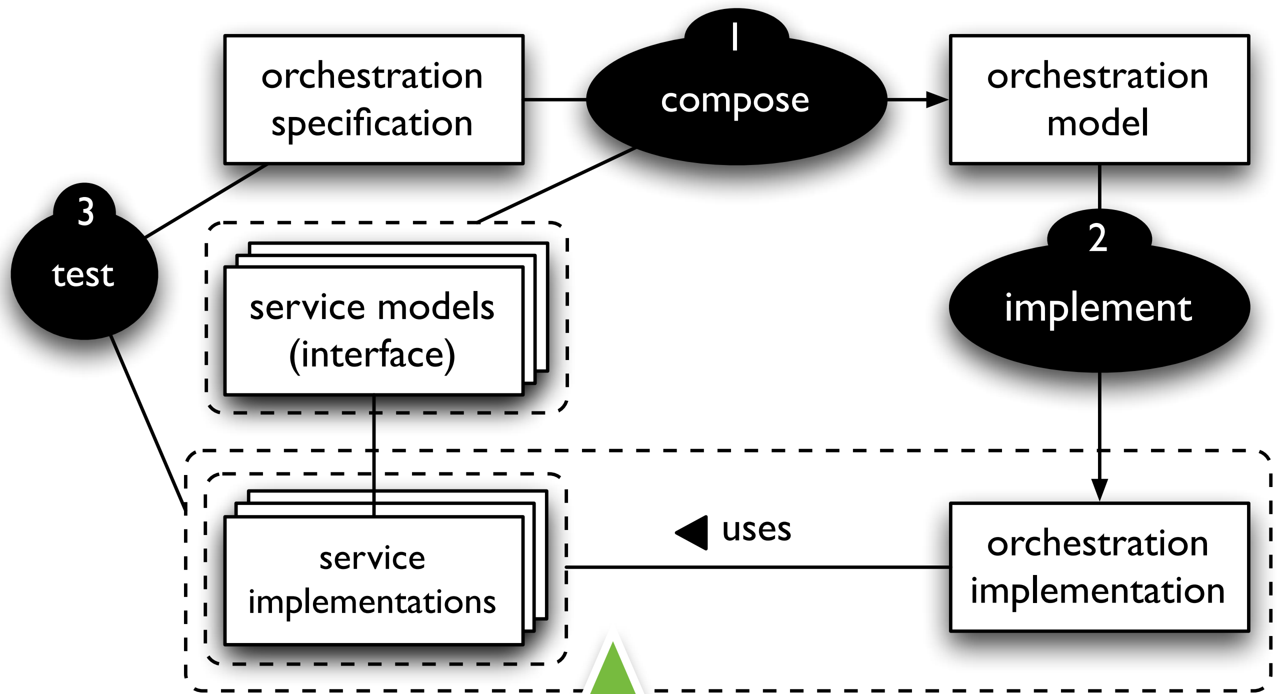
Software Architectures	Web Services
basic component	simple service
composite component (centralized / distributed)	composite service (orchestration / choreography)
signature interface	WSDL interface
behavioural interface	conversation, e.g., in ABPEL
semantic interface	semantic annotations within WSDL files

Development processes

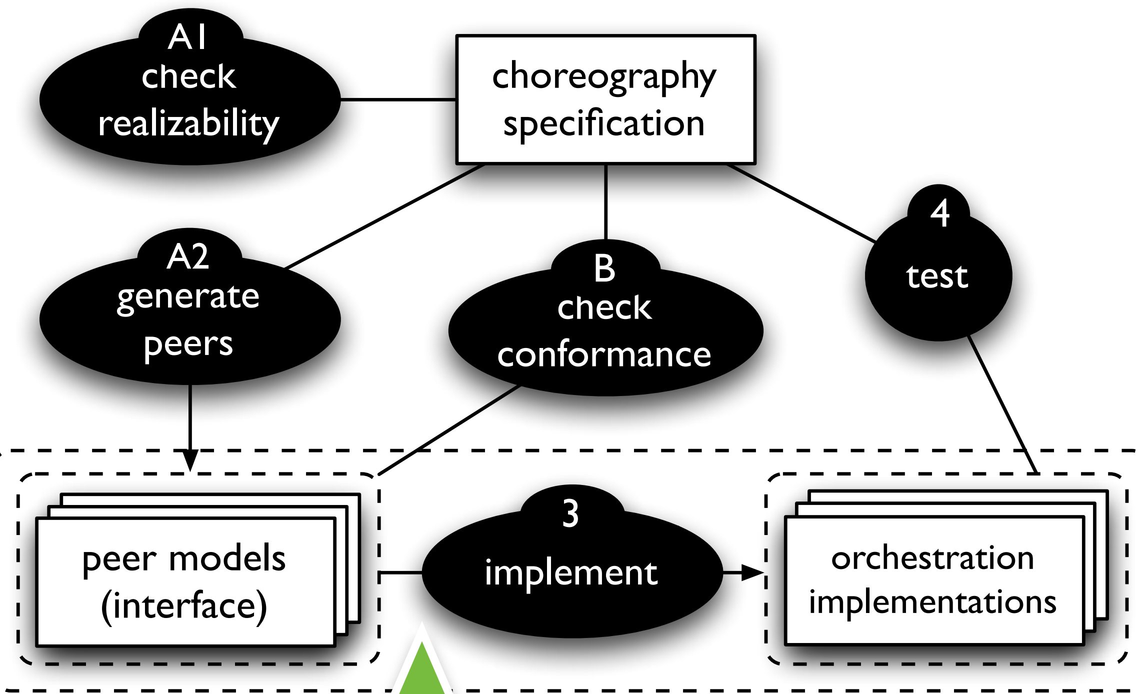




Development processes



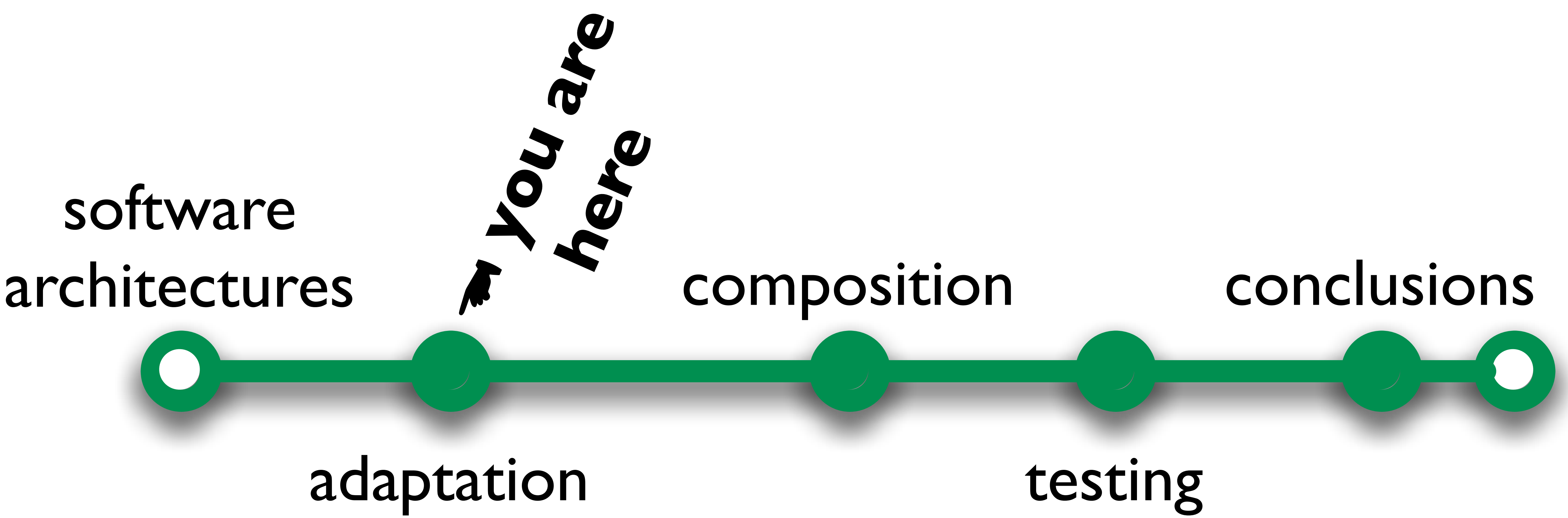
bottom-up centralized



top-down distributed

- both processes are based on **3 main activities**:
  - adaptation
  - composition
  - testing

Agenda


software architectures
you are here
adaptation
composition
testing
conclusions

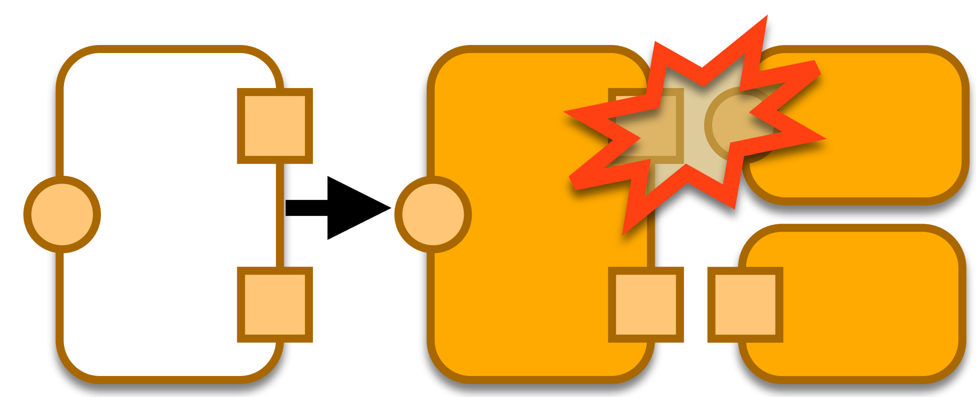
Issue

- **fact:** components are **developped separately** by different third parties

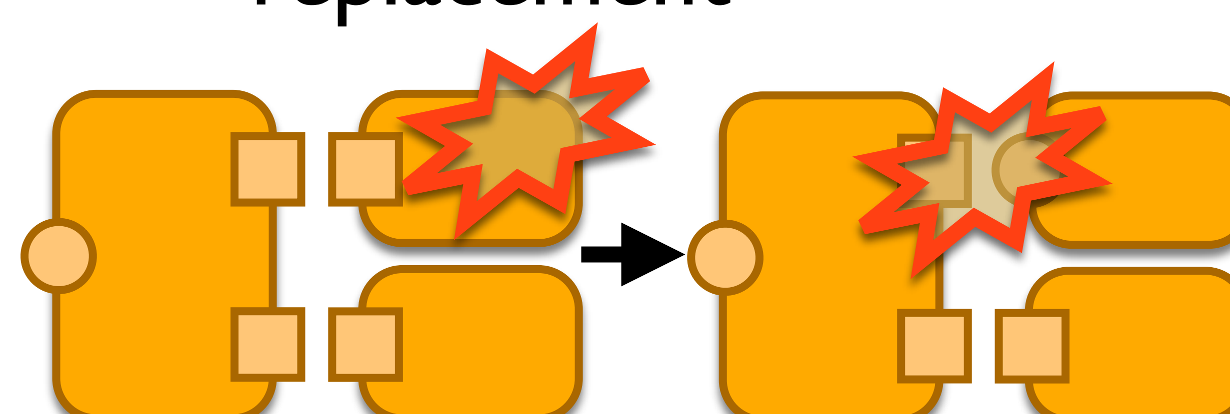
- **consequence: mismatch** between provided and required interfaces

- mismatch **prevents**:
  - reuse

  - composition



  - replacement



Mismatch

- mismatch **categories**:

  - name mismatch (1-1)
    print
    vs. imprimer

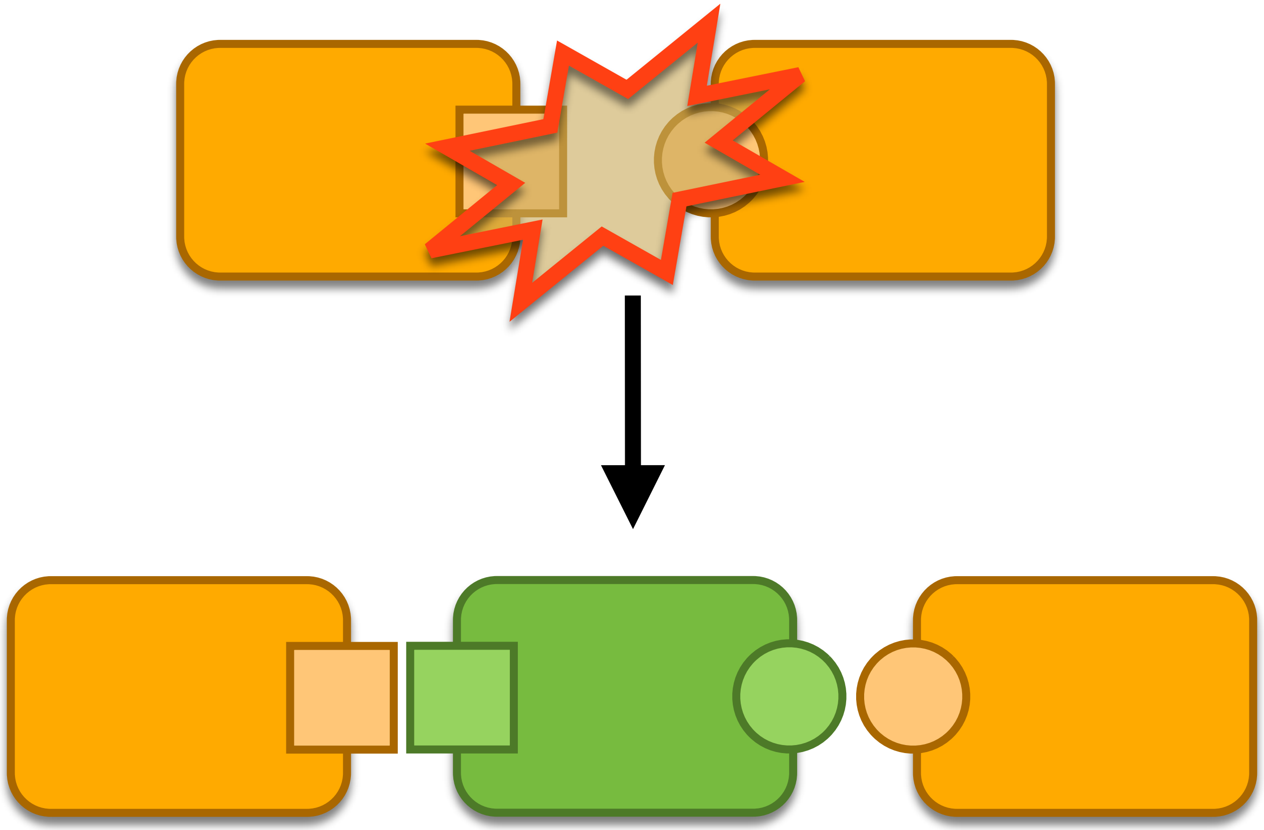
  - unspecified send/rec (0-1 / 1-0)
    login ; request* ; logout
    vs. (login ; request ; logout)*

  - generalized mismatch (n-m)
    res := query(x,y)
    vs. id := q1(x); res := q2(id,y)

  - reordering mismatch
    set(file); res := do(action)
    vs. set(action); res := do(file)

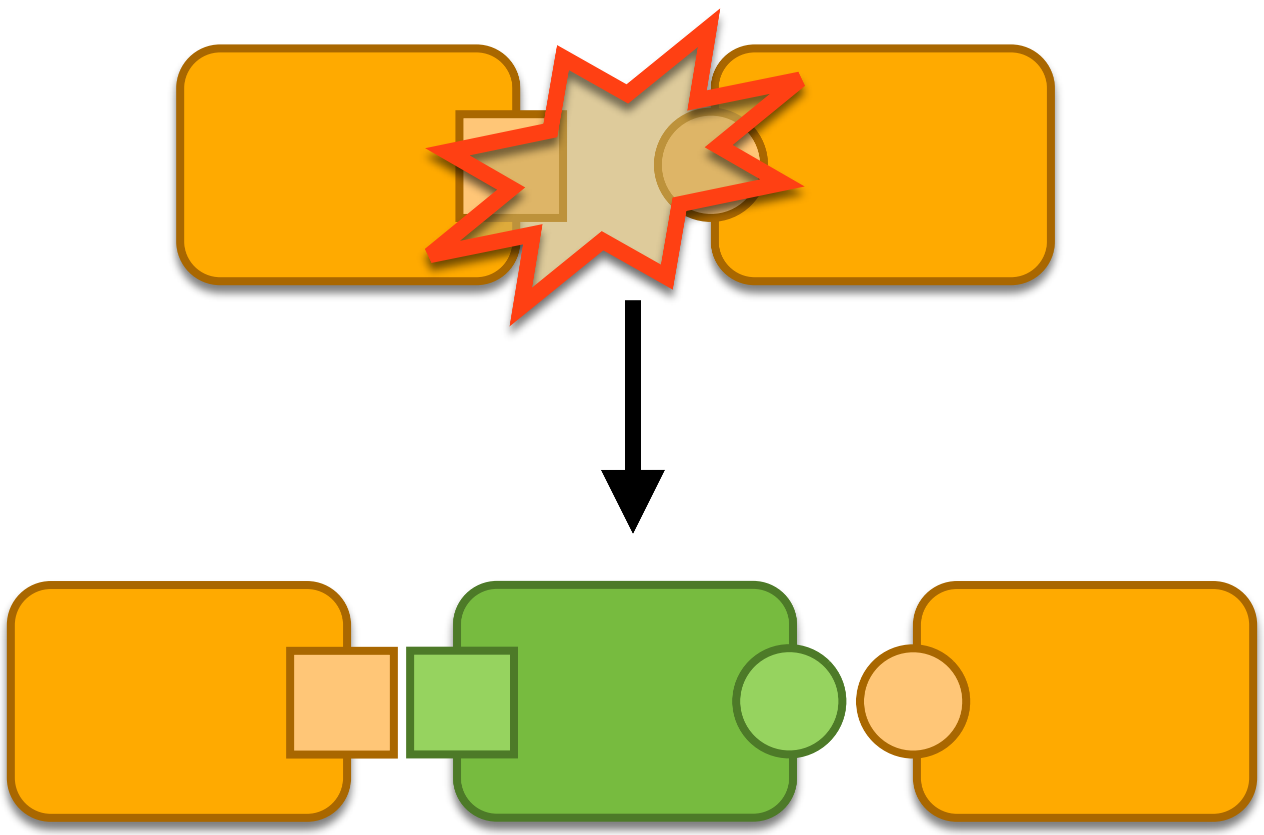
Adaptation

- addresses mismatch by generating **adaptors**



Adaptation

- addresses mismatch by generating **adaptors**



- is not **evolution**
  non-intrusiveness
- is not **customizing**
  no envisionned context
- is not **middleware**
  models/code generation rather than technical support
- is not **control synthesis**
  data & message **buffering**

Adaptation Approaches

- **restrictive** adaptation techniques
  prune interactions leading to deadlocks
  [Inverardi and Tivoli, 2003]
  +: fully automatic, **n-ary adaptation**, system **properties**
  -: support fewer adaptation scenario
- **generative** adaptation techniques
  rename and reorder exchanged messages
  [Yellin and Strom, 1997], [Bracciali et al, 2005], [Brogi et al, 2006]
  +: **extended mismatch support**
  -: generally purely theoretical and binary approaches
- **ad-hoc** adaptation techniques
  [Schmidt and Reussner, 2002], [Benatallah et al, 2005], [Dumas et al, 2006]
  -: specific mismatch (patterns/algebras), often manual

Approach: Contracts

- **adaptation contracts** are used to specify what an adaptor can do

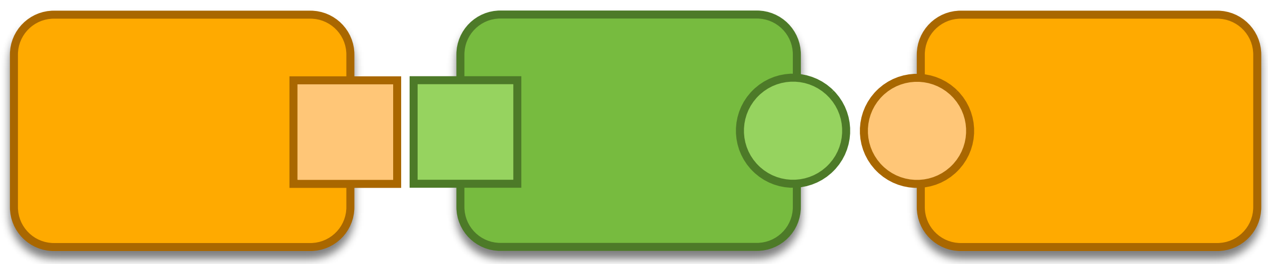
- possible correspondences
  ➠ **vectors**
  - absorbtion: <PDA:shutdown!; ROOM:_>
  - renaming: <PDA:pdf?; ROOM:text!>
  - generation: <PDA:_; ROOM:text_request?>

built on the component LTS alphabets + ‘_’

- possible orderings, simple form of system properties
  ➠ **vector-labelled transition systems**
  (v-LTS, LTS whose labels are vectors)

Approach: Constraints

- an adaptor acts **in-between** components

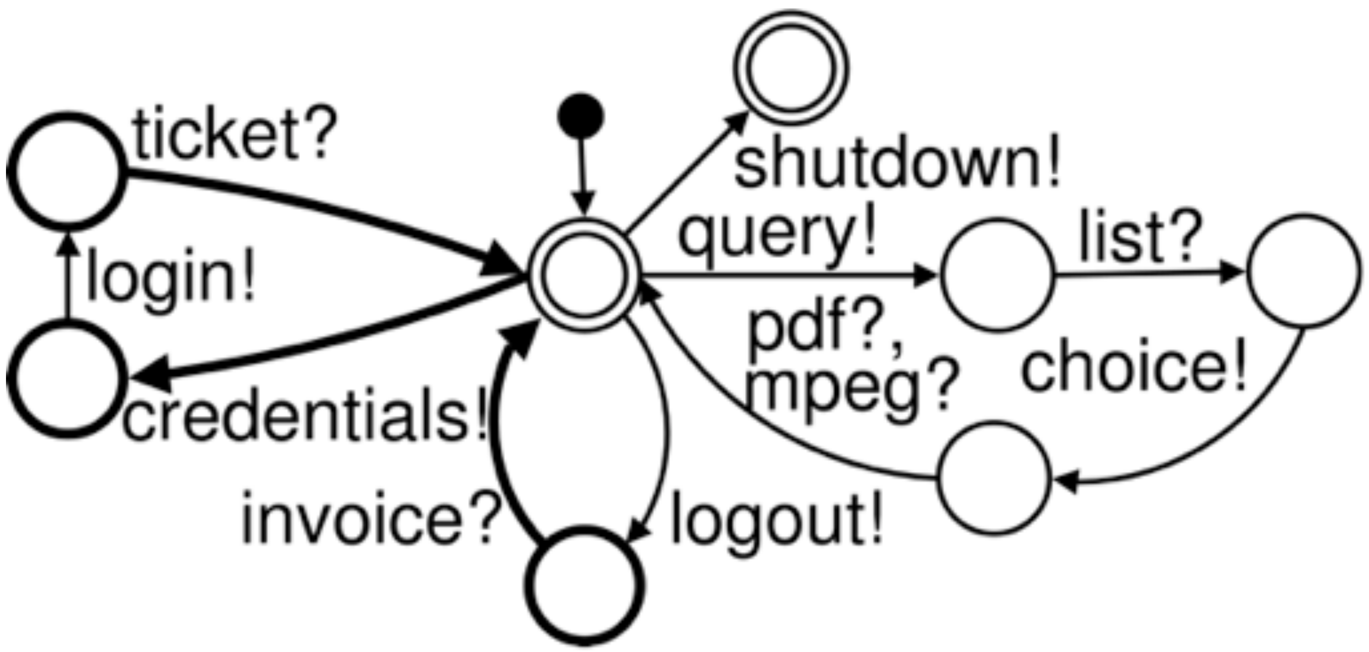


- it **respects the component behaviours**
  - ➠ event mirroring (sending ↔ reception)
- it **gets messages when sent**
  - ➠ messages are resources saved for later use
- it **sends messages when required**
  - ➠ use of owned resources

Petri nets are well-suited for this

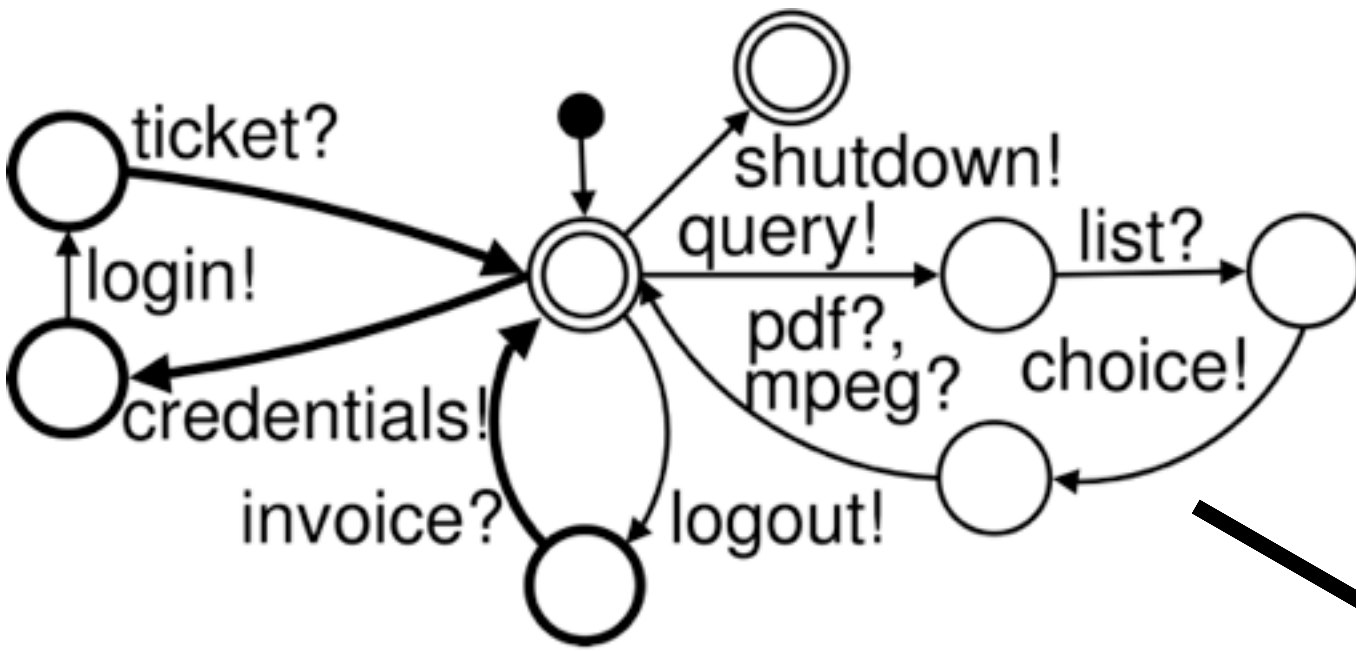
Approach: Technique (1/3)

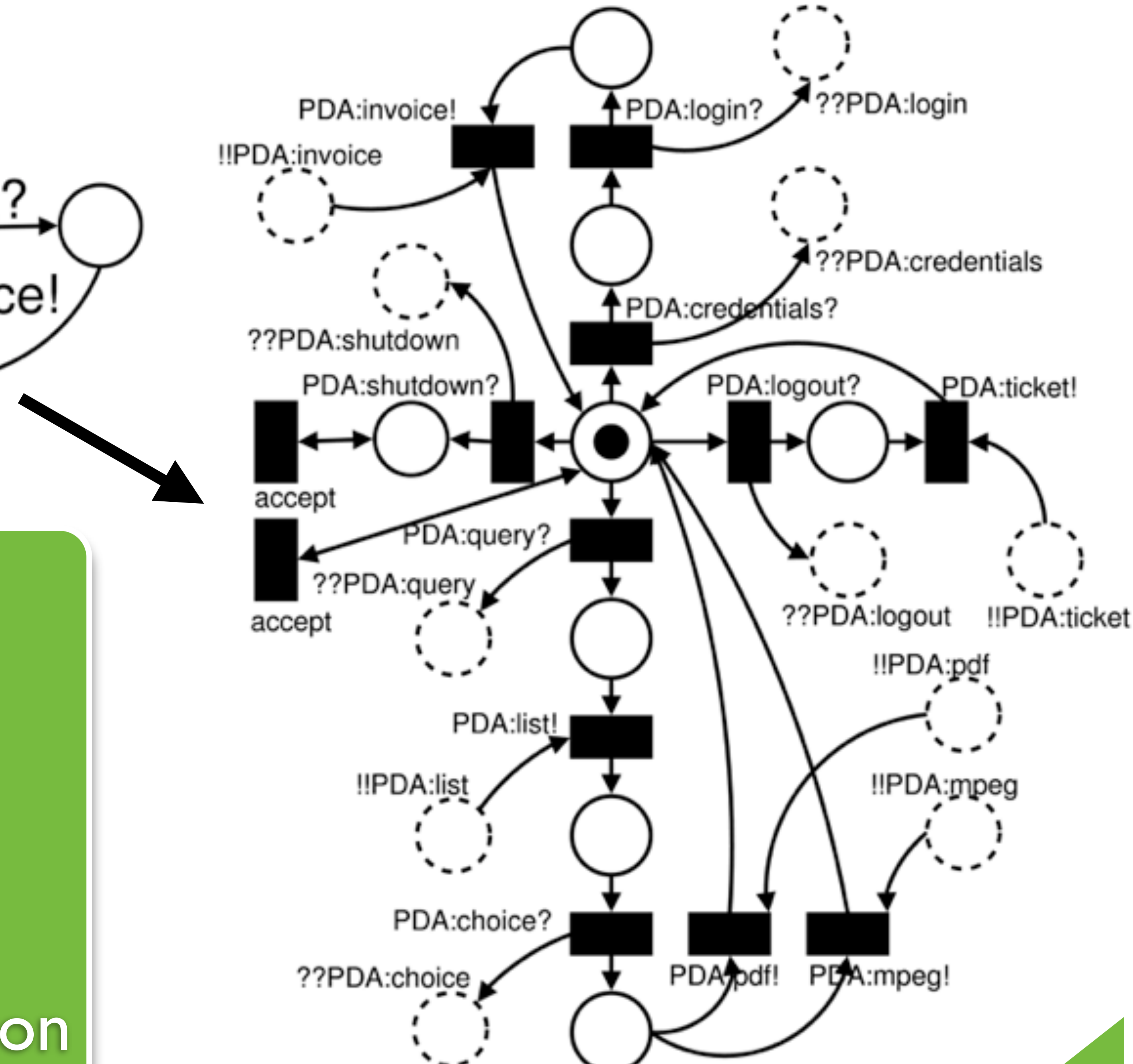
- transformation of component LTS models into Petri nets



Approach: Technique (1/3)

- transformation of component LTS models into Petri nets

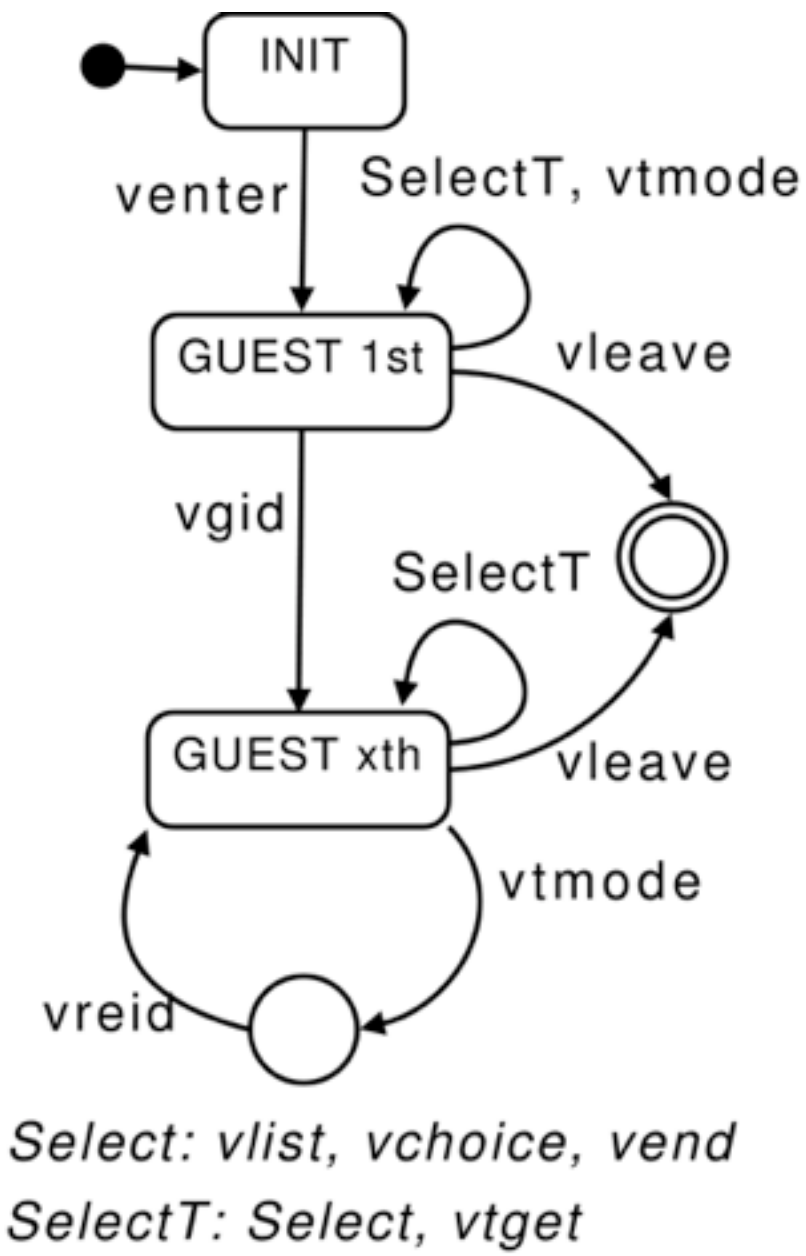




e! ➠ ??e place (store)
e? transition (mirror)
e? ➠ !!e place (store)
e! transition (mirror)
initial state ➠ token
final state ➠ accept transition

Approach: Technique (2/3)

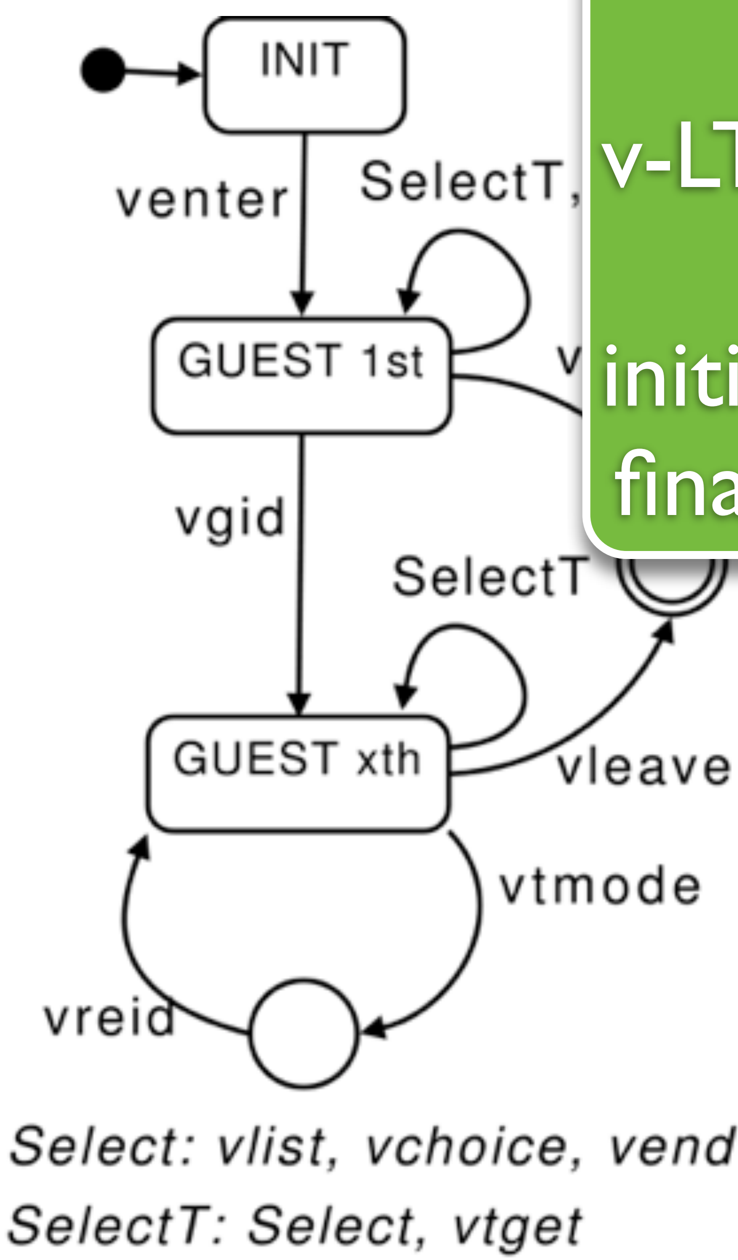
- transformation of the v-LTS contract into Petri nets

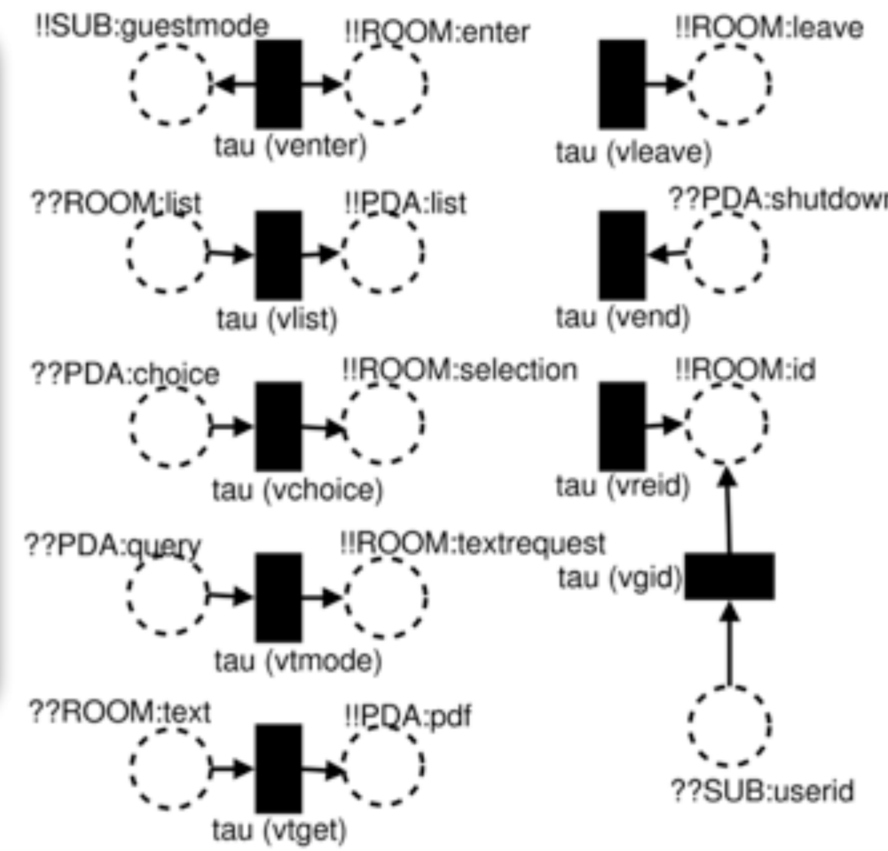


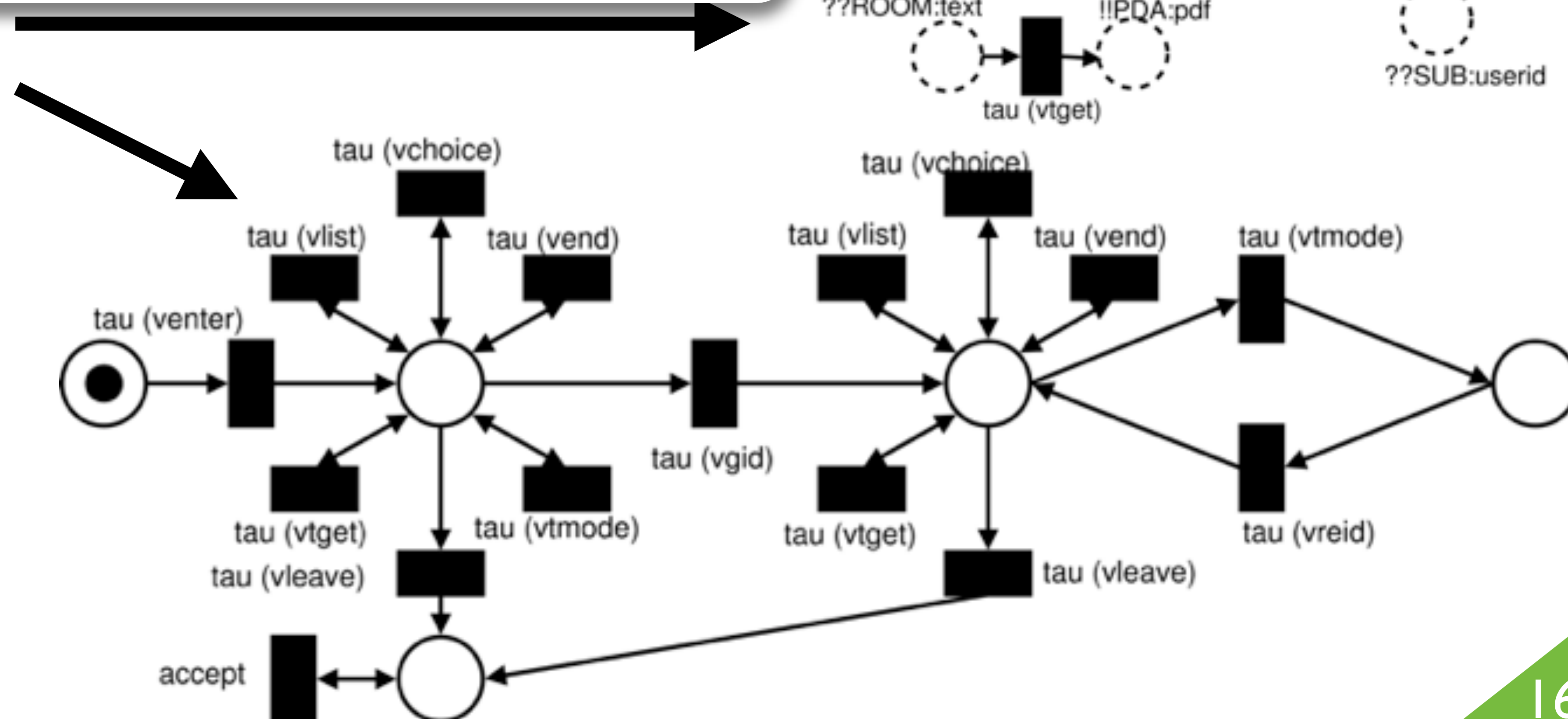
Approach: Technique (2/3)

- transformation of the v-LTS contract into Petri nets

vector ➠ tau transition (transfer)
v-LTS ➠ ordering of tau transitions
initial state ➠ token
final state ➠ accept transition



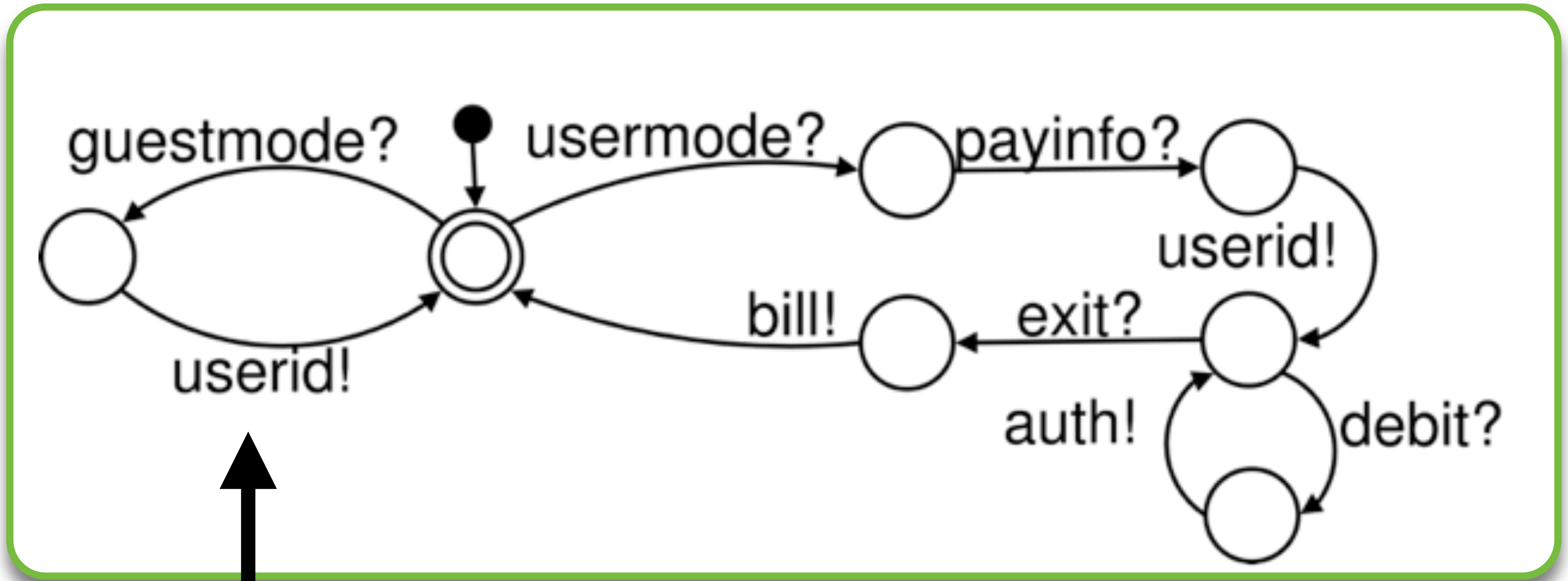


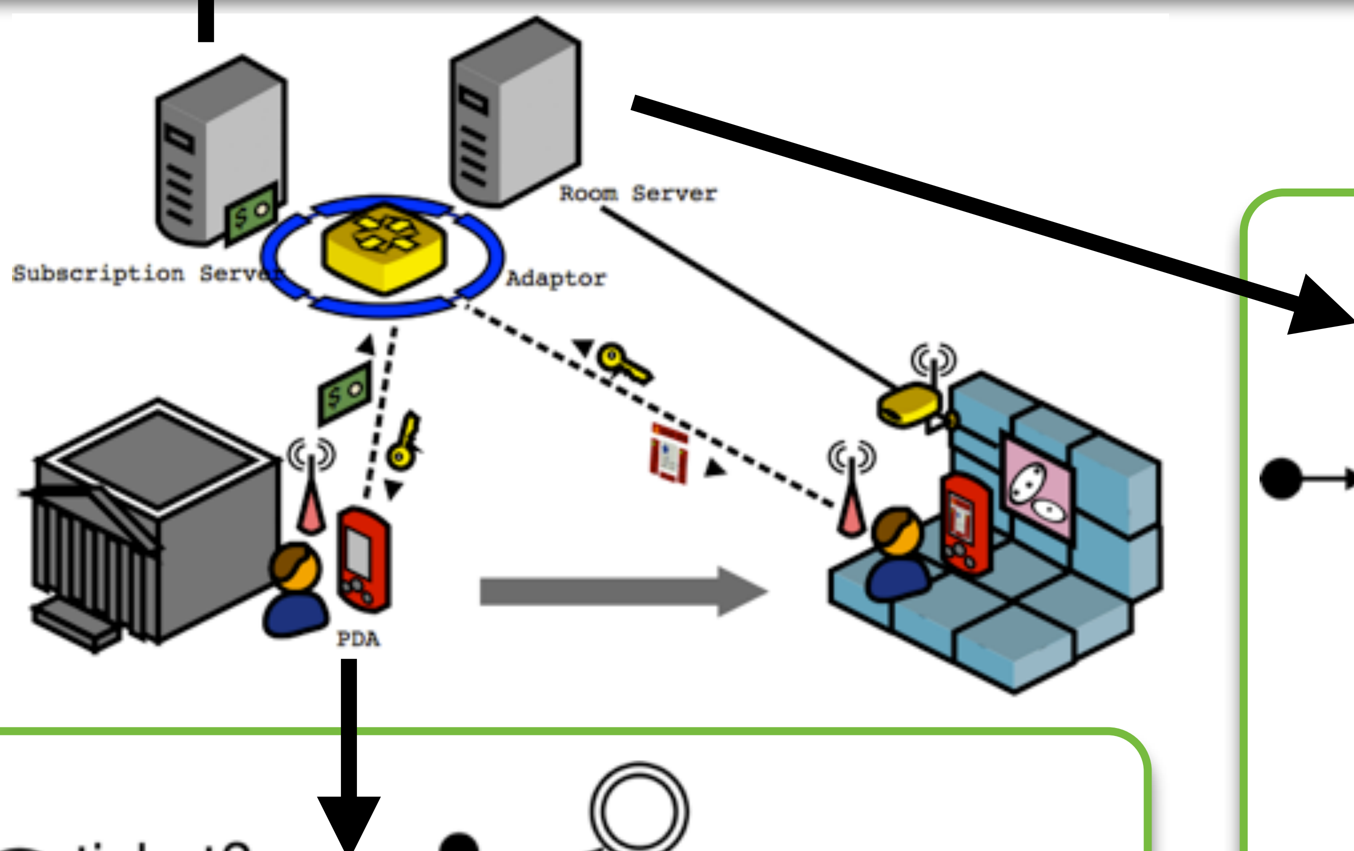


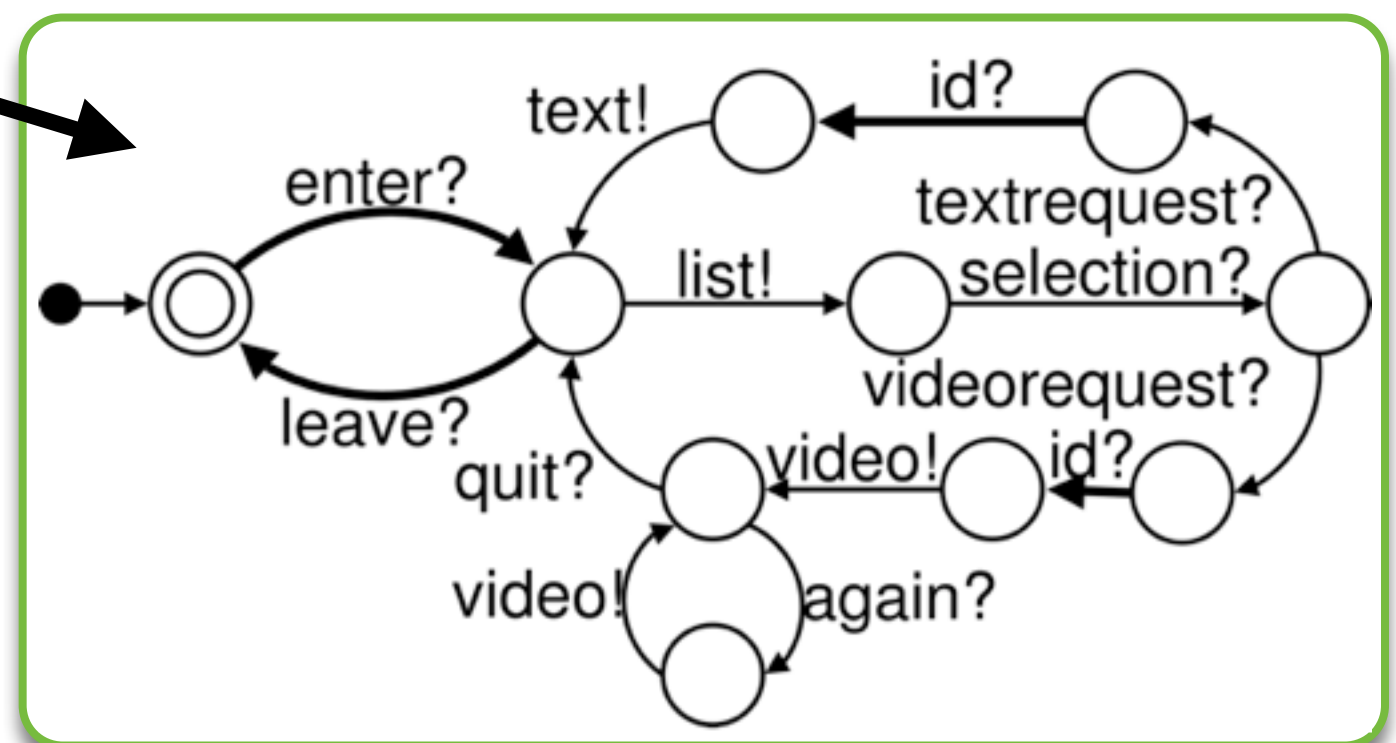
Approach: Technique (3/3)

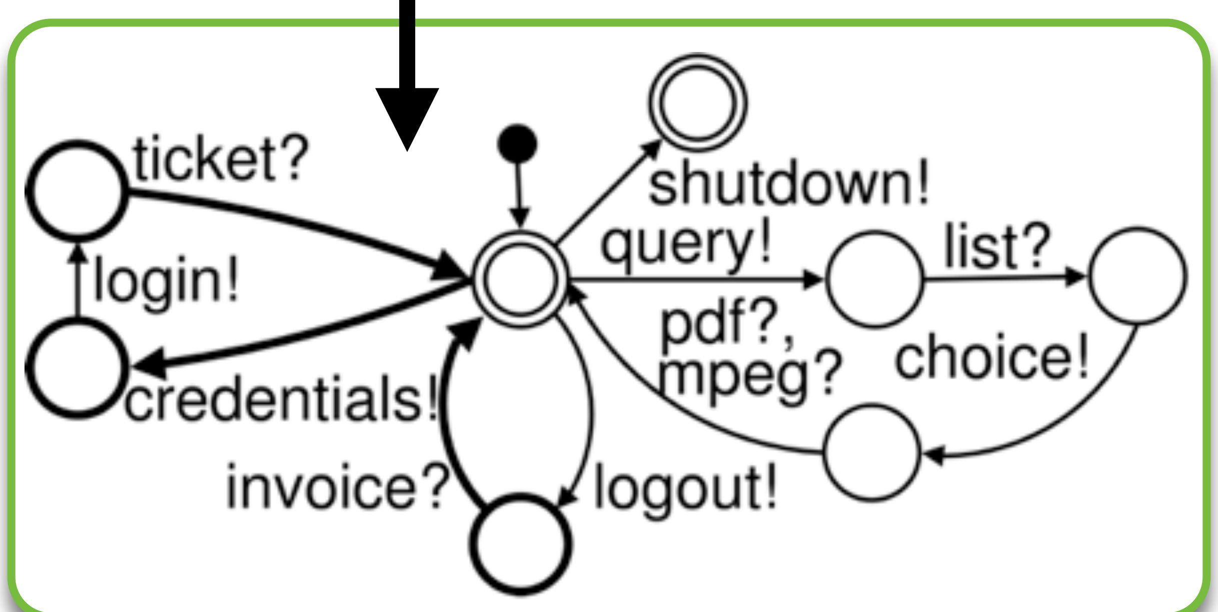
- **fusion** of the Petri nets on shared places
  messages are resources produced/consumed
- computation of the Petri net **marking graph**
  all possible interactions of adapted system
- **pruning** paths leading to deadlocks
  following *[Inverardi and Tivoli, 2003]*
- behavioural **reduction** to remove internal transitions
  introduced by adaptation basic steps (vectors)

eMuseum



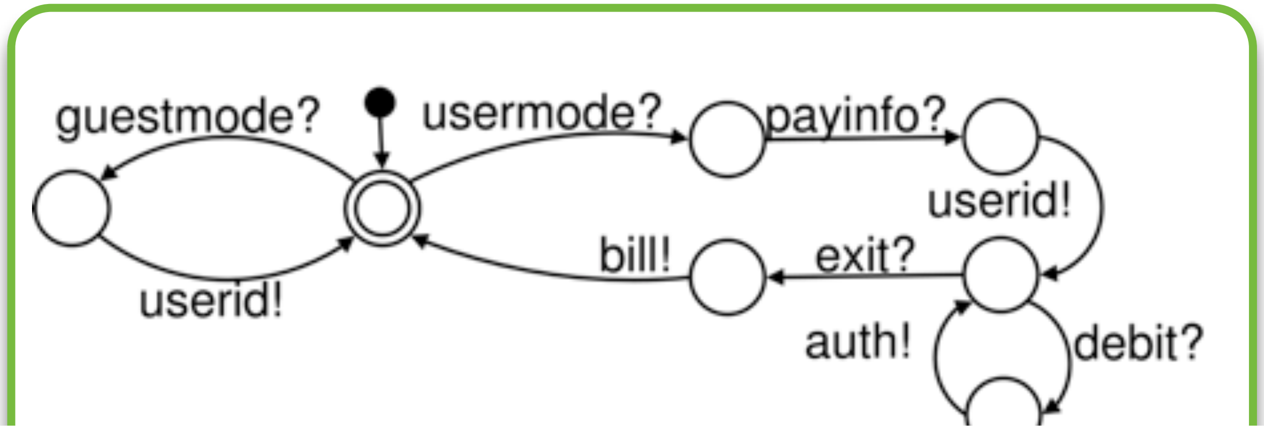


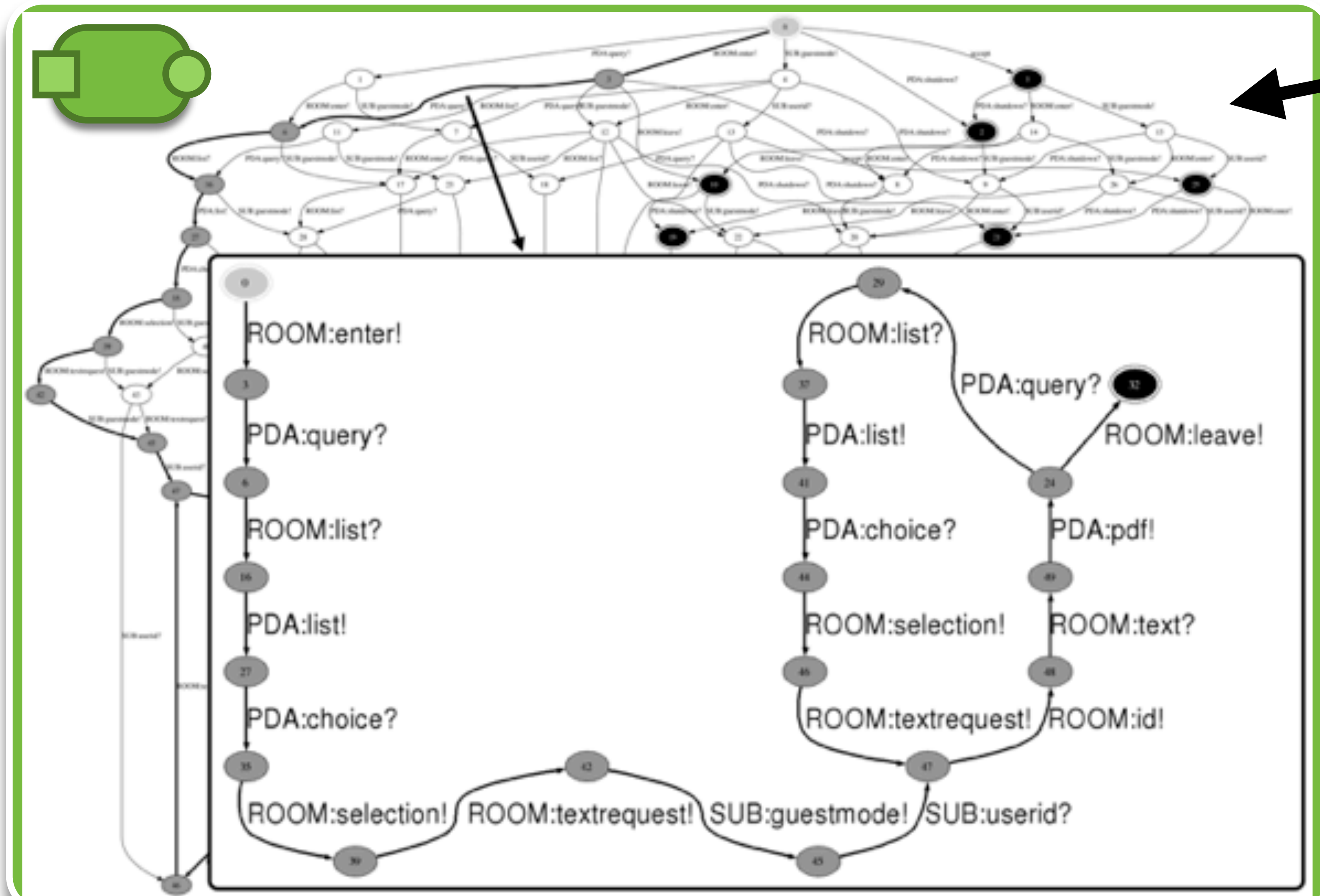


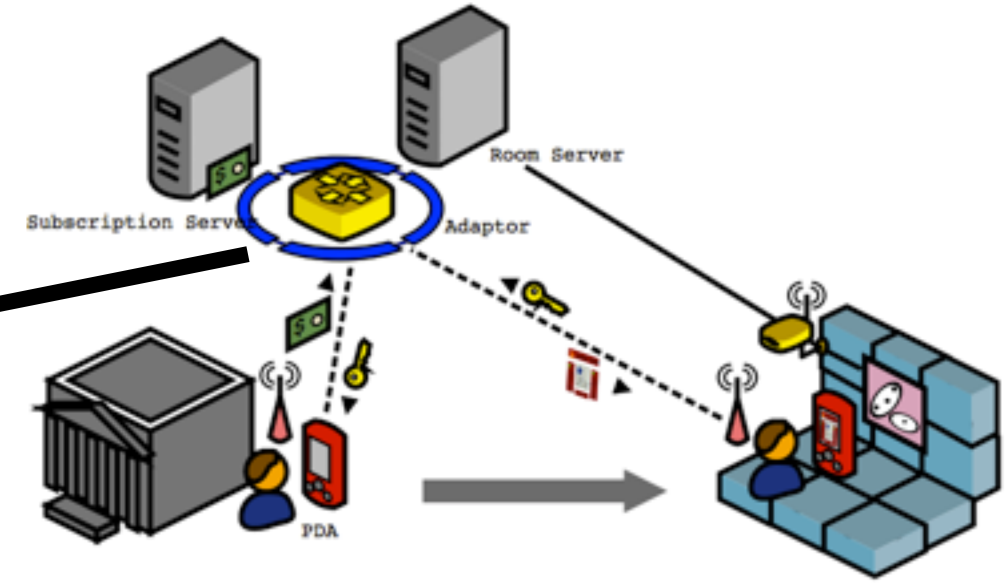


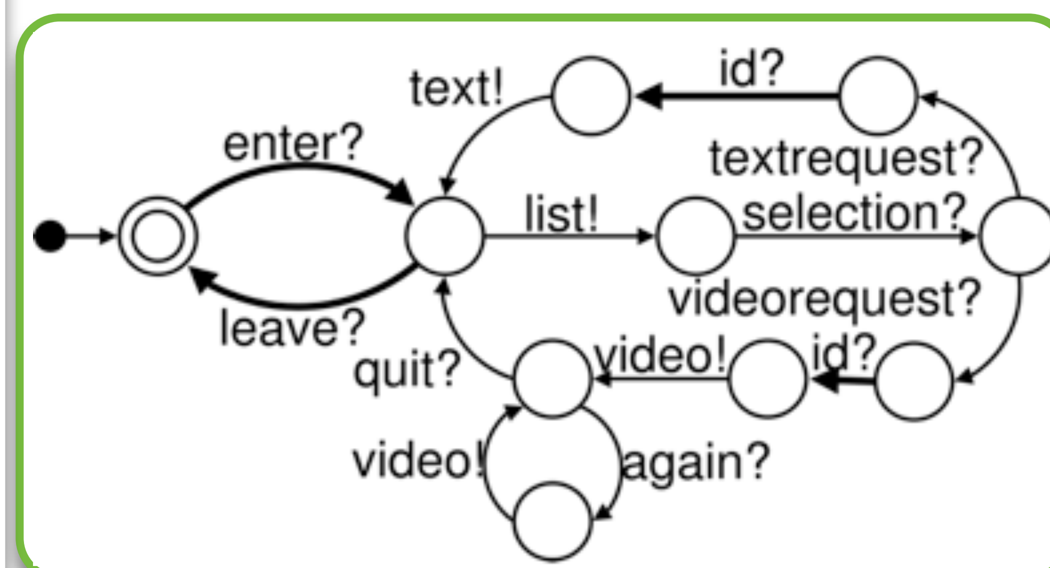
this does not work

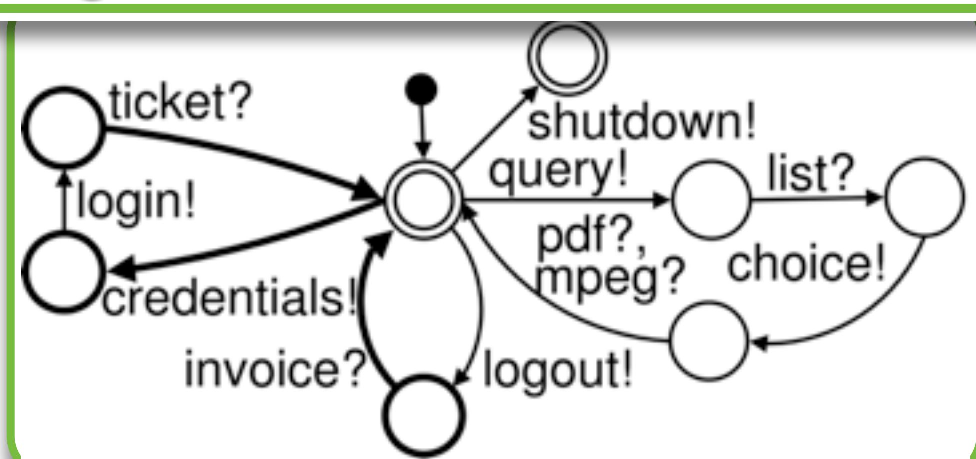
eMuseum









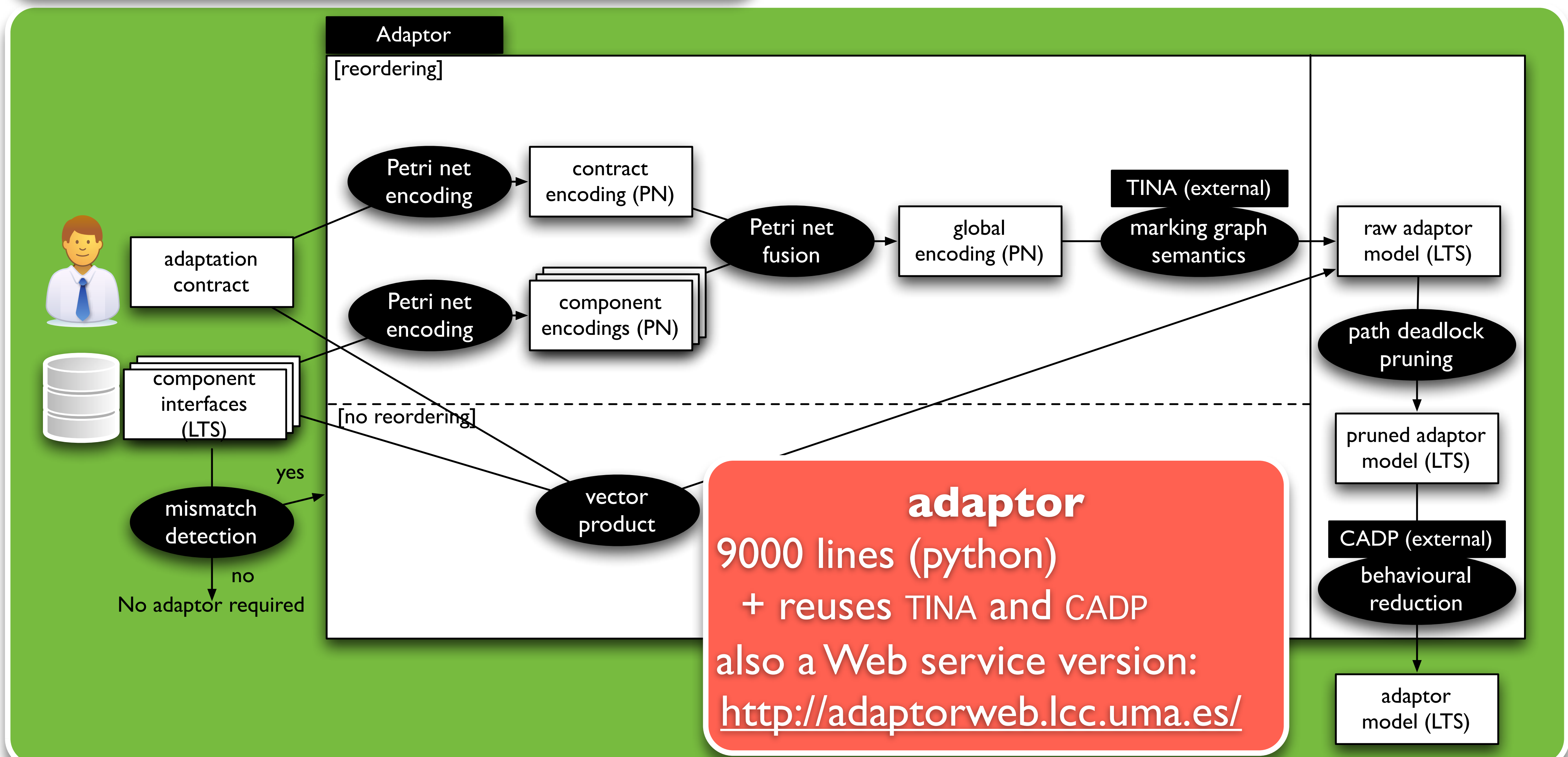


this works

Contributions on Adaptation

restrictive **and** generative
n-ary
simple **properties**
application to **WWF**

FMOODS'06, WCOP'07
IEEE TSE 34(4), 2008



adaptor
9000 lines (python)
+ reuses TINA and CADP
also a Web service version:
http://adaptorweb.lcc.uma.es/

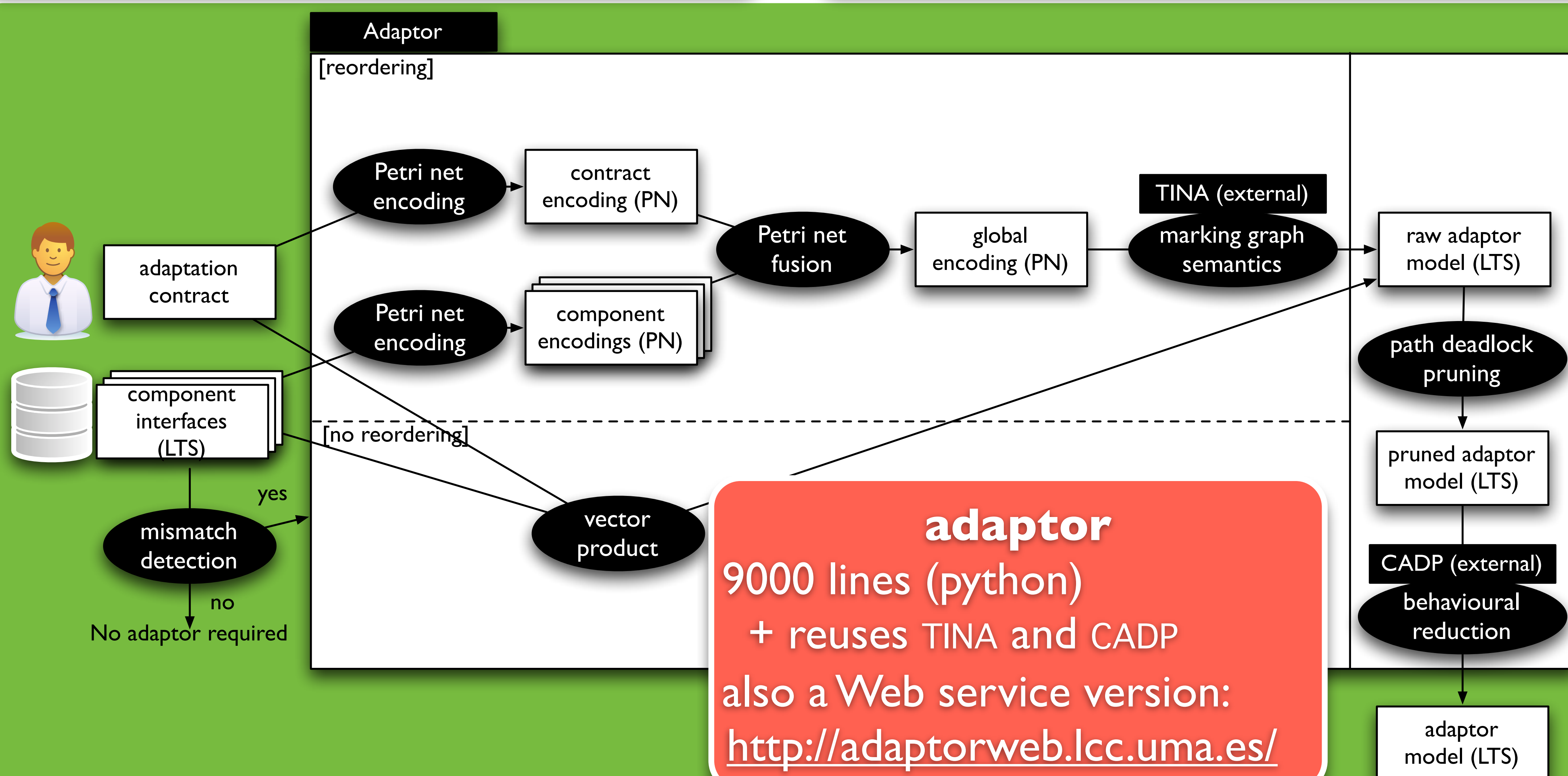
Contributions on Adaptation

restrictive **and** generative
n-ary
simple **properties**
application to **WWF**

FMOODS'06, WCOP'07
IEEE TSE 34(4), 2008

issues

pruning and reduction on **complete** state space
data is **not directly supported**
application to WWF is **partly manual**



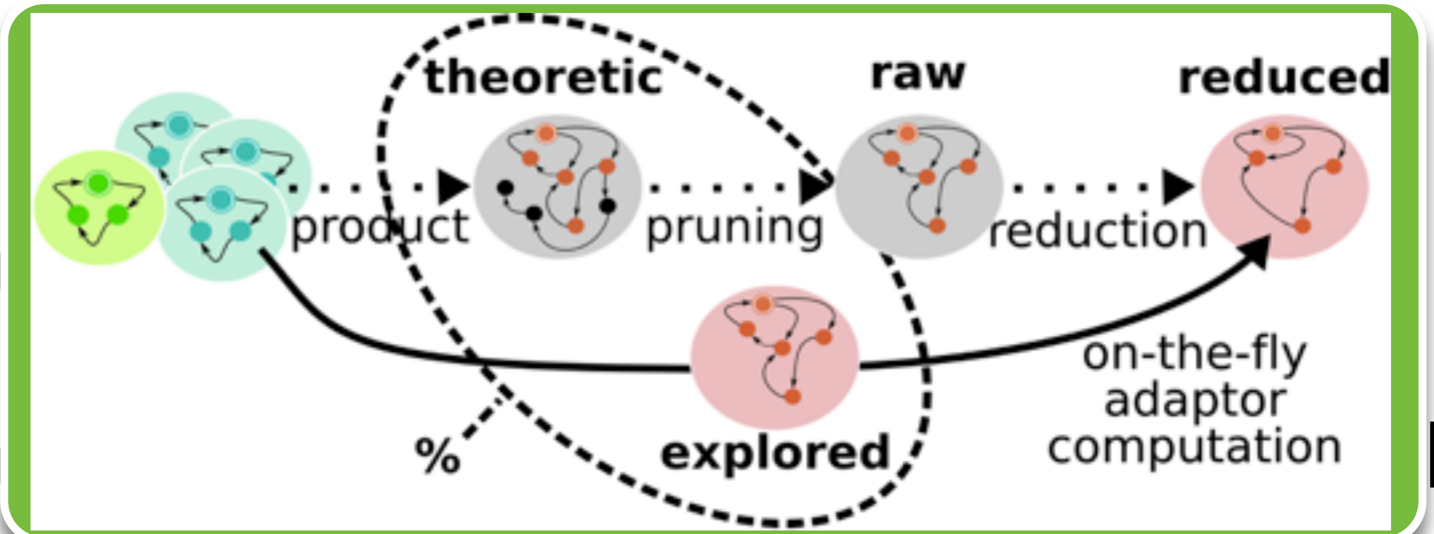
adaptor
9000 lines (python)
+ reuses TINA and CADP
also a Web service version:
http://adaptorweb.lcc.uma.es/

Extensions

- behavioural interfaces and vectors are **extended** to support data (LTS ➠ STS)
- implicit model encoding: **STS network**
- **deadlock-freeness** $\mu X . \langle \text{ FINAL } \rangle \, true \vee \langle \, true \, \rangle \, X$ encoded as a boolean equation system (BES)
- use of the Caesar.Solve library to perform pruning by **solving the BES on-the-fly** on the states of the implicit model
- model **extraction** (BPEL ➠ STS)
  model **implementation** (STS ➠ BPEL)
  - filtering to remove unimplementable parts
  - state machine pattern

Extensions

- behavioural interfaces and vectors are **extended**



- **network**
- FINAL $\rangle\ true \vee \langle\ true\ \rangle X$

Application	Adaptor LTS				State space portion explored for reduced adaptor generation			
	raw		reduced					
	states	trans.	states	trans.	states	%	trans.	%
eMuseum	21418	48692	978	2382	29026	72.8	17075	18.7
music-system	1720	4368	49	60	14805	85.9	32923	74.5
sql-server	1720	4264	22	26	2337	57.1	3427	32.9
multi-file query	1,542	3,709	61	79	6,269	99.95	11,623	69.76
mail-system	418	1059	418	1059	13630	99.7	23946	70.1
pc-store	253	472	16	16	782	88.2	1208	66.8
rate-service	241	483	28	32	400	52.6	675	37.2
video-on-demand	149	231	17	22	251	97.6	260	63.5
batchsql	137	239	31	43	429	67.1	276	21.6
restau-booking	94	108	33	37	264	99.6	280	83.1
pc-store	17	17	17	17	237	91.5	249	64.3

- state machine pattern

Contributions on Adaptation

restrictive **and** generative
n-ary
simple **properties**
application to **WWF**

FMOODS'06, WCOP'07
IEEE TSE 34(4), 2008

Contributions on Adaptation

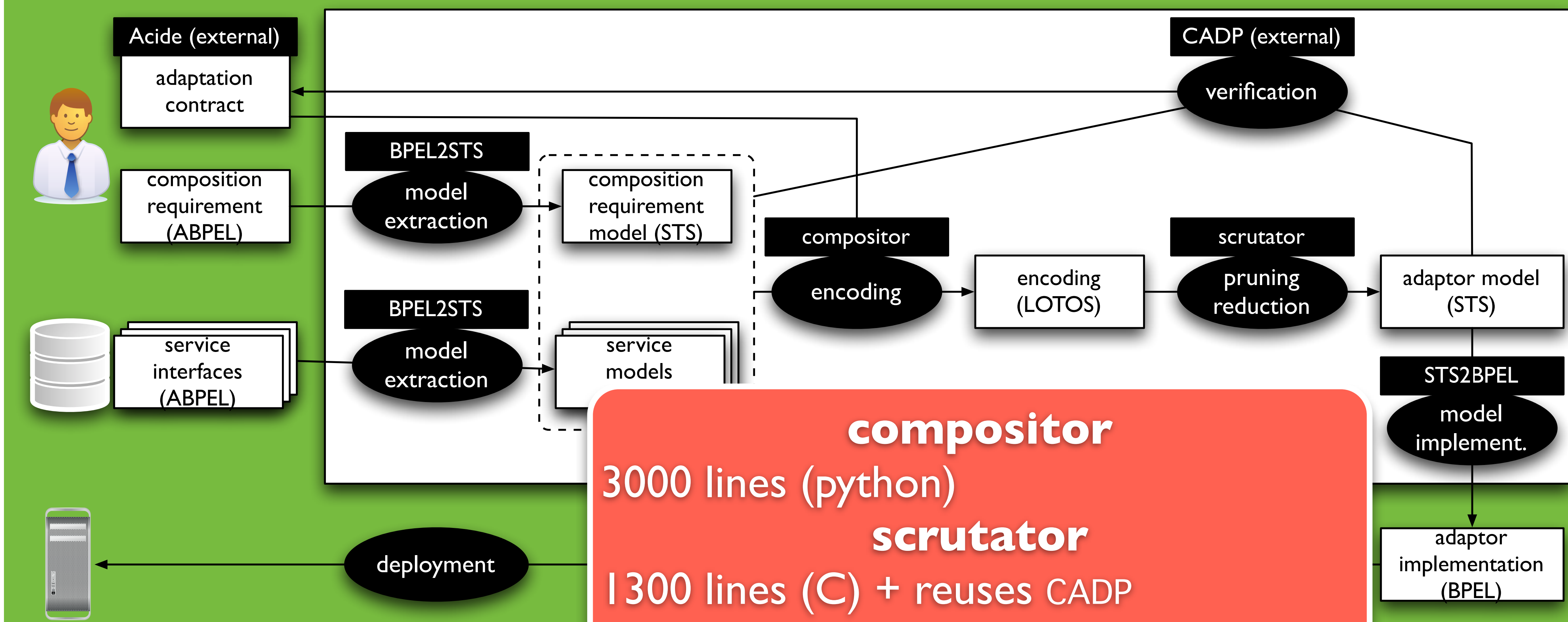
restrictive **and** generative
n-ary
simple **properties**
application to **WWF**

FMOODS'06, WCOP'07
IEEE TSE 34(4), 2008

→

on-the-fly approach
direct support for **data**
verification of adaptation properties
(general, application-specific, controllability)
application to **WS**

ICSOC'08
IEEE TSE under press



compositor
3000 lines (python)
scrutator
1300 lines (C) + reuses CADP
BPEL2STS / STS2BPEL
1500 lines (python) / 4600 lines (Java)

Contributions on Adaptation

restrictive **and** generative
n-ary
simple **properties**
application to **WWF**

FMOODS'06, WCOP'07
IEEE TSE 34(4), 2008

on-the-fly approach
direct support for **data**
verification of adaptation properties
(general, application-specific, controllability)
application to **WS**

ICSOC'08
IEEE TSE under press

open systems, **local** adaptors
incremental adaptation

FACS'06, FMOODS'07

Contributions on Adaptation

restrictive **and** generative
n-ary
simple **properties**
application to **WWF**

FMOODS'06, WCOP'07
IEEE TSE 34(4), 2008

on-the-fly approach
direct support for **data**
verification of adaptation properties
(general, application-specific, controllability)
application to **WS**

ICSOC'08
IEEE TSE under press

open systems, **local** adaptors
incremental adaptation

FACS'06, FMOODS'07

distributed approach
fully-automatic (no contract)
application to semantic WS

FASE'08

Contributions on Adaptation

restrictive **and** generative
n-ary
simple **properties**
application to **WWF**

FMOODS'06, WCOP'07
IEEE TSE 34(4), 2008

on-the-fly approach
direct support for **data**
verification of adaptation properties
(general, application-specific, controllability)
application to **WS**

ICSOC'08
IEEE TSE under press

open systems, **local** adaptors
incremental adaptation

FACS'06, FMOODS'07

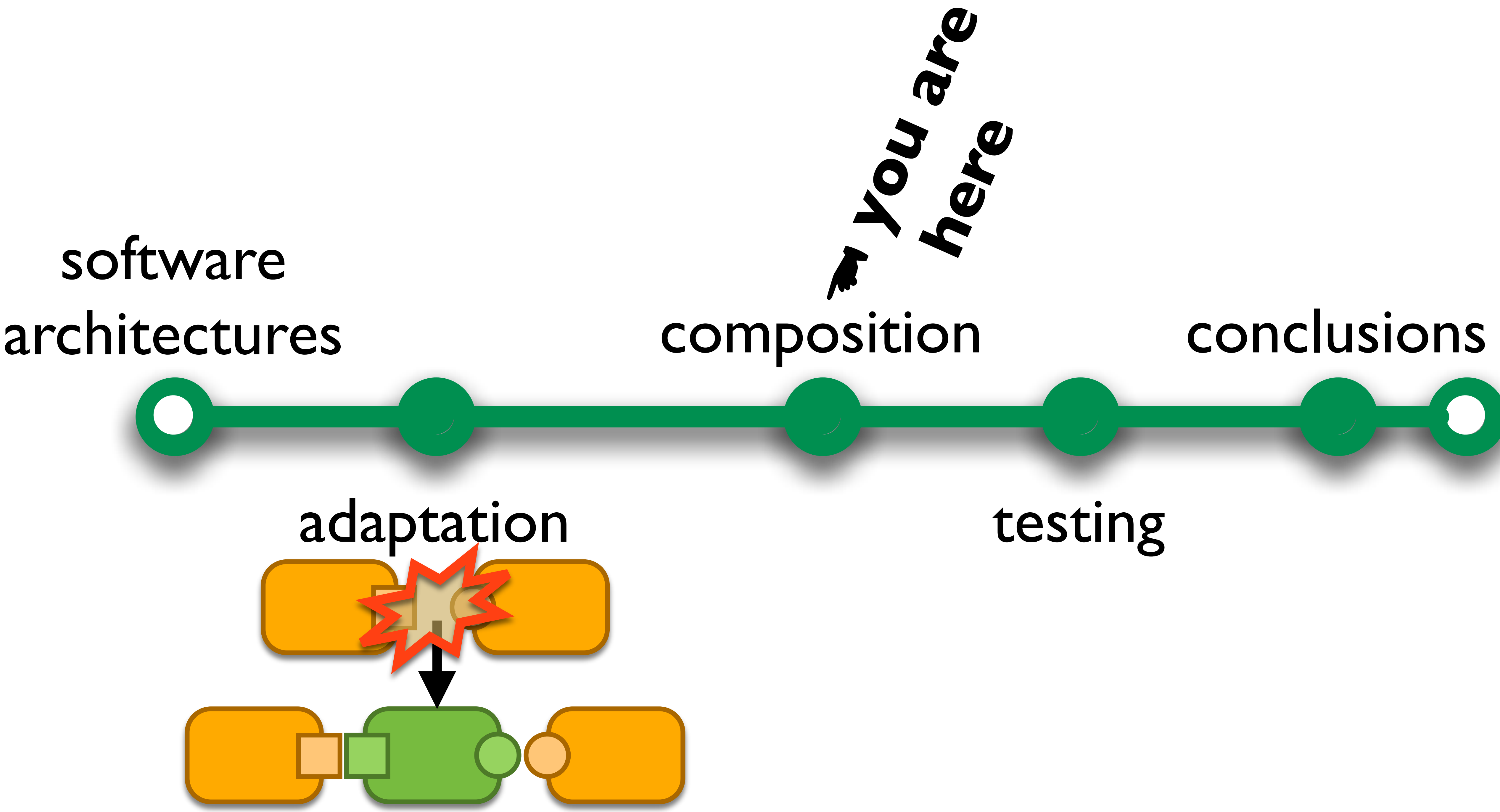
behavioural reduction
transactional equivalence

FORTE'07

distributed approach
fully-automatic (no contract)
application to semantic WS

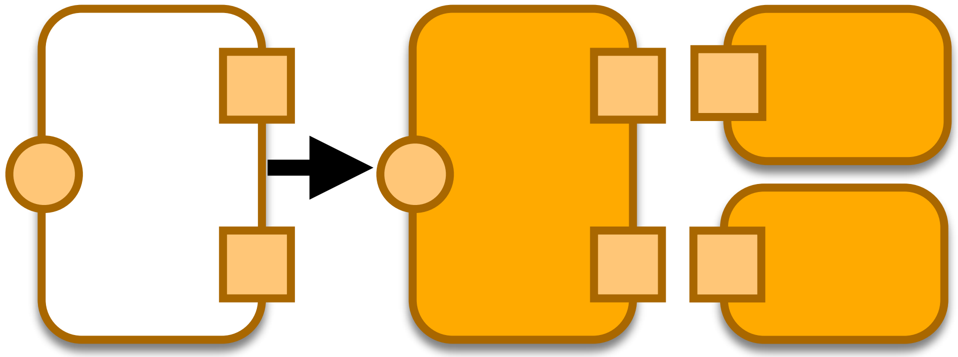
FASE'08

Agenda

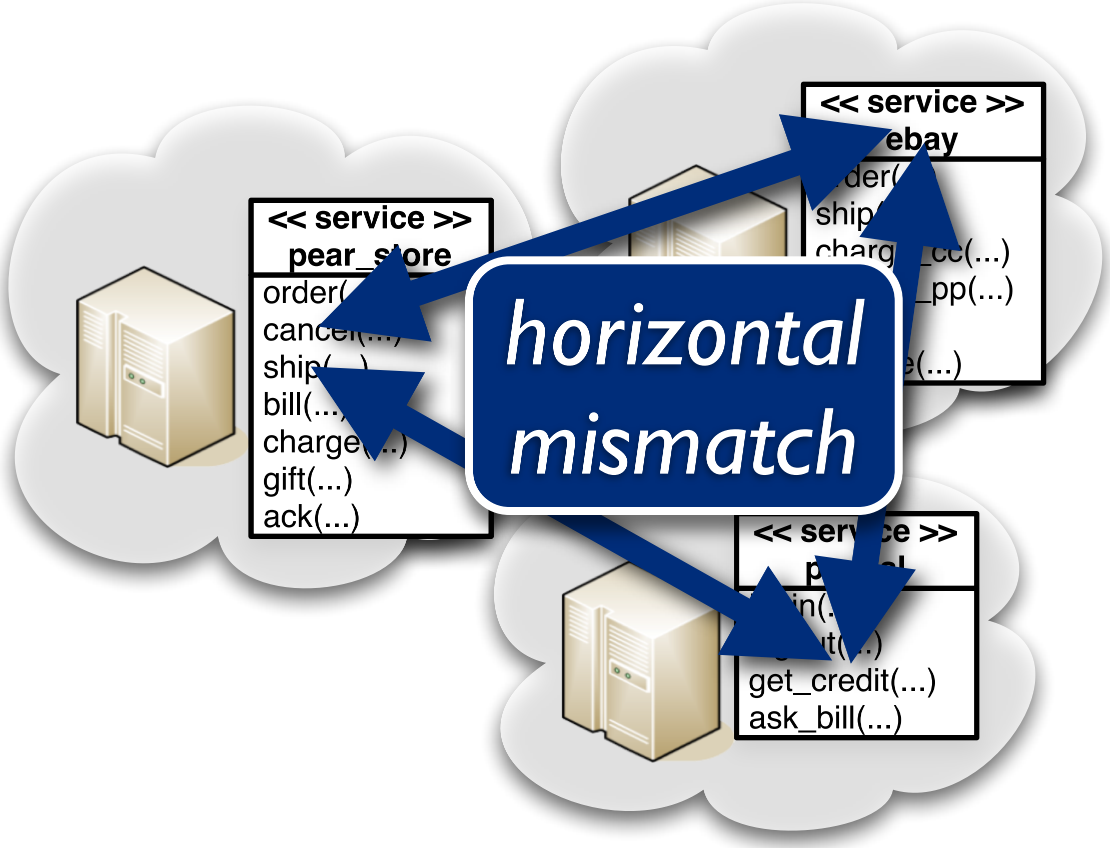

you are here
software architectures
composition
conclusions
adaptation
testing

Issue

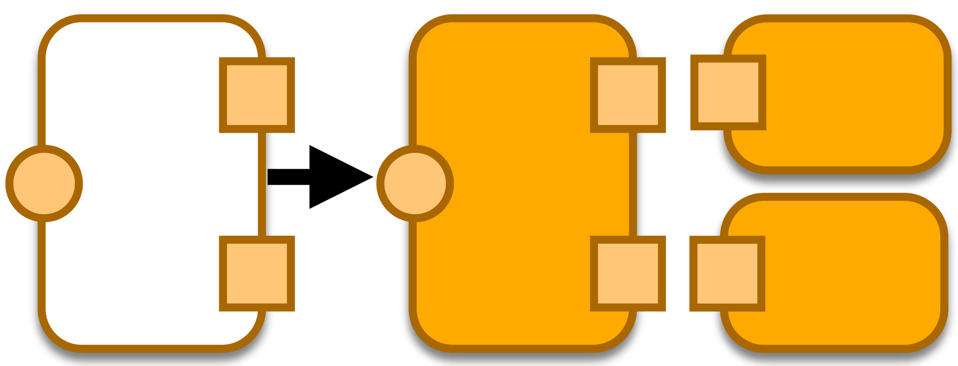
- going **beyond adaptation** and mismatch bridging
- composing **automatically** services from **requirements** both with **conversations**



Issue

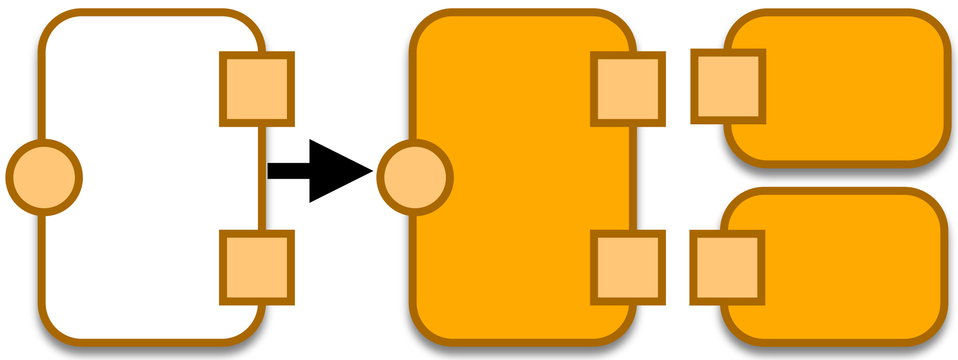


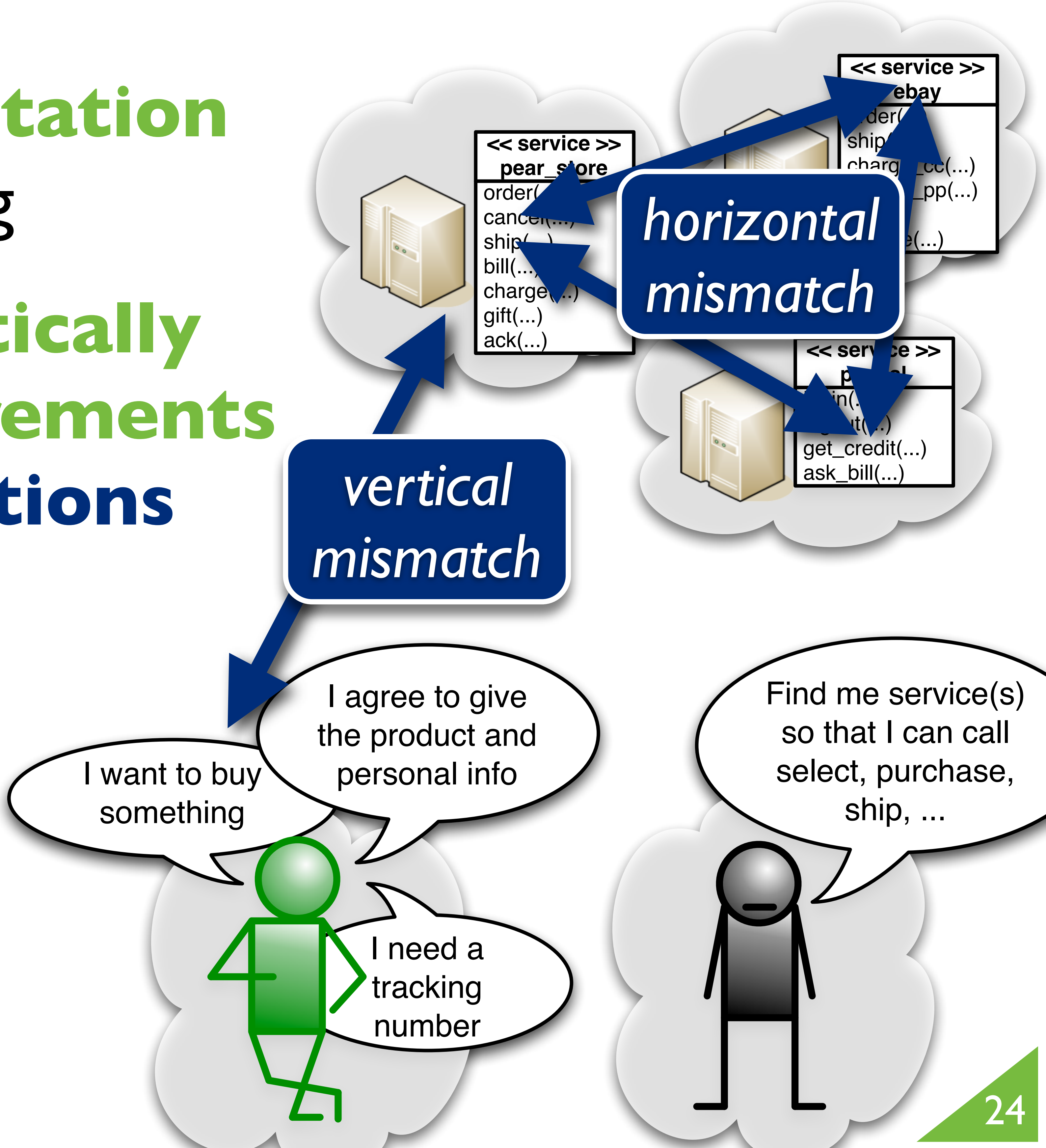
- going **beyond adaptation** and mismatch bridging
- composing **automatically** services from **requirements** both with **conversations**



Issue

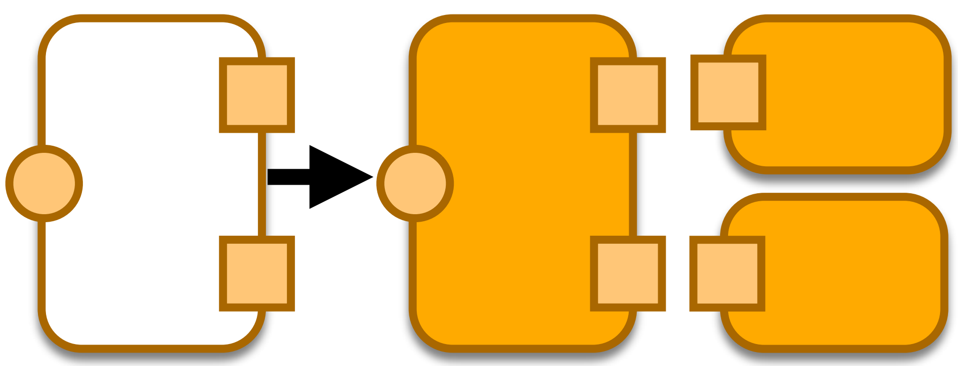
- going **beyond adaptation** and mismatch bridging
- composing **automatically** services from **requirements** both with **conversations**

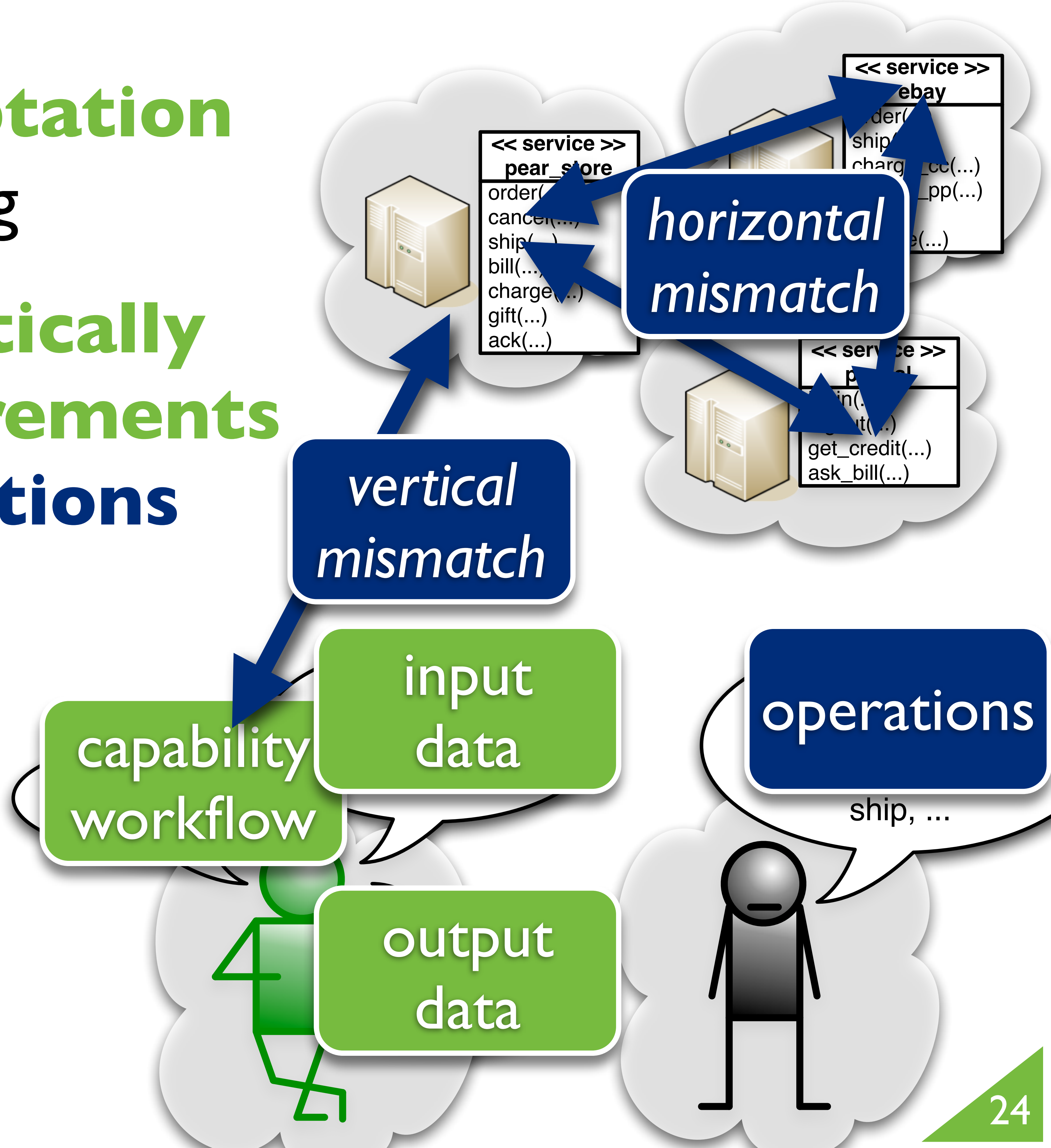




Issue

- going **beyond adaptation** and mismatch bridging
- composing **automatically** services from **requirements** both with **conversations**

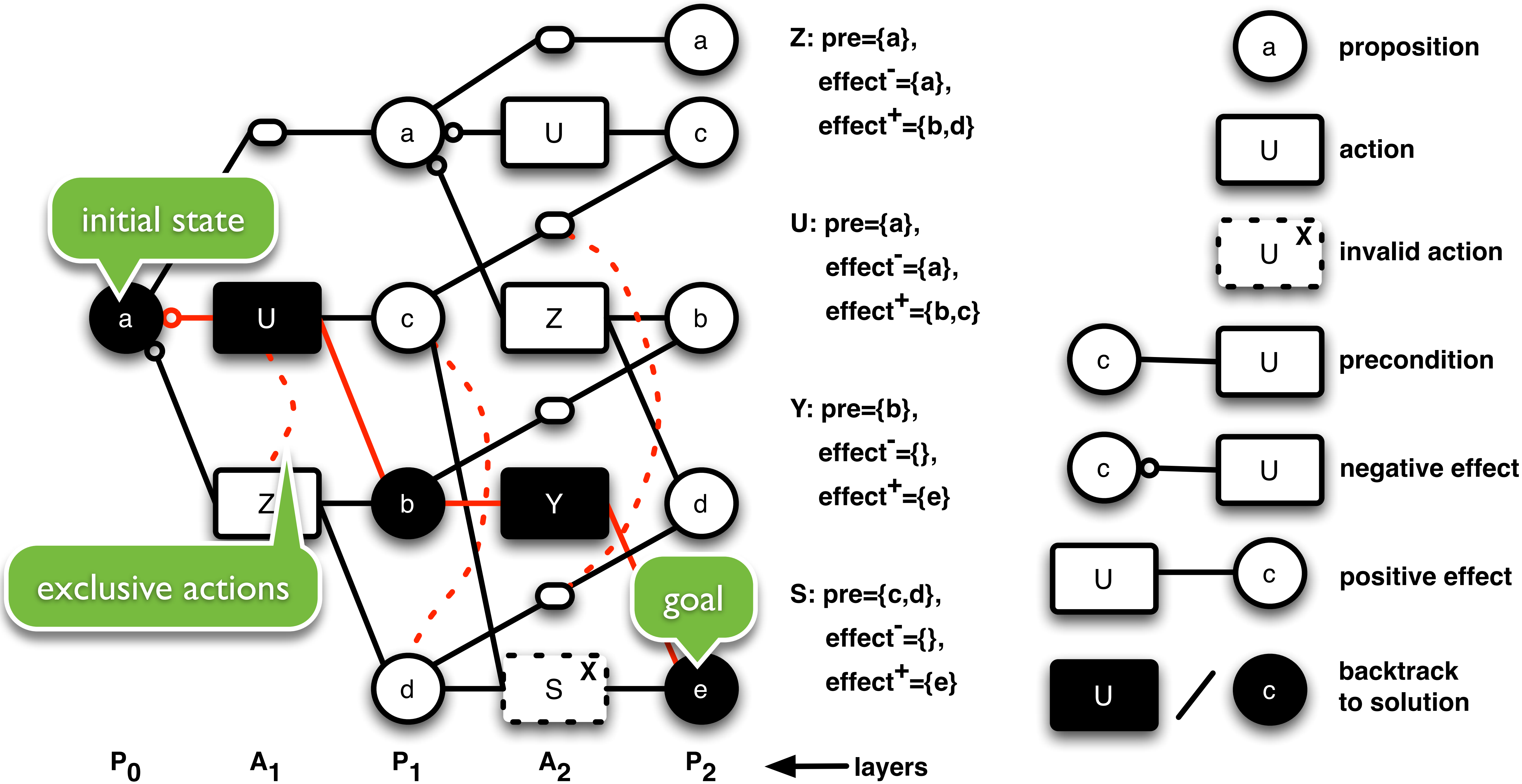




Composition Approaches

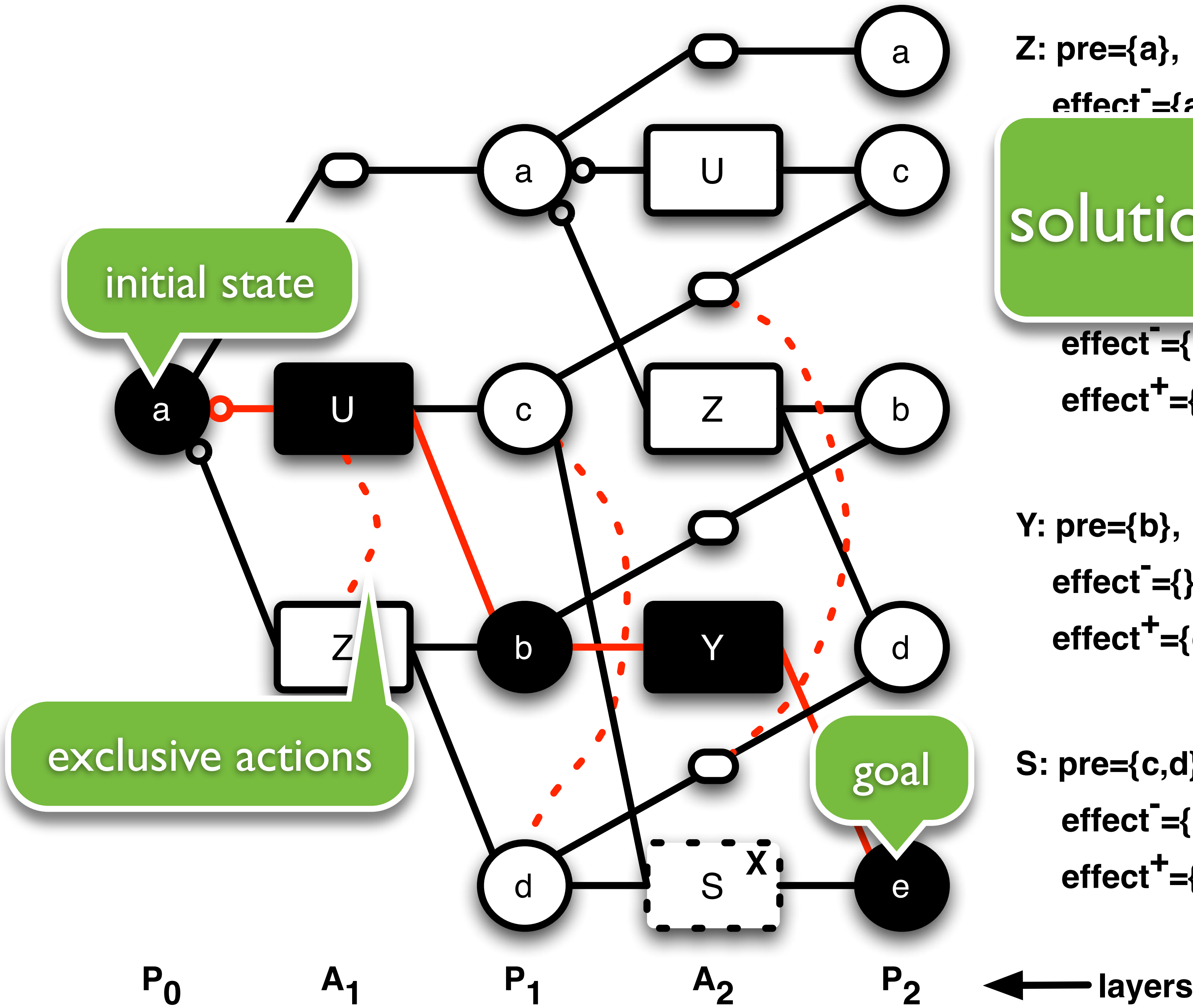
- studied under **different assumptions**
  [Marconi and Pistore, 2009]
- **conversations** for services **and** for requirements
  support for **input/output** and for **capabilities**
  in few approaches:
  [Ben Mokhtar et al, 2007], [Bertoli et al, 2010]
- however, **only horizontal mismatch** supported in
  [Ben Mokhtar et al, 2007] (using semantics)
  and with simple assumption on service integration
- increasing use of **planning** for underspecification
  [Peer, 2005], [Chan et al, 2007]

Approach: Graph Planning



- polynomial construction
- all solutions of length n
- efficient tools available

Approach: Graph Planning



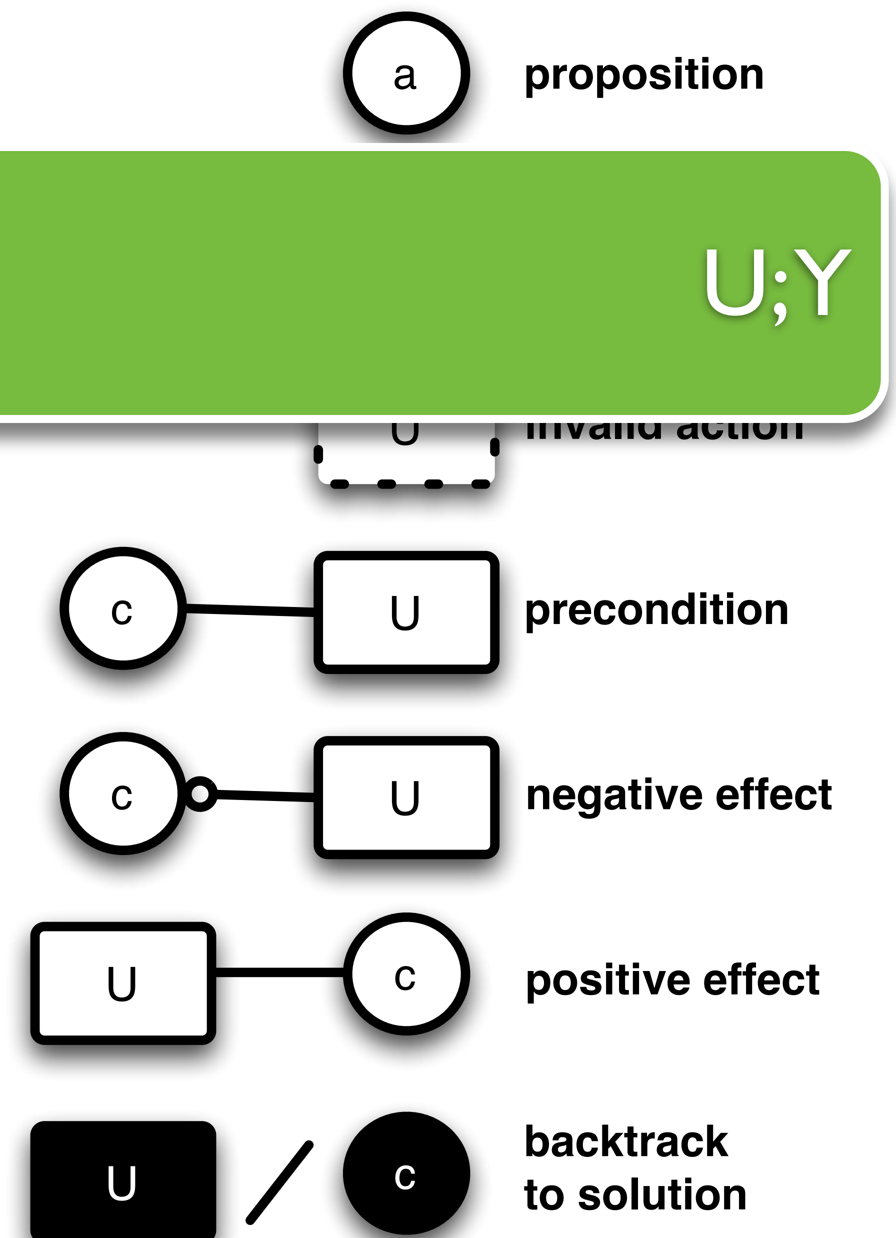
Z: pre={a},
effect$^-$={a},

solution: U;Y

effect$^-$={a},
effect$^+$={b,c}

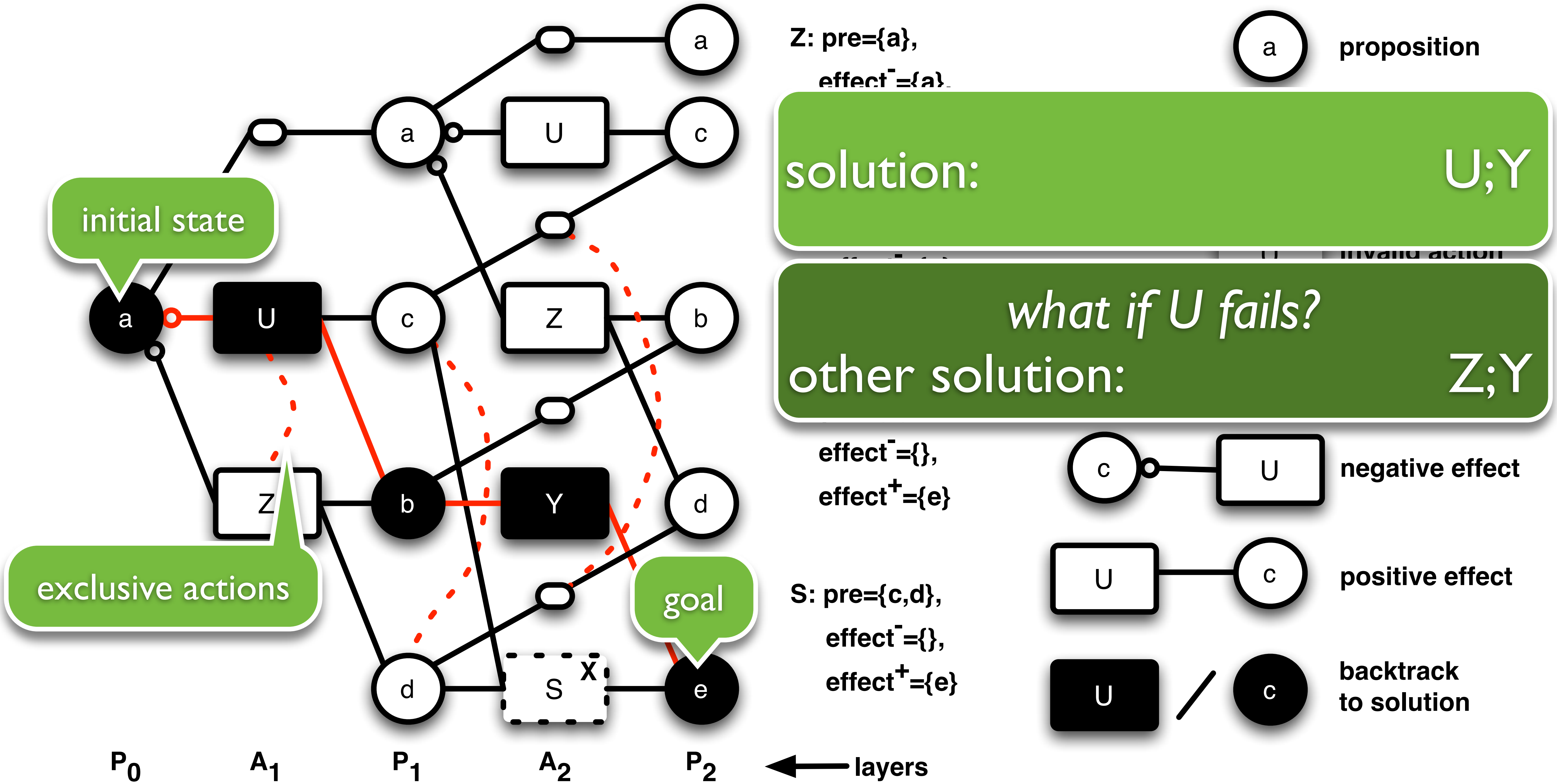
Y: pre={b},
effect$^-$={},
effect$^+$={e}

S: pre={c,d},
effect$^-$={},
effect$^+$={e}



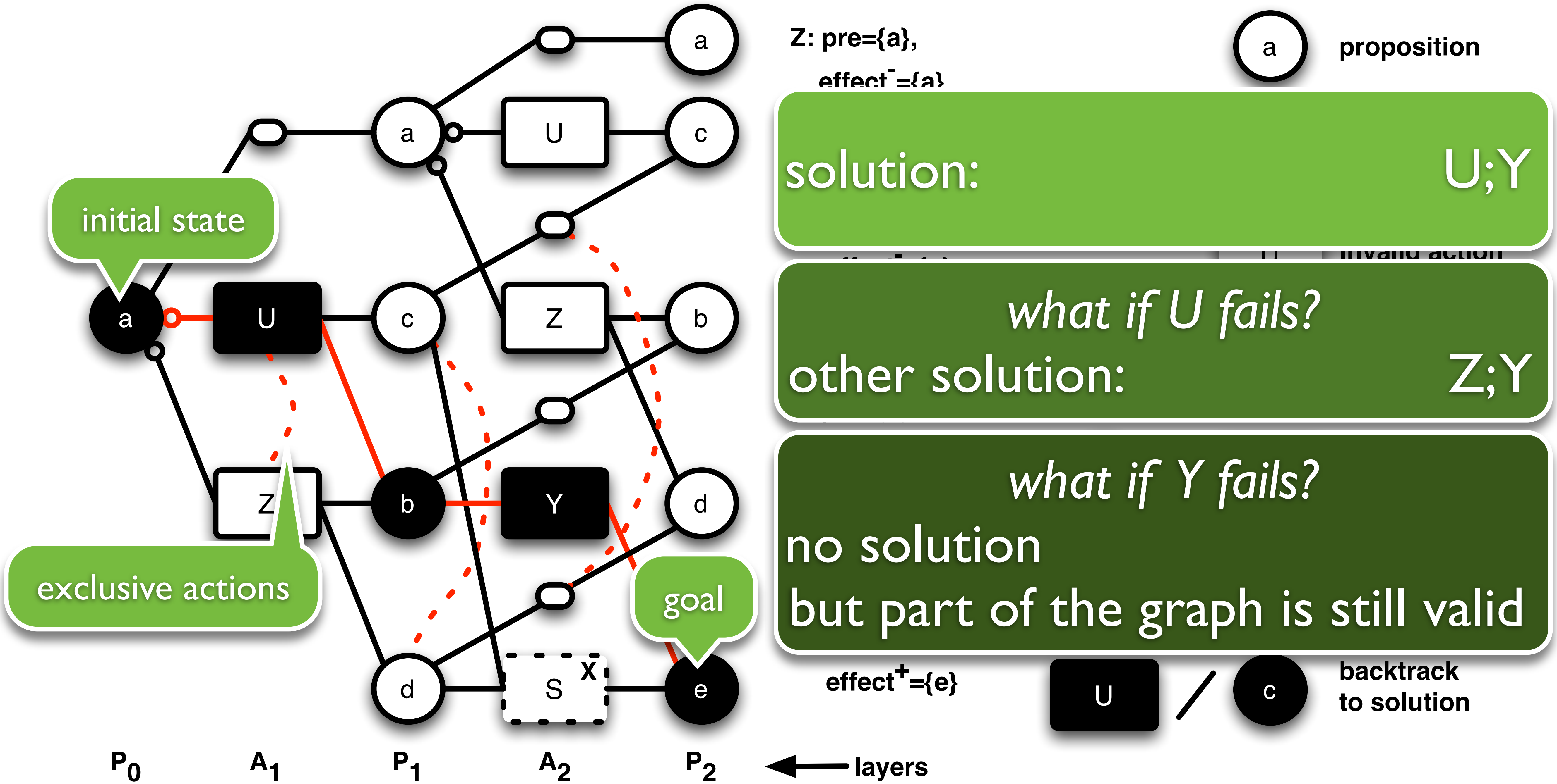
- polynomial construction
- all solutions of length n
- efficient tools available

Approach: Graph Planning



- polynomial construction
- all solutions of length n
- efficient tools available

Approach: Graph Planning



- polynomial construction
- efficient tools available
- all solutions of length n

Approach: Technique

- **semantic typing** of service operations
  input data + output data + capabilities

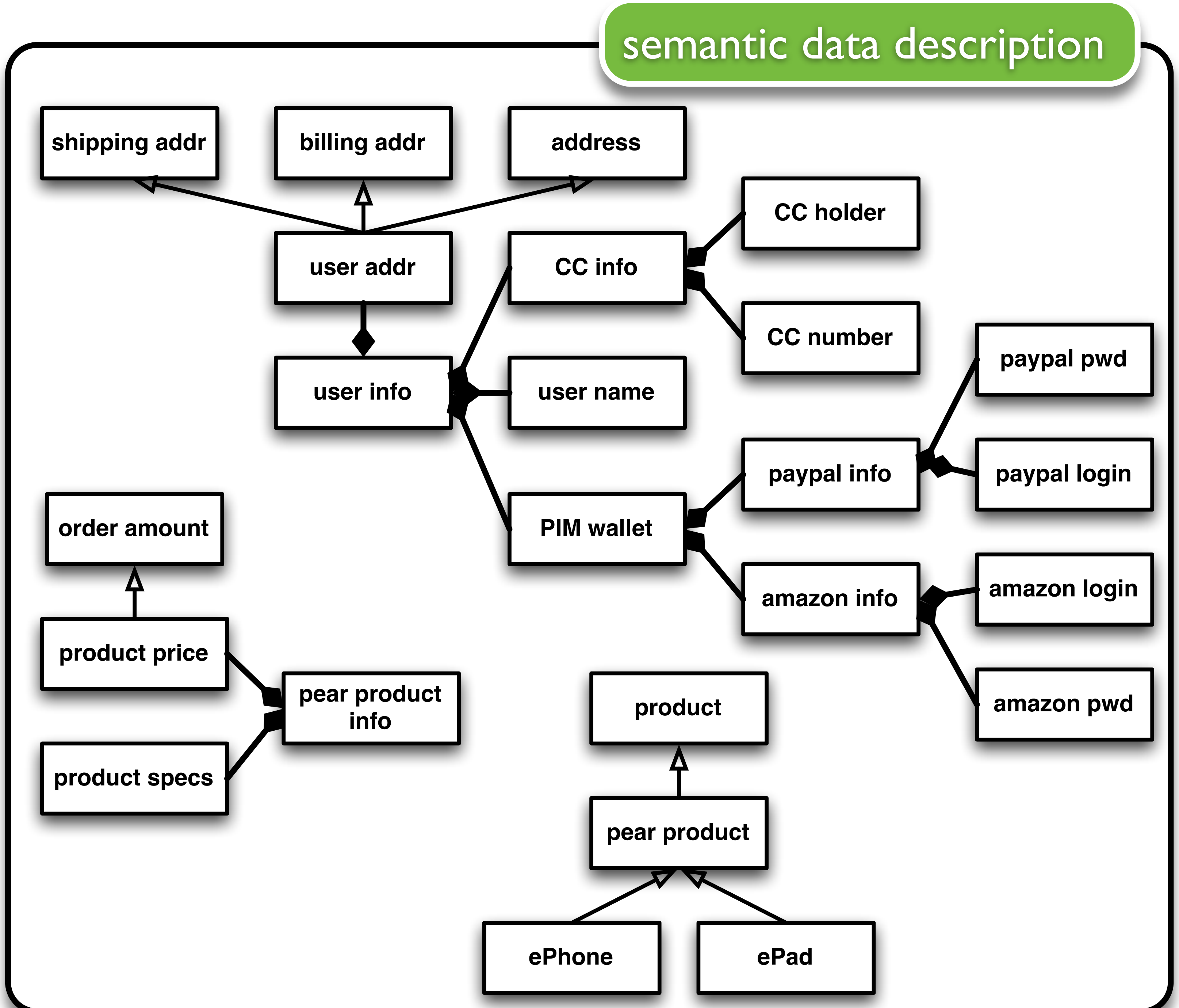
- encoding **data adaptation**
  casting $d < d'$ *enables* cast: $d \longrightarrow d'$
  (de)composition $d=\{d_i\}$ *enables* (de)comp: $\{d_i\} \longleftrightarrow d$

- encoding **conversations**
  workflow to Petri net mapping *[Kiepuszewski, 2003]*
  adapted
  - to map workflows to graph planning actions
  - to enable/disable capabilities (requirement + services)
  - to enable/disable operations (services)

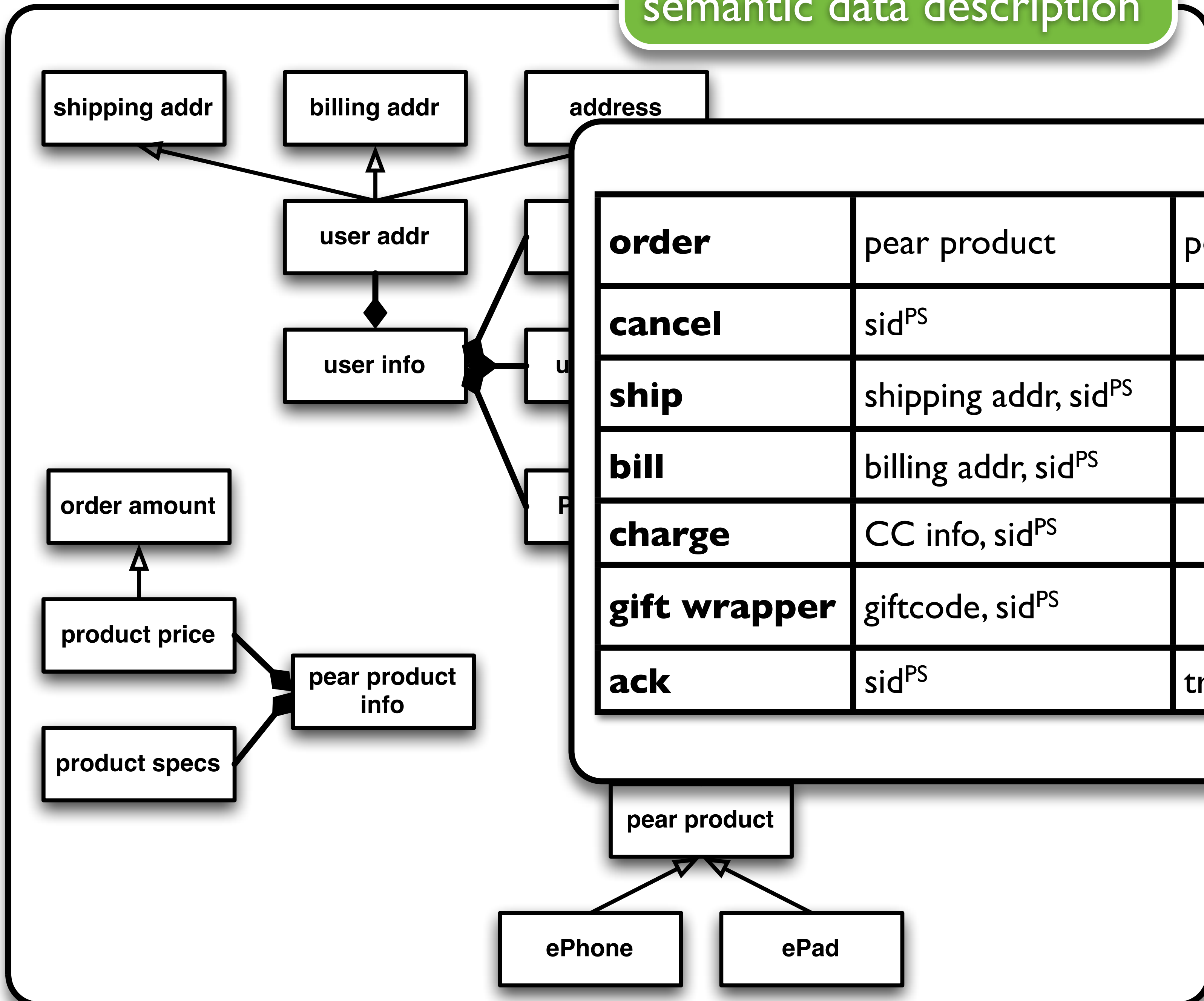
- encoding **operations**
  capability-enabled + inputs $\longrightarrow$ capability-done + outputs

eShopping

eShopping


semantic data description
shipping addr
billing addr
address
CC holder
user addr
CC info
CC number
paypal pwd
user info
user name
paypal info
paypal login
order amount
PIM wallet
amazon info
amazon login
product price
pear product info
product
amazon pwd
product specs
pear product
ePhone
ePad

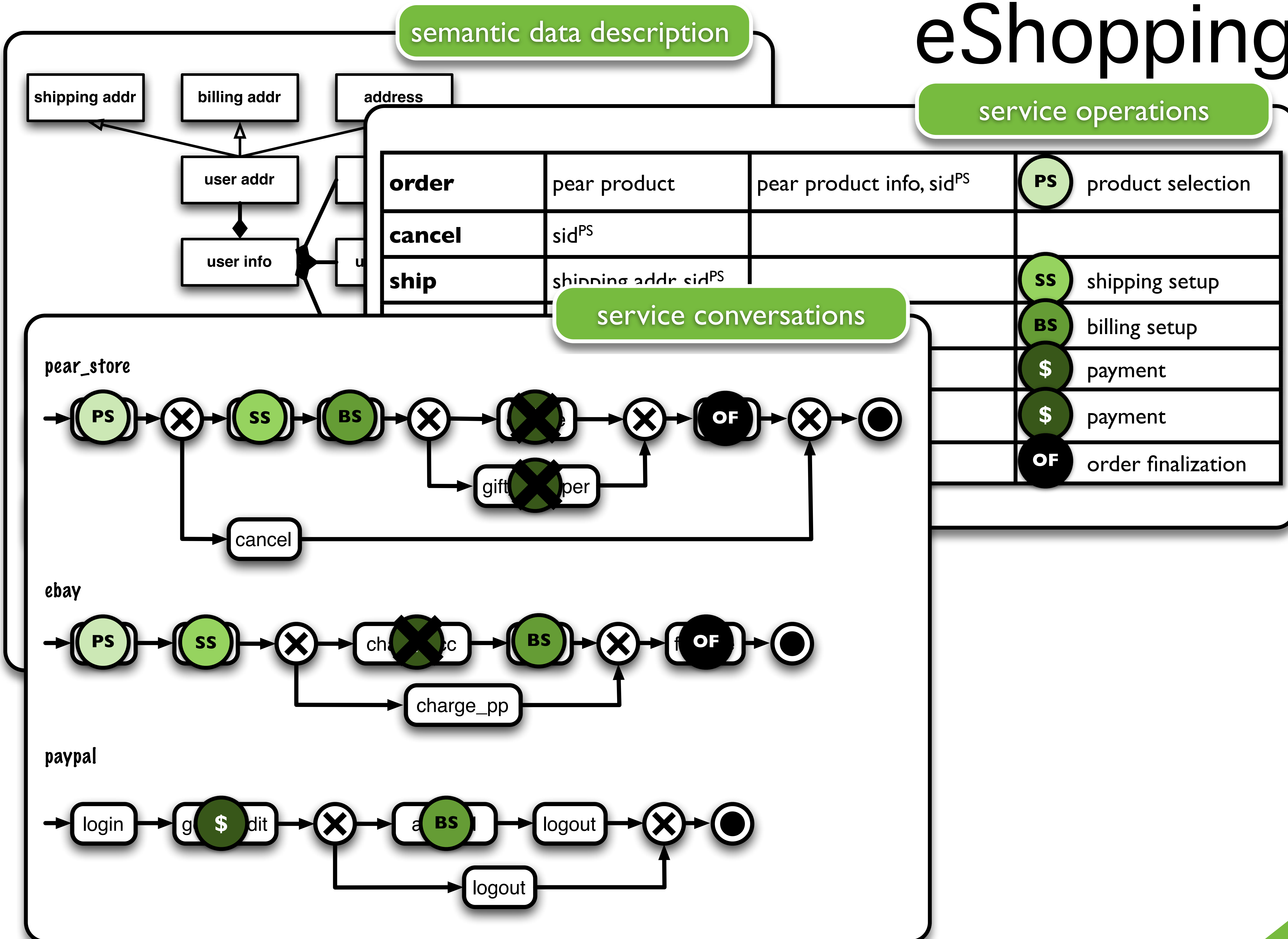
eShopping



service operations

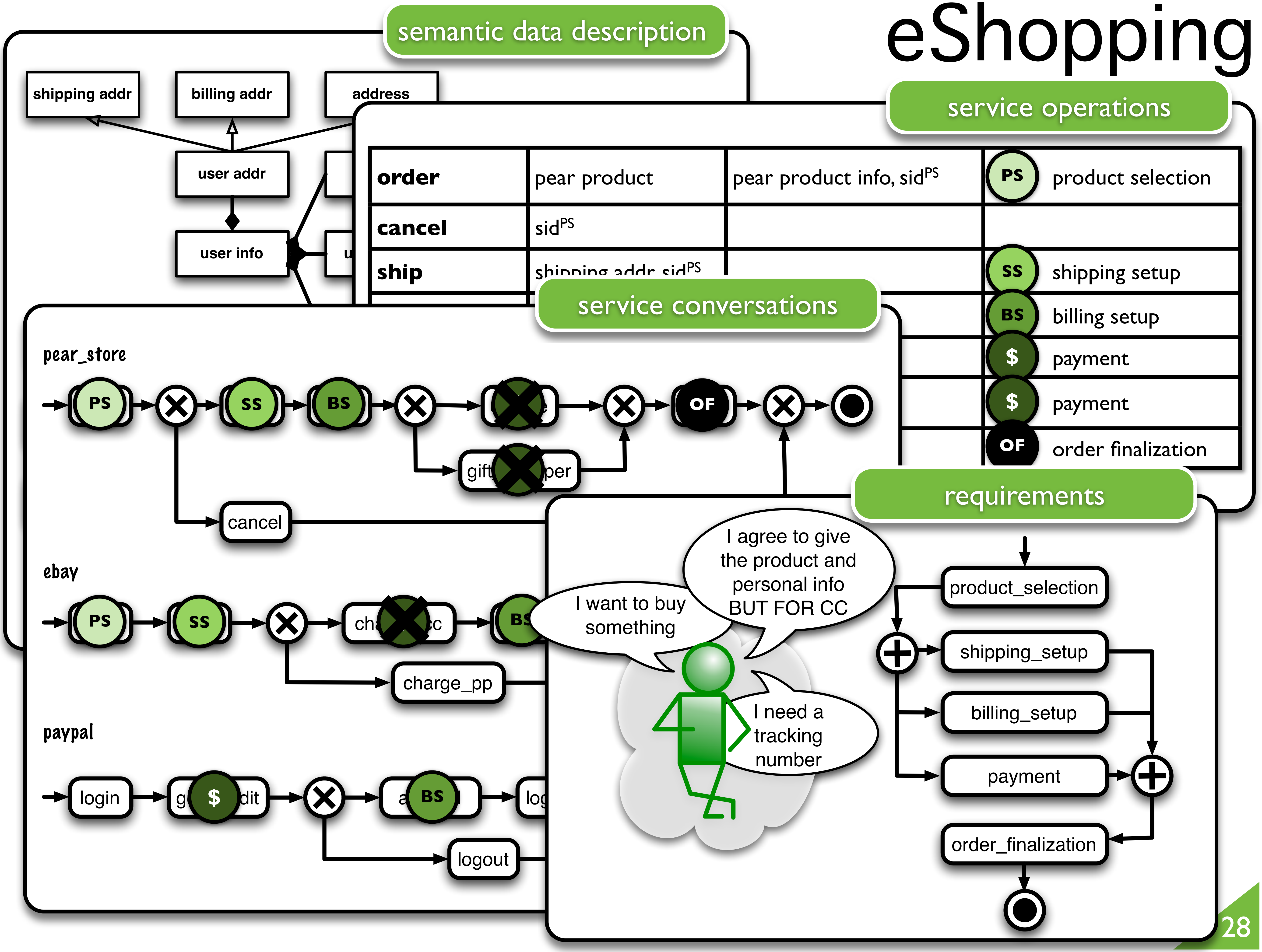
order	pear product	pear product info, sid^{PS}	PS product selection
cancel	sid^{PS}		
ship	shipping addr, sid^{PS}		SS shipping setup
bill	billing addr, sid^{PS}		BS billing setup
charge	CC info, sid^{PS}		$ payment
gift wrapper	giftcode, sid^{PS}		$ payment
ack	sid^{PS}	tracking number	OF order finalization

eShopping

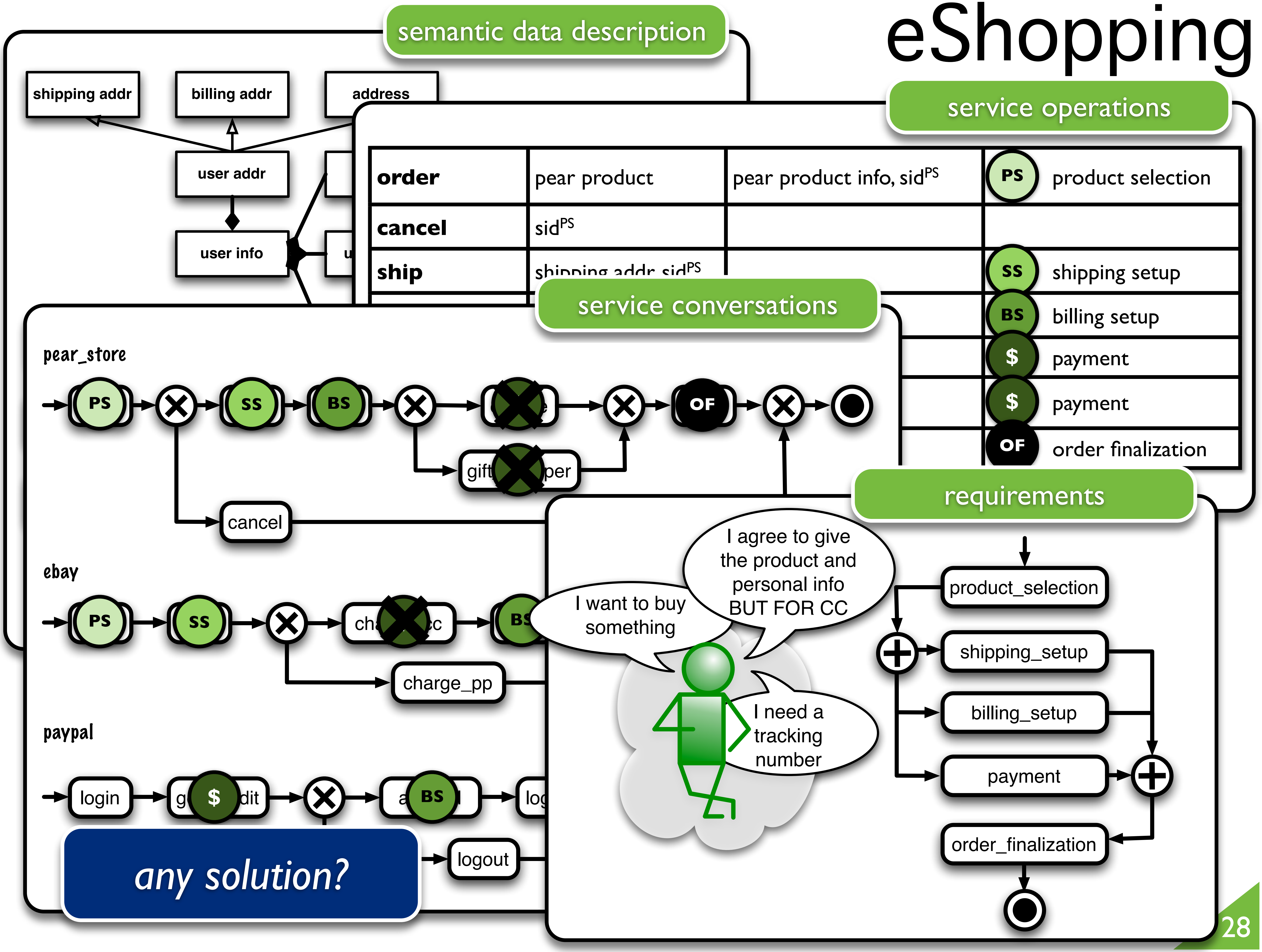


semantic data description
shipping addr
billing addr
address
user addr
user info
service operations
order
pear product
pear product info, sid^PS
PS
product selection
cancel
sid^PS
ship
shipping addr, sid^PS
SS
shipping setup
BS
billing setup
$
payment
$
payment
OF
order finalization
service conversations
pear_store
PS
SS
BS
OF
gift
per
cancel
ebay
PS
SS
BS
OF
charge_pp
paypal
login
$
dit
BS
logout
logout

eShopping

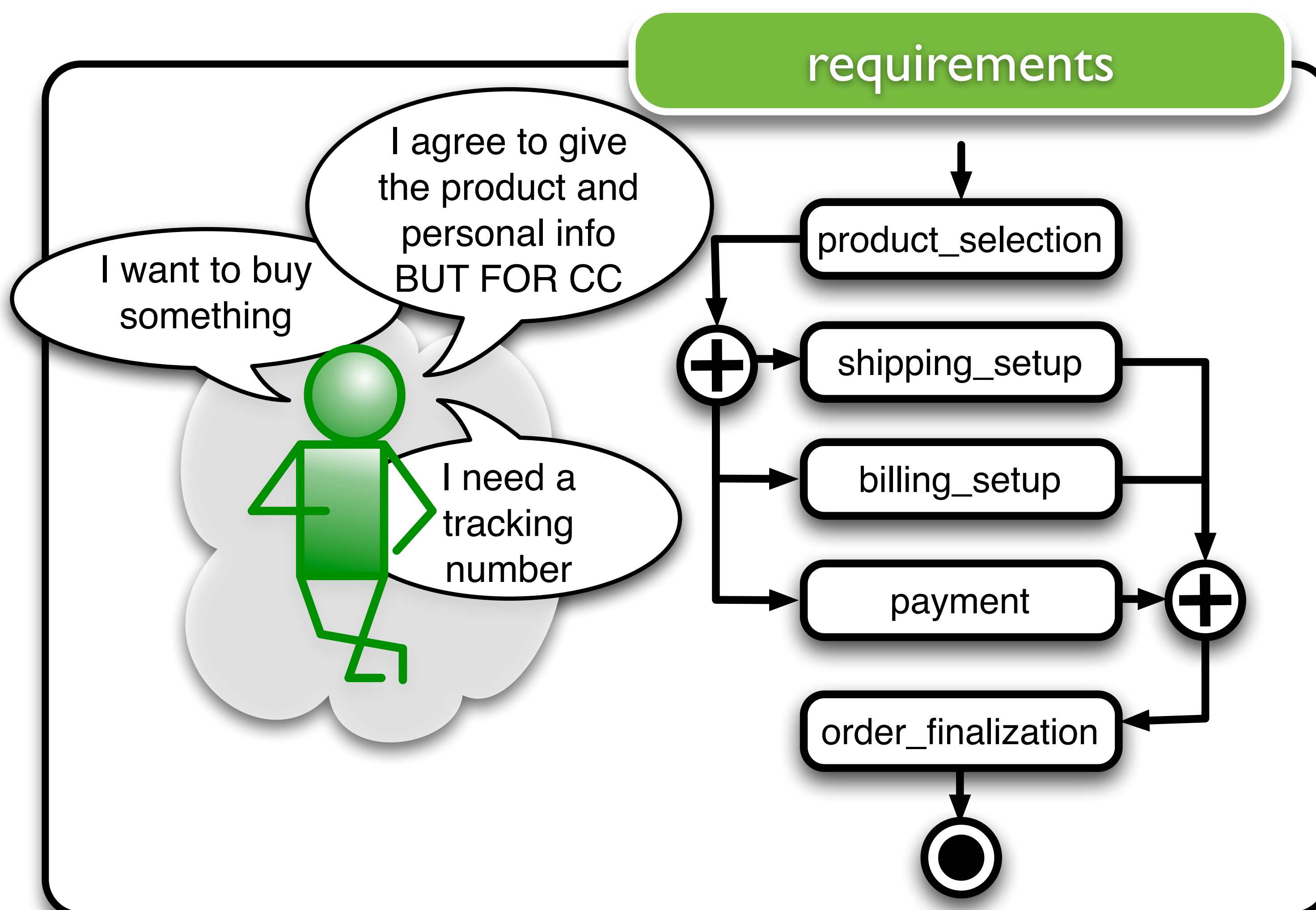

semantic data description
shipping addr
billing addr
address
user addr
user info
service operations
order
pear product
pear product info, sid^{PS}
PS
product selection
cancel
sid^{PS}
ship
SS
shipping setup
BS
billing setup
$
payment
$
payment
OF
order finalization
service conversations
pear_store
PS
SS
BS
OF
gift
per
cancel
ebay
PS
SS
charge_pp
BS
paypal
login
$
dit
BS
logout
requirements
I want to buy something
I agree to give the product and personal info BUT FOR CC
I need a tracking number
product_selection
shipping_setup
billing_setup
payment
order_finalization

eShopping



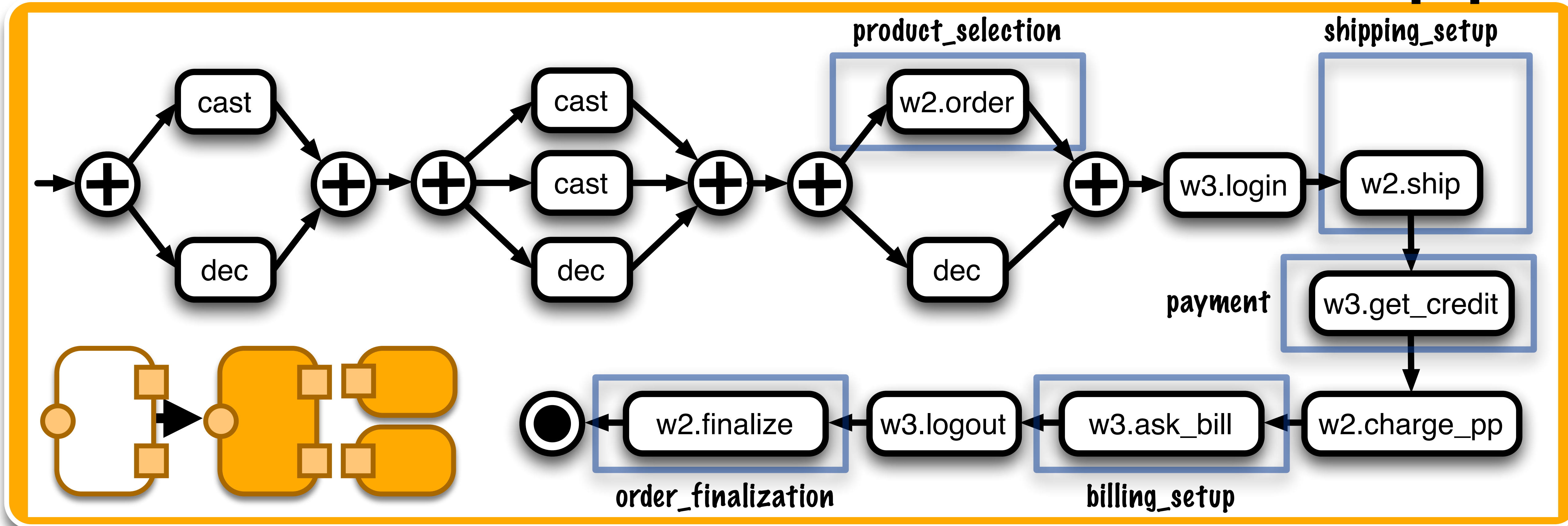
semantic data description
shipping addr
billing addr
address
user addr
user info
service operations
order
pear product
pear product info, sid PS
PS product selection
cancel
sid PS
ship
shipping addr, sid PS
SS shipping setup
BS billing setup
$ payment
$ payment
OF order finalization
service conversations
pear_store
cancel
ebay
charge_pp
paypal
login
logout
requirements
I agree to give the product and personal info BUT FOR CC
I want to buy something
I need a tracking number
product_selection
shipping_setup
billing_setup
payment
order_finalization
any solution?

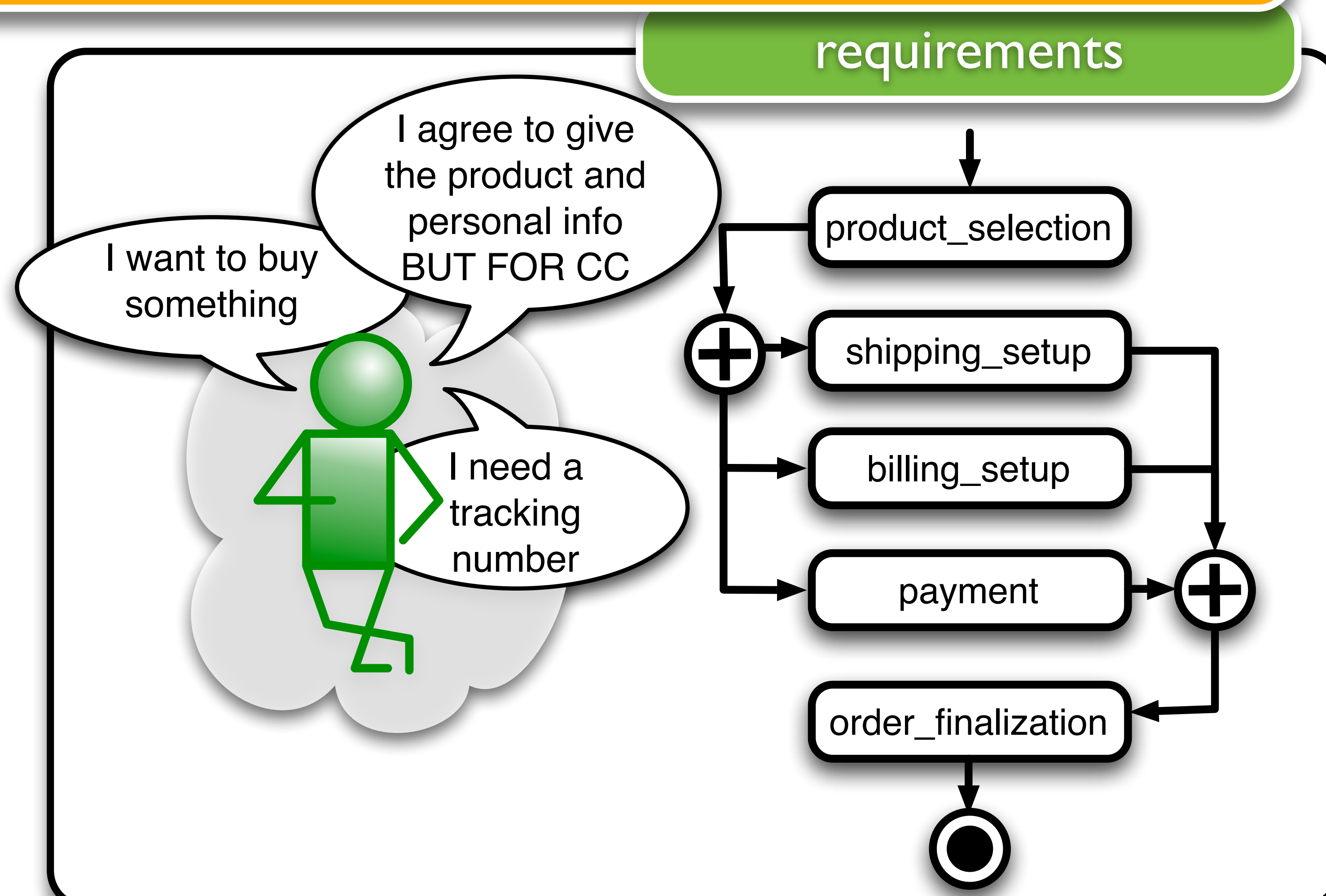
eShopping



yes

eShopping

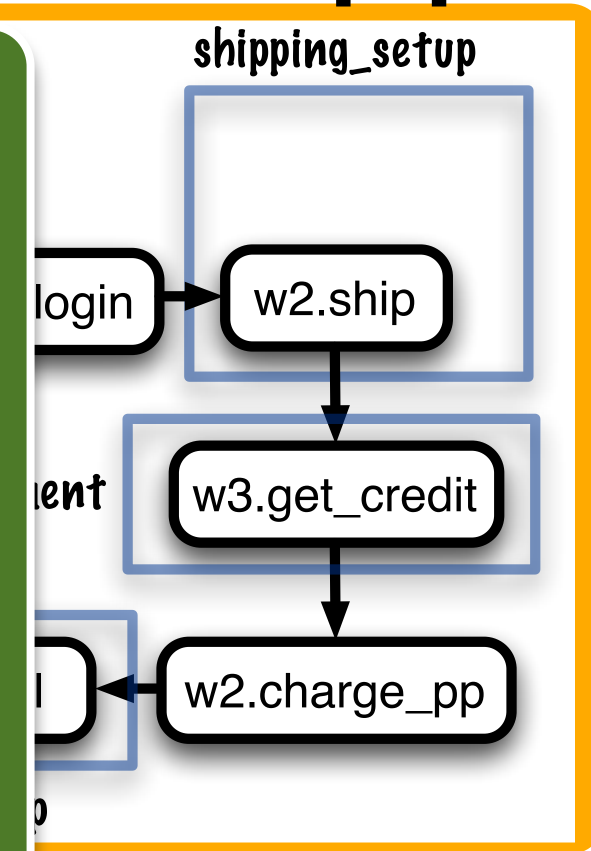


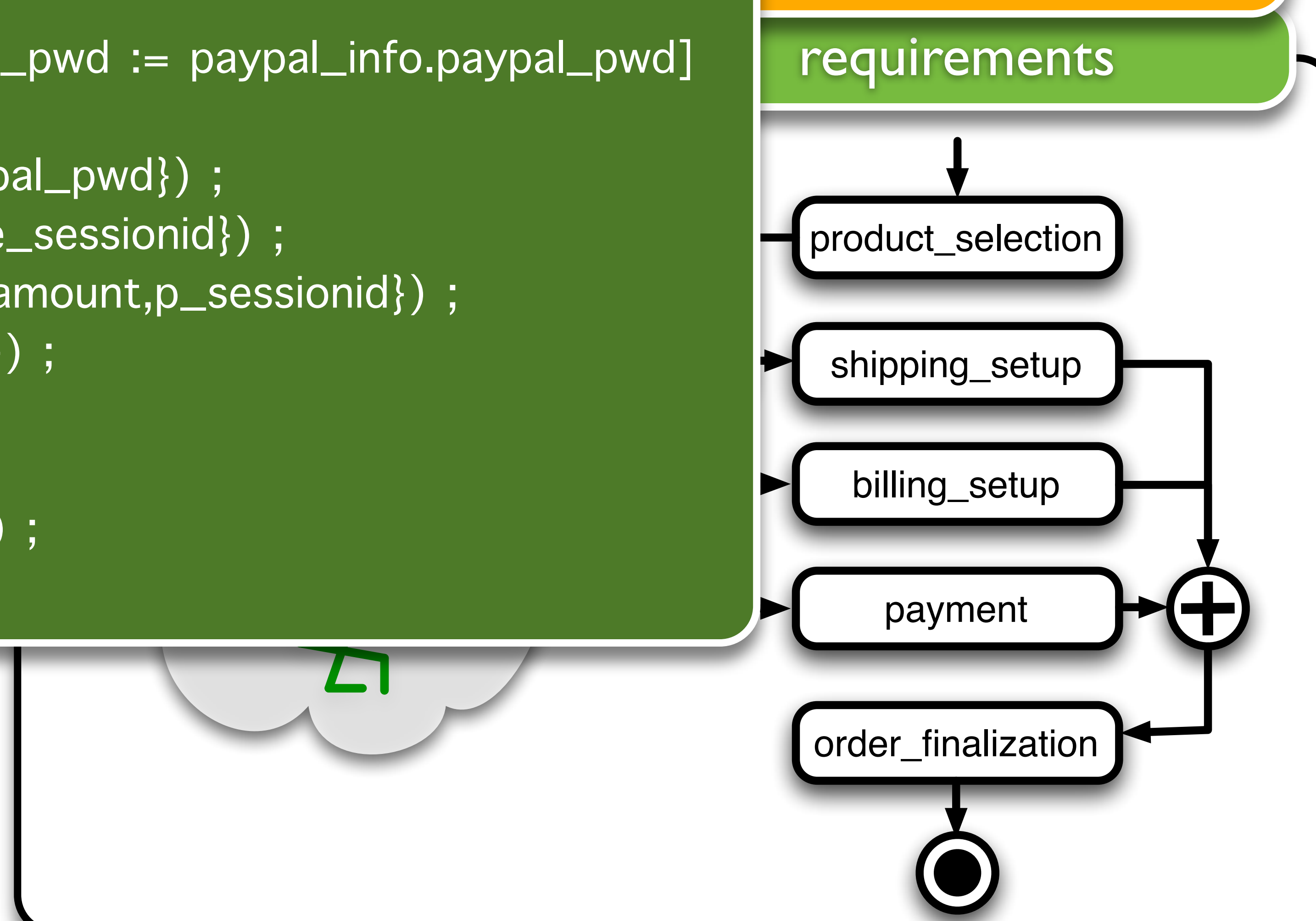


yes

eShopping

```
receive(user,op,{ePad,user_info}) ;
flow {
  [pear_product := cast(ePad)],
  [user_name := user_info.user_name; user_address := user_info.user_address;
   pim_wallet := user_info.pim_wallet]
} ;
flow {
  [product := cast(pear_product)],
  [shipping_addr := cast(user_address)],
  [paypal_info := pim_wallet.paypal_info; amazon_info := PIM_wallet.amazon_info]
} ;
flow {
  [{e_sessionid} := invoke(w2,order,{product})],
  [paypal_login := paypal_info.paypal_login; paypal_pwd := paypal_info.paypal_pwd]
} ;
{p_sessionid} := invoke(w3,login,{paypal_login,paypal_pwd}) ;
{order_amount} := invoke(w2,ship,{shipping_addr,e_sessionid}) ;
{paypal_trans_id} := invoke(w3,get_credit,{order_amount,p_sessionid}) ;
invoke(w2,charge_pp,{paypal_trans_id,e_sessionid}) ;
invoke(w3,ask_bill,{user_address,p_sessionid}) ;
invoke(w3,logout,{p_sessionid}) ;
{tracking_num} := invoke(w2,finalize,{e_sessionid}) ;
reply(user,op,{tracking_num});
```



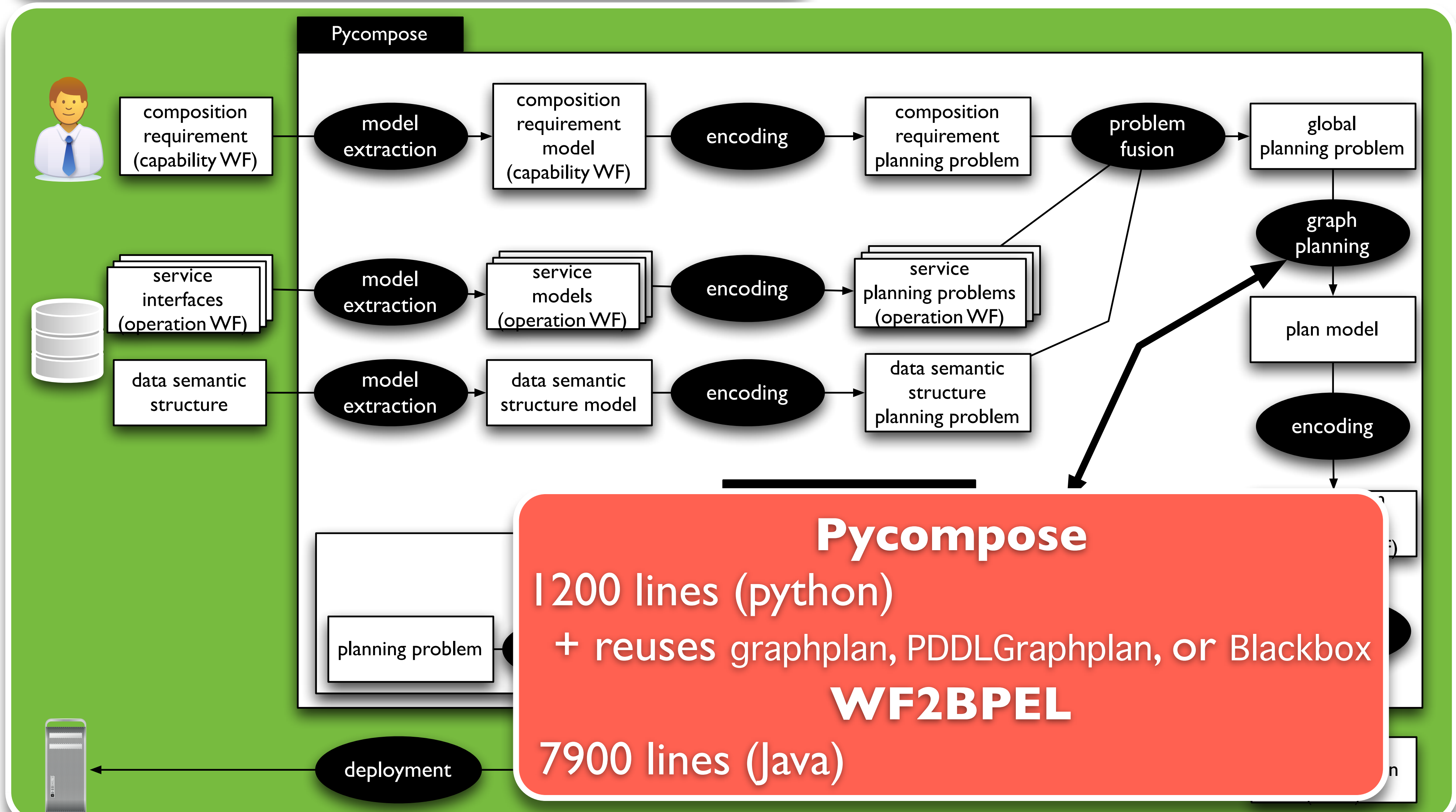


yes

Contributions on Composition

services with **conversations**
requirement with **conversation**
data flow and **control** flow
horizontal + vertical **adaptation**
application to **WS**

ICSOC'08, ISoLa'10



Pycompose
1200 lines (python)
+ **reuses** graphplan, PDDLGraphplan, or Blackbox
WF2BPEL
7900 lines (Java)

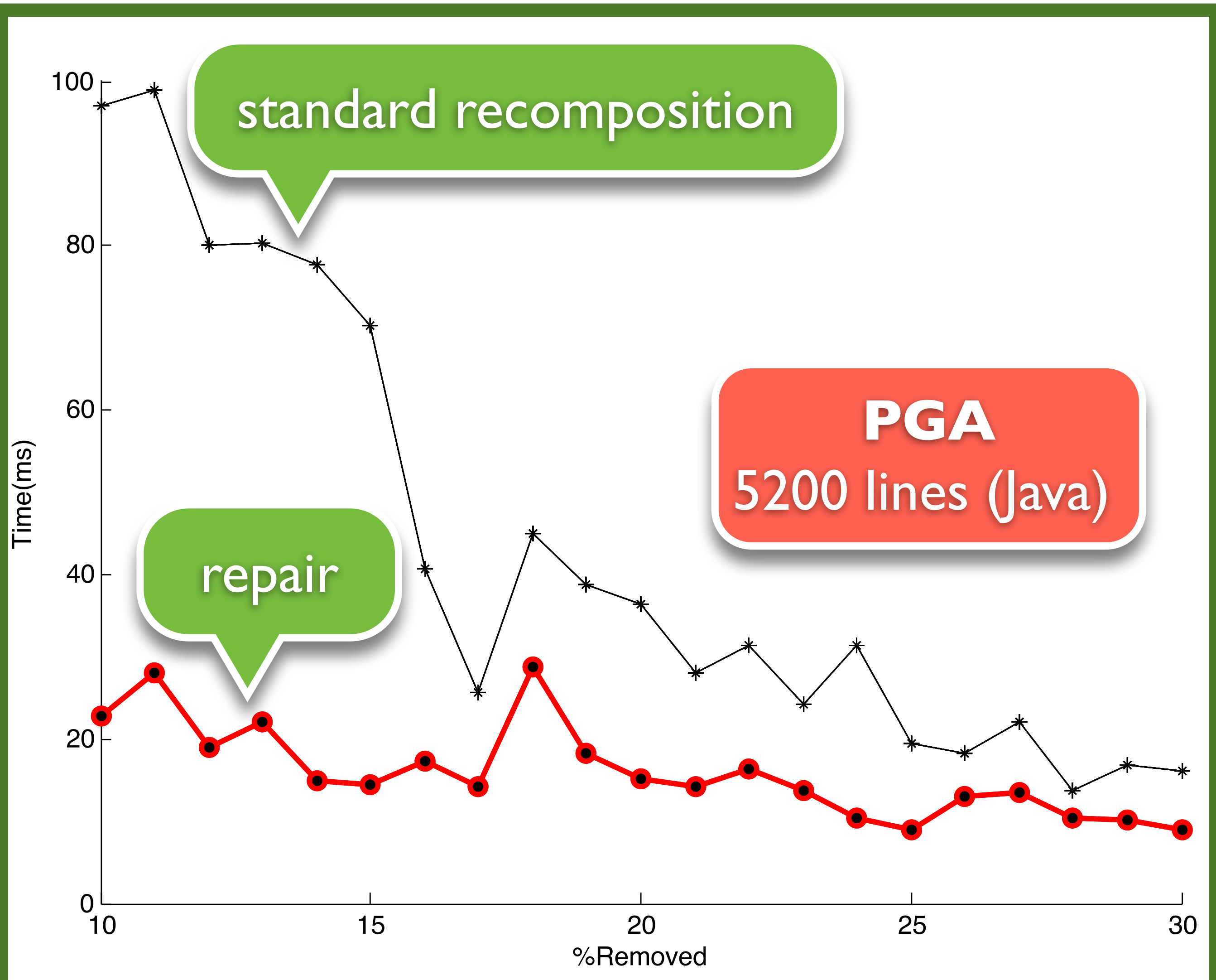
Contributions on Composition

services with **conversations**
requirement with **conversation**
data flow and **control** flow
horizontal + vertical **adaptation**
application to **WS**

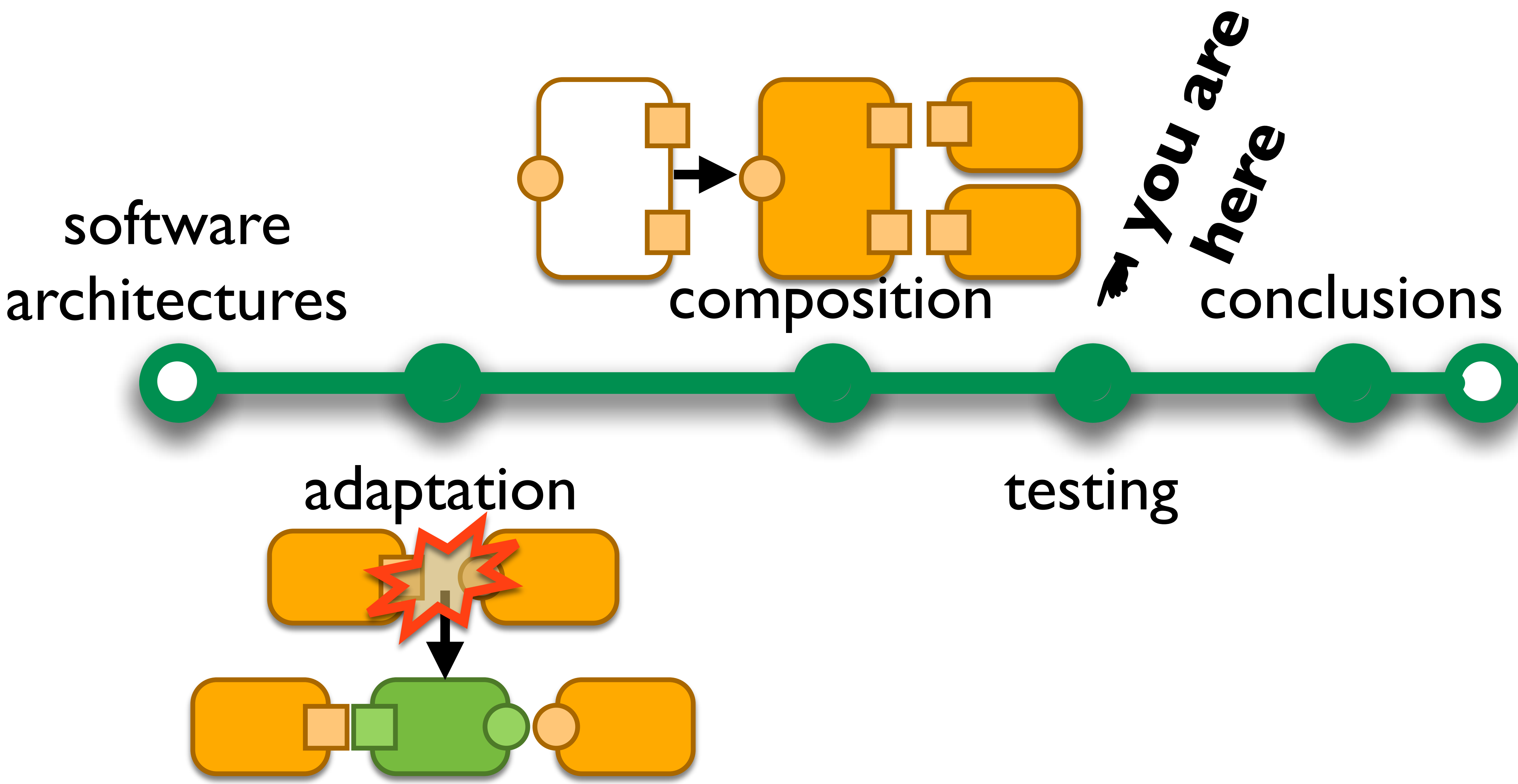
ICSOC'08, ISoLa'10

repair *vs.* recomposition:
repaired solution quality =
computation time ↘
application to **WS**

ICWS'10, ICSOC'10

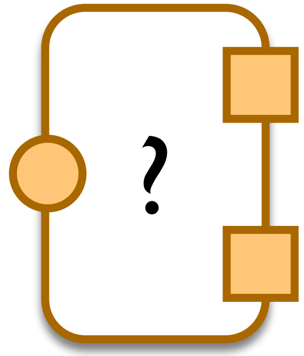


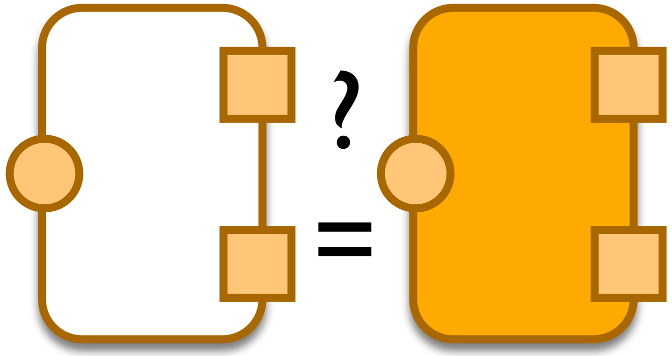
Agenda


software
architectures
composition
you are
here
conclusions
adaptation
testing

Issue

- verification (of design artifacts) and testing are **complementary**



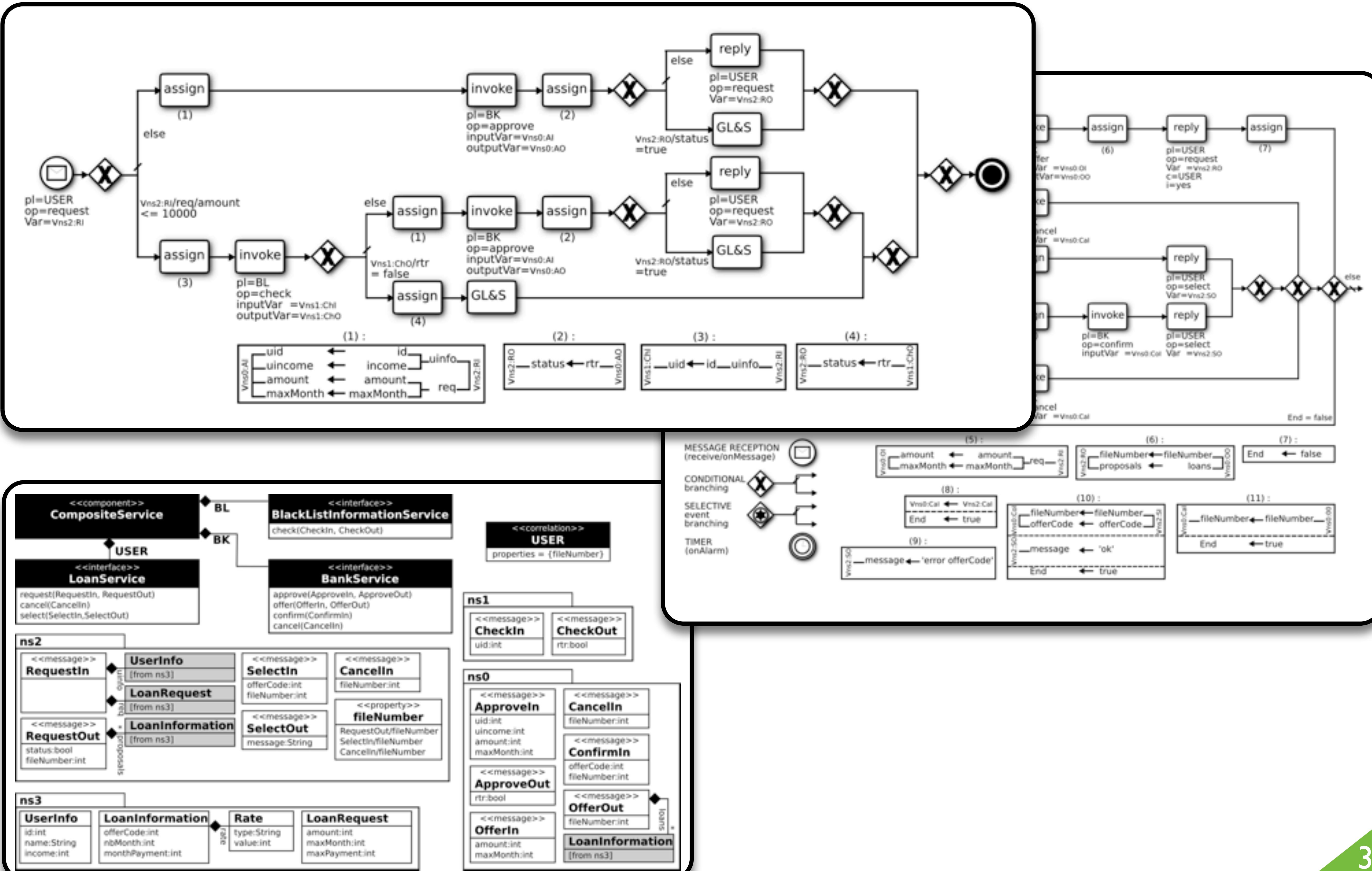


verification
requirements ⟷ model

testing
model ⟷ implementation

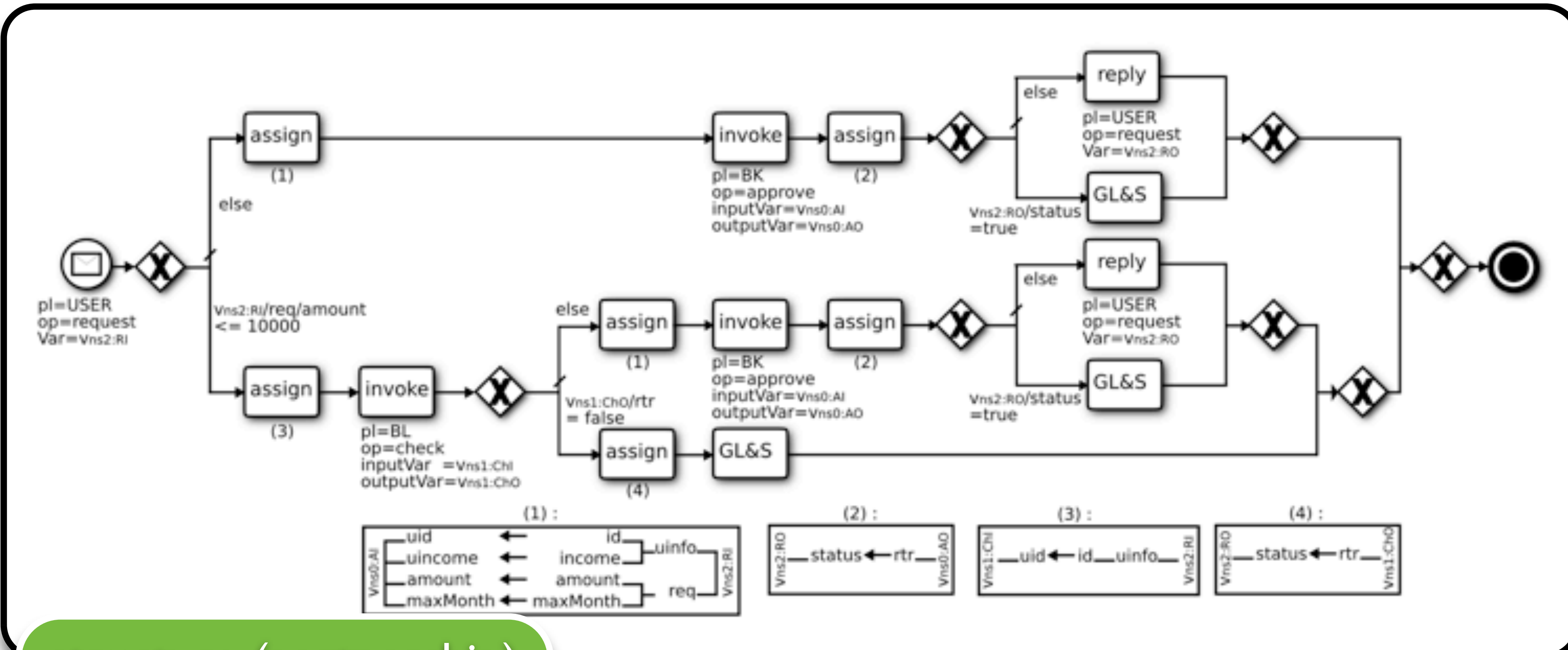
- the need for testing **increases** in a development process based on **reuse** and with **dynamic binding**

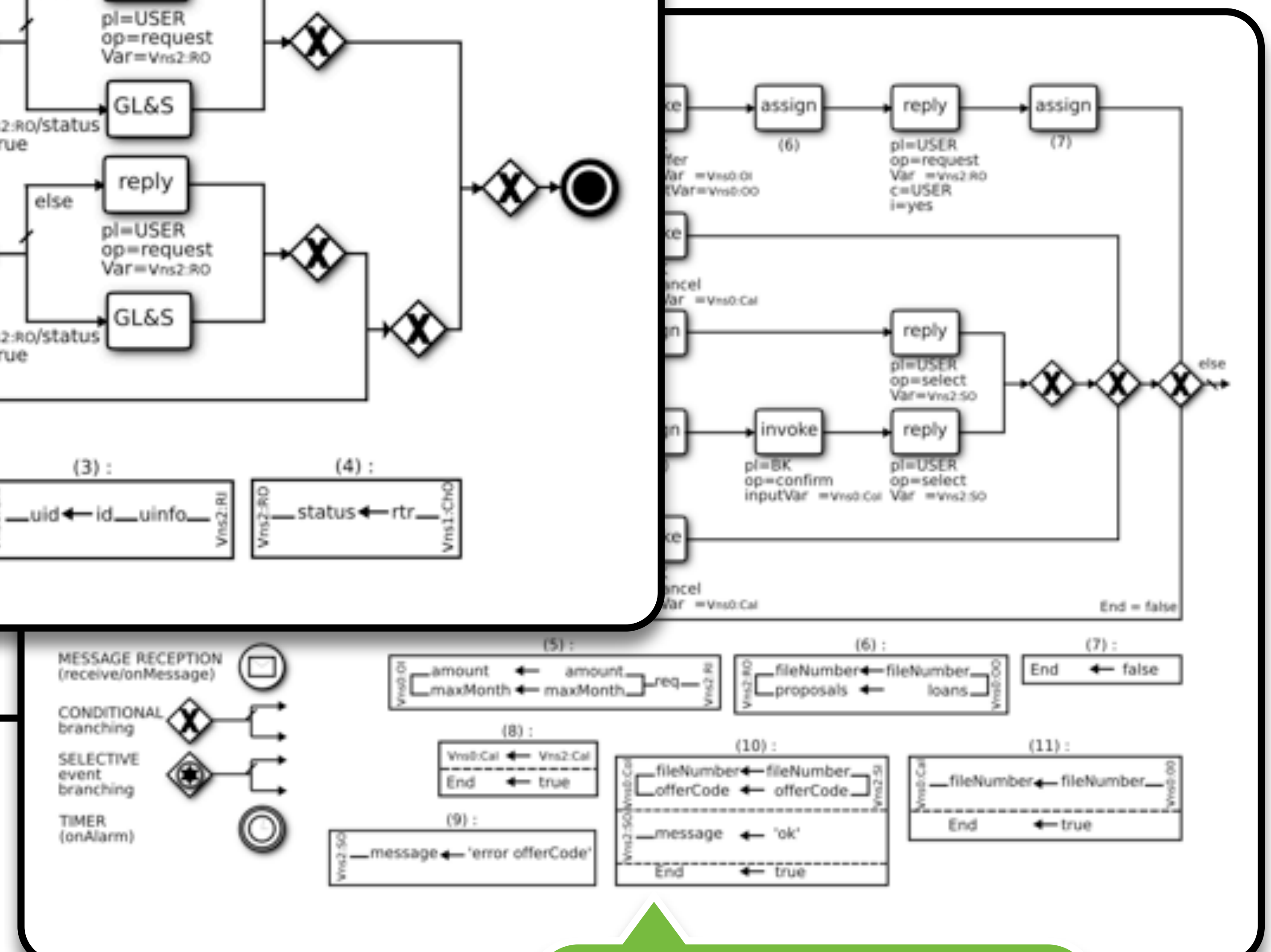
Issue



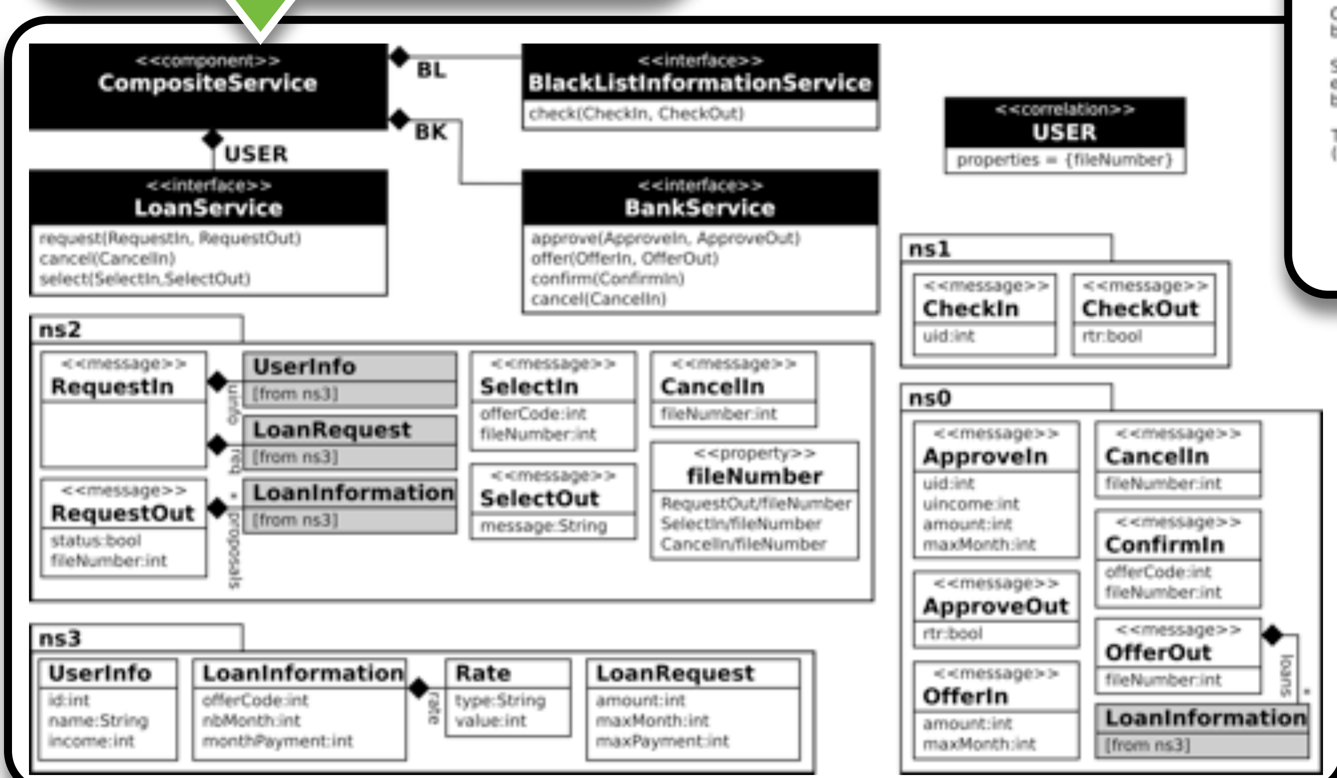
assign
invoke
reply
GL&S
else
pl=USER
op=request
Var=Vns2:RI
Vns2:RI/req/amount
<= 10000
pl=BL
op=check
pl=BK
op=approve
Vns2:RO/status
=true
Vns1:ChO/rtr
= false
(1) :
(2) :
(3) :
(4) :
uid
uincome
amount
maxMonth
income
uinfo
req
status
rtr
id
MESSAGE RECEPTION
(receive/onMessage)
CONDITIONAL
branching
SELECTIVE
event
branching
TIMER
(onAlarm)
fileNumber
proposals
loans
End
false
true
offerCode
message
'ok'
'error offerCode'
<<component>>
CompositeService
BL
BK
USER
<<interface>>
BlackListInformationService
check(CheckIn, CheckOut)
<<interface>>
LoanService
request(RequestIn, RequestOut)
cancel(CancelIn)
select(SelectIn,SelectOut)
<<interface>>
BankService
approve(ApproveIn, ApproveOut)
offer(OfferIn, OfferOut)
confirm(ConfirmIn)
cancel(CancelIn)
<<correlation>>
USER
properties = {fileNumber}
ns1
CheckIn
CheckOut
ns2
RequestIn
UserInfo
LoanRequest
LoanInformation
RequestOut
SelectIn
SelectOut
CancelIn
<<property>>
fileNumber
ns0
ApproveIn
ApproveOut
OfferIn
CancelIn
ConfirmIn
OfferOut
LoanInformation
ns3
UserInfo
LoanInformation
Rate
LoanRequest

Issue



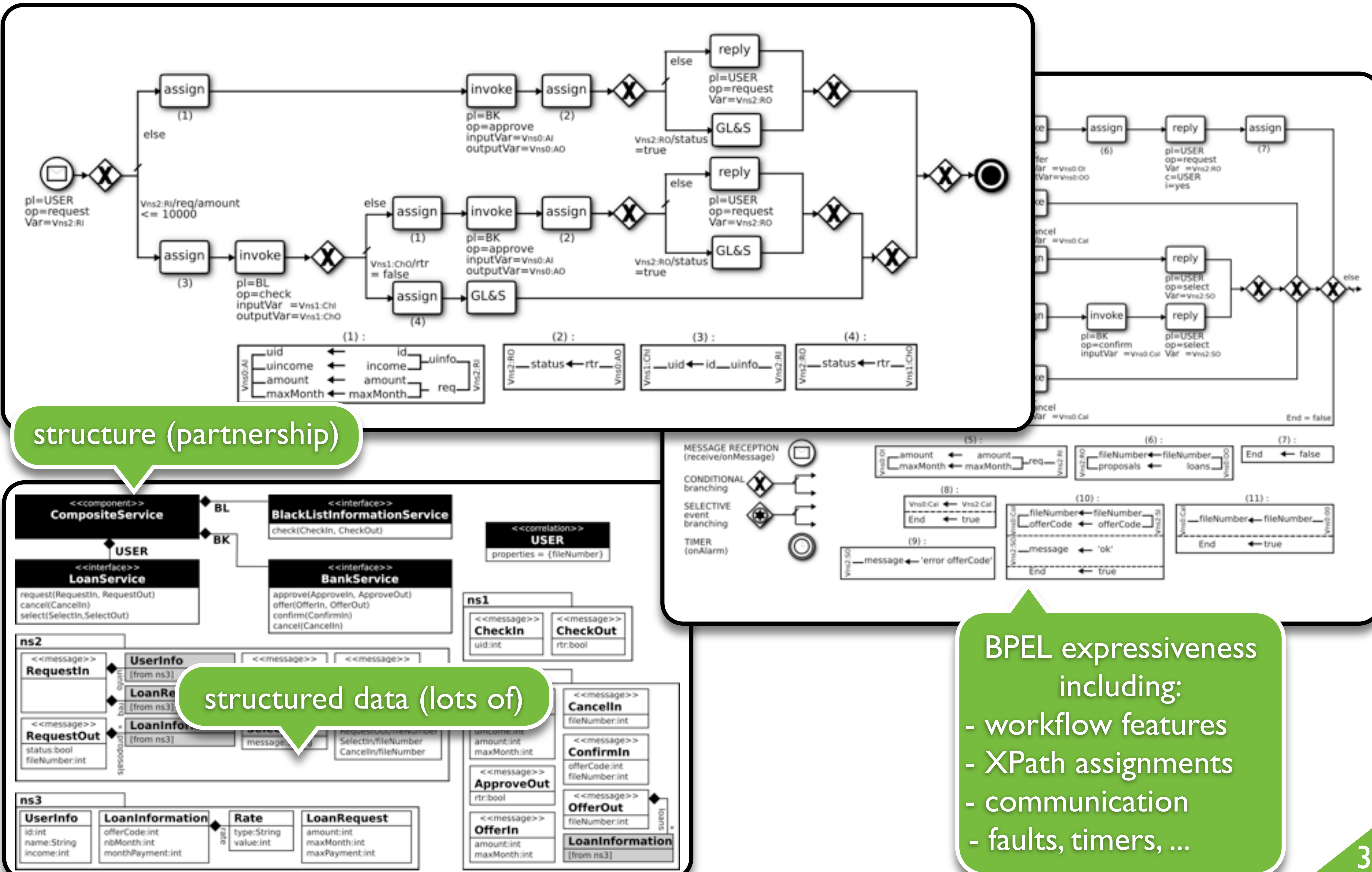


structure (partnership)



BPEL expressiveness including:
- workflow features
- XPath assignments
- communication
- faults, timers, ...

Issue

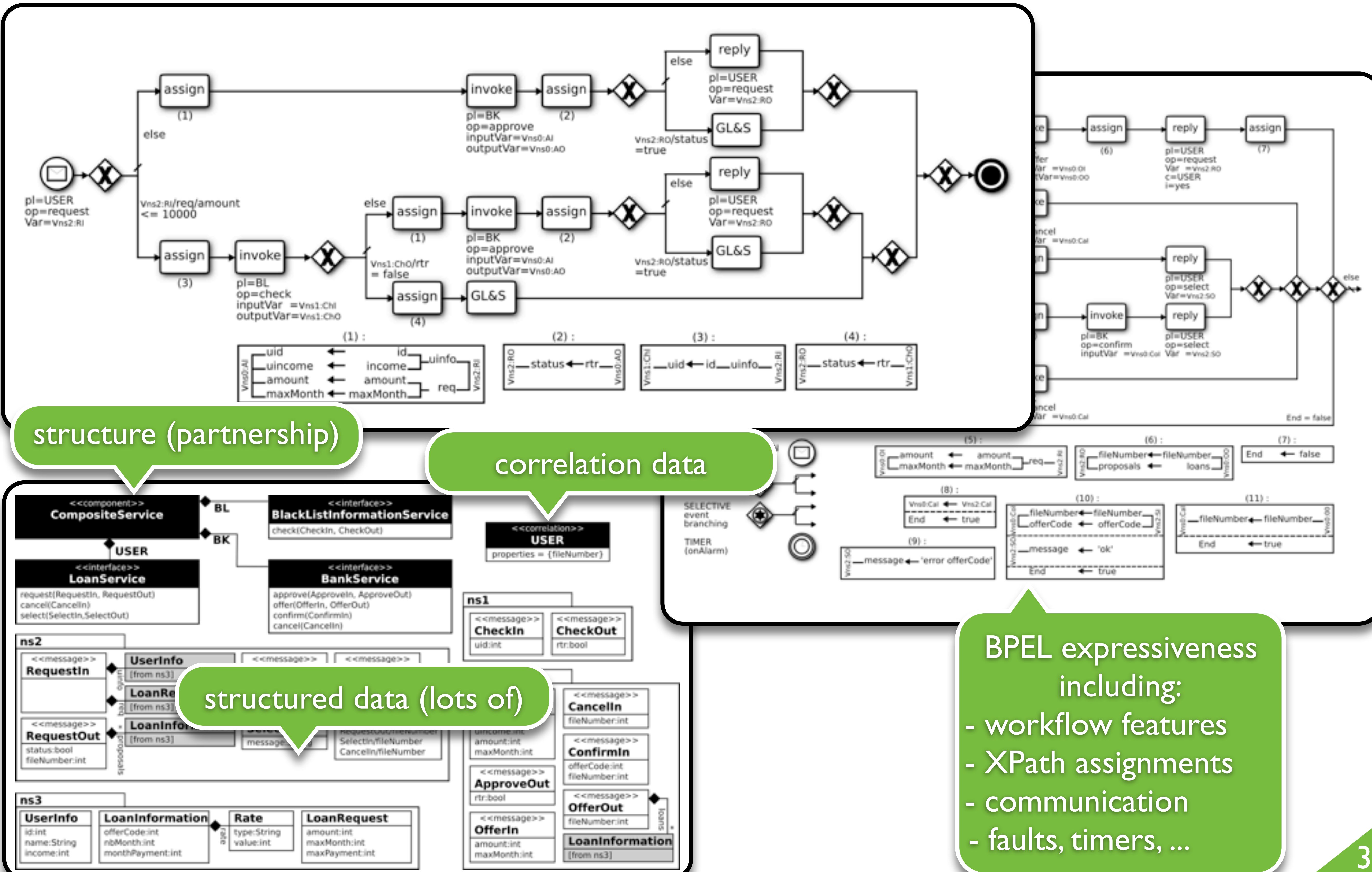


structure (partnership)

structured data (lots of)

BPEL expressiveness including:
- workflow features
- XPath assignments
- communication
- faults, timers, ...

Issue



structure (partnership)

correlation data

structured data (lots of)

BPEL expressiveness including:
- workflow features
- XPath assignments
- communication
- faults, timers, ...

Testing Approaches

- **involvment** of the testing community
  [Bozkurt et al, 2010], [Russli et al, 2011]

- WSDL-testing does not supports **conversations**
  white-box testing is **not realistic** wrt. reuse

- the **treatment of data** makes the difference
  WSDL-S ➠ EFSM + theorem prover *[Sinha and Paradkar, 2006]*
  BPEL ➠ CFG + symbolic execution + solver *[Yan et al, 2006]*
  BPEL ➠ Promela + model-checker *[Zheng et al, 2007]*
  UML ➠ STS + online approach *[Frantzen et al, 2009]*

- **test passing** in *[Zheng et al, 2007]* and *[Frantzen et al, 2009]*

- **combining** on-line approach + symbolic execution
  perspective of *[Frantzen et al, 2009]*

Approach: Technique (1/4)

- transformation ABPEL specification → STS

based on an earlier work by Mateescu and Rampacek (2008)

use of STS instead of dtLTS

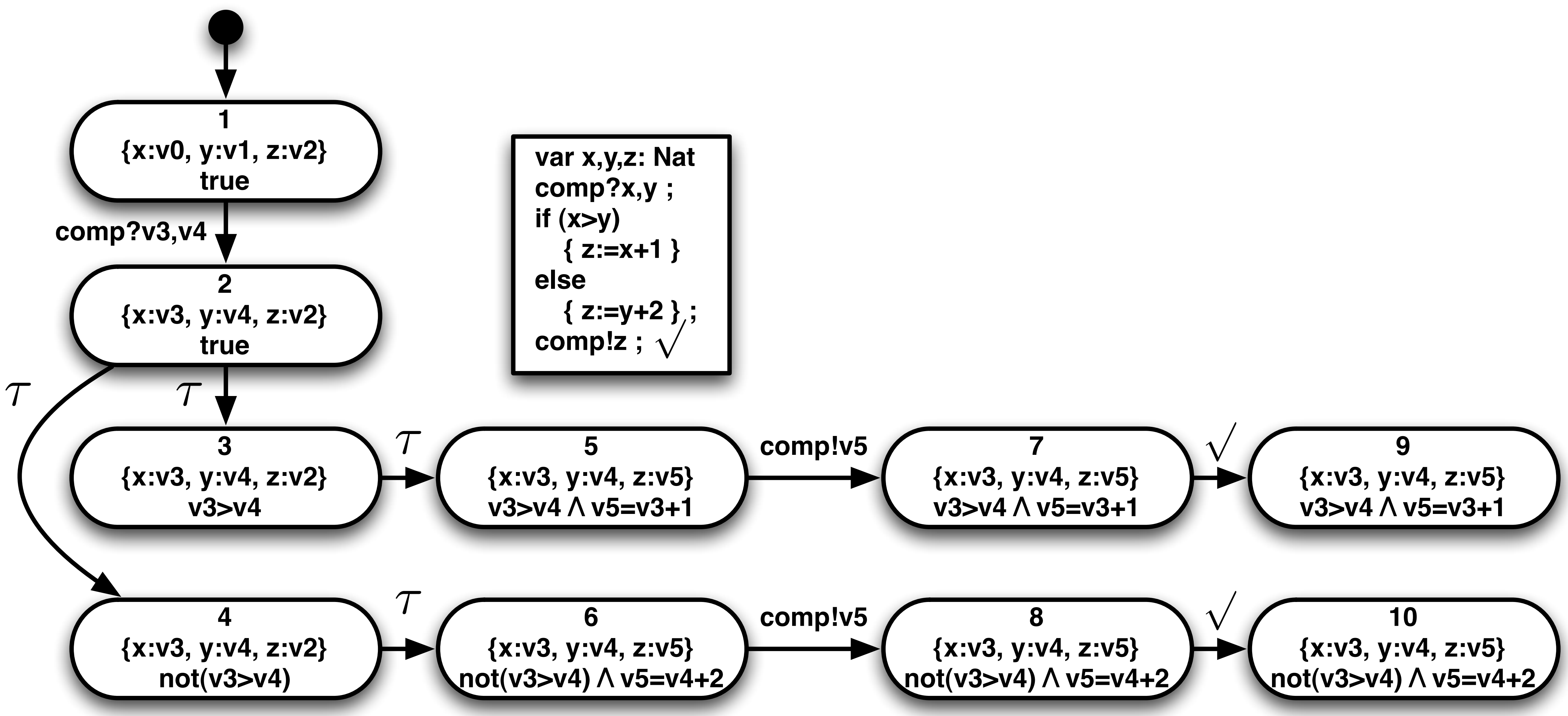
reused as is: time, throw, sequence

extensions:
data in constructs
correlation and m. faults

BPEL	STS	
empty	$\mathsf{empty} \xrightarrow{\surd} 0$	
time	$p \xrightarrow{\chi} p$ with $p \in \{\mathsf{time}, \mathsf{rec}(pl,o,v_{in}), \mathsf{send}(pl,o,v_{out})\}$	
assign^{+}	$p1{:=}p2 \xrightarrow{\tau\ /\ p1:=p2} \mathsf{empty}$	
throw	$\forall e \in Ex$ throw $e \xrightarrow{e} 0$	
rec^{+}	$\mathsf{rec}(pl,o,v_{in}) \xrightarrow{pl.o?va_m\ /\ v_{in}:=va_m} \mathsf{empty}$ with $\exists o \in \mathcal{O}(\Sigma_{pl}),\ in(o) = m$	
send^{+}	$\mathsf{send}(pl,o,v_{out}) \xrightarrow{\tau\ /\ va_m:=v_{out}} \_ \xrightarrow{pl.o!va_m} \mathsf{empty}$ with $\exists o \in \mathcal{O}(\Sigma_{pl}),\ out(o) = m$	
receive*	$\mathsf{receive}(pl,o,v_{in}) = \mathsf{rec}(pl,o,v_{in})$	
reply*	$\mathsf{reply}(pl,o,v_{out}) = \mathsf{send}(pl,o,v_{out})$	
invoke^{+}	$\mathsf{invoke}(pl,o,v_{in}) = \mathsf{send}(pl,o,v_{in})$	$\mathsf{invoke}(pl,o,v_{in},v_{out}) = \mathsf{send}(pl,o,v_{in});\mathsf{rec}(pl,o,v_{out})$
sequence*	$\forall a \in Ev \backslash \{\surd\},\ \dfrac{P \xrightarrow{[g]\ a\ /\ A} P'}{P;Q \xrightarrow{[g]\ a\ /\ A} P';Q}$	$\forall a \in Ev,\ \dfrac{P \xrightarrow{\surd} P' \wedge Q \xrightarrow{[g]\ a\ /\ A} Q'}{P;Q \xrightarrow{[g]\ a\ /\ A} Q'}$
if*	$\text{if c then P else Q} \xrightarrow{[c]\ \tau} P$	$\text{if c then P else Q} \xrightarrow{[\neg c]\ \tau} Q$
while*	$\text{while c } \{P\} \xrightarrow{[c]\ \tau} P;\text{while c } \{P\}$	$\text{while c } \{P\} \xrightarrow{[\neg c]\ \tau} \mathsf{empty}$
scope*	let $EH^{d} = [\{((pl_i,o_i,v_i),P_i)_{i\in I}\},(d,Q),\{(e_j,R_j)_{j\in J}\}]$, $O_I = \{(pl_i,o_i,v_i)_{i\in I}\}$, $\overline{O_I} = \{pl_i.o_i \mid (pl_i,o_i,v_i) \in O_I\}$, $E_J = \{e_{j,j\in J}\}$ in:	
event handler	$\forall (pl_i,o_i,v_i) \in O_I,\ \dfrac{\forall a \in Ex \cup \{\chi,\surd\},\ \neg(P \xrightarrow{a})}{\mathsf{scope}(P,EH^{d}) \xrightarrow{pl_i.o_i?va_m\ /\ v_i:=va_m} P_i}$ with $\exists o_i \in \mathcal{O}(\Sigma_{pl_i}),\ in(o_i) = m$	
passing	$\forall d > 1,\ \dfrac{P \xrightarrow{\chi} P' \wedge \forall a \in Ex \cup \{\tau,\surd\},\ \neg(P \xrightarrow{a})}{\mathsf{scope}(P,EH^{d}) \xrightarrow{\chi} \mathsf{scope}(P,EH^{d-1})}$	$\dfrac{P \xrightarrow{\chi} P' \wedge \forall a \in Ex \cup \{\tau,\surd\},\ \neg(P \xrightarrow{a})}{\mathsf{scope}(P,EH^{1}) \xrightarrow{\chi} Q}$
m		
handler	$\forall e_j \in E_J,\ \dfrac{P \xrightarrow{e_j}}{\mathsf{scope}(P,EH^{d}) \xrightarrow{\tau} R_j}$	$\forall e \in Ex \backslash E_J,\ \dfrac{P \xrightarrow{e}}{\mathsf{scope}(P,EH^{d}) \xrightarrow{e} 0}$
upported fault		
termination	$\dfrac{P \xrightarrow{\surd}}{\mathsf{scope}(P,EH^{d}) \xrightarrow{\surd} 0}$	$\forall a \in Ev,\ \dfrac{hd(a) \not\in (\{\chi,\surd\} \cup Ex \cup \overline{O_I}) \wedge P \xrightarrow{[g]\ a\ /\ A} P'}{\mathsf{scope}(P,EH^{d}) \xrightarrow{[g]\ a\ /\ A} \mathsf{scope}(P',EH^{d})}$
e internals		
pick	$\mathsf{pick}(E) = \mathsf{scope}(\mathsf{time},E)$	
message faults^{+}, flow^{+}, until^{+}: see [14]		

Approach: Technique (2/4)

- **unfolding** STS → SET
- Symbolic Execution
  King (1976), Kurshid et al (2003), Frantzen et al (2006)

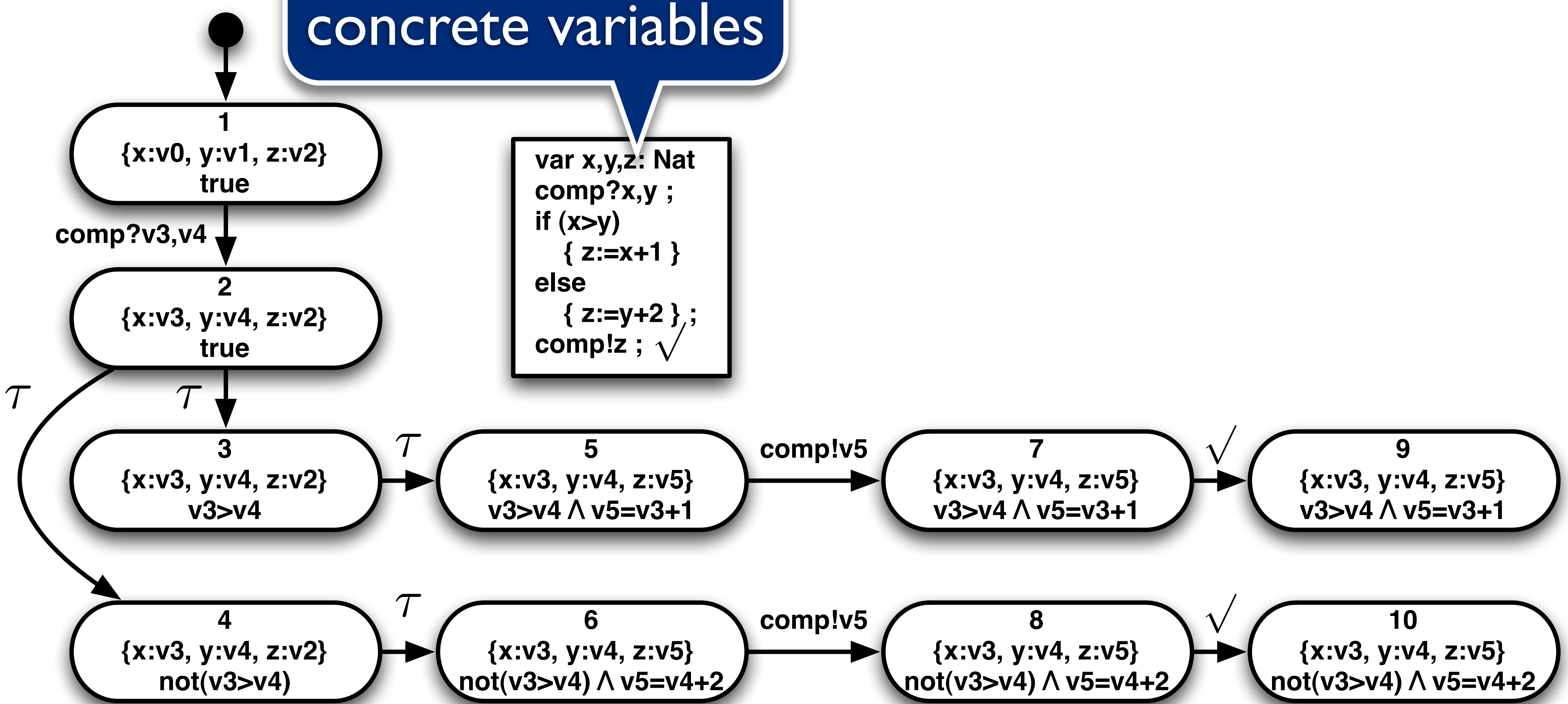


Approach: Technique (2/4)

- **unfolding** STS → SET
- Symbolic Execution
  King (1976), Kurshid et al (2003), Frantzen et al (2006)

size without symbolic execution: [66,565; 132,612]

program with concrete variables




```
var x,y,z: Nat
comp?x,y ;
if (x>y)
   { z:=x+1 }
else
   { z:=y+2 } ;
comp!z ; √
```

Approach: Technique (2/4)

- **unfolding** STS → SET
- Symbolic Execution
  King (1976), Kurshid et al (2

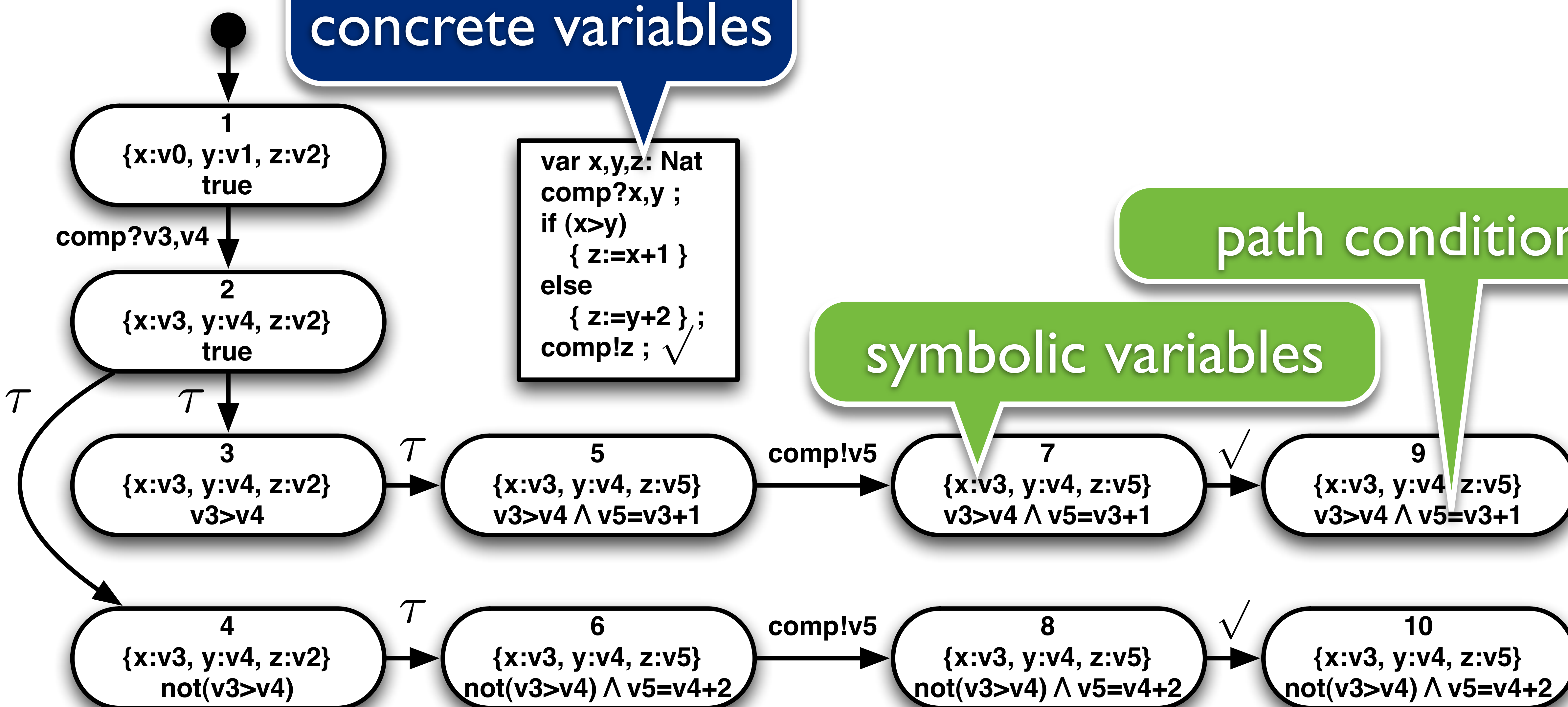
size without symbolic execution: [66,565; 132,612]

size with symbolic execution: [10; 9]

program with concrete variables

symbolic variables

path condition



Approach: Technique (2/4)

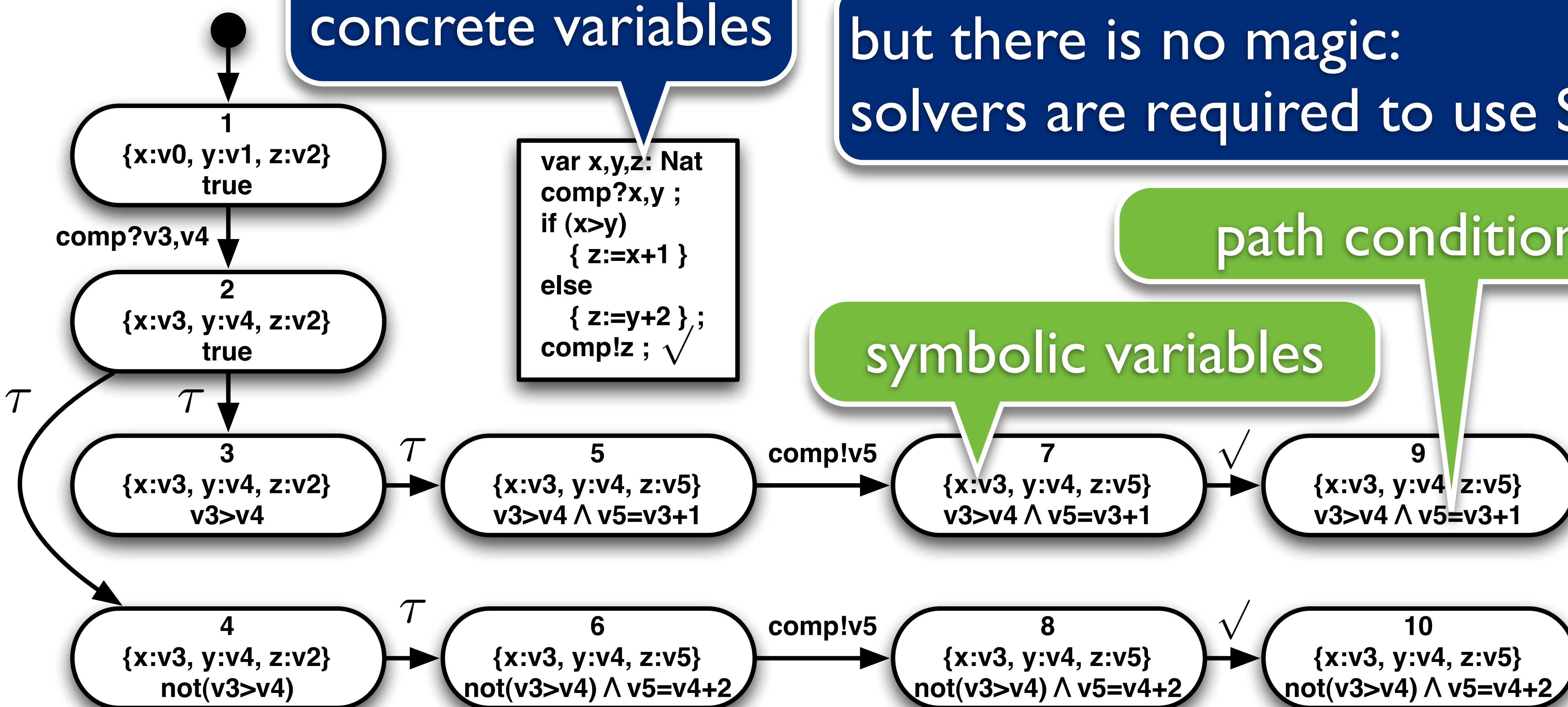
- **unfolding** STS → SET
- Symbolic Execution
  King (1976), Kurshid et al (2

size without symbolic execution:
[66,565; 132,612]

size with symbolic execution:
[10; 9]

but there is no magic:
solvers are required to use SET

program with concrete variables



symbolic variables

path condition

Approach: Technique (3/4)

- **unfolding** STS → SET
- Symbolic Execution
  King (1976), Kurshid et al (2003), Frantzen et al (2006)
- SET size issue:
  - unfolding up to some length k
  - cutting **infeasible paths**
  - use of an **inclusion criterion** over SET nodes
- **online** algorithm to avoid false positives

Approach: Technique (4/4)

- online testing

Algorithm 1: Online Testing Algorithm

```
Data: SET + a distinguished path p, path p = n1 l1 n2 l2 ... l_{k-1} n_k ;
begin
    π = π_k; i := 1; rtr := Pass ;
    while i < k and rtr = Pass do
        switch l_i do
            case USER.e?x_s
                val := (SOLVE(π)[x_s]);
                try {send (e(val)); π := π ∧ x_s = val;}
                catch (e ∈ Ex) { rtr := Fail; }
            case USER.e!x_s
                start TAC;
                try {receive (e(val)); π = π ∧ (x_s = val);
                    if ¬SOLVE(π) then rtr := Fail; }
                catch (timeout_TAC) {rtr := Fail;}
                catch (receive e') { if e' ∈ may(η_i) then rtr := Inconclusive;
                                     else rtr := Fail;}
            case χ
                wait(1 unit of time);
            otherwise
                skip;
        i := i + 1;
    return rtr;
end
```

Approach: Technique (4/4)

- online testing

test passing (SoapUI)

Algorithm 1: Online Testing Algorithm

```
Data: SET + a distinguished path p, path p = n1 l1 n2 l2 ... l(k-1) nk ;
begin
  π = πk; i := 1; rtr := Pass ;
  while i < k and rtr = Pass do
    switch li do
      case USER.e?xs
        val := (SOLVE(π)[xs]);
        try {send (e(val)); π := π ∧ xs = val;}
        catch
      case USER
        start TAC;
        try {receive (e(val)); π = π ∧ (xs = val);
            if ¬SOLVE(π) then rtr := Fail; }
        catch (timeout_TAC) {rtr := Fail;}
        catch (receive e') { if e' ∈ may(ηi) then rtr := Inconclusive;
                             else rtr := Fail;}
      case χ
        wait(1 unit of time);
      otherwise
        skip;
    i := i + 1;
  return rtr;
end
```

path cond. solving (Z3)


```
<soapenv:Envelope xsi:...="http:... >
<soapenv:Body>
<ns2:RequestIn>
  <ns3:uInfo>
    <id>1</id>
    <name>Simpson</name>
    <income>10002</income>
  </ns3:uInfo>
  <ns3:req>
    <amount>10001</amount>
    <maxMonth>12</maxMonth>
    <maxPayment>1000</maxPayment>
  </ns3:req>
</ns2:RequestIn>
</soapenv:Body>
</soapenv:Envelope>
```

Approach: Technique (4/4)

- online testing

test passing (SoapUI)

Algorithm 1: Online Testing Algorithm

```
Data: SET + a distinguished path p, path p = n_1 l_1 n_2 l_2 ... l_{k-1} n_k ;
begin
  π = π_k; i := 1; rtr := Pass ;
  while i < k and rtr = Pass do
    switch l_i do
      case USER.e?x_s
        val := (SOLVE(π)[x_s]);
        try {send ...
        catch ...
      case USE...
        start TAC;
        try {receive (e(val)); π = π ∧ (x_s = val);
            if ¬SOLVE(π) then rtr := Fail; }
        catch (time... TAC) {rtr := Fail;}
        catch ... clusive;
      case χ
        wait(1 unit of time);
      otherwise
        skip;
    i := i + 1;
  return rtr;
end
```

path cond. solving (Z3)

output checking (Z3)


```
<soapenv:Envelope xsi:...="http:... >
<soapenv:Body>
<ns2:RequestIn>
  <ns3:uInfo>
    <id>1</id>
    <name>Simpson</name>
    <income>10002</income>
  </ns3:uInfo>
  <ns3:req>
    <amount>10001</amount>
    <maxMonth>12</maxMonth>
    <maxPayment>1000</maxPayment>
  </ns3:req>
</ns2:RequestIn>
</soapenv:Body>
</soapenv:Envelope>
```

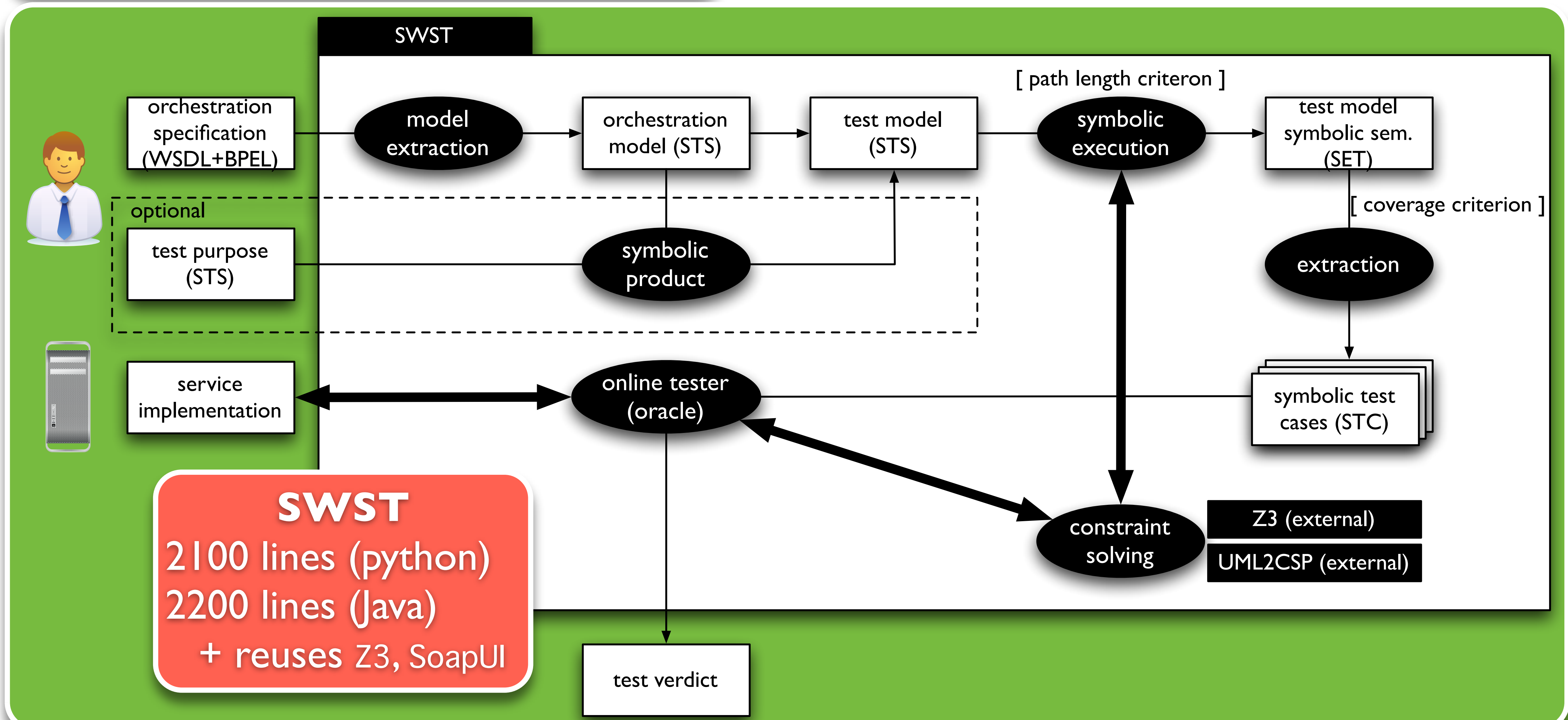
```
<soapenv:Envelope xsi:...="http:... >
<soapenv:Body>
<ns2:RequestOut>
  <status>true</status>
  <fileNumber>1</fileNumber>
  <ns3:proposals>
    <offerCode>1</offerCode>
    <nbMonths>12</nbMonths>
    <monthPayment>918</monthPayment>
    <ns3:rate>
      <type>fixed</type>
      <value>10</value>
    </ns3:rate>
  </ns3:proposals>
</ns2:RequestOut>
</soapenv:Body>
</soapenv:Envelope>
```

Contributions on Testing

L. Bentakouk PhD thesis

black-box symbolic approach
active **online** testing
application to **WS orchestration**

TESTCOM/FATES'09
TAP'11



SWST
2100 lines (python)
2200 lines (Java)
+ reuses Z3, SoapUI

Contributions on Testing

L. Bentakouk PhD thesis

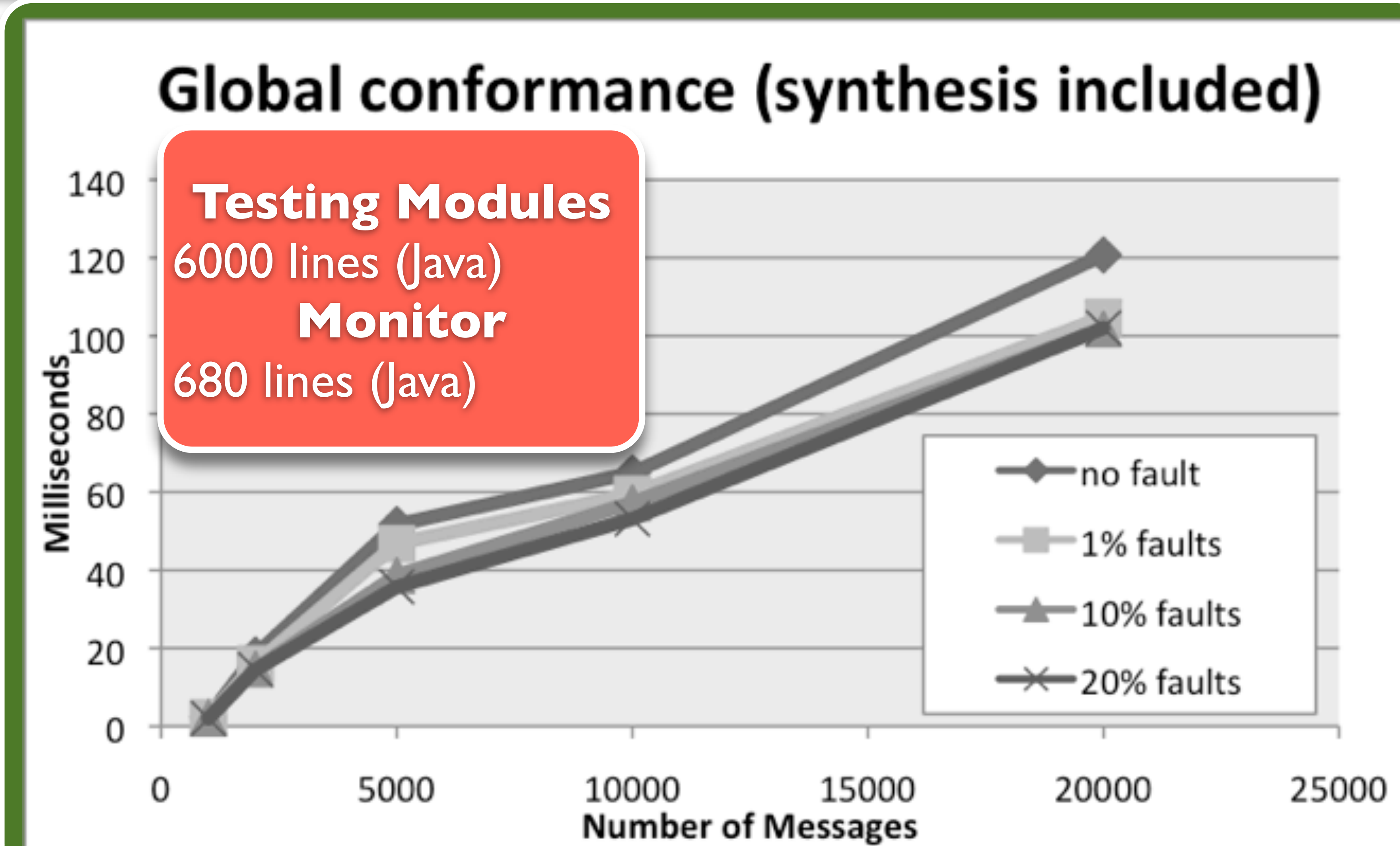
black-box symbolic approach
active **online** testing
application to **WS orchestration**

TESTCOM/FATES'09
TAP'11

H.N. Nguyen PhD thesis

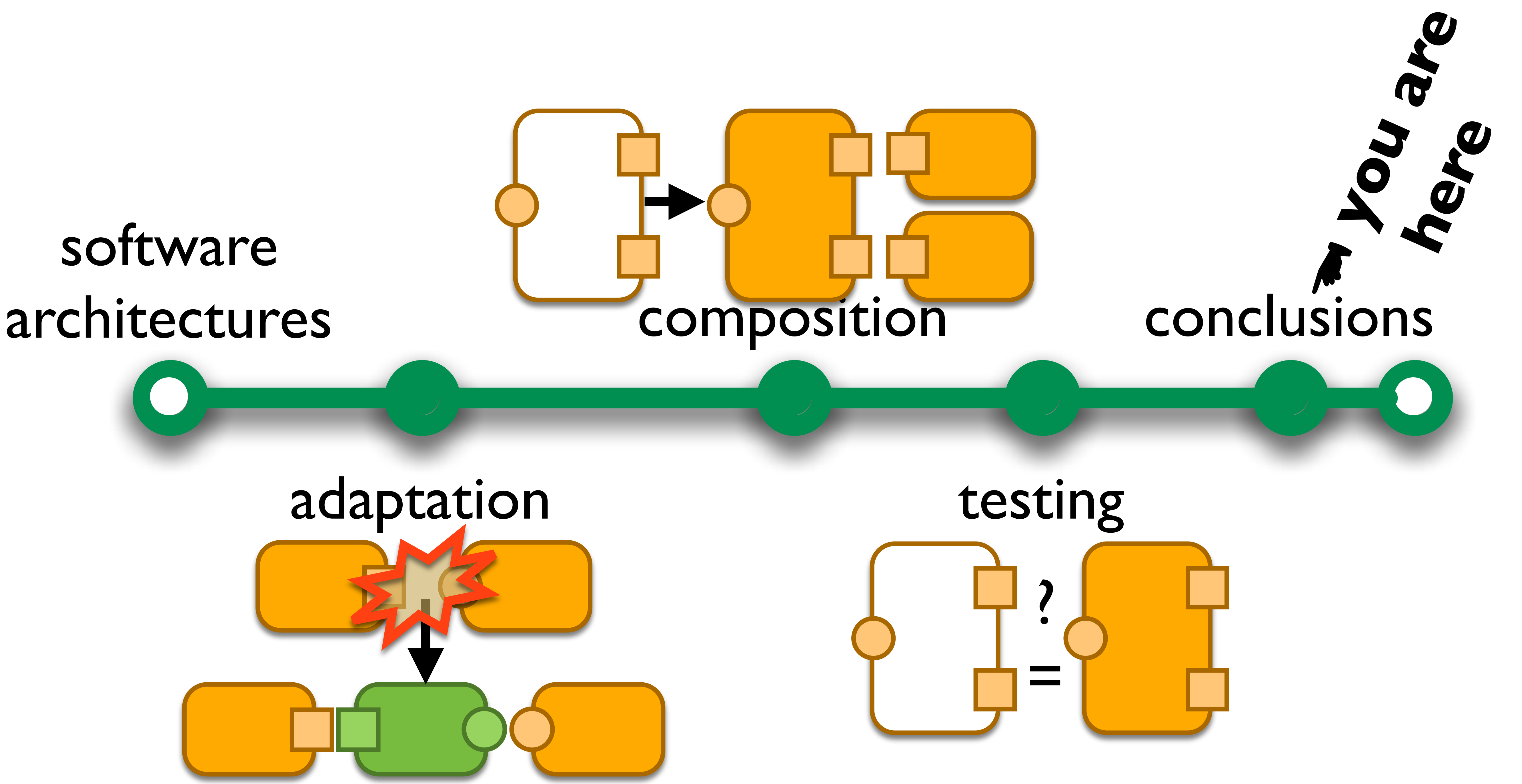
black-box testing
passive offline testing
application to **WS choreography**
local and global conformance

SAC'12



Testing Modules
6000 lines (Java)
Monitor
680 lines (Java)

Agenda



you are here
software architectures
composition
conclusions
adaptation
testing
?
=

Conclusions

suported by

Project PERSO (leader)
adaptation & composition
Project WebMoV
testing

«in-the-large» works

FMOODS'06, FACS,06, FMOODS'07, FACS'07, FORTE'07, FASE'08, ICSOC'08
IEEE TSE 34(4), 2008
IEEE TSE under press

ICSOC'08, ISoLa'10
ICWS'10, ICSOC'10

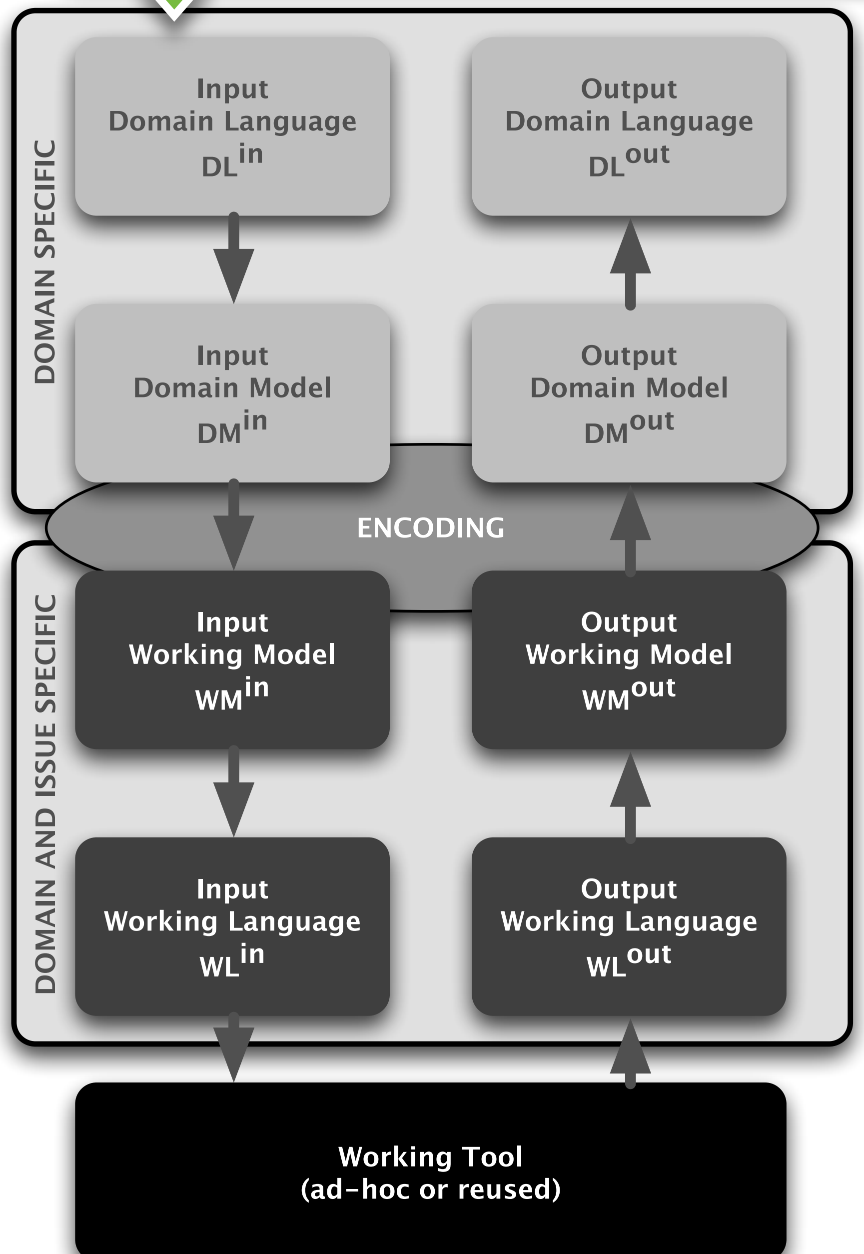
TESTCOM'09, TAP'11
SAC'12

SAC'12

issue	type	DL^{in}	DM^{in}	WM	DM^{out}	DL^{out}
adaptation	centralised	WWF	Trans. Syst.	Trans. Syst. Petri Net	Trans. Syst.	WWF
		WSDL BPEL	Symbolic Trans. Syst.	Process Algebra	Symbolic Trans. Syst.	BPEL
	distributed	SAWSDL BPEL	Trans. Syst.	Trans. Syst.	Trans. Syst.	BPEL
		Workflow	Event Structure	Petri Net	Event Structure	
composition + repair	centralised	WSDL OWL Workflow	Planning	Planning	Planning	BPEL
testing	centralised	ABPEL	Symbolic Trans. Syst.	Symbolic Exec. Tree	Symbolic Test Cases	SOAP
	distributed	Chor SOAP logs	Trace	Trace	n/a	n/a
realisability	distributed	BPMN 2.0	Workflow	Process Algebra	n/a	n/a

Conclusions

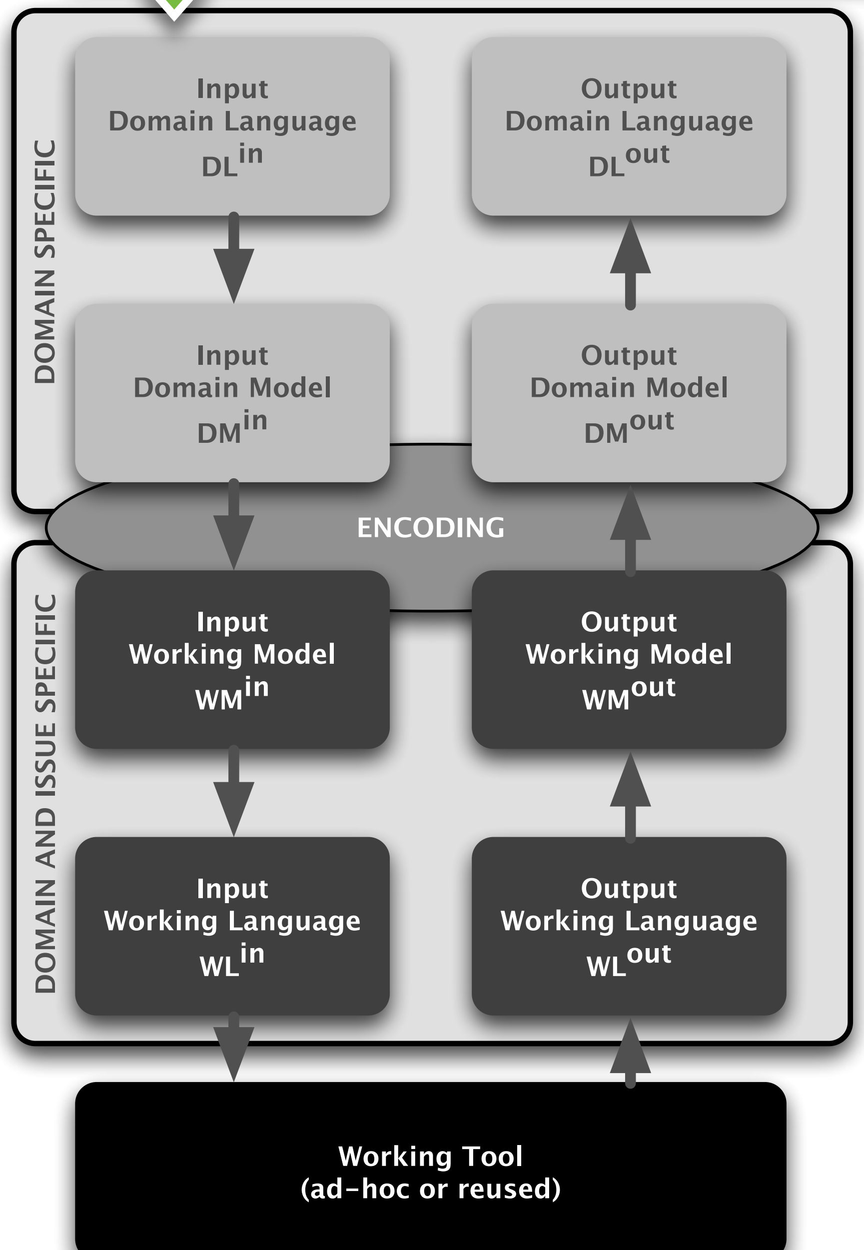
1. a general approach



issue	type	DL^{in}	DM^{in}	WM	DM^{out}	DL^{out}
adaptation	centralised	WWF	Trans. Syst.	Trans. Syst. / Petri Net	Trans. Syst.	WWF
		WSDL BPEL	Symbolic Trans. Syst.	Process Algebra	Symbolic Trans. Syst.	BPEL
	distributed	SAWSDL BPEL	Trans. Syst.	Trans. Syst.	Trans. Syst.	
		Workflow	Event Structure	Petri Net	Event Structure	
composition + repair	centralised	WSDL OWL Workflow	Planning	Planning	Planning	BPEL
testing	centralised	ABPEL	Symbolic Trans. Syst.	Symbolic Exec. Tree	Symbolic Test Cases	SOAP
	distributed	Chor SOAP logs	Trace	Trace	n/a	n/a
realisability	distributed	BPMN 2.0	Workflow	Process Algebra	n/a	n/a

Conclusions

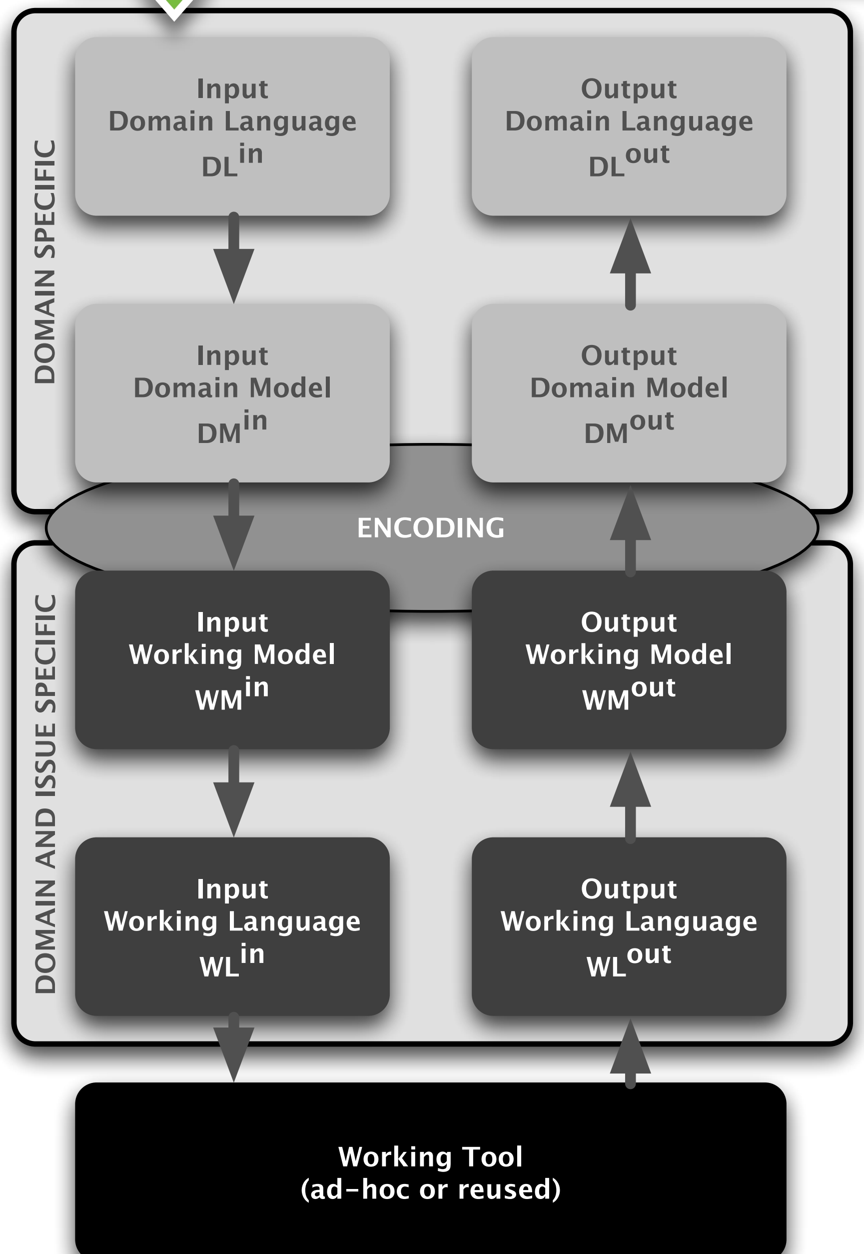
1. a general approach
- Domain Specific Languages / Models



issue	type	DL^{in}	DM^{in}	WM	DM^{out}	DL^{out}
adaptation	centralised	WWF	Trans. Syst.	Trans. Syst. / Petri Net	Trans. Syst.	WWF
		WSDL BPEL	Symbolic Trans. Syst.	Process Algebra	Symbolic Trans. Syst.	BPEL
	distributed	SAWSDL BPEL	Trans. Syst.	Trans. Syst.	Trans. Syst.	
		Workflow	Event Structure	Petri Net	Event Structure	
composition + repair	centralised	WSDL OWL Workflow	Planning	Planning	Planning	BPEL
testing	centralised	ABPEL	Symbolic Trans. Syst.	Symbolic Exec. Tree	Symbolic Test Cases	SOAP
	distributed	Chor SOAP logs	Trace	Trace	n/a	n/a
realisability	distributed	BPMN 2.0	Workflow	Process Algebra	n/a	n/a

Conclusions

1. a general approach
- Domain Specific Languages / Models
- Issue Specific Languages / Models

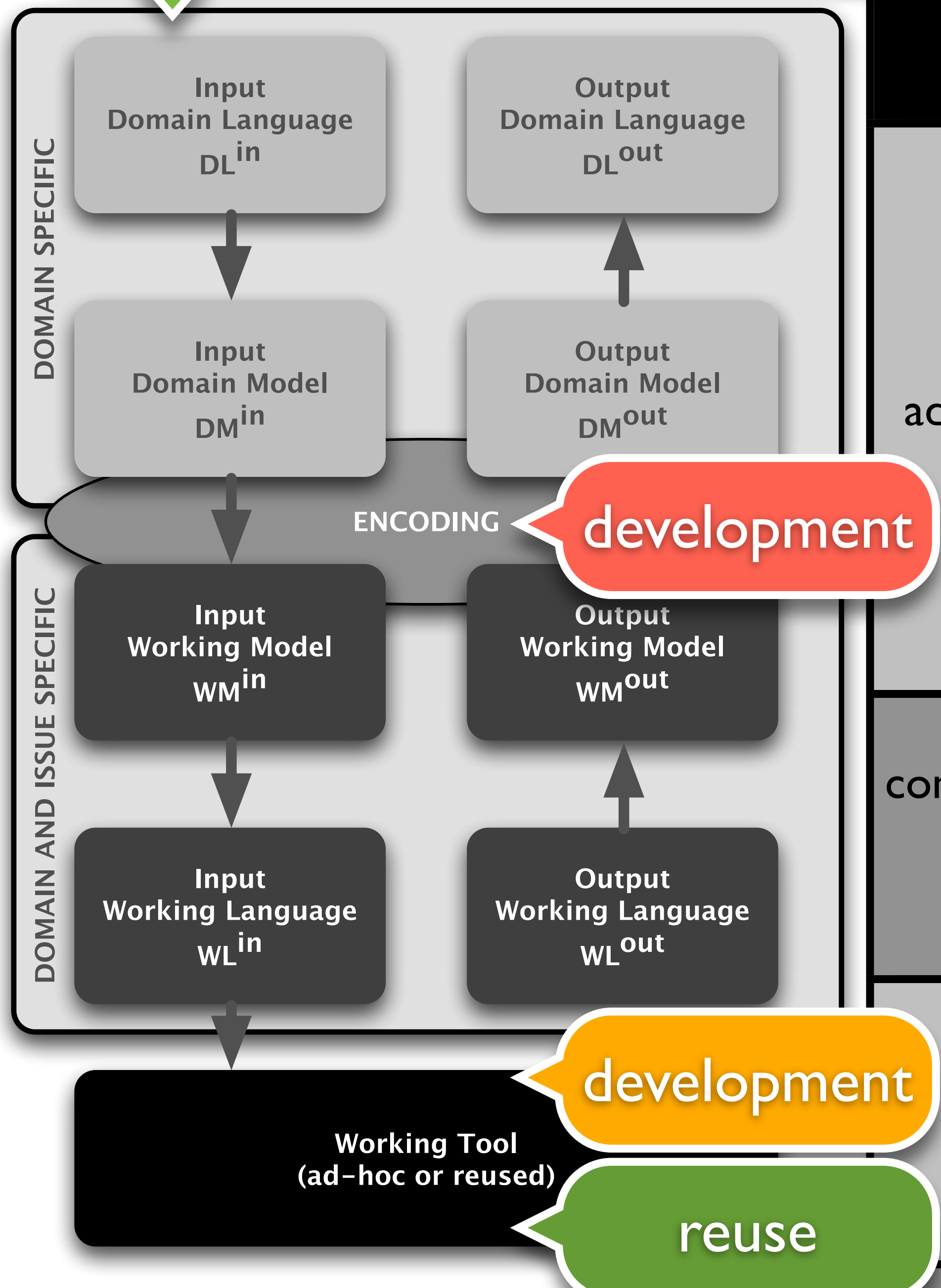


issue	type	DL^{in}	DM^{in}	WM	DM^{out}	DL^{out}
adaptation	centralised	WWF	Trans. Syst.	Trans. Syst. Petri Net	Trans. Syst.	WWF
		WSDL BPEL	Symbolic Trans. Syst.	Process Algebra	Symbolic Trans. Syst.	BPEL
	distributed	SAWSDL BPEL	Trans. Syst.	Trans. Syst.	Trans. Syst.	
		Workflow	Event Structure	Petri Net	Event Structure	
composition + repair	centralised	WSDL OWL Workflow	Planning	Planning	Planning	BPEL
testing	centralised	ABPEL	Symbolic Trans. Syst.	Symbolic Exec. Tree	Symbolic Test Cases	SOAP
	distributed	Chor SOAP logs	Trace	Trace	n/a	n/a
realisability	distributed	BPMN 2.0	Workflow	Process Algebra	n/a	n/a

Conclusions

1. a general approach
- Domain Specific Languages / Models
- Issue Specific Languages / Models

encodings DM ⟷ WM



issue	type	DL^{in}	DM^{in}	WM	DM^{out}	DL^{out}
adaptation	centralised	awf2lts (U. Malaga)		adaptor CADP, TINA	lts2awf (U. Malaga)	
		BPEL2STS compositor		scrutator CADP	STS2BPEL	
	distributed	(ongoing)		LOLA	(ongoing)	
composition + repair	centralised	pycompose PGA		Graphplan PDDLGP Blackbox	WF2BPEL	
testing	centralised	BPEL2STS		SWST Z3	SWST Z3, SoapUI	
	distributed	Chor, Monitoring, and Synthesis Modules		Conformance Modules	n/a	
realisability	distributed	BPMN2Py Py2LNT		CADP	n/a	

Conclusions

II. no unique language
no unique model

- we have to face existing DL^{in}/DL^{out}
- DM^{in}/WM/DM^{out} have specificities

transition systems

+ simplicity, symbolic extensions
+ tool support
- //ism implementation

process algebras

+ on-the-fly tool support
- non symbolic models

event structures

+ workflow encoding
+ //ism implementation
- loops in behaviours
- tool support

Petri nets

+ interleaving or true concurrency
+ interaction or resource viewpoint
+ workflow encoding
+ //ism implementation
+ tool support

ssue	type	DL^{in}	DM^{in}	WM	DM^{out}	DL^{out}
ptation	centralised	WWF	Trans. Syst.	Trans. Syst. / Petri Net	Trans. Syst.	WWF
	centralised	WSDL BPEL	Symbolic Trans. Syst.	Process Algebra	Symbolic Trans. Syst.	BPEL
	distributed	SAWSDL BPEL	Trans. Syst.	Trans. Syst.	Trans. Syst.	BPEL
	distributed	Workflow	Event Structure	Petri Net	Event Structure	BPEL
position + epair	centralised	WSDL OWL Workflow	Planning	Planning	Planning	BPEL
esting	centralised	ABPEL	Symbolic Trans. Syst.	Symbolic Exec. Tree	Symbolic Test Cases	SOAP
esting	distributed	Chor SOAP logs	Trace	Trace	n/a	n/a
isability	distributed	BPMN 2.0	Workflow	Process Algebra	n/a	n/a

Perspectives

H.N. Nguyen PhD thesis

eternal peer composition

online and **distributed** approach
using test, diagnosis, and repair
model retrieval (concurrency + data)

submitted project

verification of BPMN 2.0 choreographies

industrial application
language **expressiveness**
compositional verification

event structures
coloured Petri nets
solvers / provers

R. Khéfifi PhD thesis

resource-centric composition

industrial application
new applicative domain (personal info.)
REST vs SOAP services

ANR

adaptation

rich semantics (pre/post)
true concurrency

tools and interconnection

WS versions
(some already exist)

Formal Model-Based Approaches for the Development of Composite Systems

MeFoSyLoMa Seminar
(originally, Habil. thesis defense, Nov. 24th, 2011)

Pascal Poizat
Université d'Evry Val d'Essonne;
LRI CNRS UMR 8623 et Université Paris-Sud 11

Evry, February 17th, 2012