SUMMERSOC Χερσόνησος June 19, 2019

Conceptual Foundations of Service Orientation and beyond

Wolfgang Reisig
Humboldt-Universität zu Berlin

Introduction: Two remarks

1. A short remark on

   Einstein Center *Digital Future*

develop embedded systems

in the current buzzwords areas:

Health, mobility, industries, services, …

Executive Board

“ When I look into the digital future ...

...I see great need of

harmonizing unlimited potential

of a planet-sized Universal Turing machine …”

What he wants to say:

...I see great need of

a Unifying Theory for really big

(“planet sized”) Digital Infrastructures

Introduction: Two remarks

2. A short remark on
yesterday's talk of Schahram
On Digital Ecosystems

"nobody is looking at the whole zoo"
… at least two do: him and me

Elements of the zoo:

Resiliance
Elasticity
Granularity
Accessability
Block chain
Elastic contract
Decentralization
Sensing
Everything as a service
Micro-services, ~ data,
~ computing, ~activities

A unifying Theory for all this
will require notions motivated by theory:

canonical, universal principles,
independent of semantical details

… inspired by nature

Conceptual Foundations of Service Orientation and beyond

Long term objective:

a unifying Theory for

The Digital infrastructure

The Digital ecosystem

Contents of this talk:

four aspects

1. Models
2. Invariants
3. Distributedness
4. Components and composition

1. Models

Models in science:

THE central concept

May be quite difficult

Distance to „the modeled“ may be large

Monday: a quantum computing model
may be quantum immplemented in
different ways

1.1 Models in Informatics

Late 1960ies:

„Software crisis“

Proposed soluton:

Better programming languages

Big ones: Algol 68, PL1

Small ones: Pascal

Later on: ADA; OO, DSLs

„Programs as models“

Focus on implementation

Not on the problem

or the algorithmic idea

"Programs as models"
Example: Dijkstra's Pebble Game

Given: An urn, containing finitely many black and white pebbles.

… how to model this behavior ?

The actions:

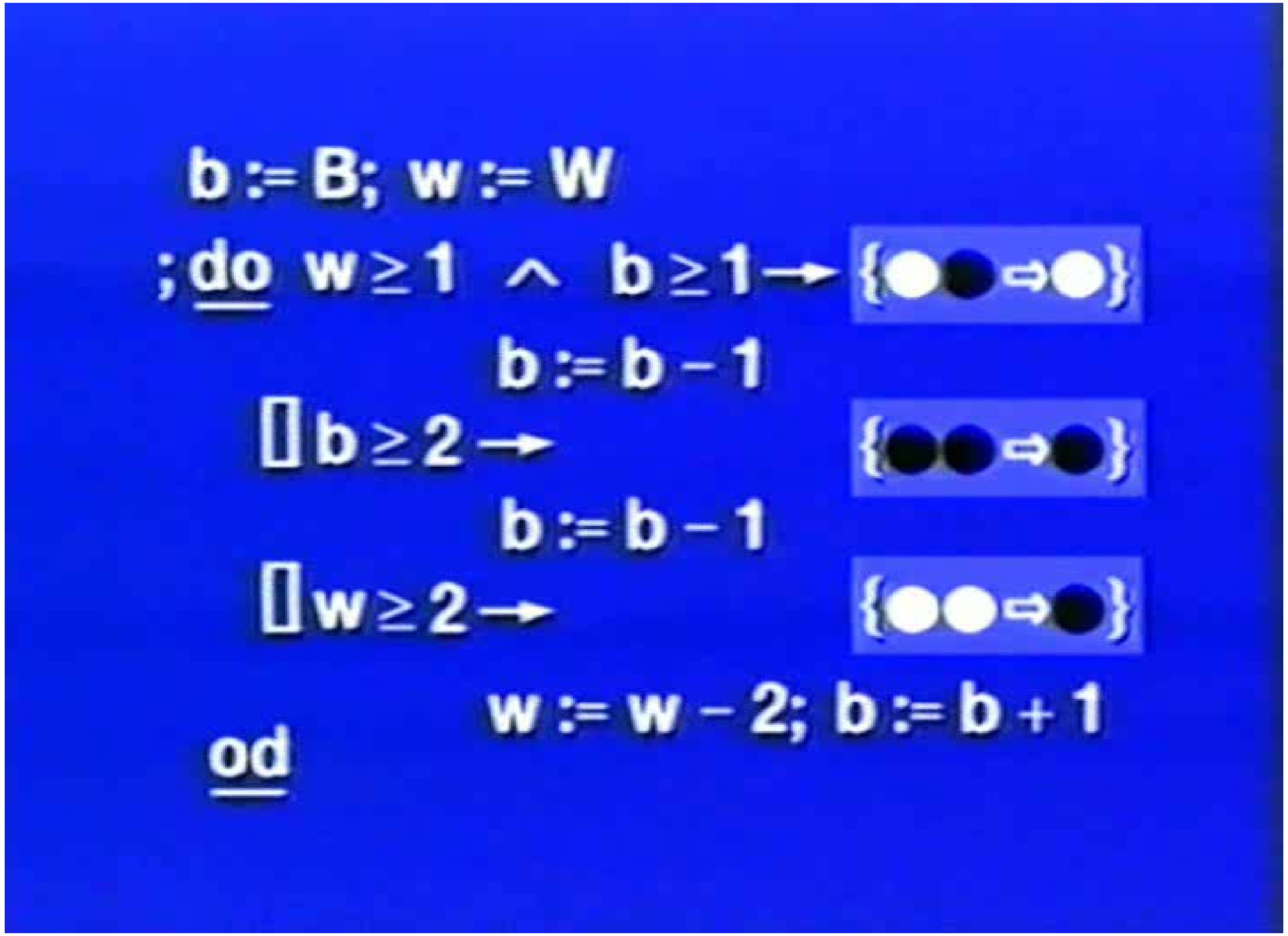
Repeat as long as possible:

(1) remove two pebbles, a, b
return one pebble, c .

(2) If colour of a *and* b *differ:*
c *is white;*
otherwise, c *is black.*

Every 5 year child can play that game.

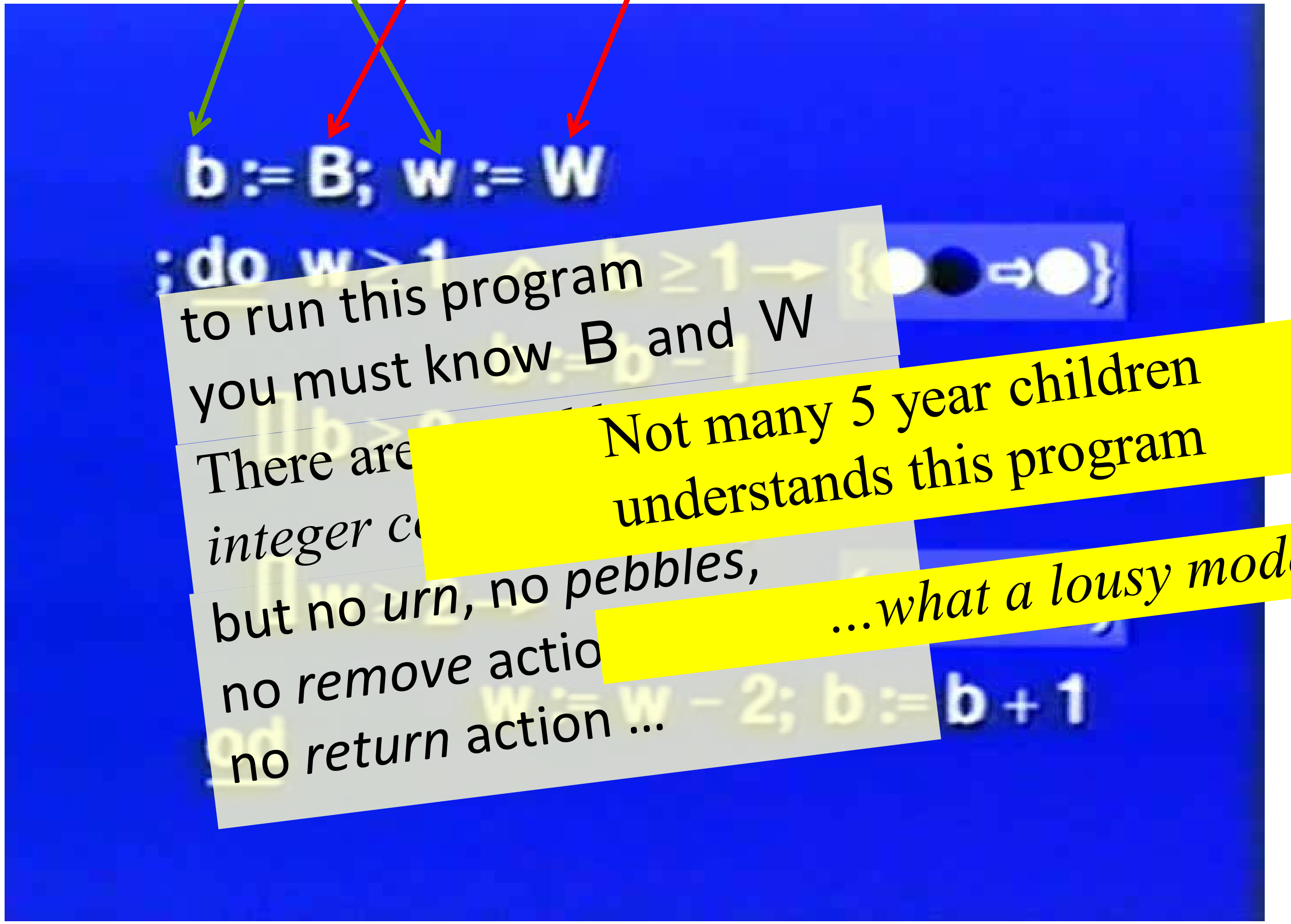
Dijkstra‘s model



Dijkstra‘s model

initial number of black and white pebbles

integer variables



to run this program
you must know B and W

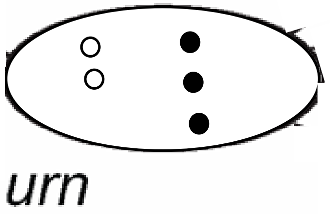
There are integer c…

Not many 5 year children
understands this program

but no *urn*, no *pebbles*,
no *remove* actio…
no *return* action …

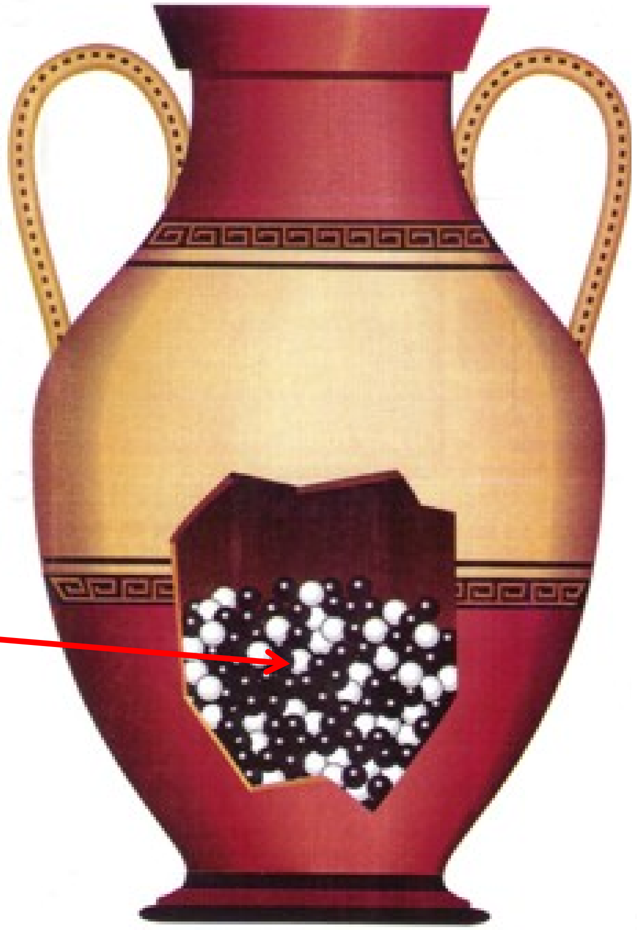
…what a lousy model!

A Petri Net model

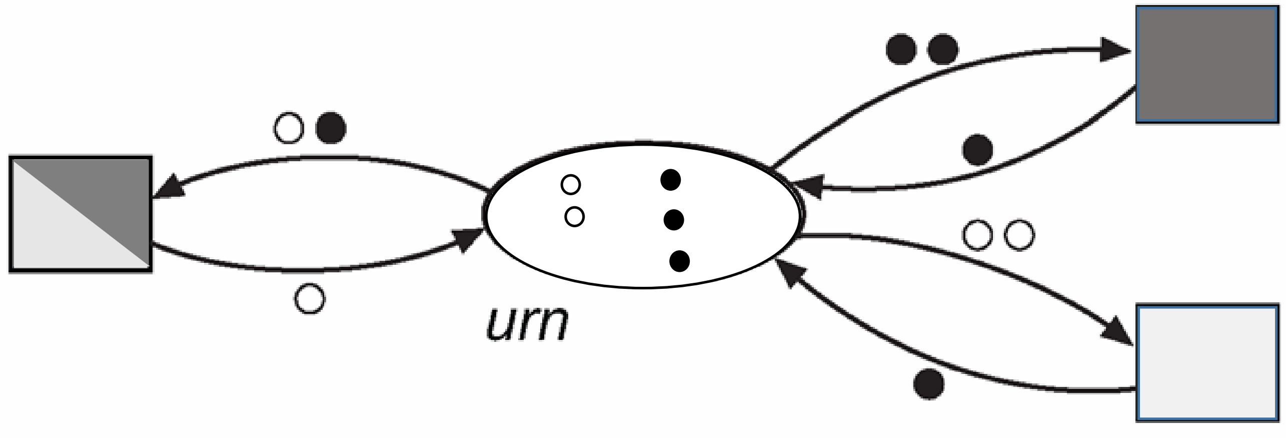


A schema for *infinitely many* initial states:

PEBBLES : a symbol, to be interpreted
by a set of black and white pebbles.

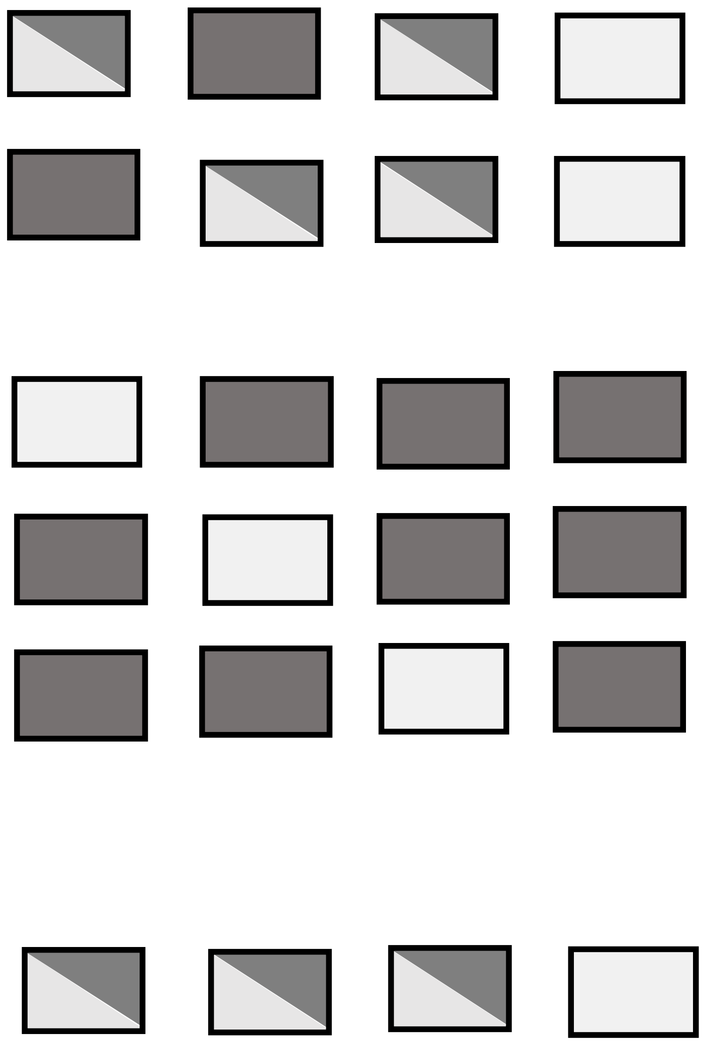


An instantiation

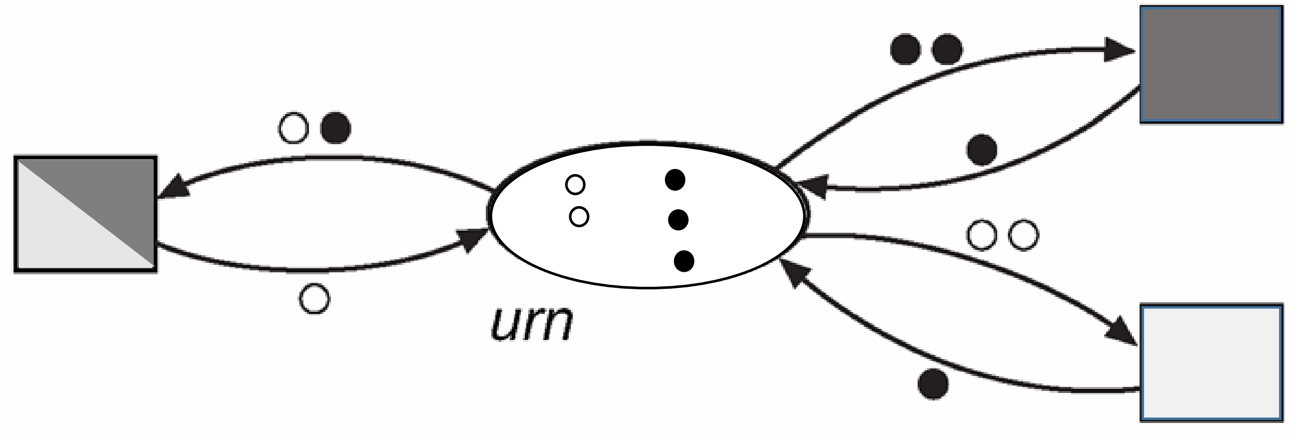


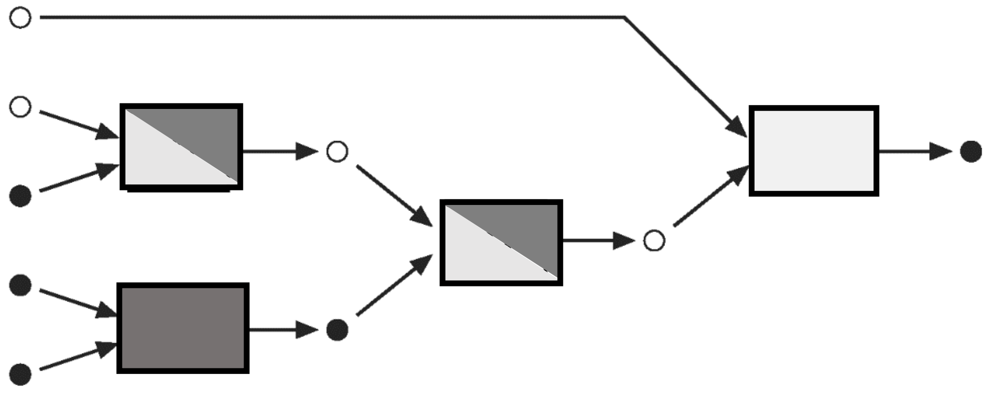
… and its
sequential behaviors

Sequ. Beh.: a sequence of *actions*

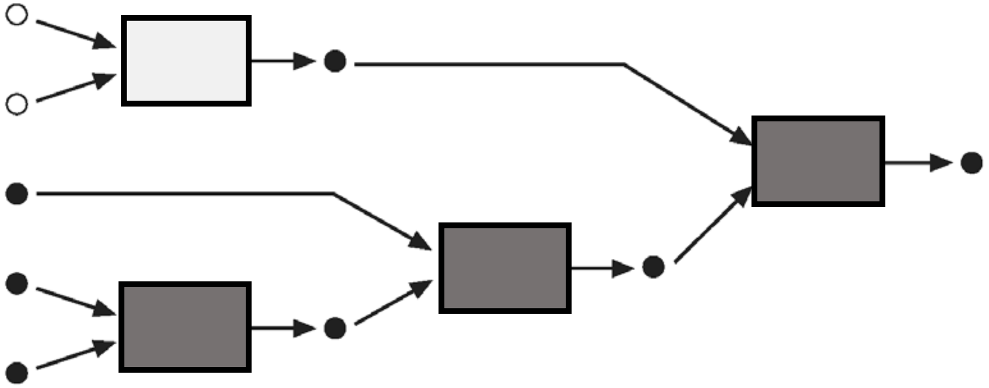


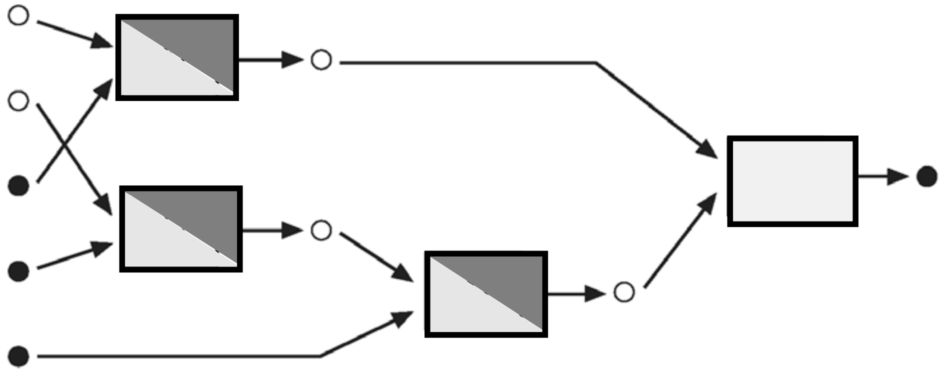
An instantiation





… and its
distributed behaviors





Modeling languages

executable modeling languages:

actors

Stream based functions

TLA

UNITY

“pure” modeling languages

SADT

Petri Nets

Statecharts

Later on: UML

UML

Grady Booch, (2004):

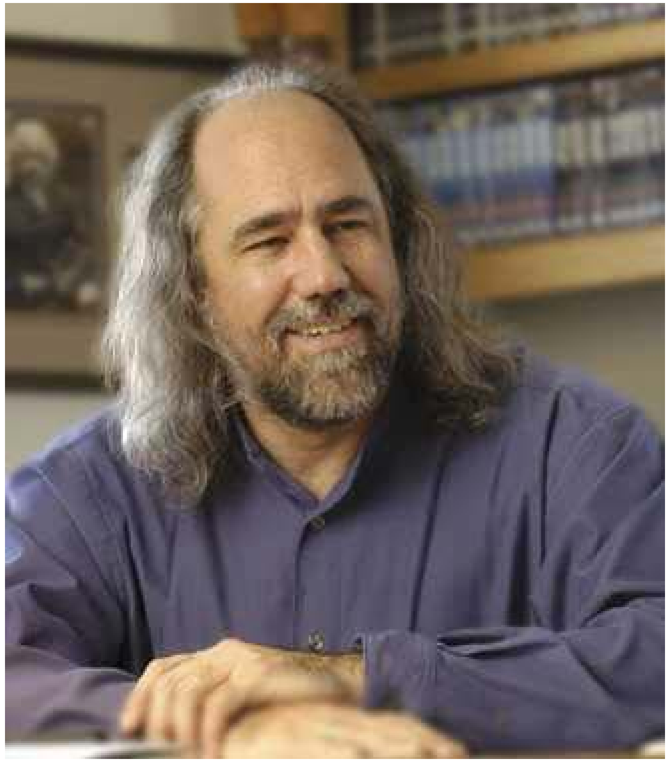
We must "*elevate models*

as to a first class citizenship ...

a peer of traditional text languages

(and potentially its master)".

"*models as products*".



THE fundamental difference to programming:

1. Modeled behavior is not necessarily implemented.
2. The modeler freely chooses the level of abstraction

2. Invariants

What is a good scientific theory?
It provides good descriptions
("Models") of the realm of interest

What is a good model?
It allows good predictions
of future behavior
and the derivation of
other interesting properties
of realm.

Central aspect:
Invariants

generally:
"what remains constant
while a system proceeds"

Invariants in sciences

A physical process preserves

the involved amount of energy.

$e = mc^2$

A motor has a maximal torsion moment

(2nd derivative of speed)

A chemical reaction preserves

the involved atoms:

$$CH_4 + 2\ O_2 \rightarrow CO_2 + 2\ H_2O.$$

Theoretical Biology ("Systems Biology"):

Metabolism retains the amount of matter

Invariants govern the design of scientific notions

2.1 Invariants in Informatics

Classical programs: C.A.R. Hoare’s invariant calculus

Petri nets: Place invariants, transition invariants

temporal logic

Not too impressive

More general invariants

What remains invariant when using

- a cash machine

Amount of
money in
account
+ in hand

- a garbage collector
- a communication protocol
- an elevator control
- a telephone switching system
- a seat booking system

?

We need basic notions based on invariants!

… a notion of *"information":*

As long as the computer
does not interact with others,
the amount of information
within the computer
remains constant.

Information is what you use
to decide an alternative.

stated differently:

During computation,
information is re-ordered,
but neither generated
nor destroyed.

This notion then may imply
a god notion of *privacy*.

3. Distributedness

In the Digital infrastructure,
In the Digital ecosystem
there is nothing “global”!

Every action/event/activity
has *local* causes and local effects.

Autonomy is inevitable

What about the
planet-sized *Universal* Turing machine ?
A distributed version …

A distributed interpreter
running on any distributed architecture
interpreting all distributed descriptions
of distributed systems that are to run
on a distributed architecture …

Does such a thing exist?

1. No
2. Yes, for the prize of probability

As Sefanie showed us on Monday
in the case of quantum computing

Paxos Protocol

3. Distributedness

Are probabilistic, distributed algorithms
the standard, basic notion
for a theory of the Digital ecosystem?

4. Components and composition

The Digital Ecosystem is *structured*.

Structure: frequently *hierarchical*

To this end, required:

Canonical, universal principles,

independent of semantical details.

Specification: abstract

Implementation: refined

Required property:

specification implies implementation

Refinement calculus, encapsulation, …

4. Components and composition

The Digital Ecosystem is *structured*.

Structure: frequently *composed*, shaped

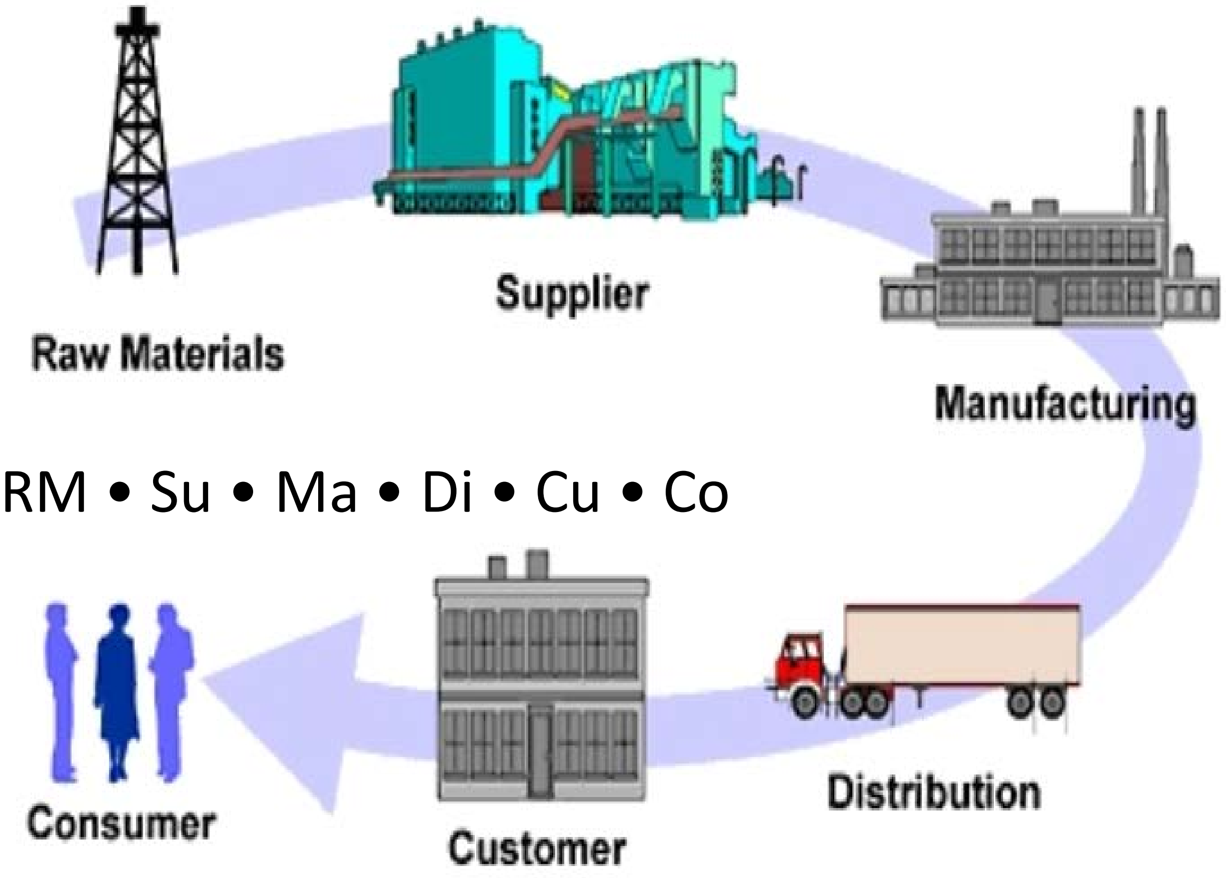
$A_1 \bullet A_2 \bullet \ldots \bullet A_n$, with components A_i .

To this end, required:

Canonical, universal principles,

independent of semantical details.

Example: A supply chain



RM • Su • Ma • Di • Cu • Co

Structure: frequently *composed*, shaped $A_1 \bullet A_2 \bullet \dots \bullet A_n$, with components A_i .

To this end, required:

Canonical, universal principles,

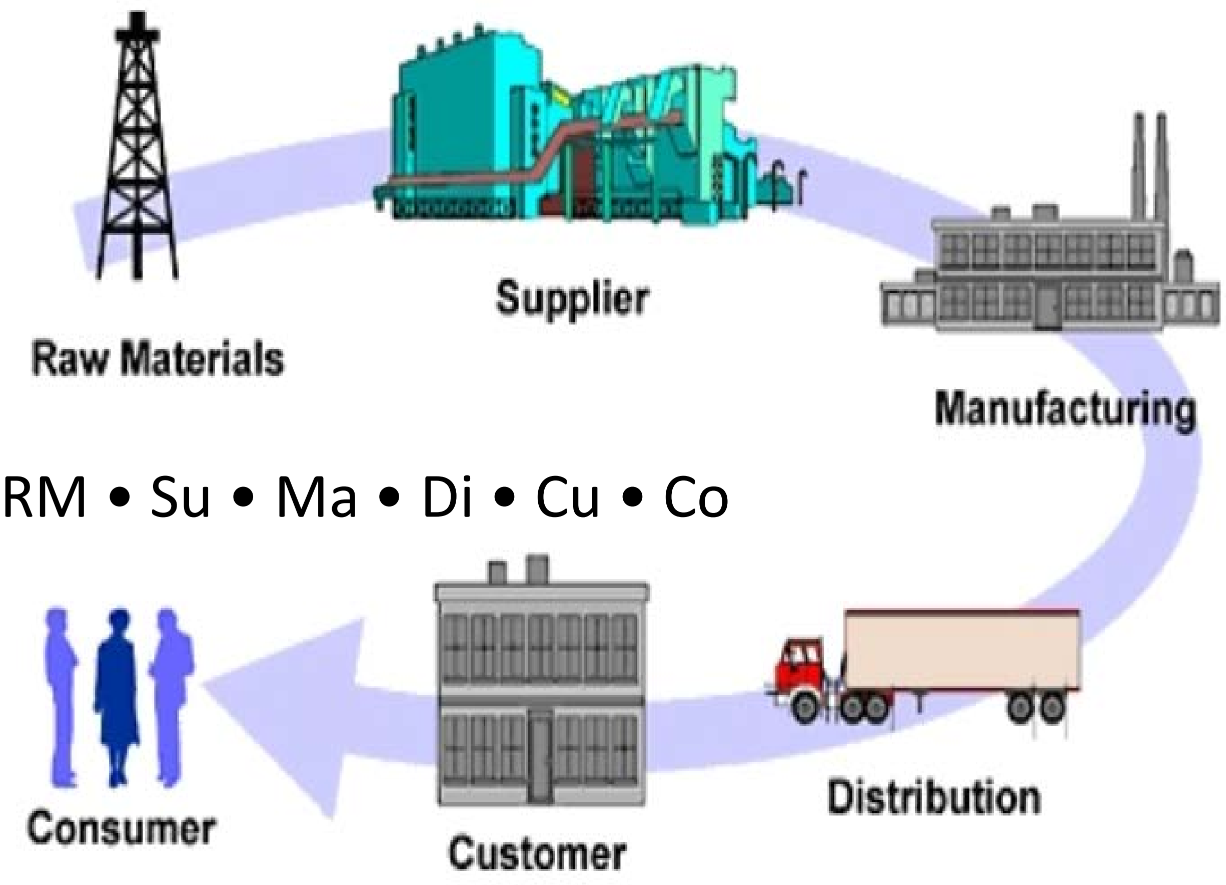
independent of semantical details.

Event based systems: PubSub

SOC: provider/requester/broker

So, what are (the) fundamentals of composition?

Let‘s start with *components*



RM • Su • Ma • Di • Cu • Co

Structure: frequently *composed*, shaped $A_1 \bullet A_2 \bullet \ldots \bullet A_n$, with components A_i .

To this end, required:

Canonical, universal principles,

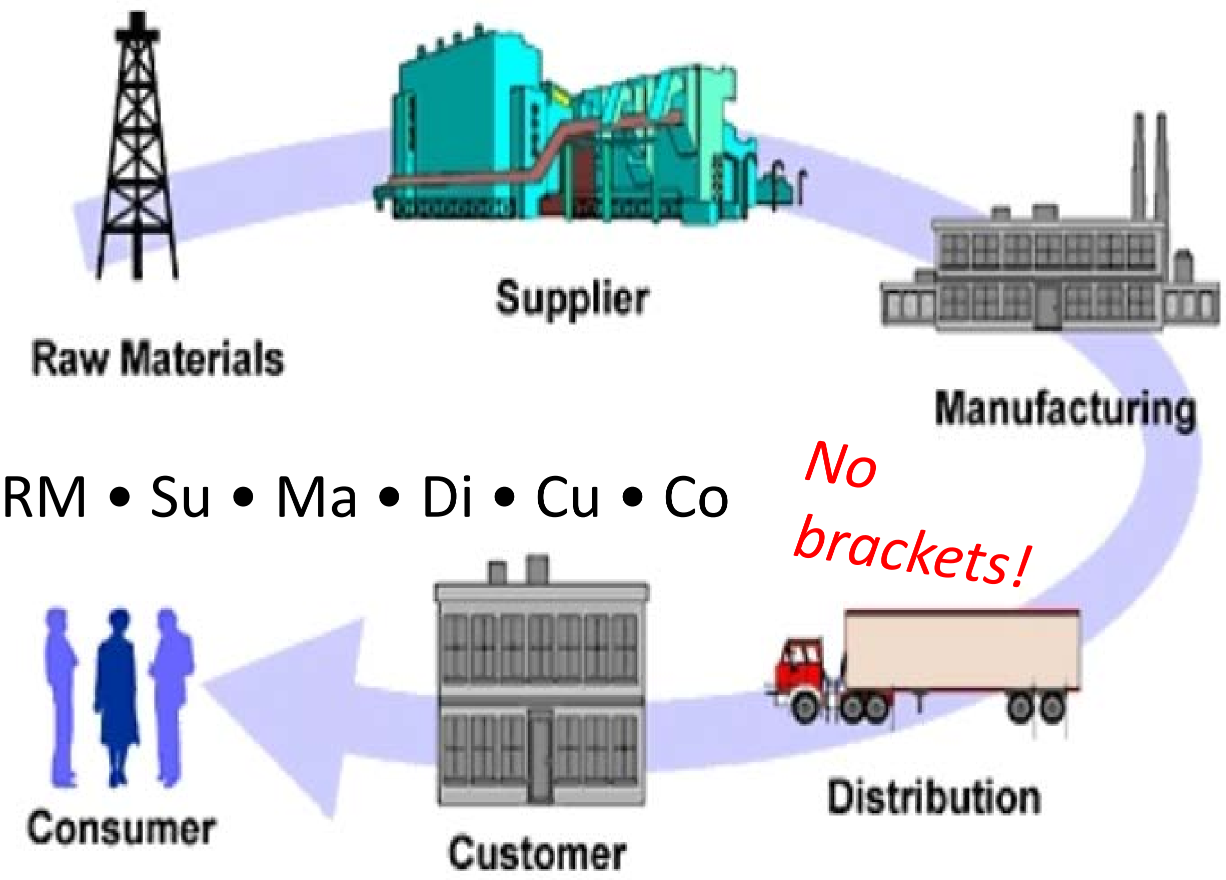
independent of semantical details.

A *component* consists of

- its *interface* and
- its *inner structure*.
- Interface: strict, formal.
- Inner structure: liberal.
- Composition: formal;

  dependig only on the interface.

Let‘s start with *components*



RM • Su • Ma • Di • Cu • Co

No brackets!

Structure: frequently *composed*, shaped $A_1 \bullet A_2 \bullet \ldots \bullet A_n$, with components A_i .

To this end, required:

Canonical, universal principles,

independent of semantical details.

Fundamental:

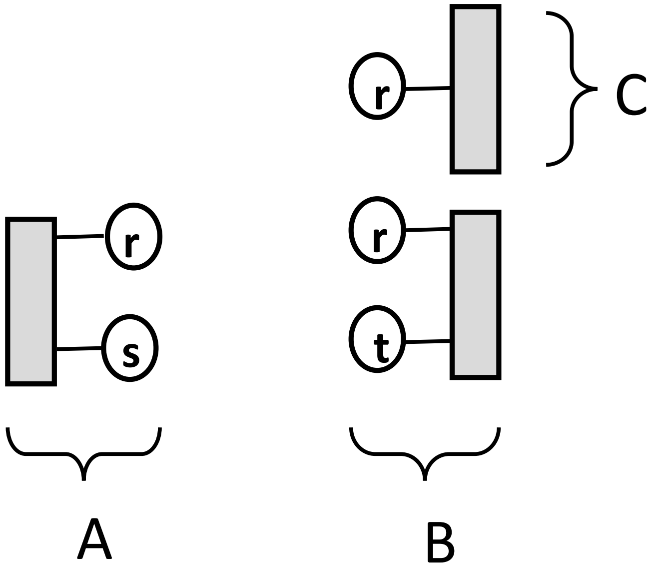
Composition is associative:

$$(A \bullet B) \bullet C \;=\; A \bullet (B \bullet C).$$

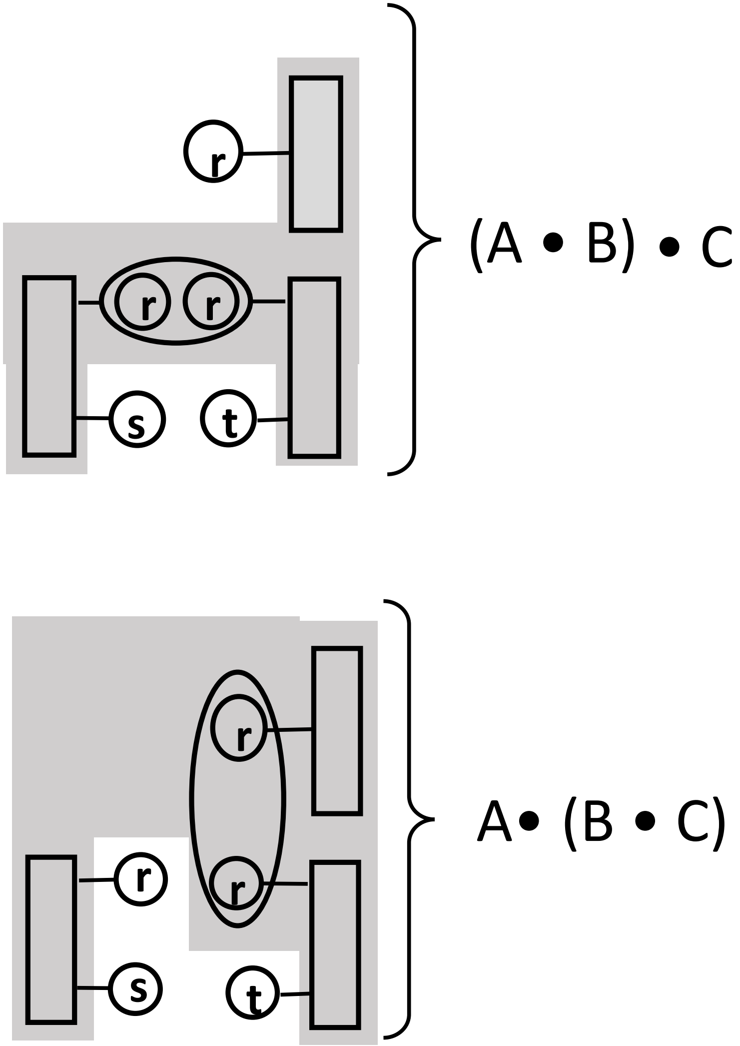
Assumed without saying

An obvious start

An interface is
a set of *labeled gates.*
Composition:
Glue equally labelled gates,
remove them from
the interfaces.



• is not associative!



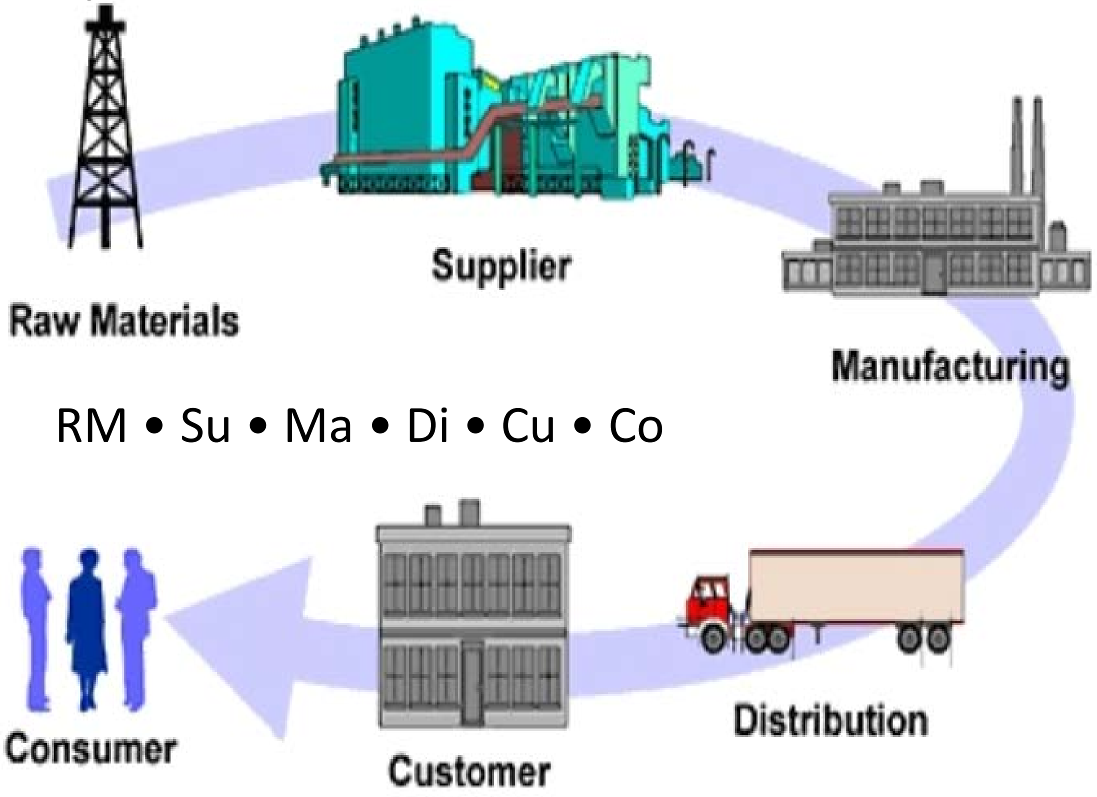
Hoe solve this problem?

- ignore it
- restrict choice of labels
- define „n-fold composition“
- …

… our proposal:

A closer look:

Beispiel:



RM • Su • Ma • Di • Cu • Co

- Frequently, the interface is 2-*partitioned:*
- ***input*** and ***output*,**
- ***customer*** and ***supplier*,**
- ***requester*** and ***provider*,**
- ***consumer*** and ***producer*,**
- ***buy side*** and ***sell side*,**
- ***predecessor*** and ***successor*,**
- ***assumptions*** and ***guaranties*,**
- ***pull*** and ***push*,**
- etc.

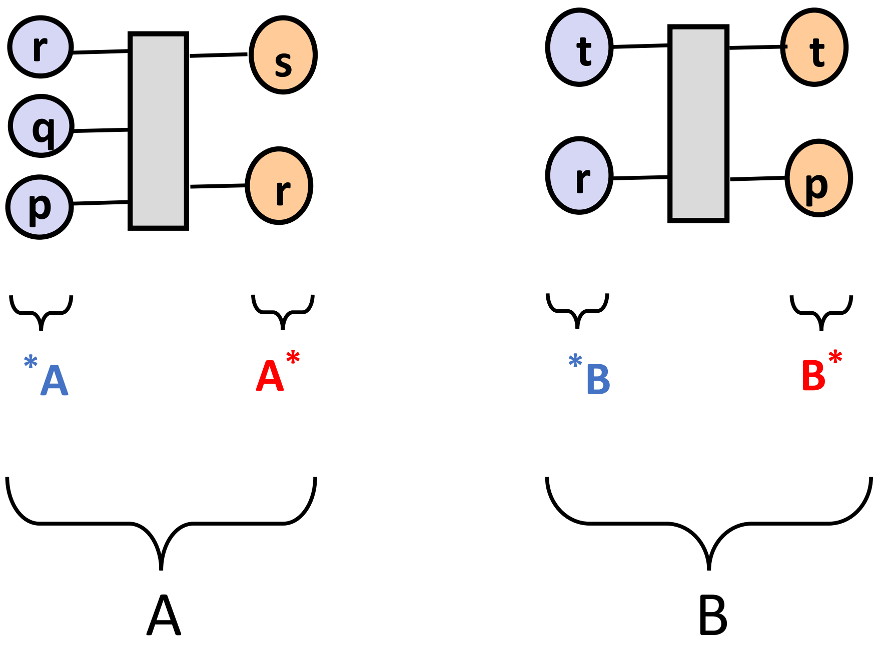
A component **C** frequently has

a *left* and a *right* interface, ***C** und **C***.

composition **A • B**:

Glue gates of **A*** und ***B** .

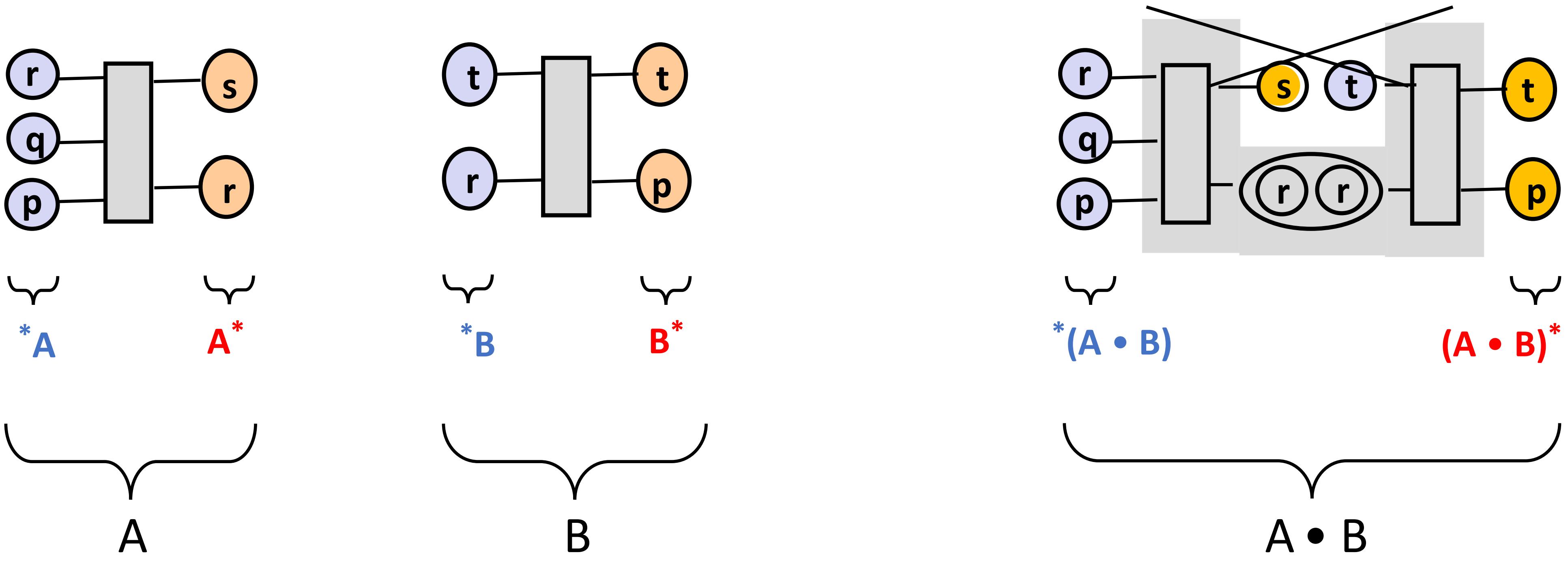
Fundamental idea



A component **C** frequently has

a *left* and a *right* interface, ***C** und **C***.

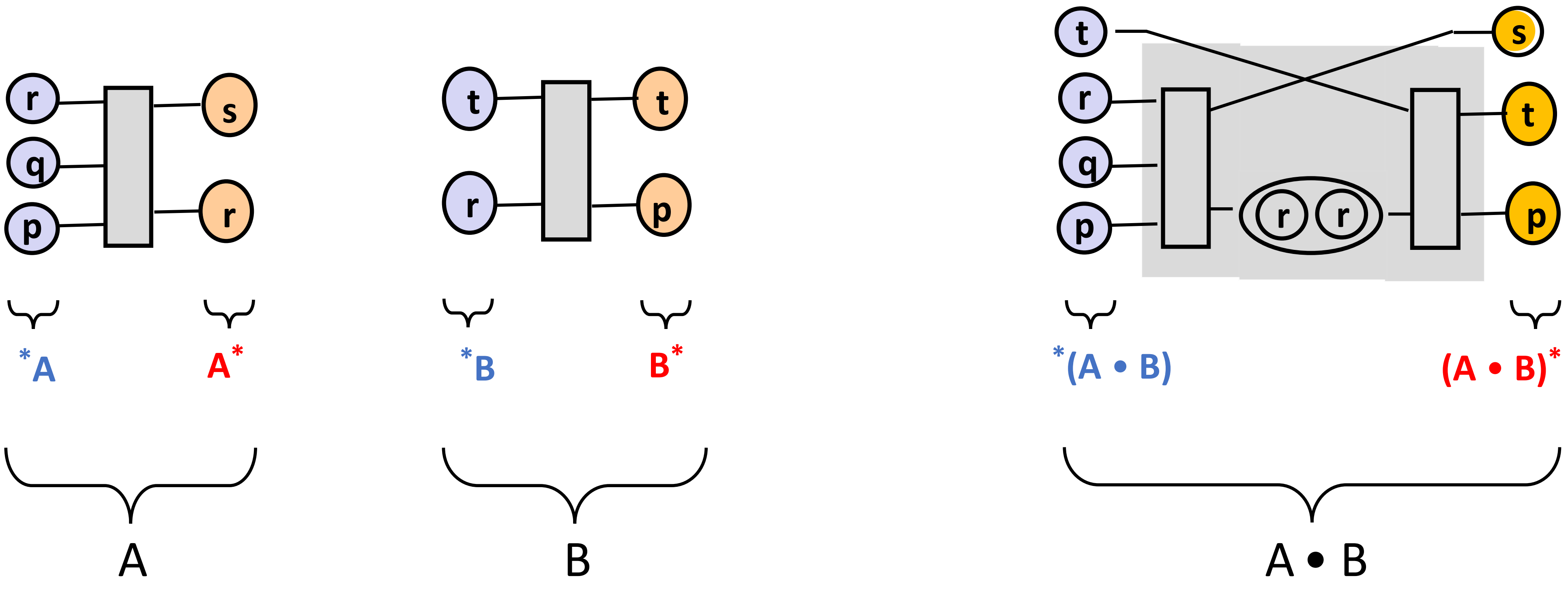
Fundamental idea



A component **C** frequently has
a *left* and a *right* interface, ***C** und **C***.

composition **A • B**:
Glue gates of **A*** und ***B** .

Fundamental idea

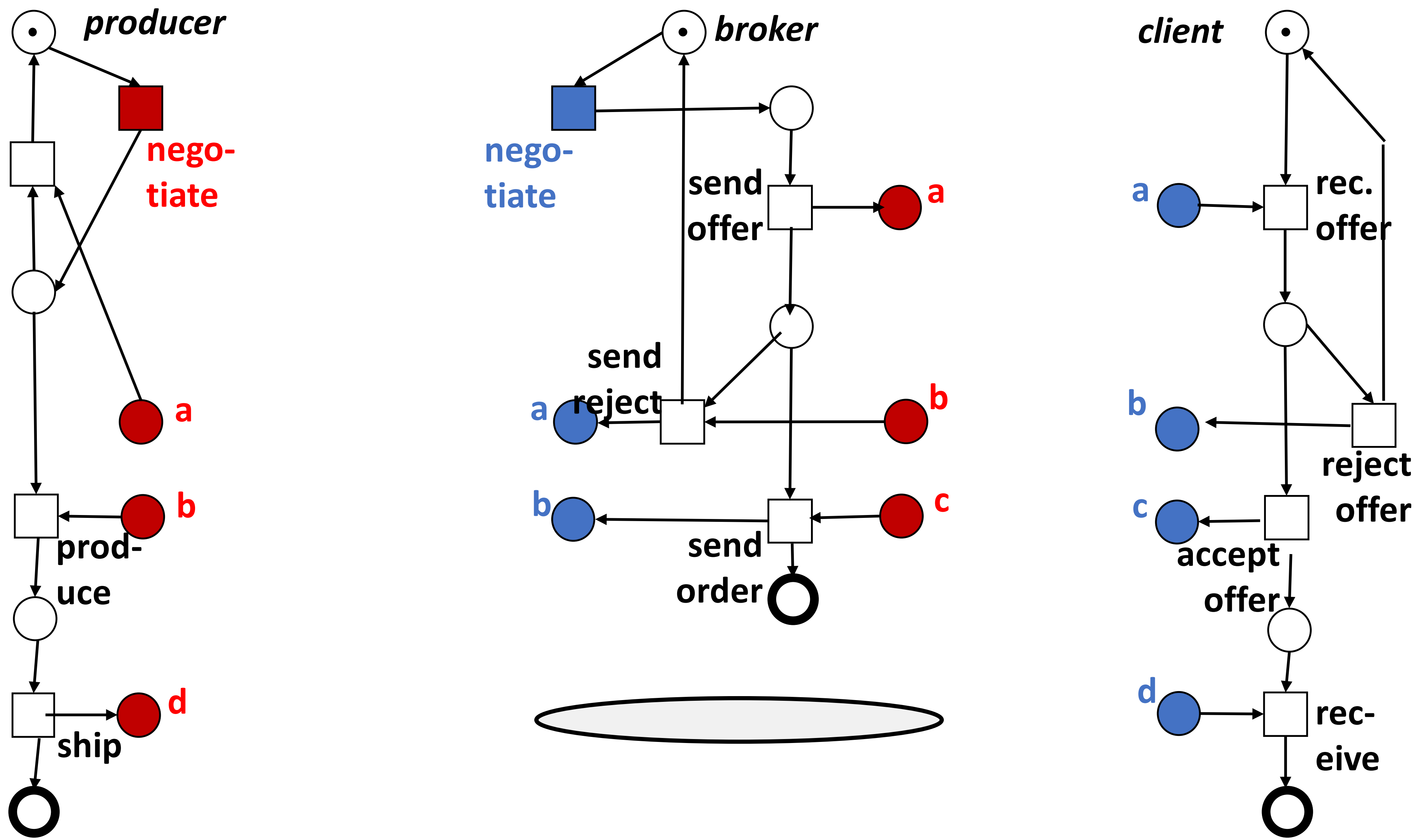


This composition is associative!

A component **C** frequently has
a *left* and a *right* interface, ***C** und **C***.

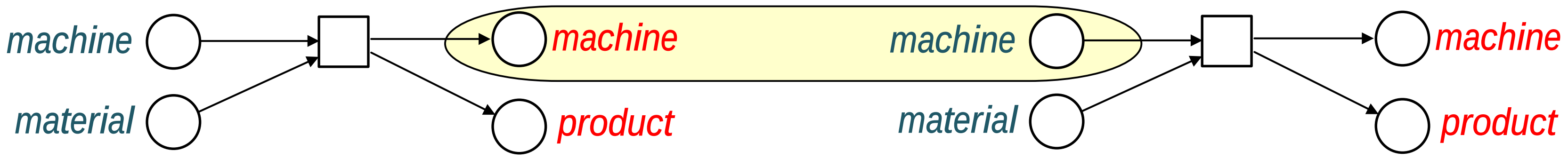
composition **A • B**:
Glue gates of **A*** und ***B** .

Example



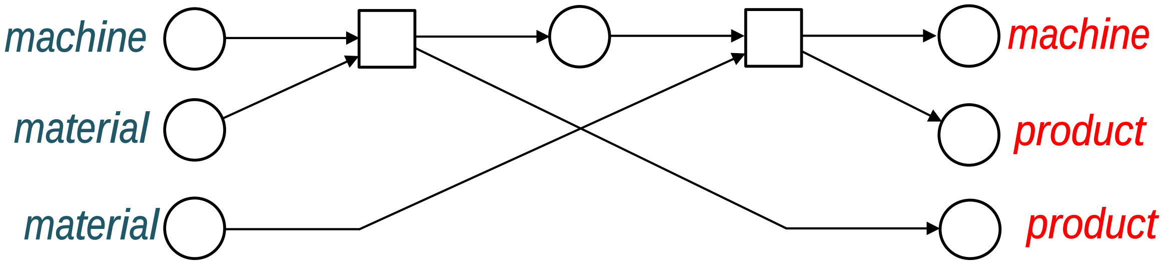
producer
nego-
tiate
a
b
prod-
uce
d
ship
broker
nego-
tiate
send
offer
a
send
reject
a
b
b
c
send
order
client
a
rec.
offer
b
reject
offer
c
accept
offer
d
rec-
eive

Composition of several instances

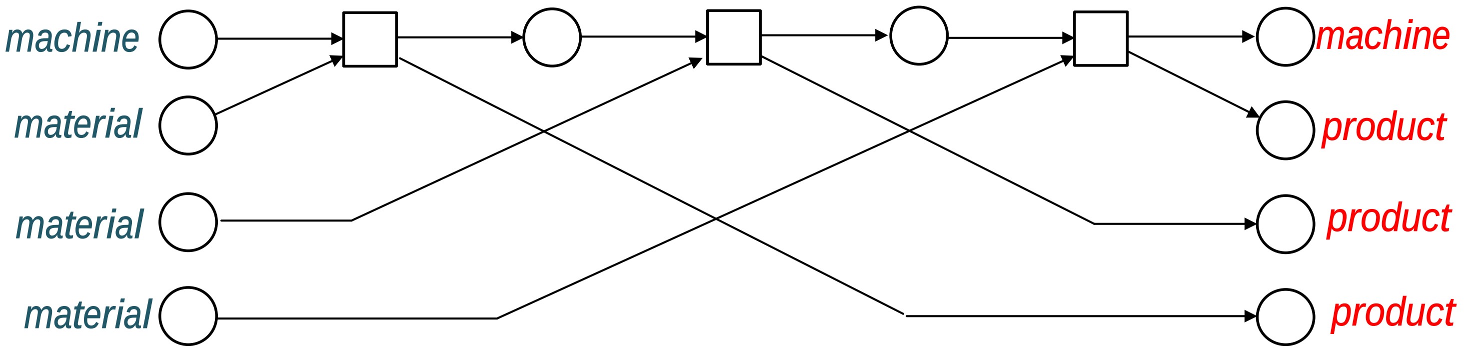


a. workflow model N

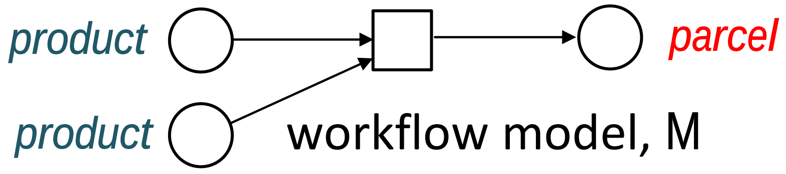
2nd instance of N,



b. composed workflow model, N • N



c. composed workflow model, N • N • N



workflow model, M

N • N • N • M

… so, in mathematical terms …

Let be given a finite set Σ of symbols "*gate labels*". Then

- an *interface over* Σ

  is a finite, ordered set, labeled over Σ.

- a *component* **C** *over* Σ

  is any structure with

  a *left* and a *right* interface,

  C** und **C.

Let $\mathbf{S}_\Sigma$ denote the class of all components over Σ.

Composition • then is defi

Exactly as in formal languages

Fundamental, not trivial:

Theorem. Composition • on $\mathbf{S}_\Sigma$ is

total and

associative.

Observation. With ***E = E* = ∅**,

A•E = E•A = A.

Corrollary.

$\mathbf{M} =_{def} (\mathbf{S}_\Sigma, \bullet, \mathbf{E})$ is a monoid

Just replace „word over Σ" by „component over Σ"

How can you do „everything“ with this?

Compose **A** and **B**

under side conditions

Conformance,

Interoperability,

Composability,

Semantic restrictions,

Data types,

Alternative gate matches,

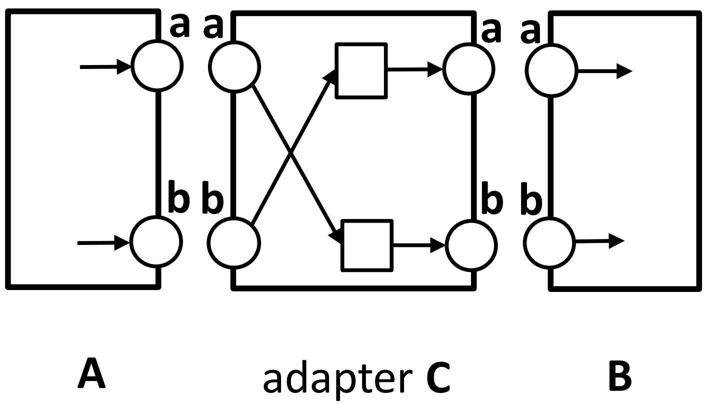
Nondeterminism, …

formulate as an *adapter* **C**,

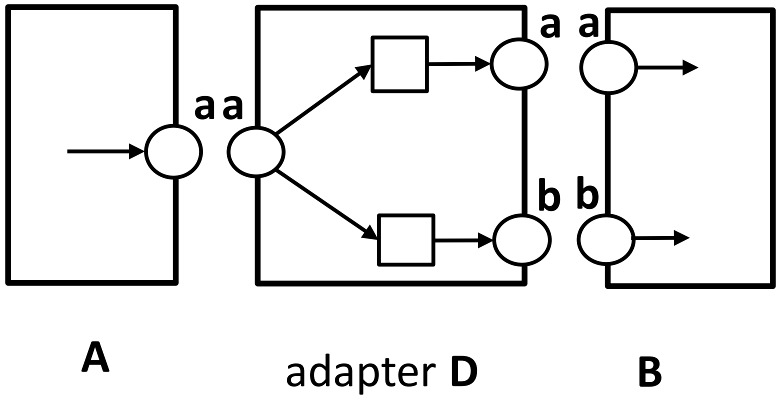
in **A • C • B**.

examples:

requirement:
want to glue non-matching gates



reqirement:
nondeterministic choice



Complex composition as *Adapter*

To express a property π

of a composition operator $\bullet_\pi$:

Construct an a*dapter* $[\pi]$;

replace $A \bullet_\pi B$ by $A \bullet [\pi] \bullet B$.

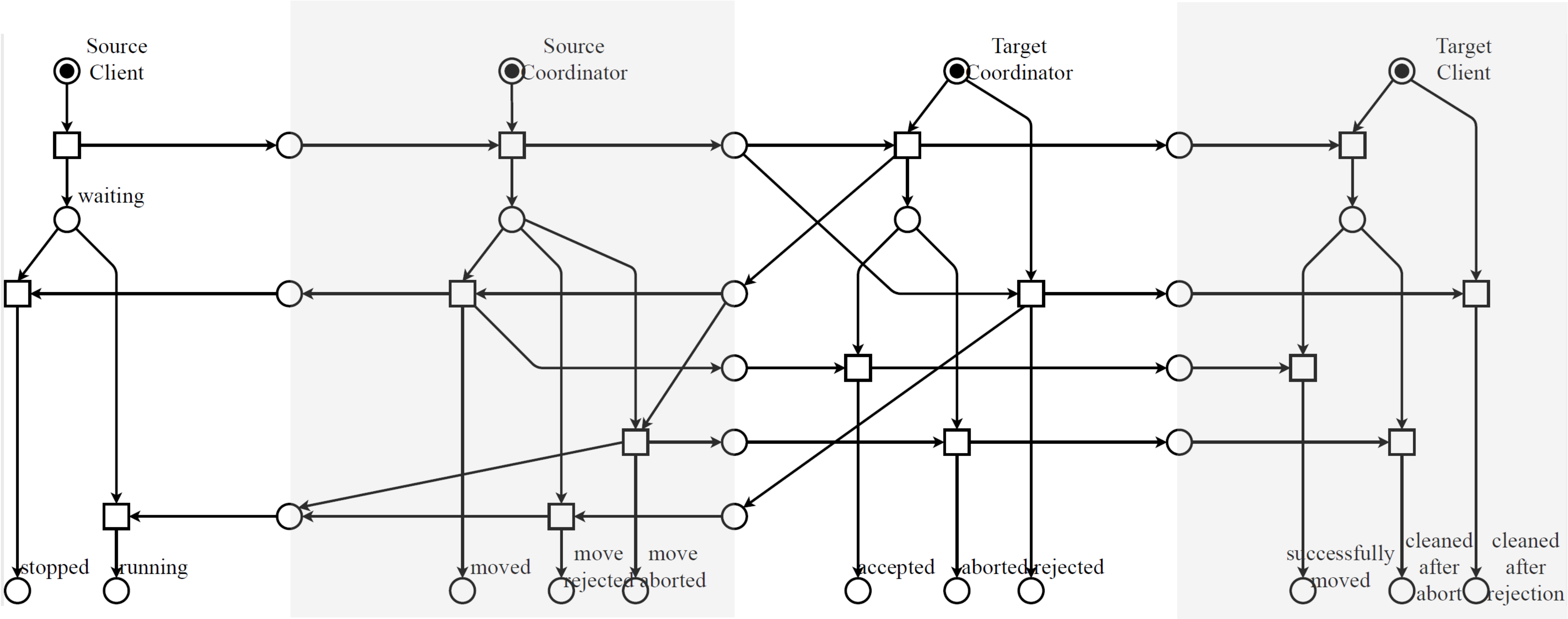
Advantage (among others):

Composition of properties:

$A \bullet [\pi] \bullet [\pi'] \bullet B$.

Example: PubSub migration

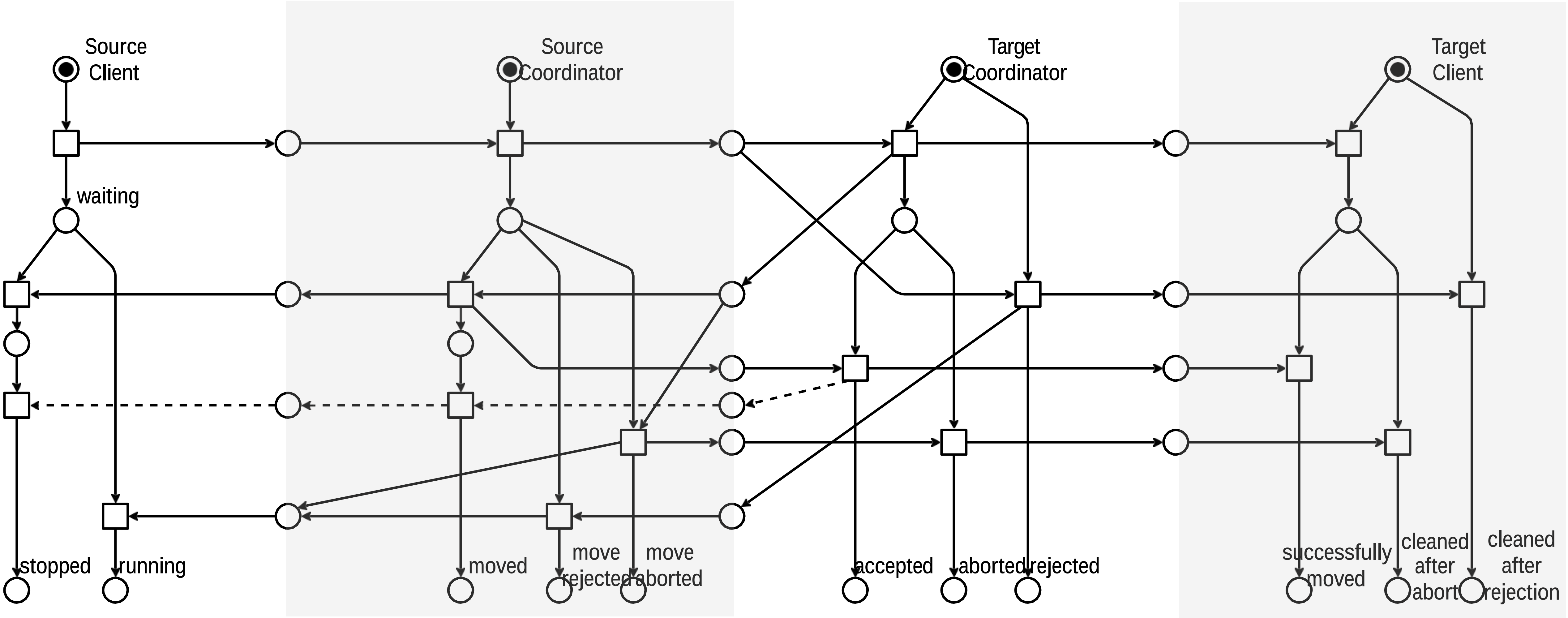
SCl • SCo • TCl • TCo



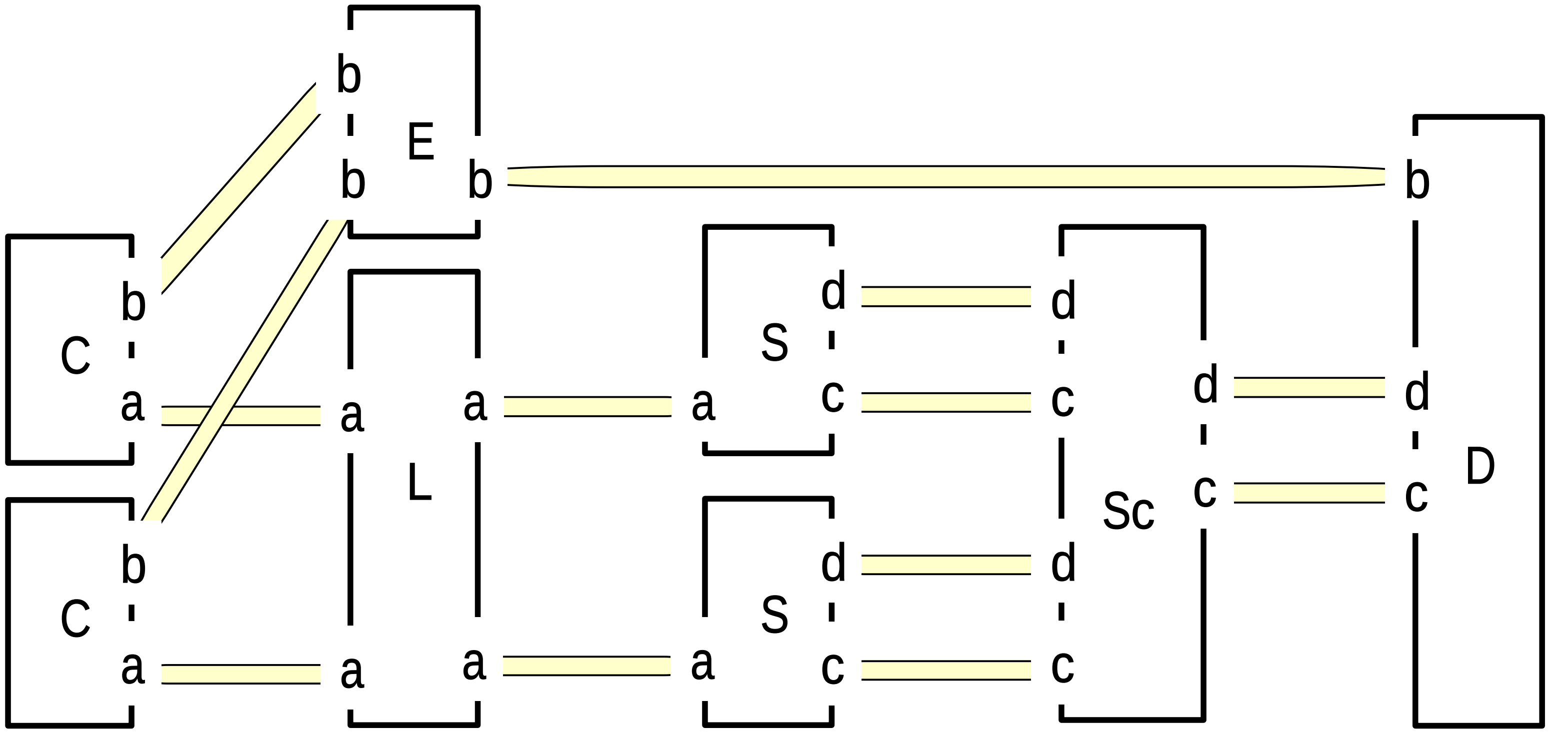
Example: PubSub migration

SCl • SCo • TCl • TCo

… redundant messages:



Example: a small computer architecture



C client
L load balancer
S service
Sc scheduler
D data base
E editor

a messages from client or load bal.
b messages to client or editor

c message from service or schedul.
d message to service or schleduler

C • C • L • S • S • E • Sc • D

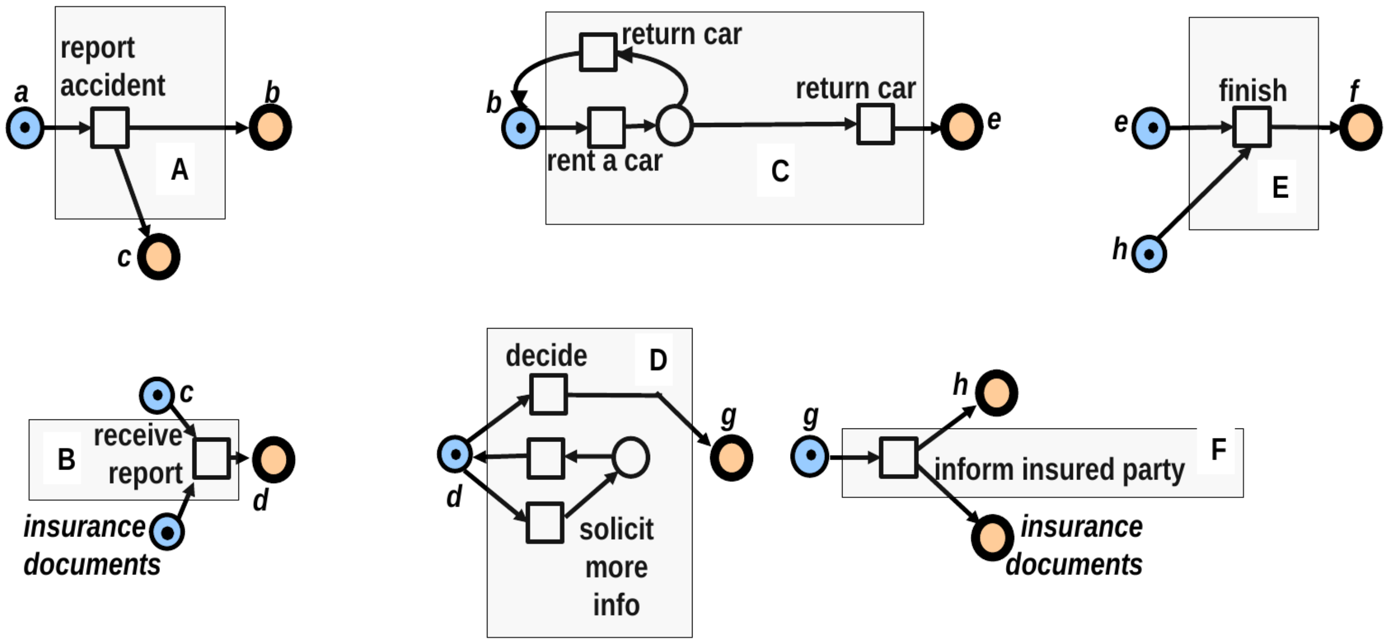
Preservation of properties

Example:

car rental

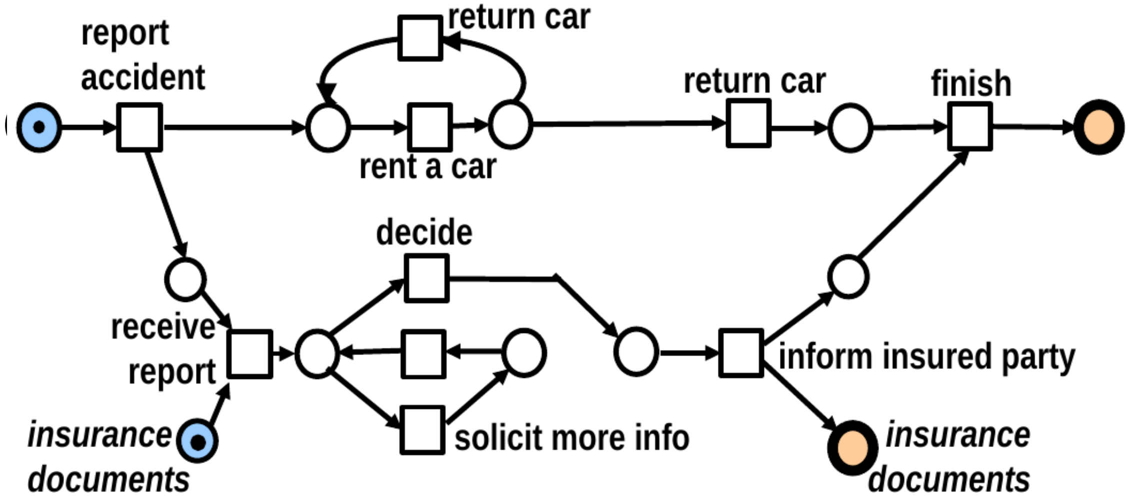
business process

at trafic accident



A, … F

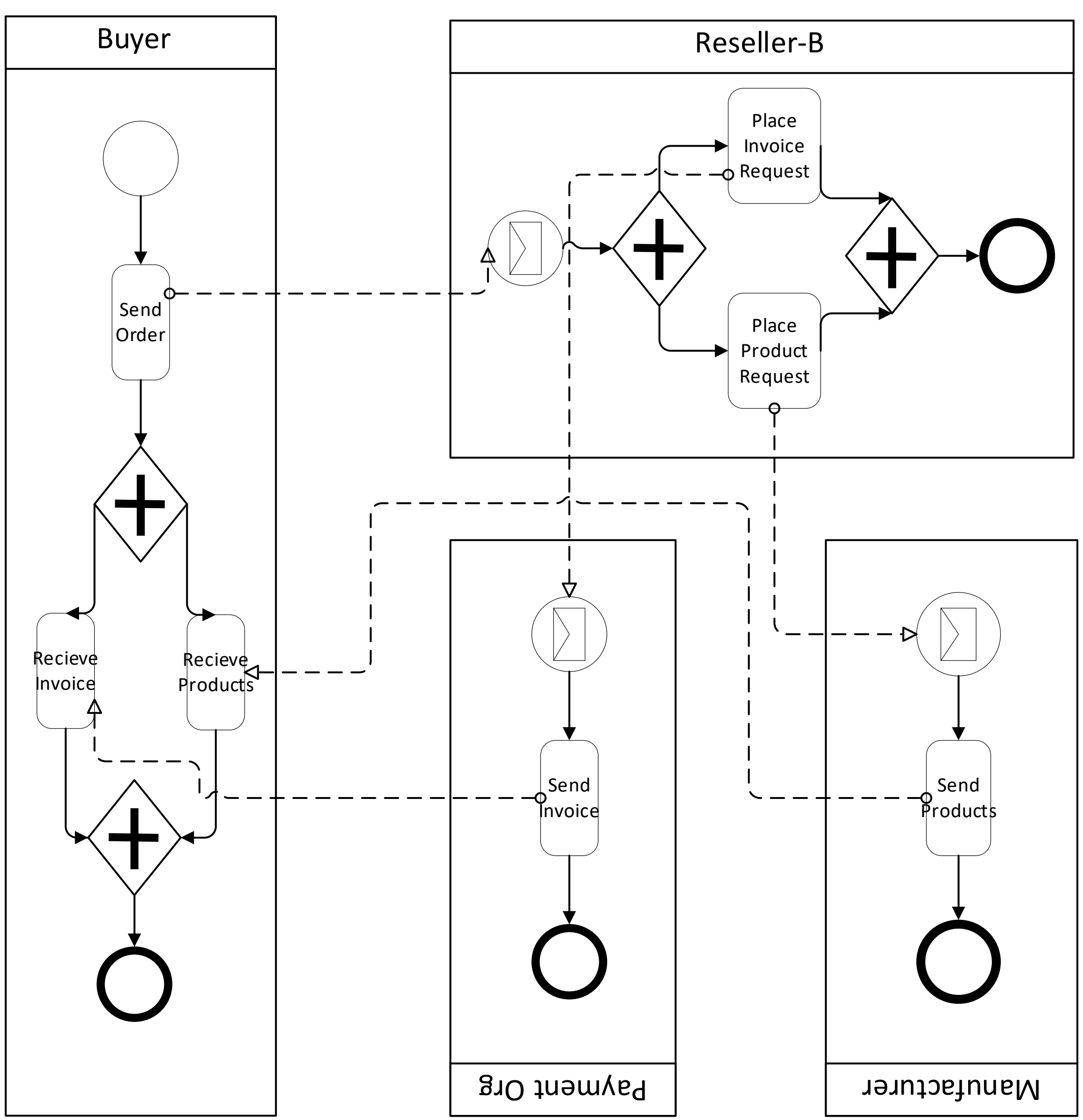
are aparently sound

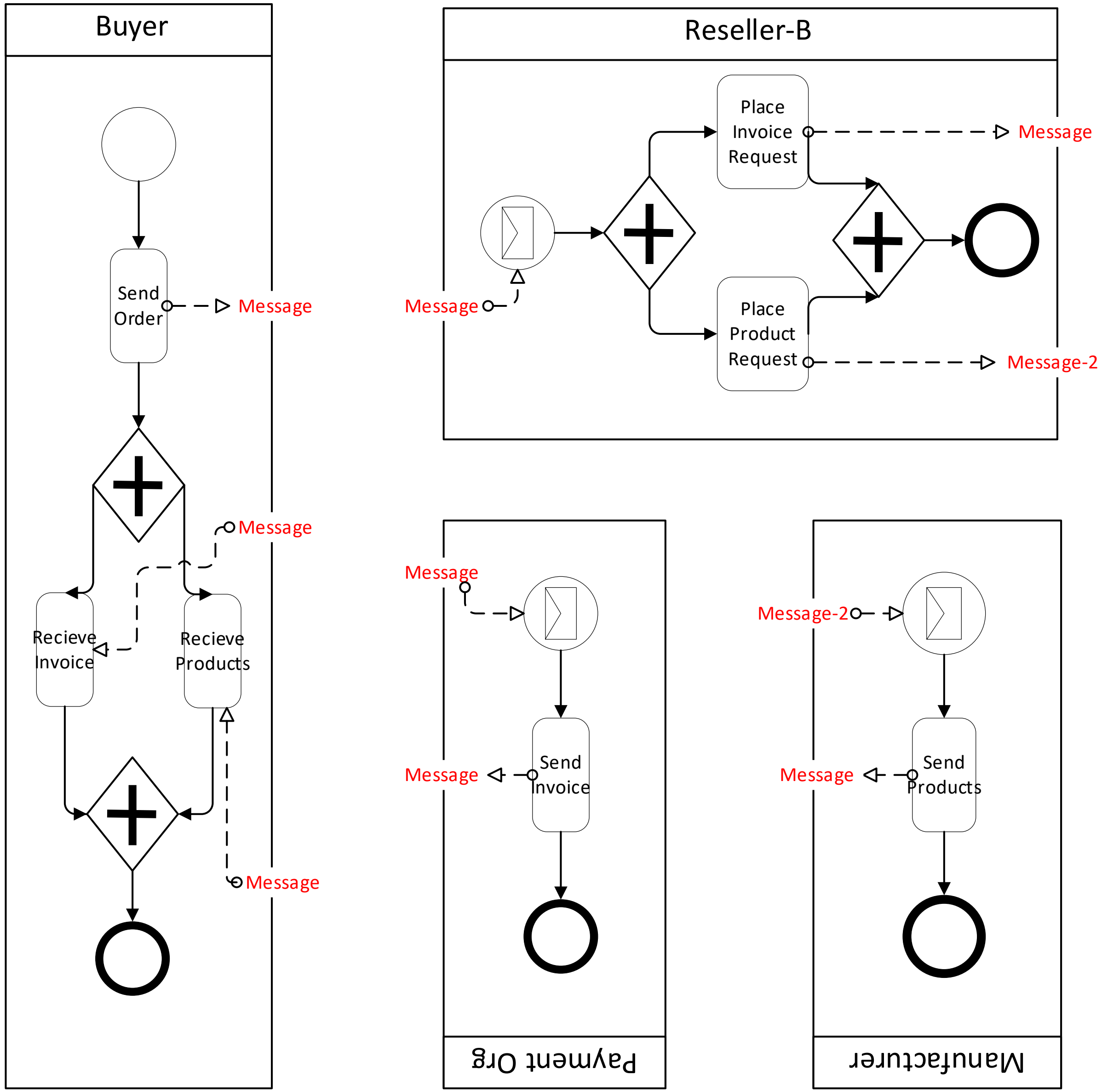


A•B•C•D•E•F

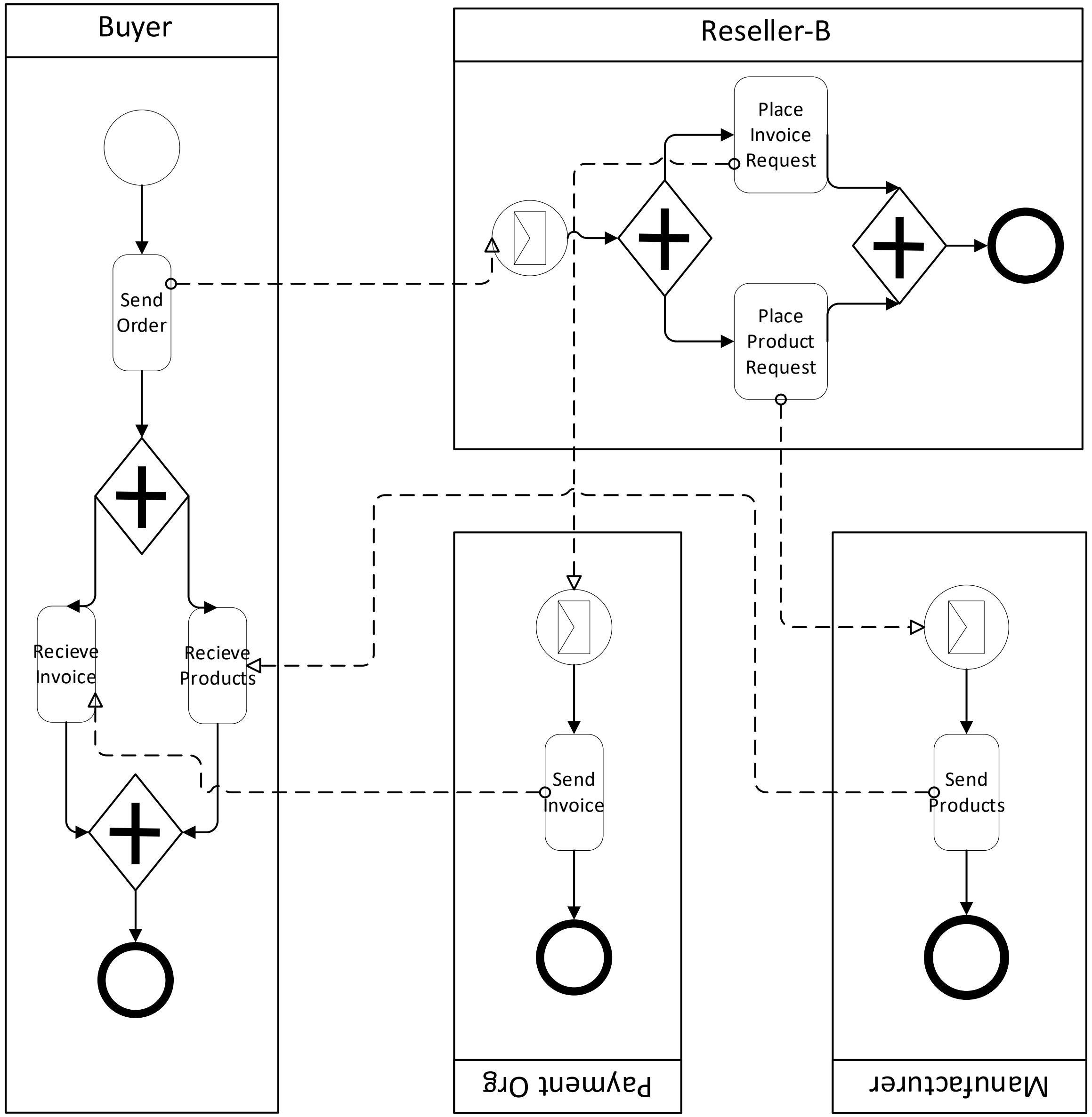
Is sound by construktion

BPMN components

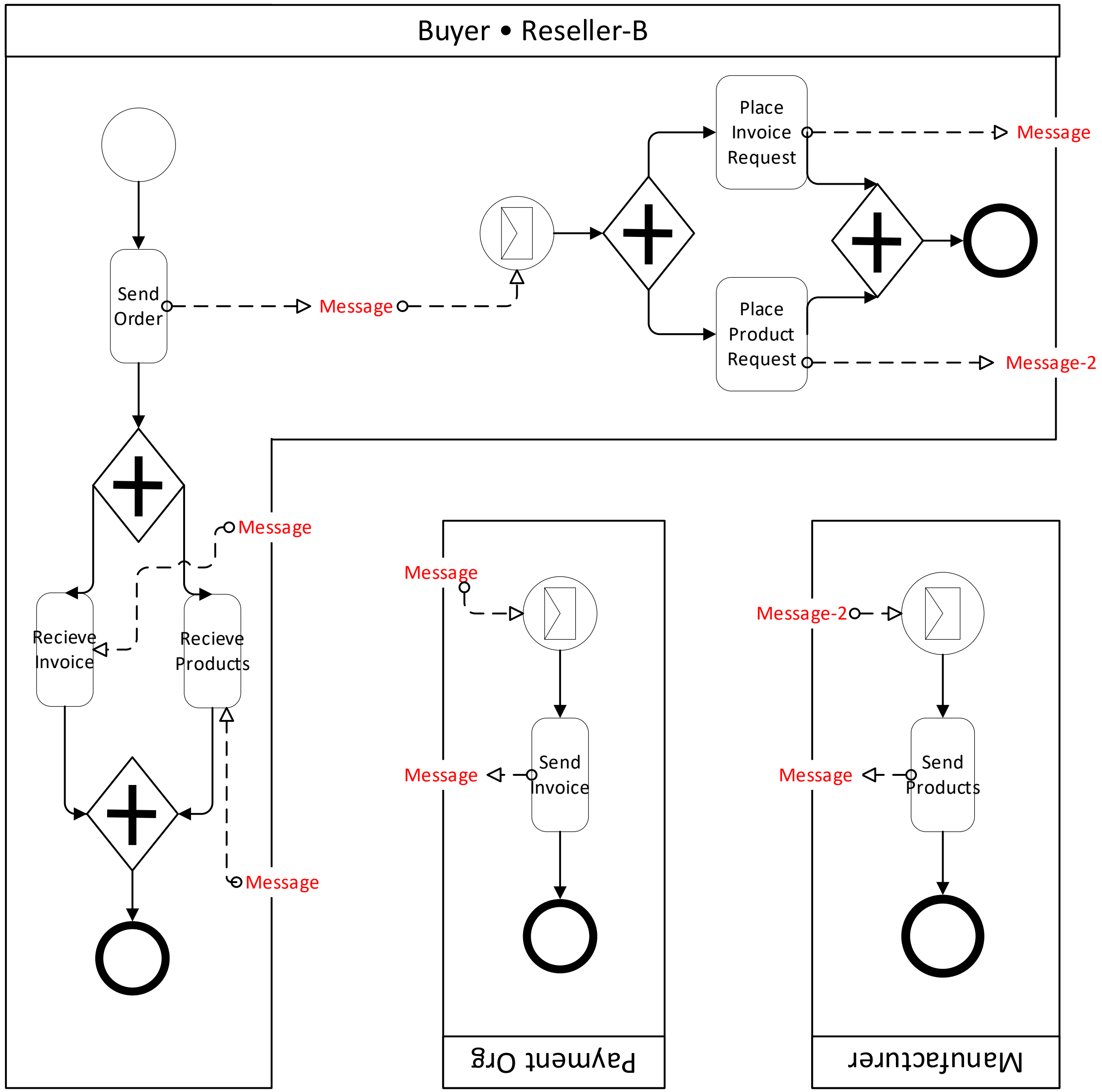




BPMN

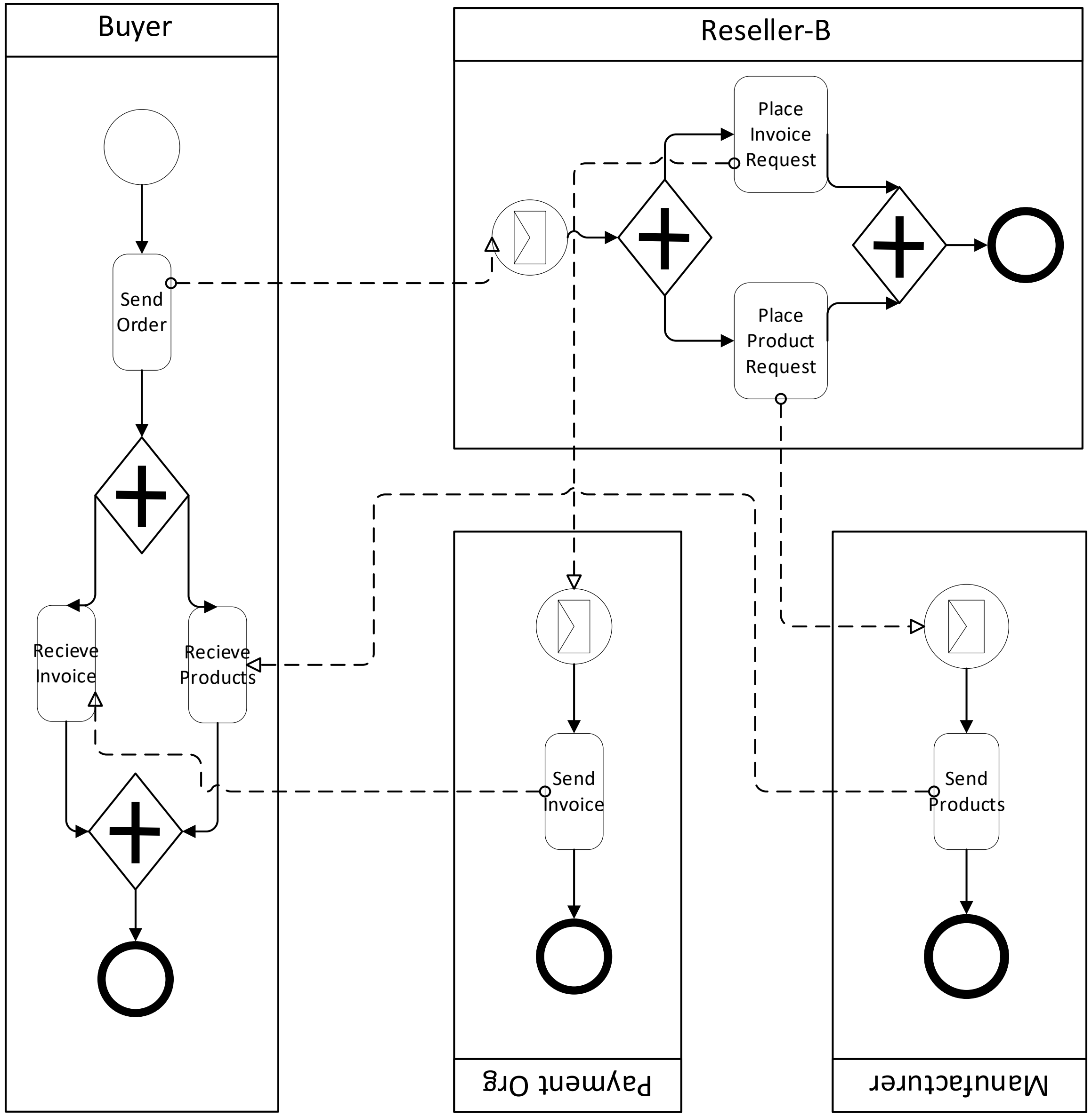


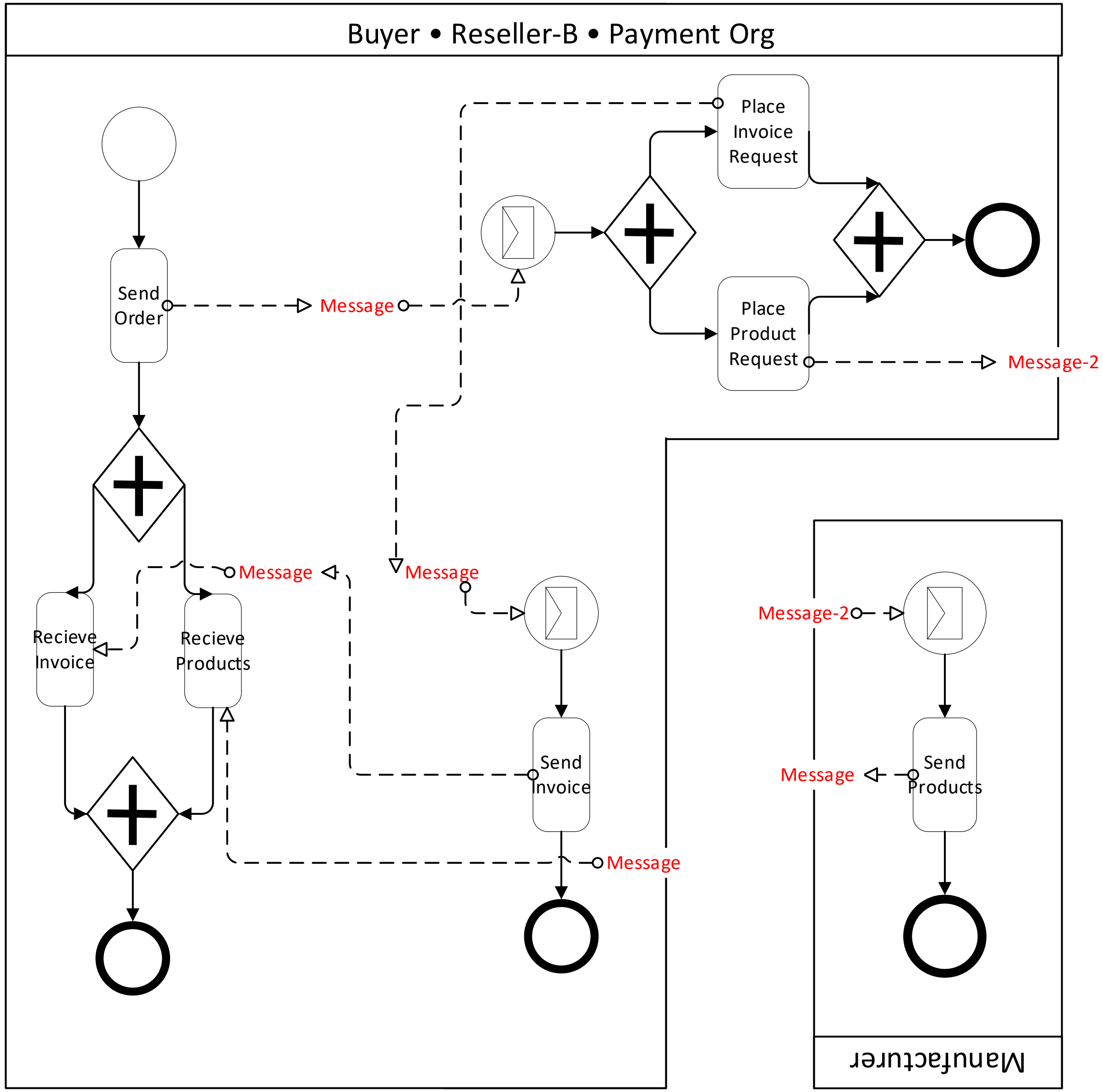
Buyer • Reseller



BPMN

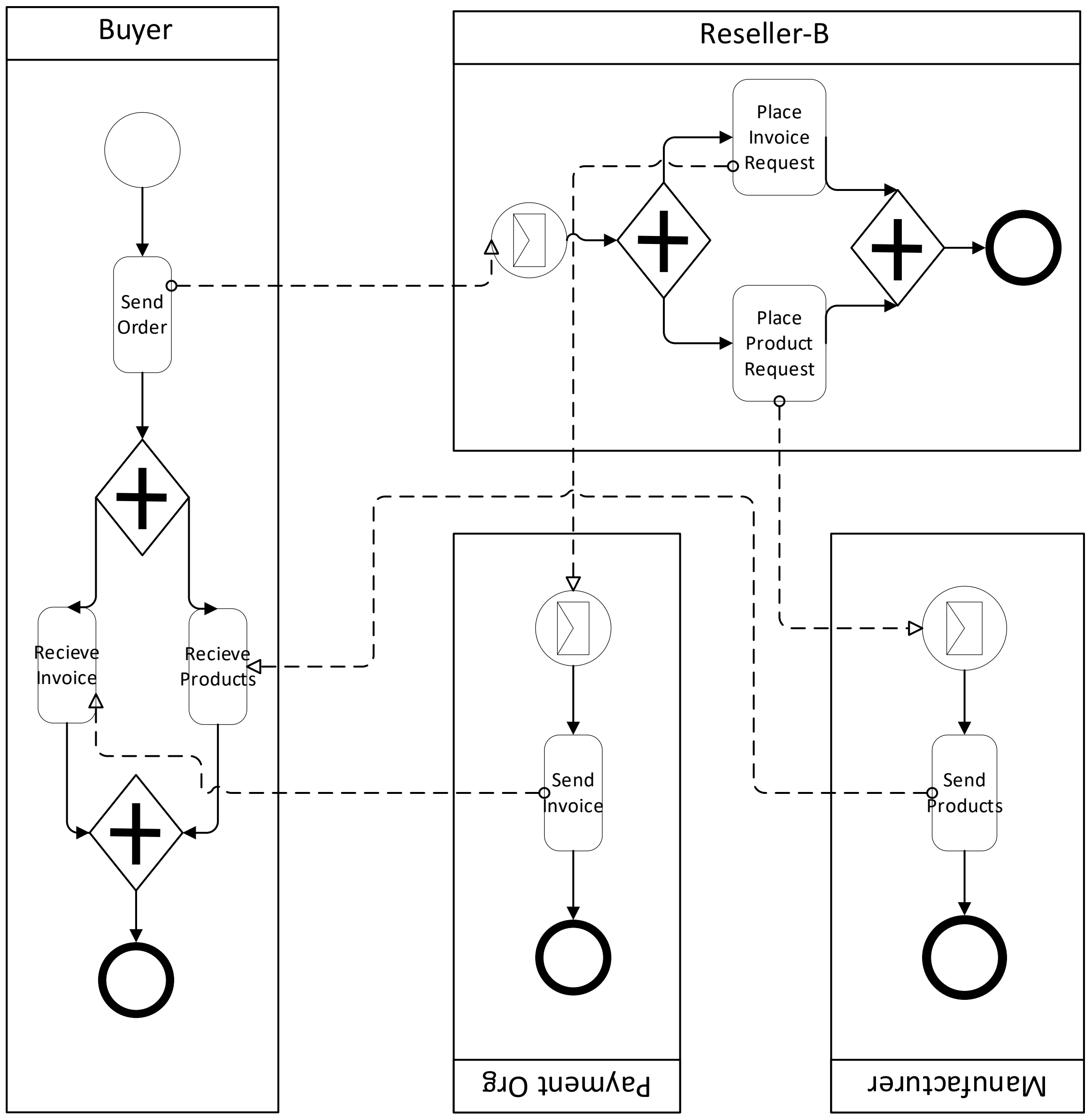
Buyer • Reseller • Payment Org

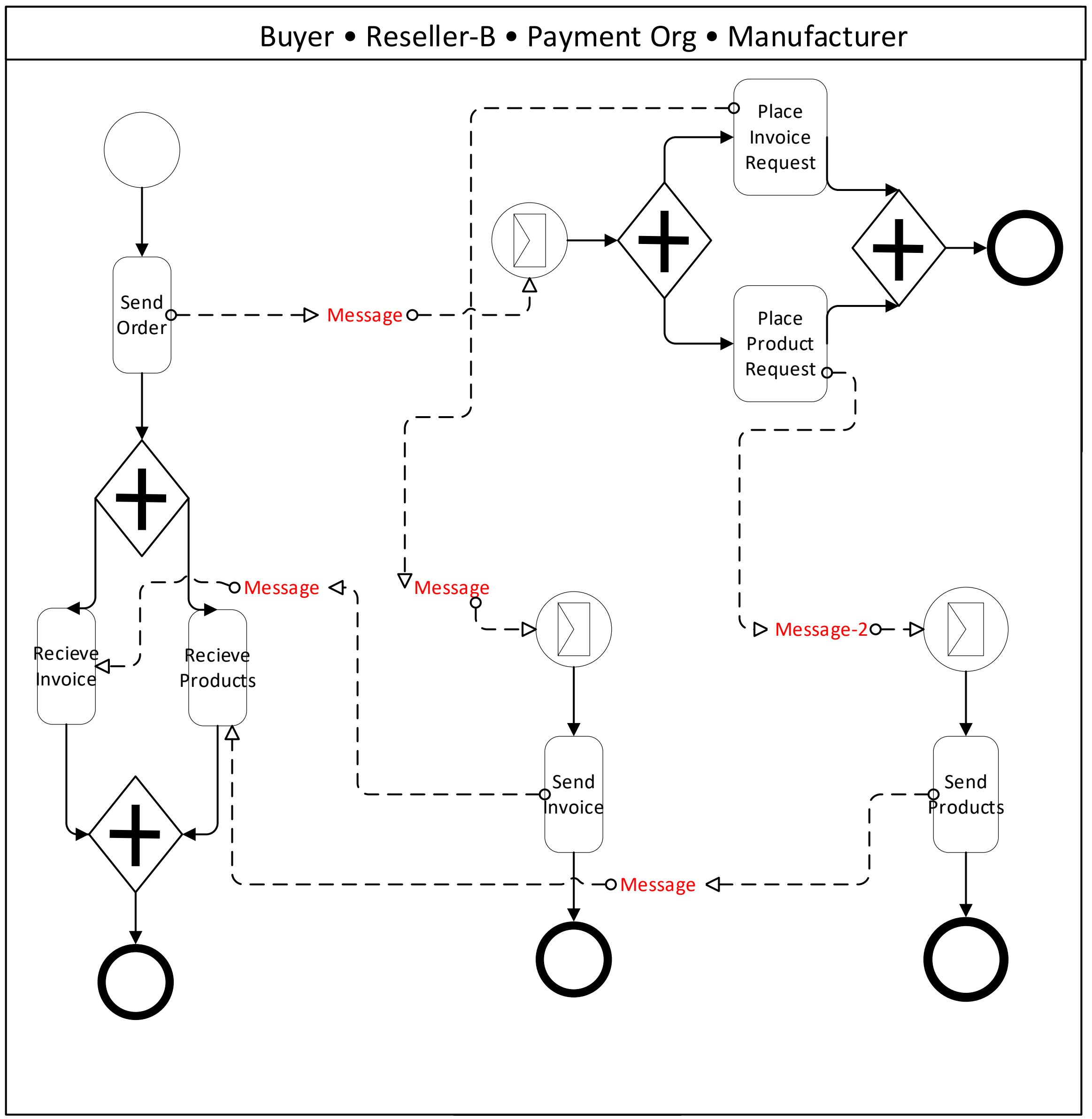




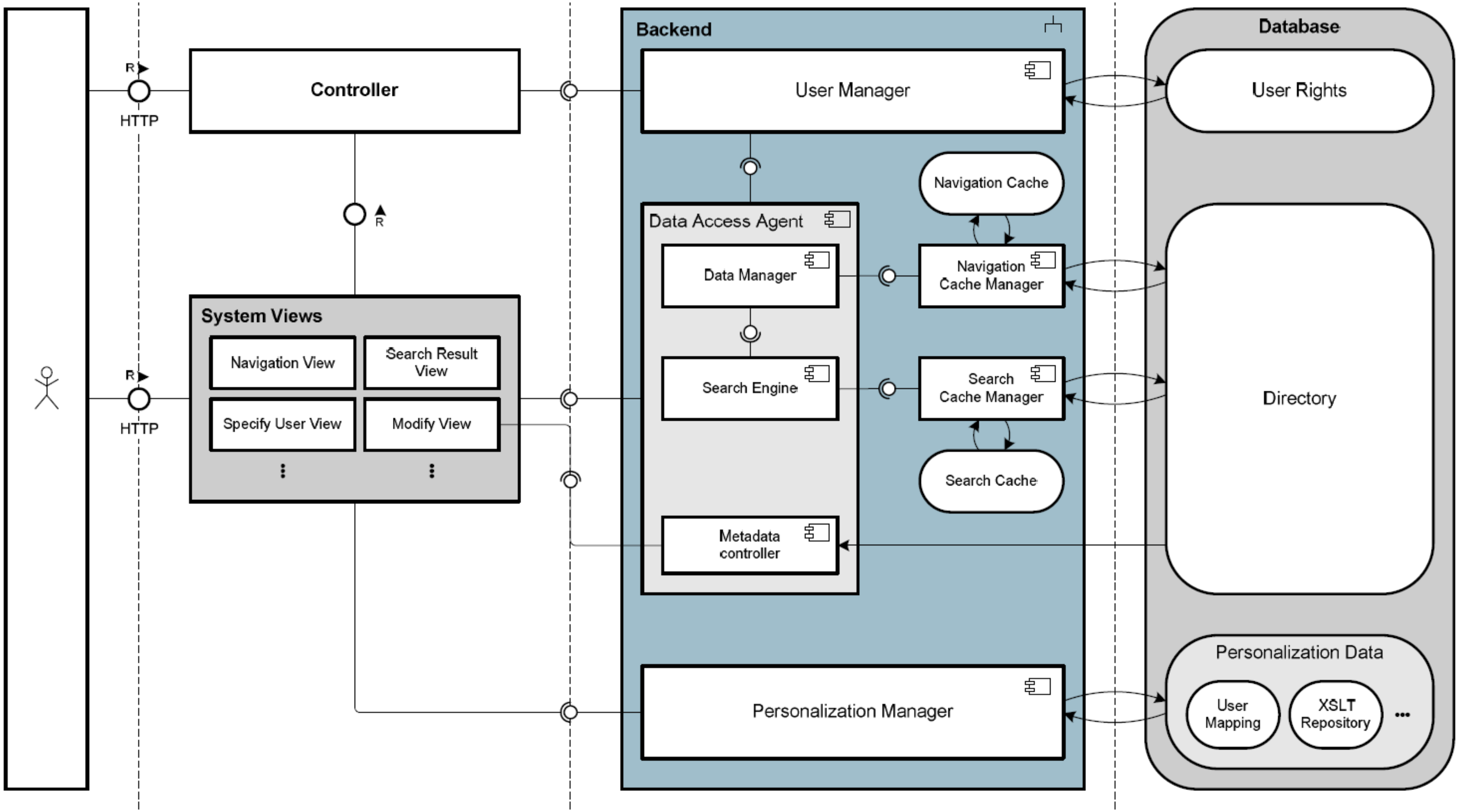
BPMN

Buyer • Reseller • Payment Org • Manufacturer

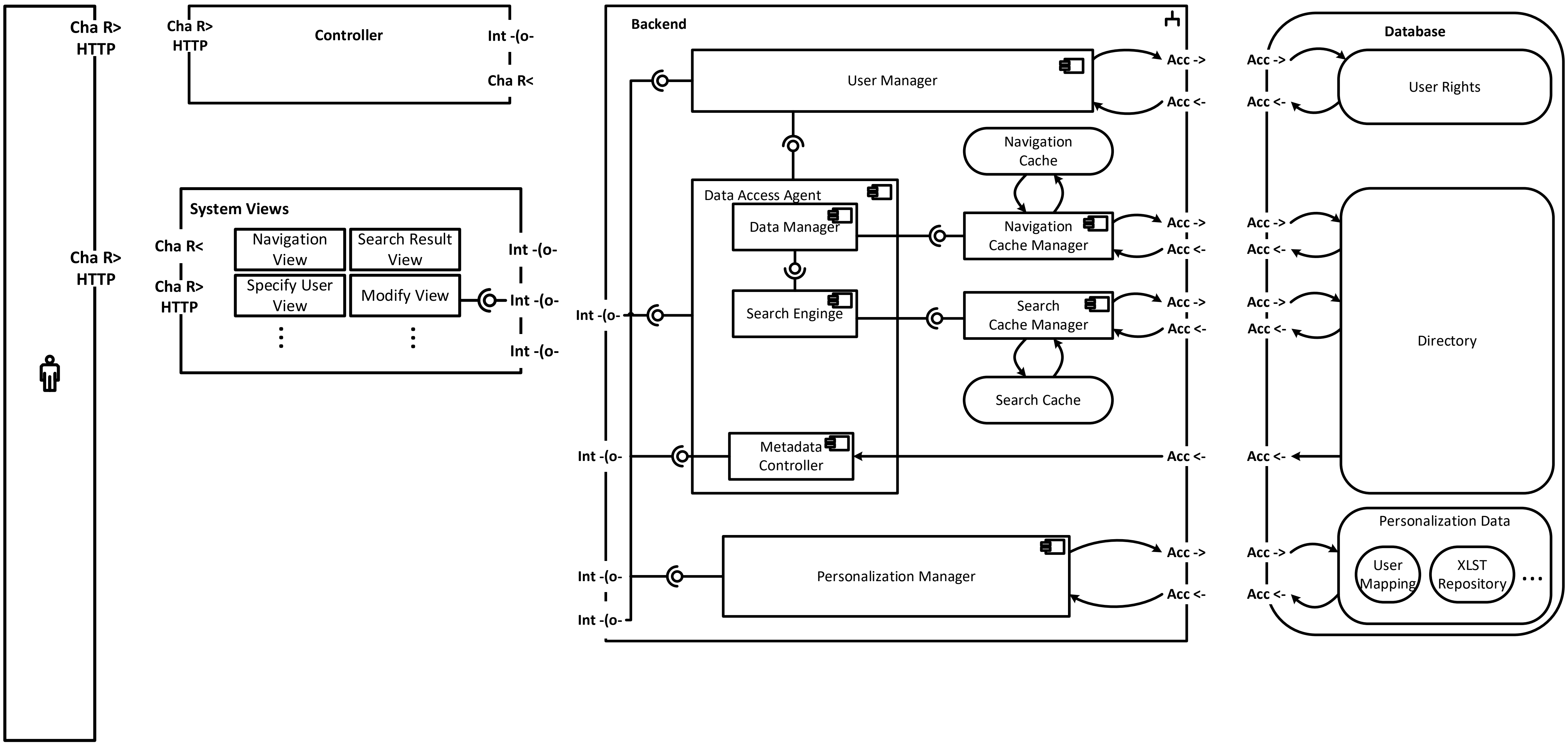




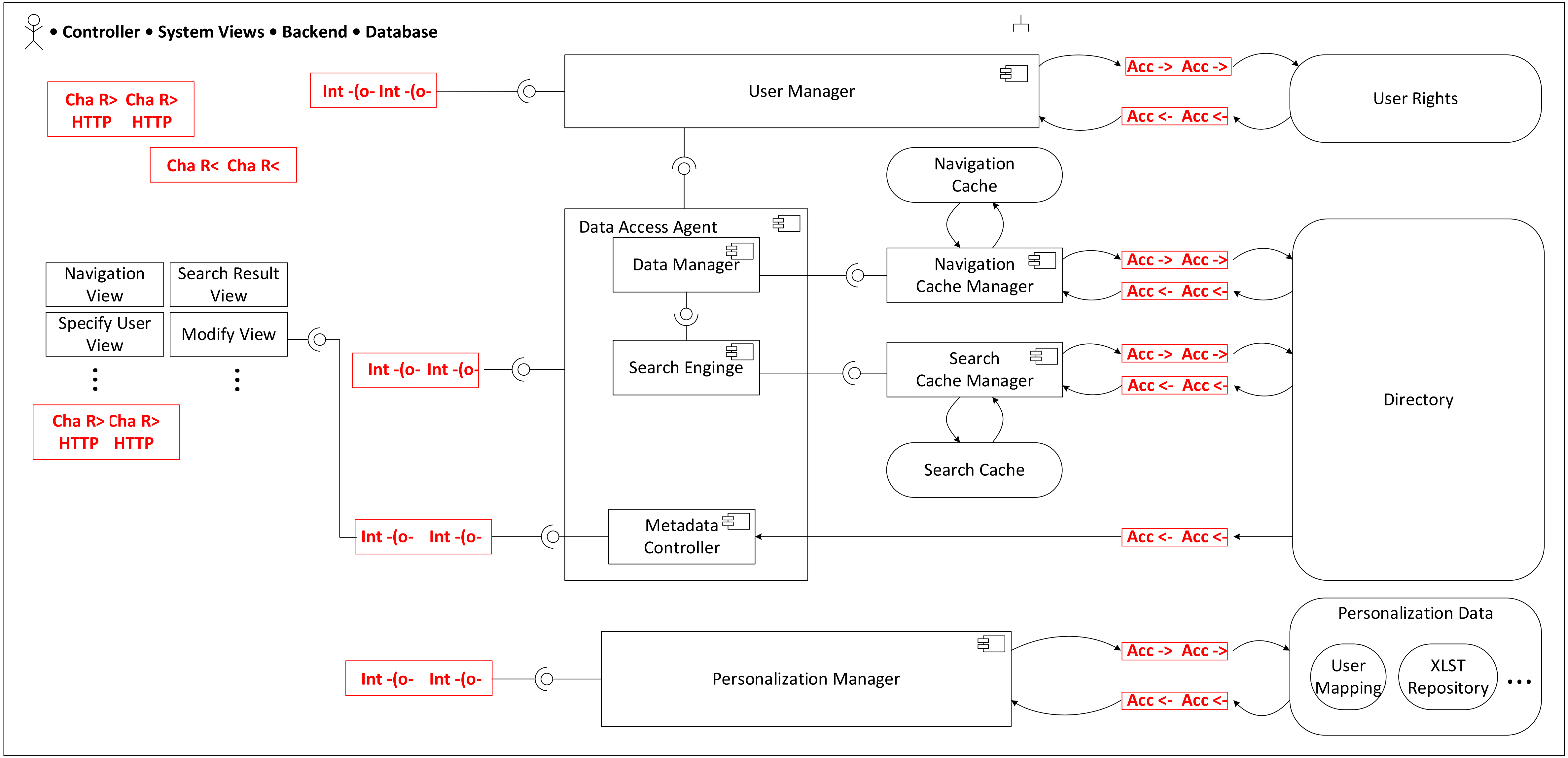
SAP:TAM Standardized Technical Architecture Modeling



Five components



Five components composed



1.2 The *seams* of a composed system *

A canonical, universal principle
for composed systems
independent of
their semantical contents.

Motivted by
Interface description languages
DARWIN
RADL
AADL

as well as
Early architecture description languages
Rapide
Wright
Weaves

Idea: specify the interface
before you program
the operations.

* Work together with Heinz Schmidt, Melbourne

The Problem

Let $A =_{def} A_1 \bullet A_2 \bullet \ldots \bullet A_n$.

How recompute from A all the A_i ?

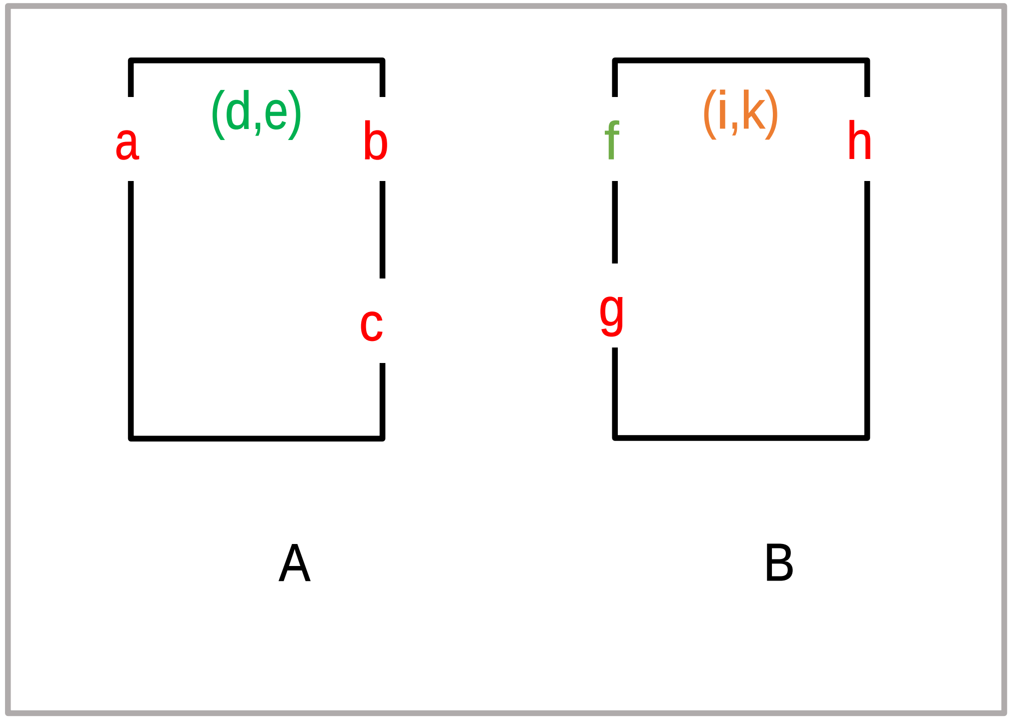
In general: not possible.

Idea: store enough information

when composing the A_i !

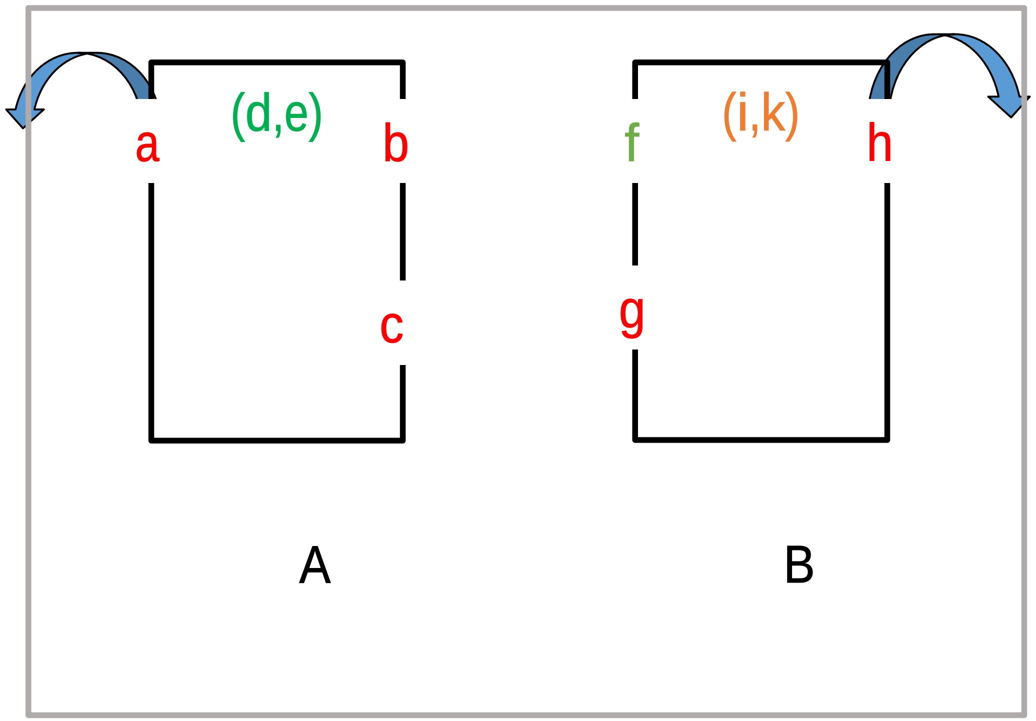
.. the *seam* !.

… what to remember upon composition?

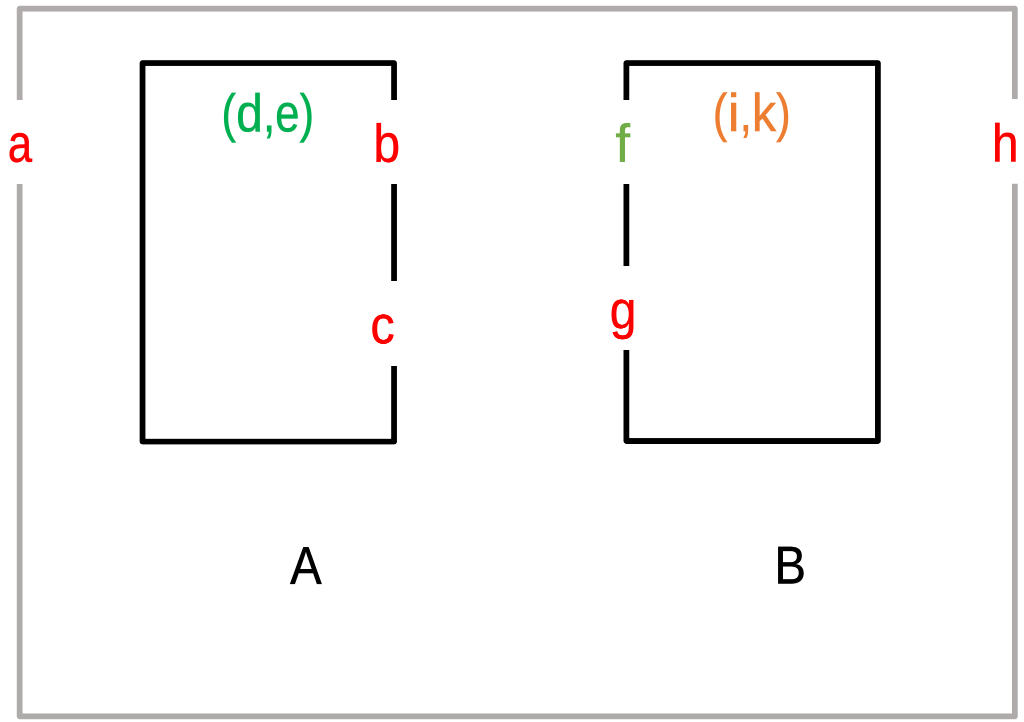


A•B

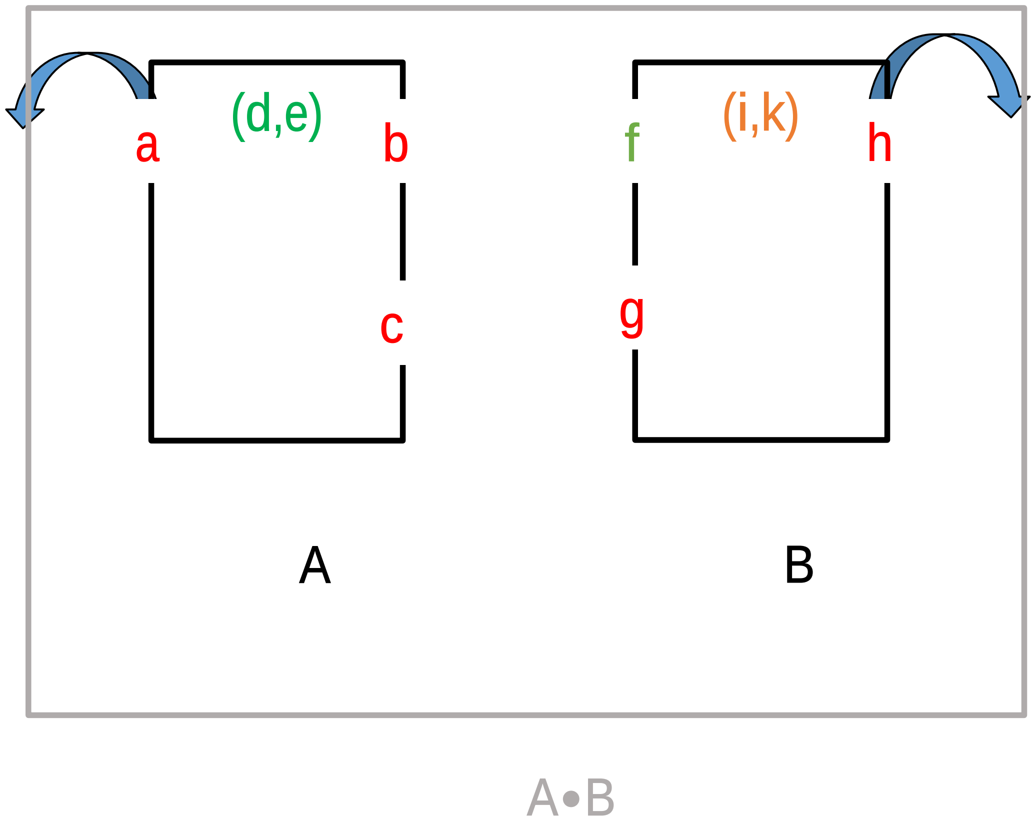
label: red, green, yellow

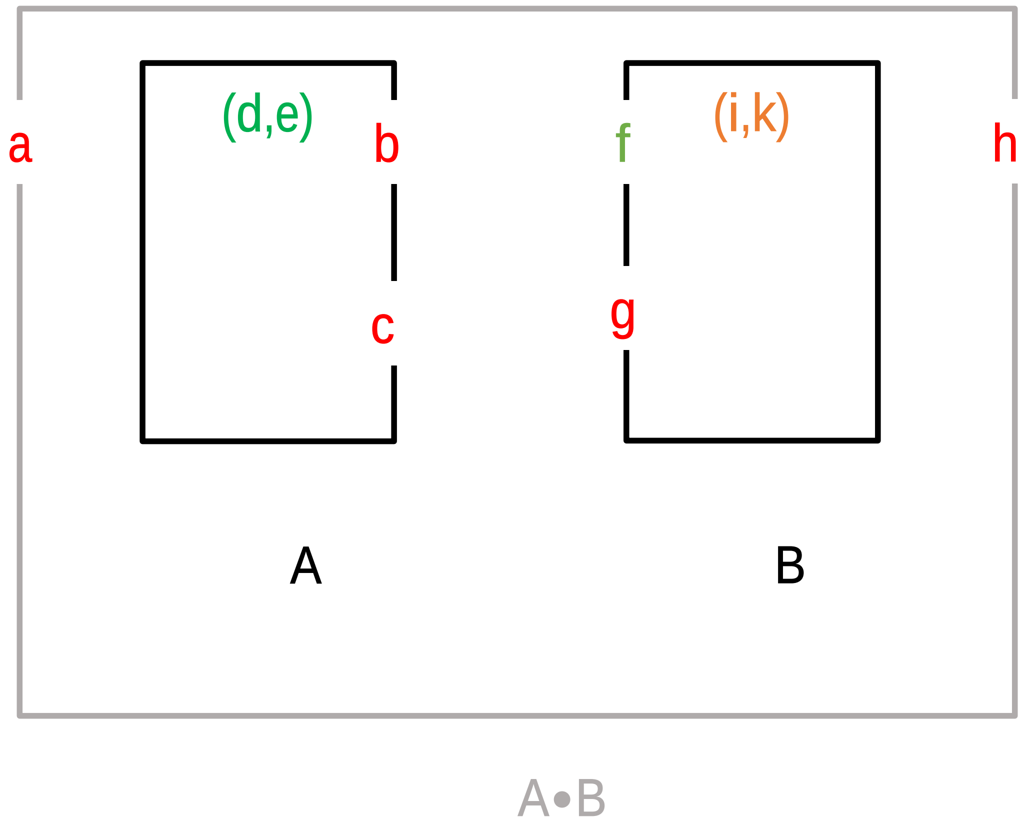


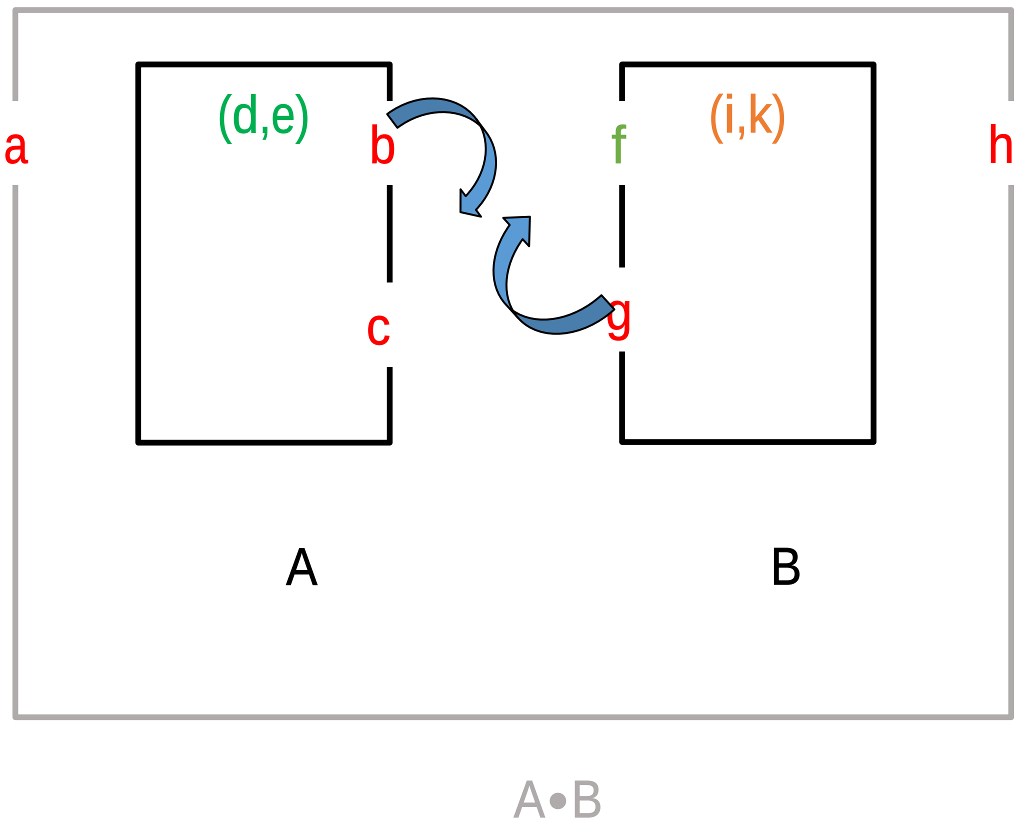
A•B

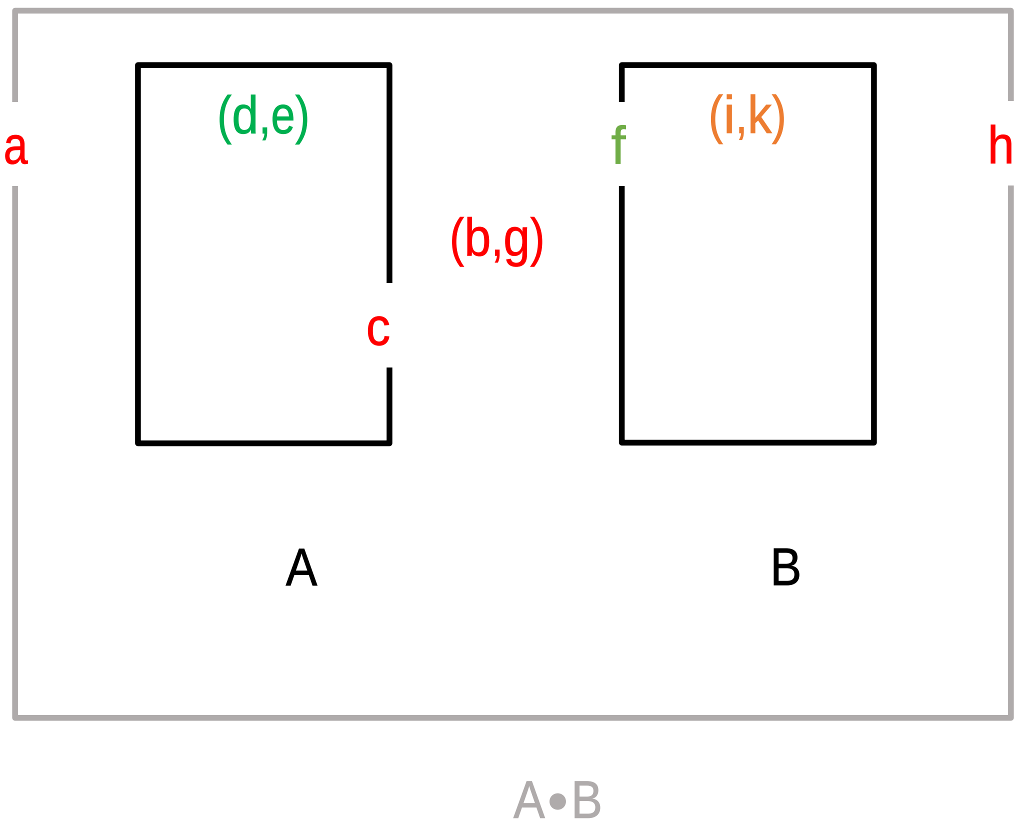


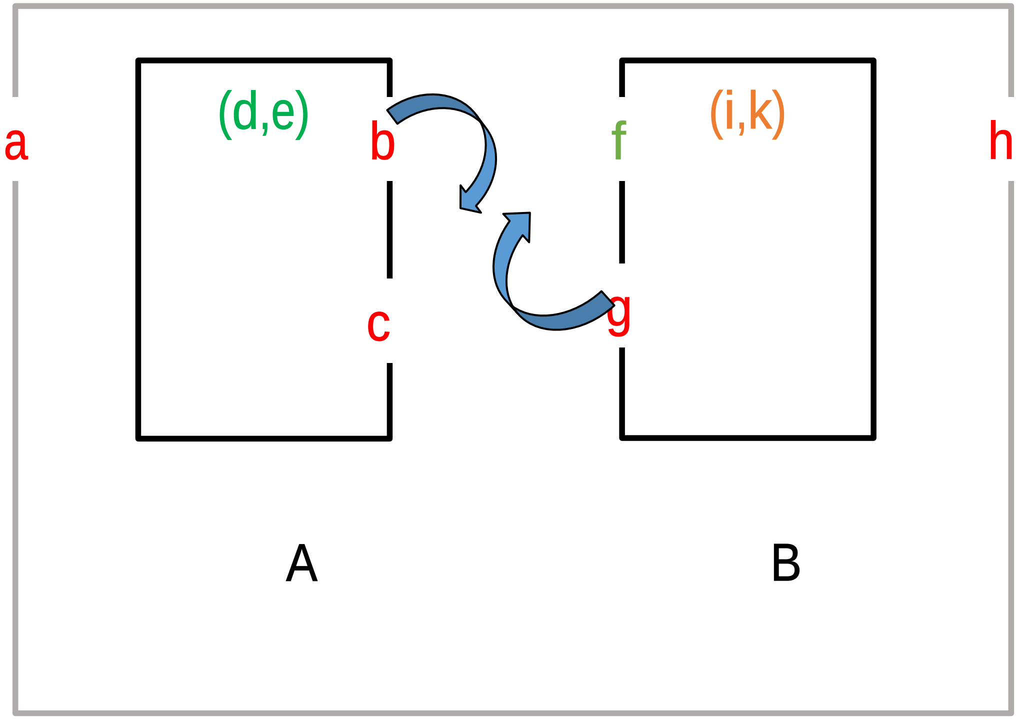
A•B


(d,e)
a
b
c
(i,k)
f
h
g
A
B
A•B

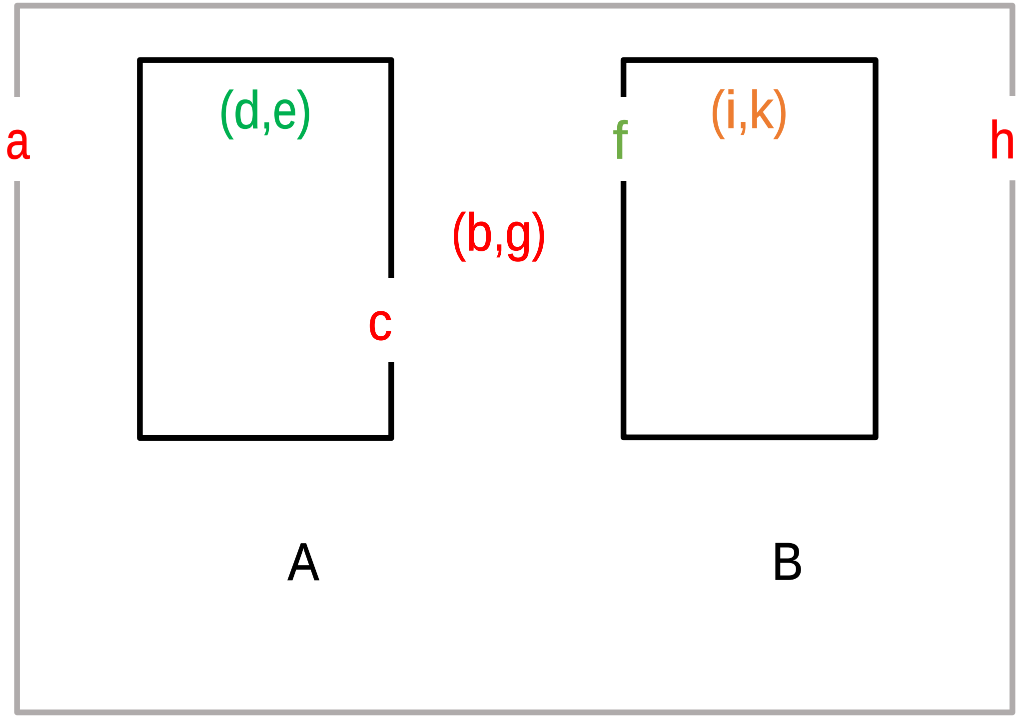

(d,e)
a
b
c
(i,k)
f
h
g
A
B
A•B


(d,e)
a
b
c
f
(i,k)
g
h
A
B
A•B

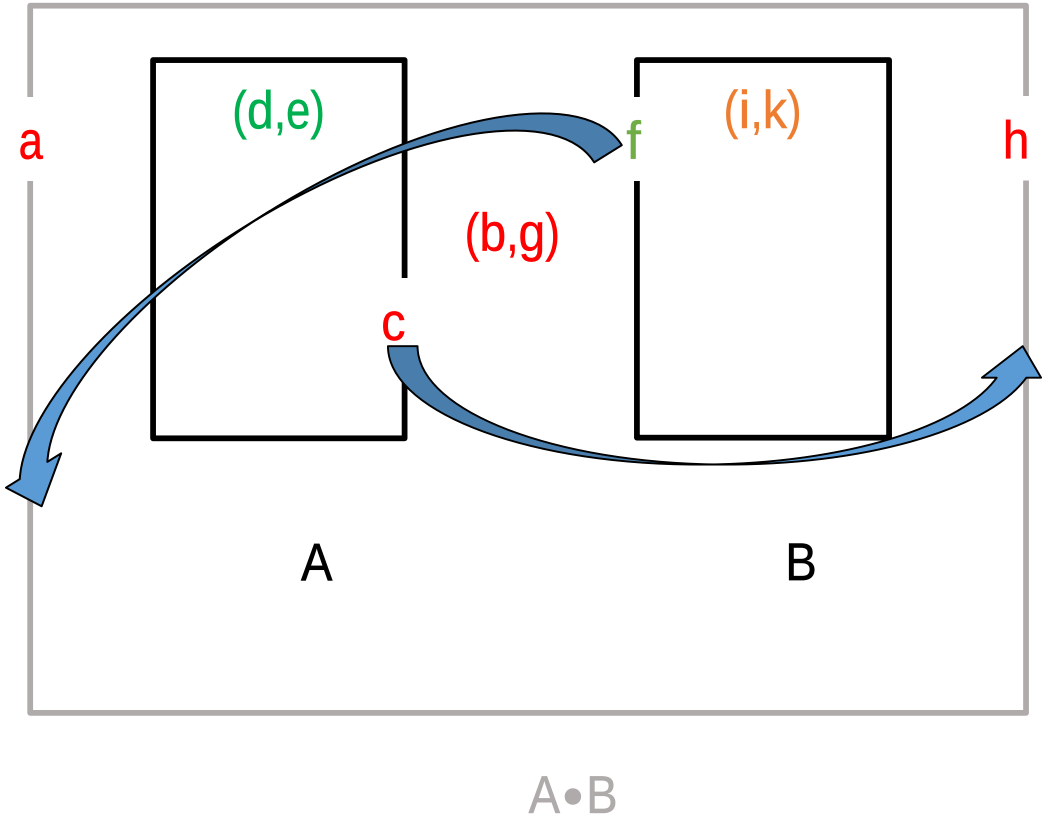

(d,e)
a
c
(b,g)
f
(i,k)
h
A
B
A•B

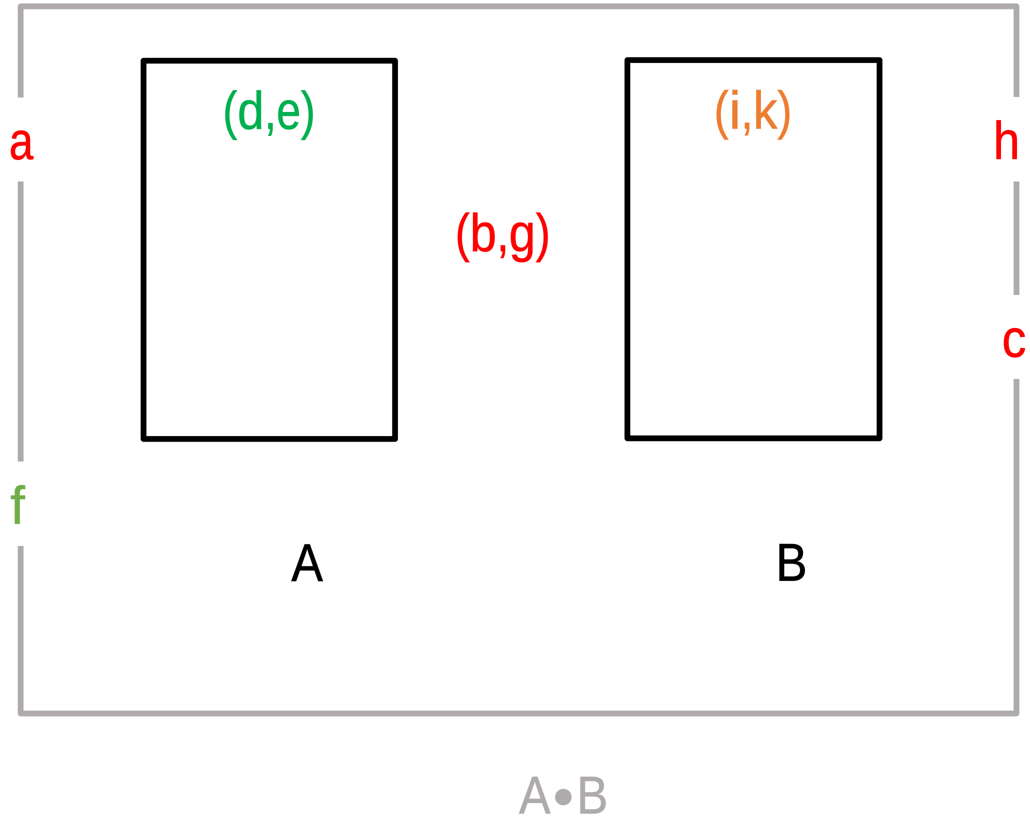


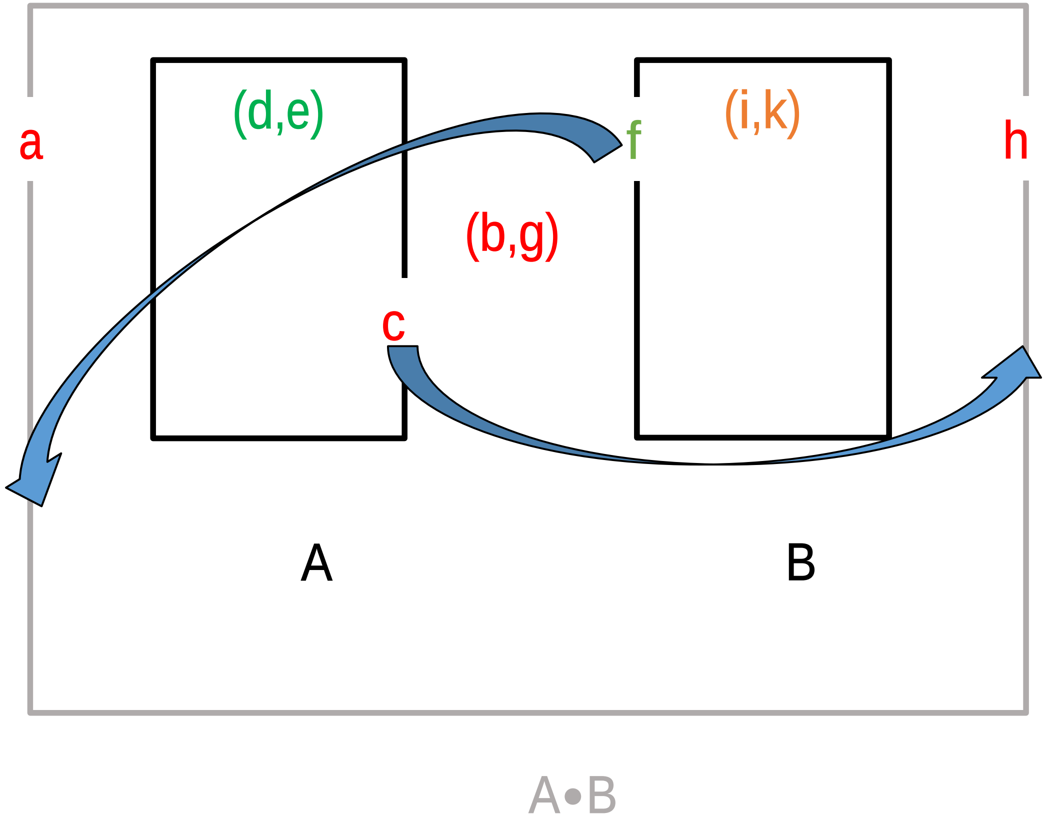
A•B



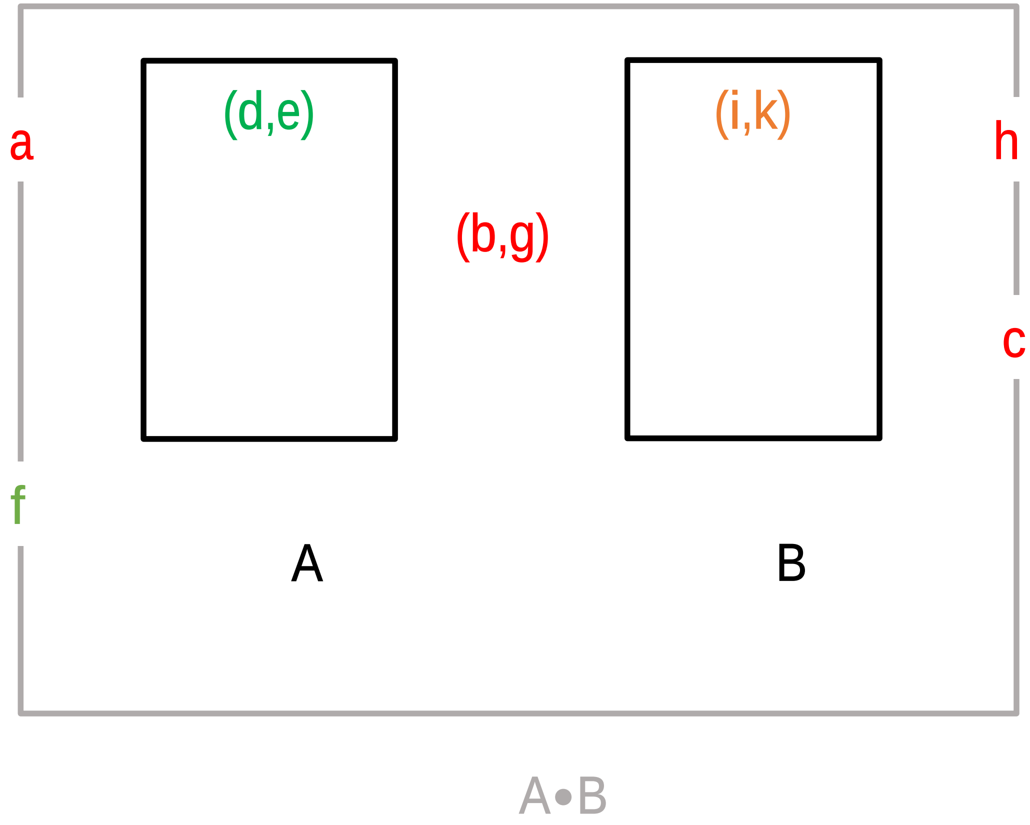
A•B


a
(d,e)
f
(i,k)
h
(b,g)
c
A
B
A•B

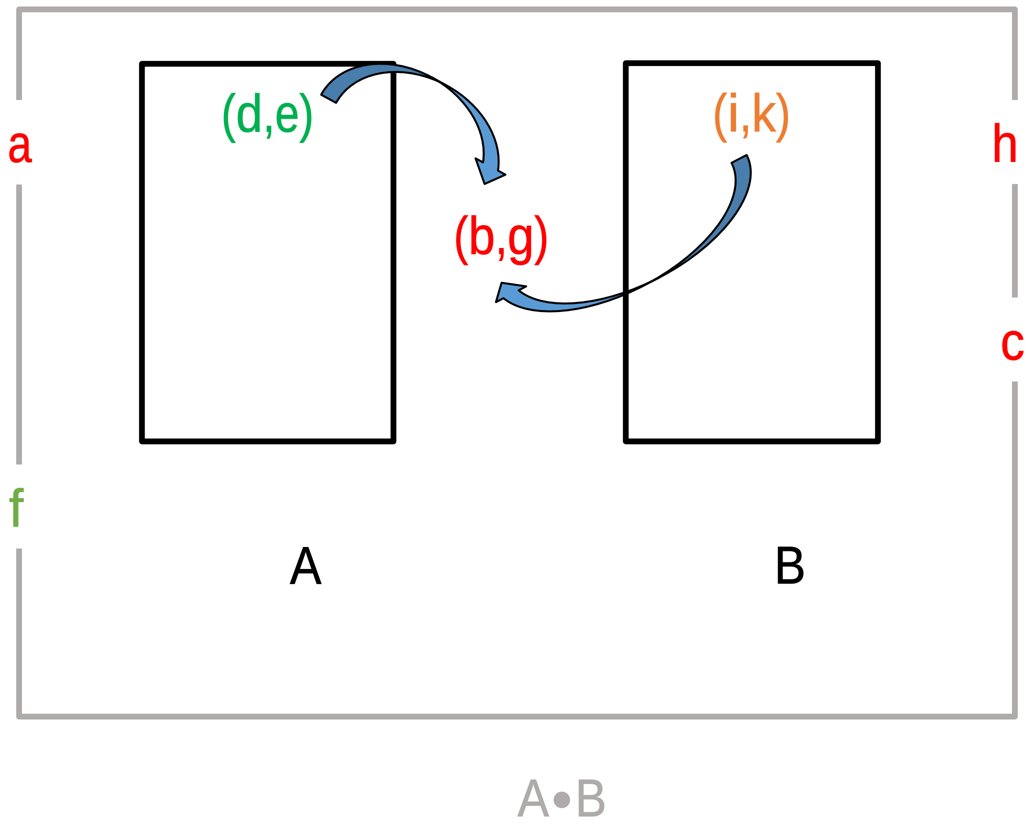

a
(d,e)
(i,k)
h
(b,g)
c
f
A
B
A•B



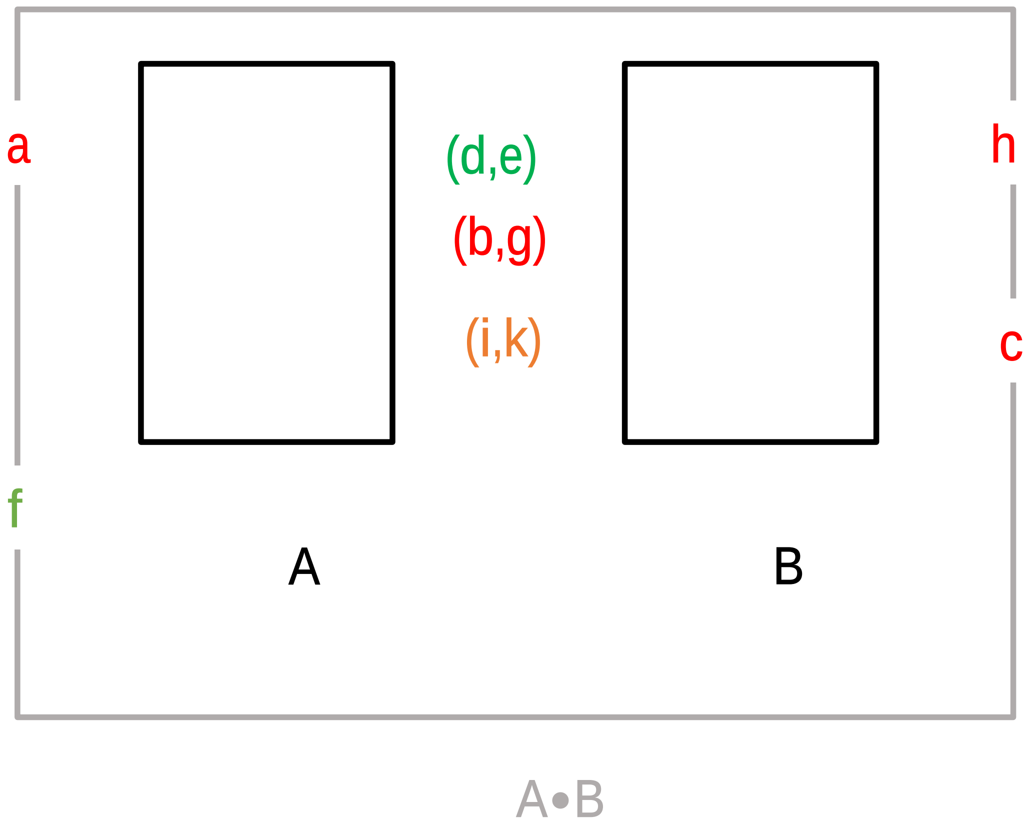
a
(d,e)
f
(i,k)
h
(b,g)
c
A
B
A•B



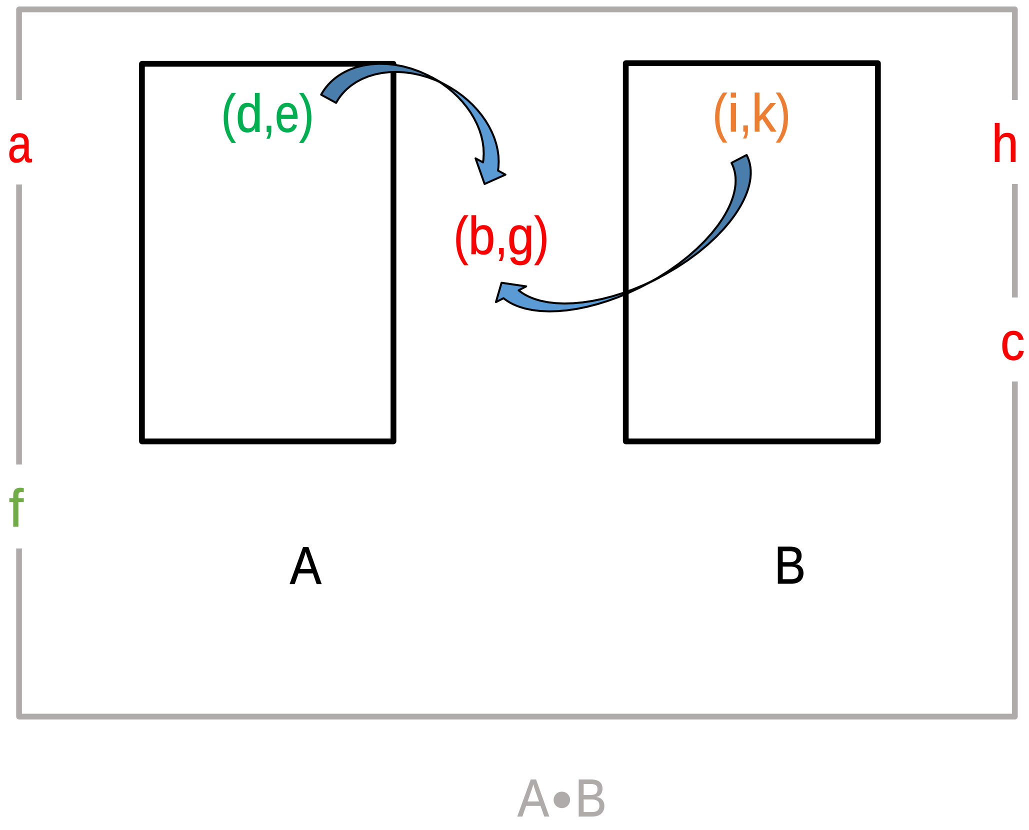
a
(d,e)
(i,k)
h
(b,g)
c
f
A
B
A•B



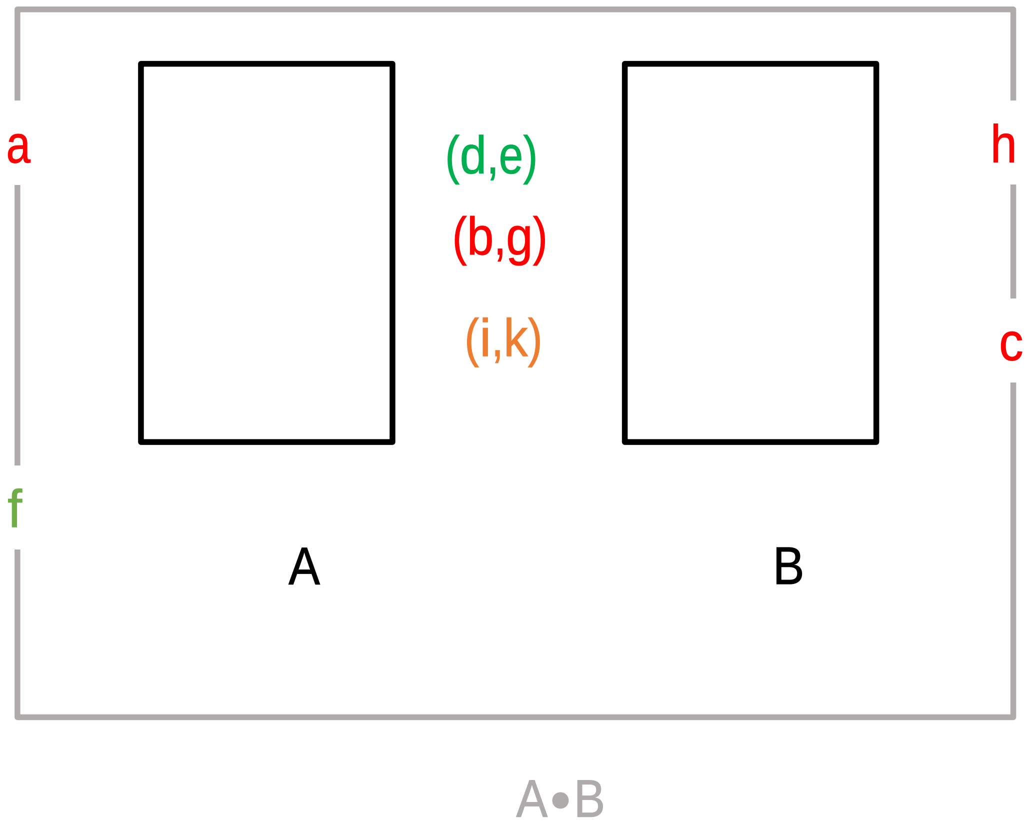
a
(d,e)
(i,k)
h
(b,g)
c
f
A
B
A•B



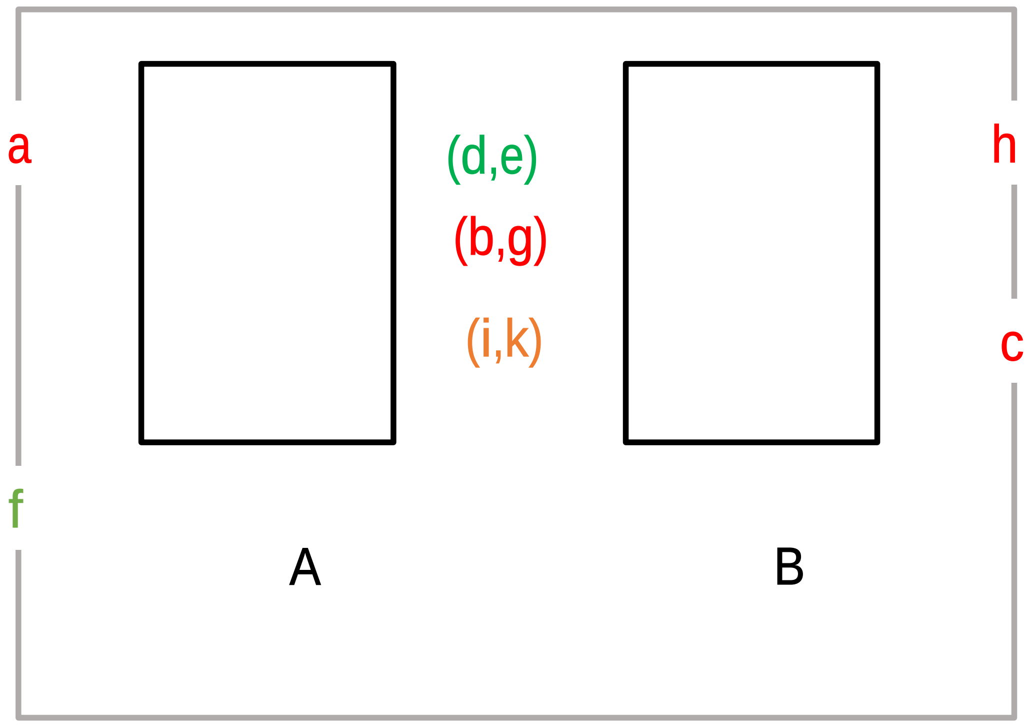
a
(d,e)
h
(b,g)
(i,k)
c
f
A
B
A•B



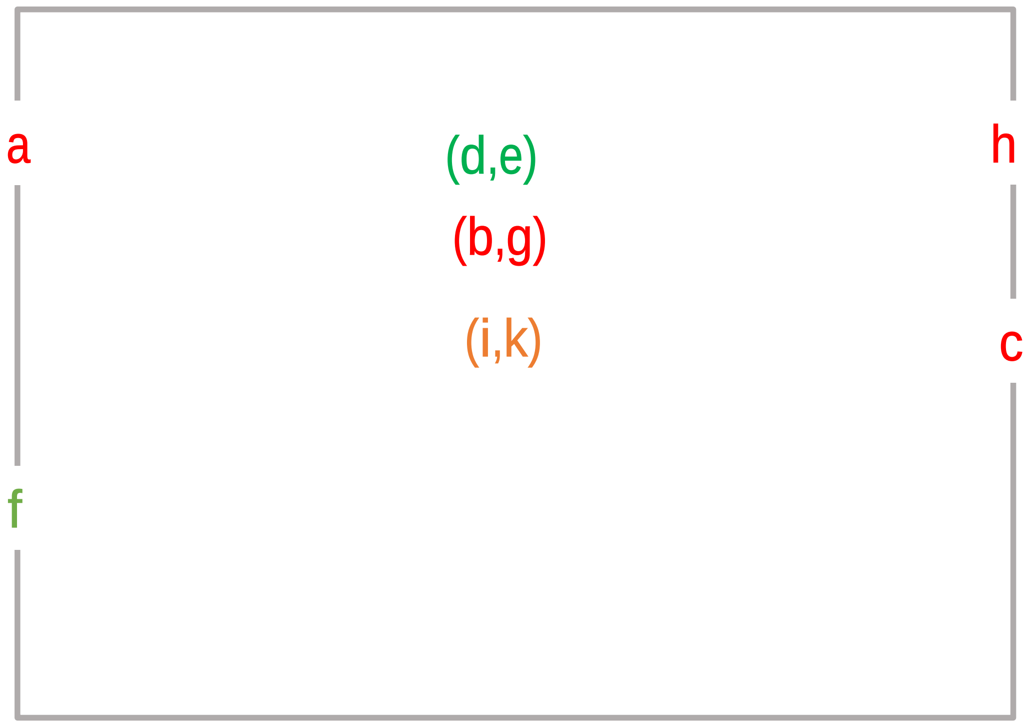
a
(d,e)
(i,k)
h
(b,g)
c
f
A
B
A•B



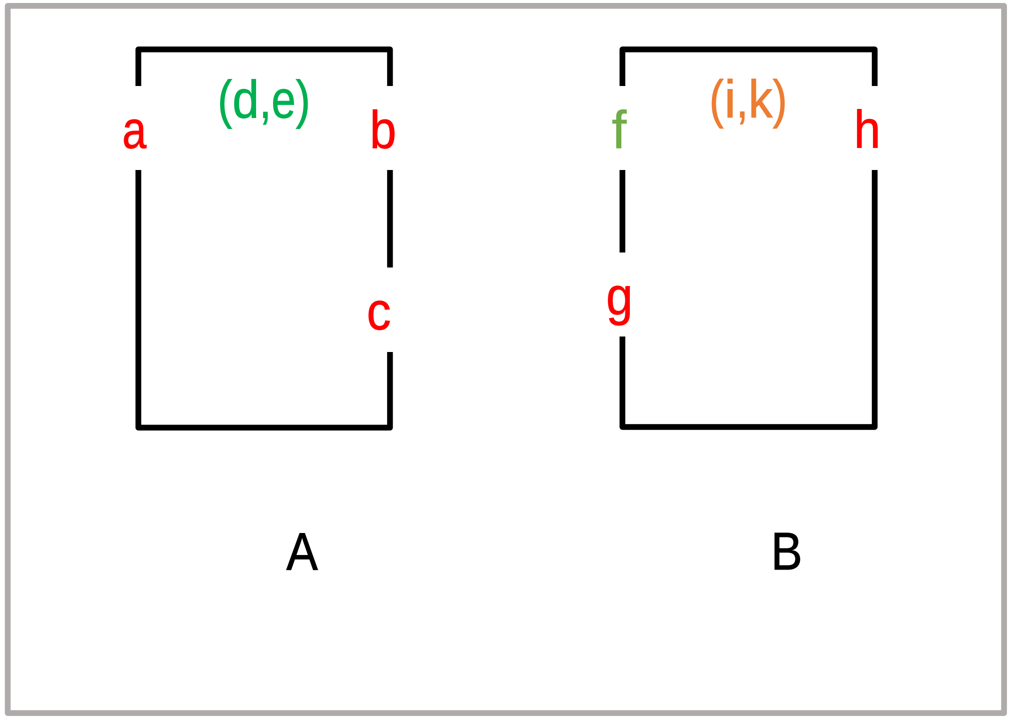
a
(d,e)
h
(b,g)
(i,k)
c
f
A
B
A•B



A•B

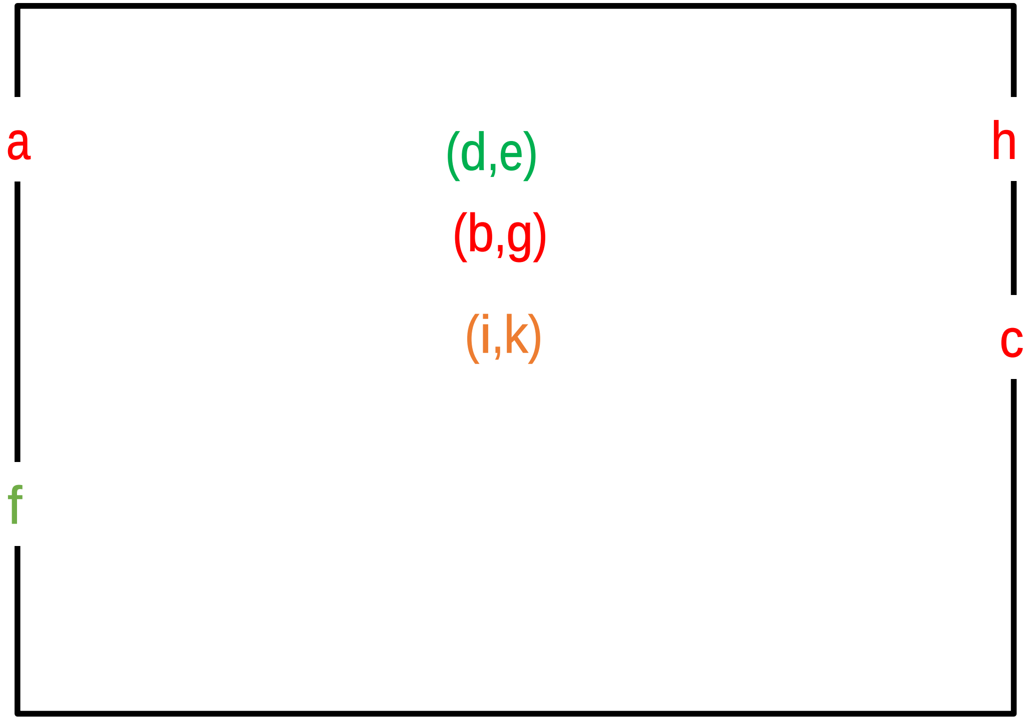


A•B



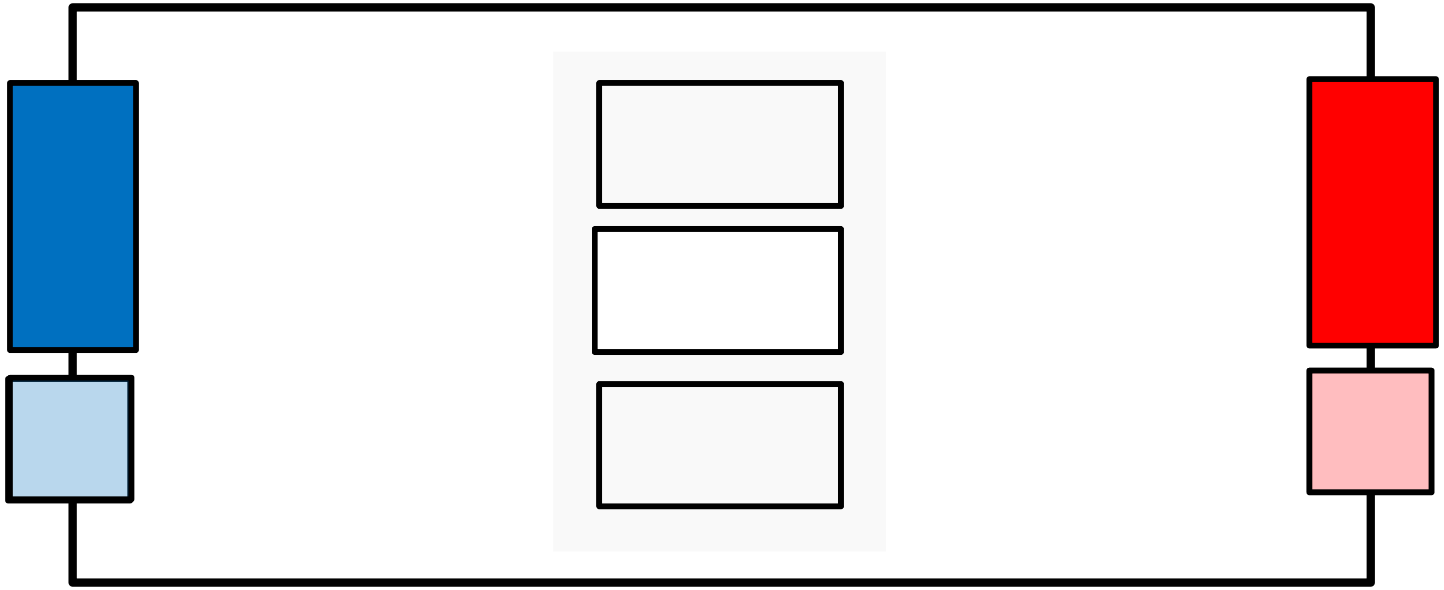
A•B

remember

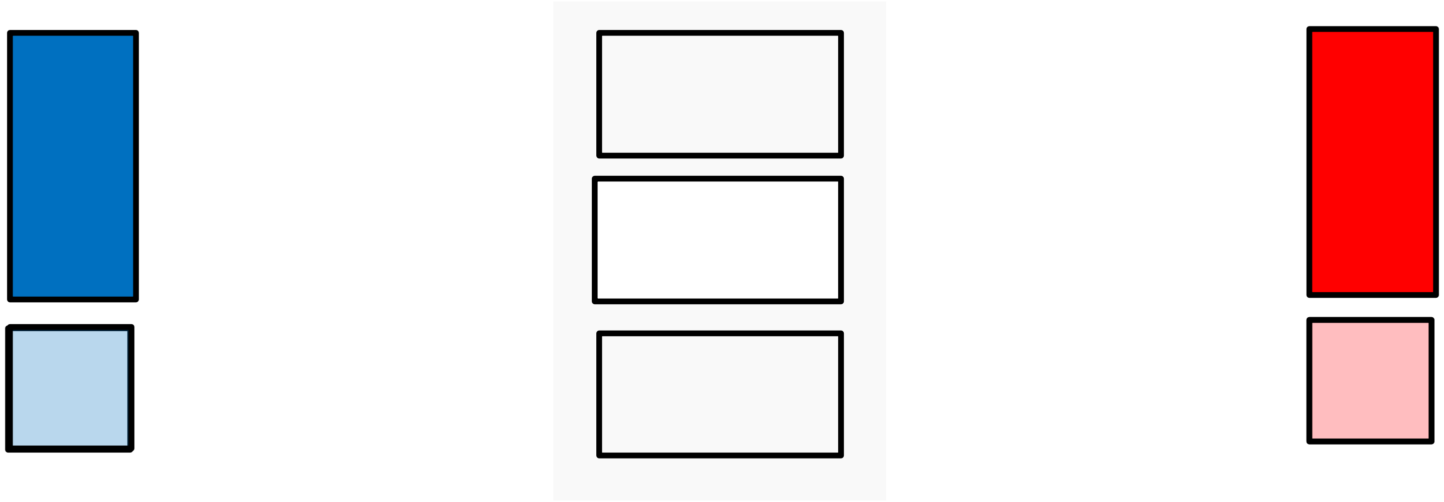


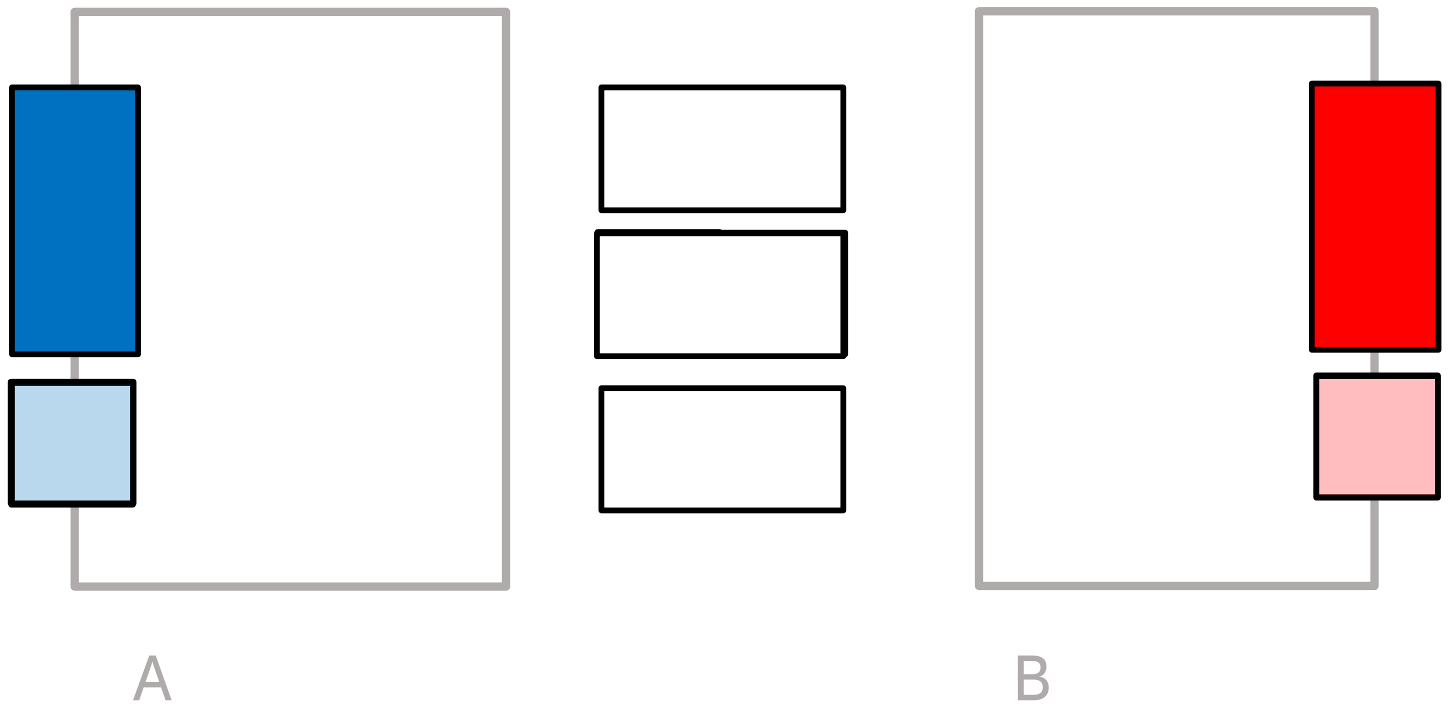
A•B

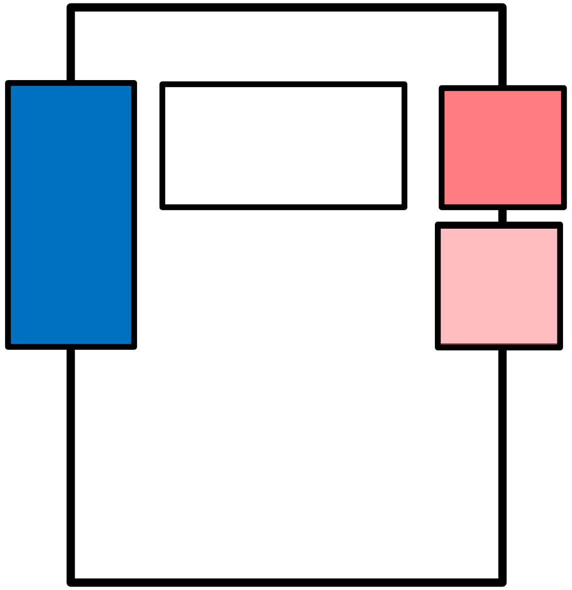
… and now we decompose



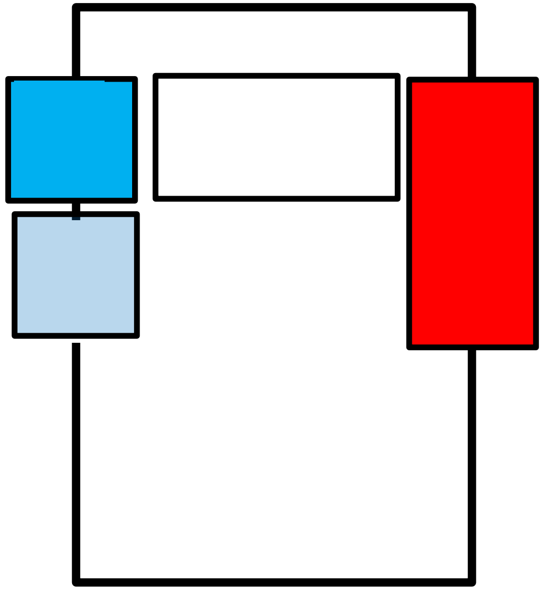
Component C
C und C are bi-partitioned.
seam(C) is 3-partitioned.




A
B



A



B

Advantages …

Early composition,

One-fits-all – definition of composition,

Semantics of components: formulated in *any* formalism,

Associativity is for free.

… so, what did we achieve, conceptually?

- Canonical, universal principles for the composition of components, independent of their semantical contents.

- Strict at the interface,
  Liberal at the inner structure.

Bernhard, this means autonomy

- Supporting horizontal scaling

As required by Gregor Hohpe

Resumé of this talk

We need a theory for the Digital infrastructure!

… deep and strong as computability theory

but for a much broader area

Potential ingrediences:

- *Model(s) that lean more on the problem side*
- *Theoretical notions based on invariants*
- *“Distributedness” from scratch,*
- *A universal composition operator*

So, let’s do it!

SUMMERSOC Χερσόνησος June 19, 2019

Conceptual Foundations of Service Orientation and beyond

- *Model(s) that lean more on the problem side*
- *Theoretical notions based on invariants*
- *"Distributedness" from scratch,*
- *A universal composition operator*

So, let's do it!

Humboldt-Universität zu Berlin

Humboldt Universität Informatik

Institut für Informatik
Humboldt-Universität zu Berlin

Prof. Dr. W. Reisig