How to model the internet of everything

SummerSoc 22 July 7, 2022

Peter Fettke

German Research Center for Artificial Intelligence (DFKI) and Saarland University, Saarbrücken, Germany

Wolfgang Reisig

Humboldt-Universität zu Berlin Germany

How to model the internet of everything

yesterday:

Rick Kazman:

Challenges in Architecting

Web 4.0 systems

“enormous design debts

in today’s software”

“How can we analyze this

kind of (software) systems?”

Don’t do it!

Start with modeling

- the real world problem involved,
- as well as its solution!

“make the company’s administration more efficient”

“How can we analyze models?”

Generate software from models!

“This has been attempt-ted several times

and always failed”

!Try harder!

Use the right

modeling concepts!

Which ones?

You will see

How to model the internet of everything

All engineering sciences

start with models!

CS starts with the product.

usual way:

First you write a lousy software,

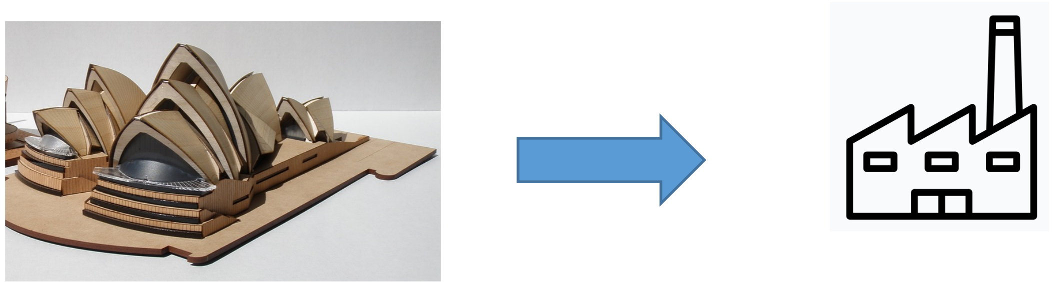
then you try to repair it.

Modeling in informatics
is about changing the perspective.

Car prodution:

Can you imagine?

Instead:

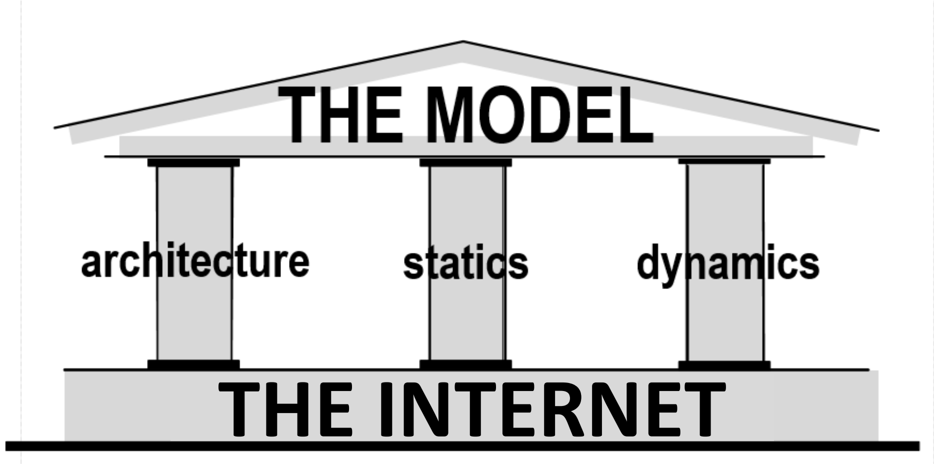


How to model the internet of everything

Three postulates:

a comprehensive model of the internet …

- is *structured* **architecture**
- includes data ***and*** *real life items* **statics**
- is *locally* updated. **dynamics**



1. Architecture

modeling techniques

with modules and

parallel composition

- process algebras,
- csp,
- statecharts

all of them focus implementability,

not (big) systems!

we need a fundamental technique more general,

more abstract,

more simple

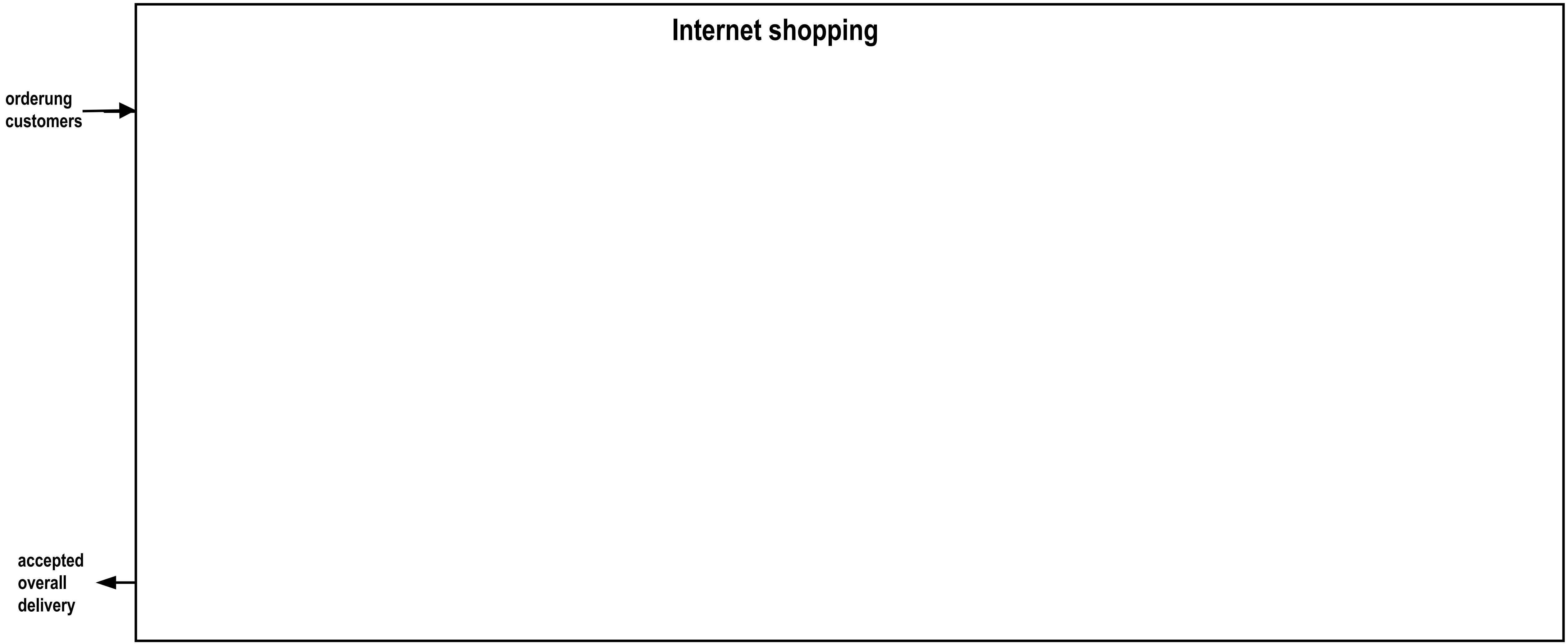
focussing the real world problem

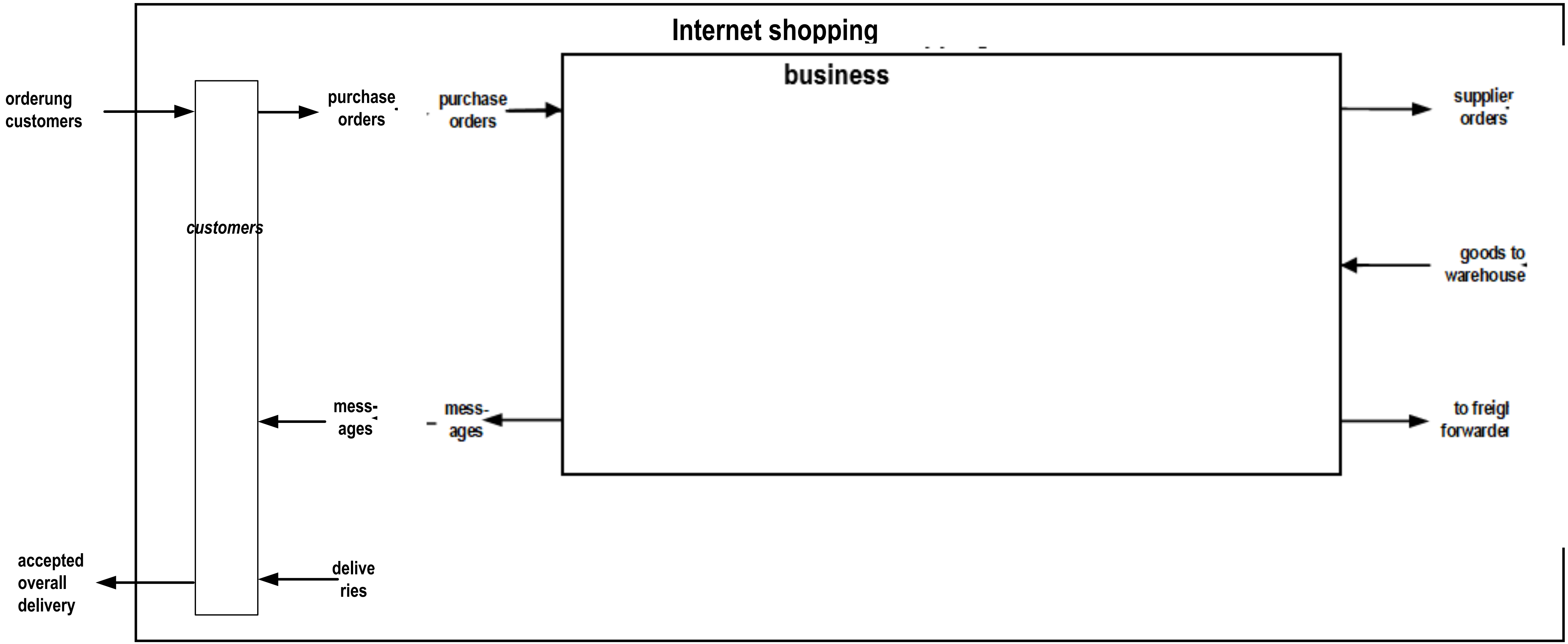
business process modeling

- EPC
- UML-AD
- RAD
- IDE

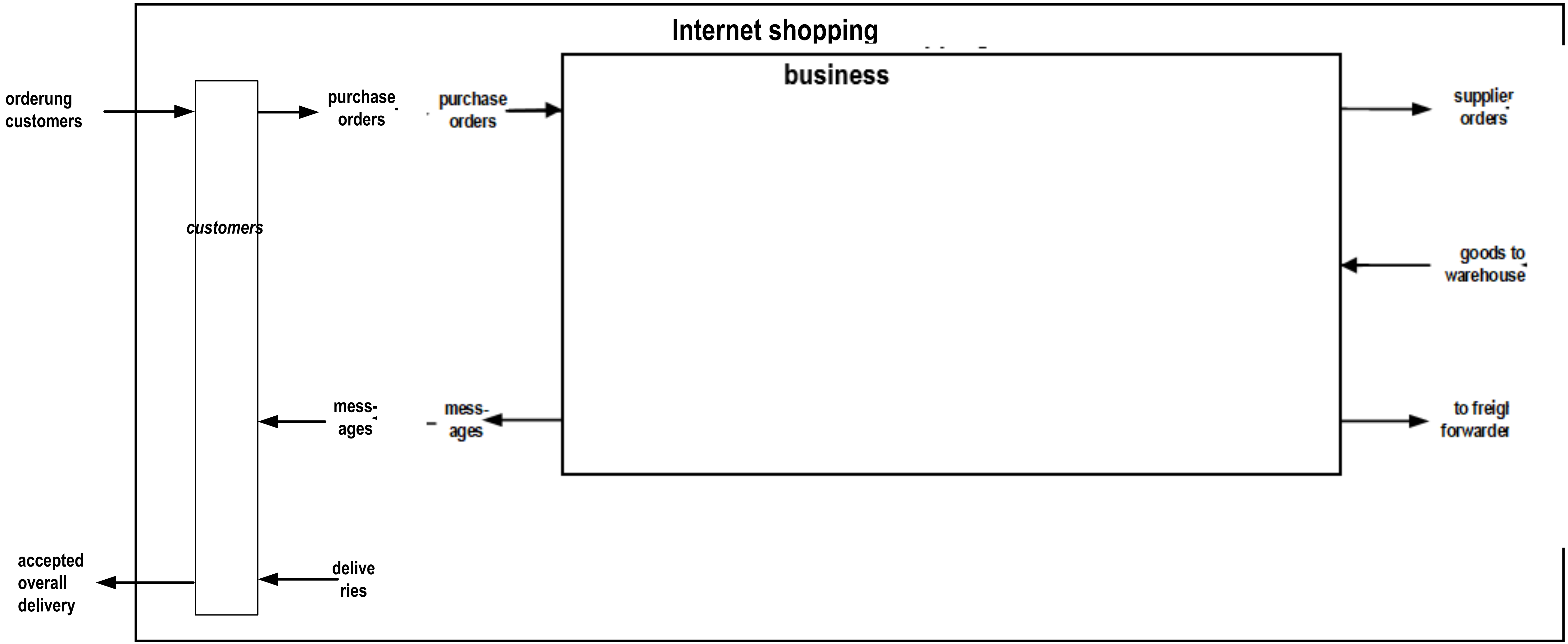
Architecture languages

- ABACUS
- ACME / ADML
- ArchiMate
- ByADL
- DEMO
- LePUS3 and Class-Z
- Rapide
- Wright

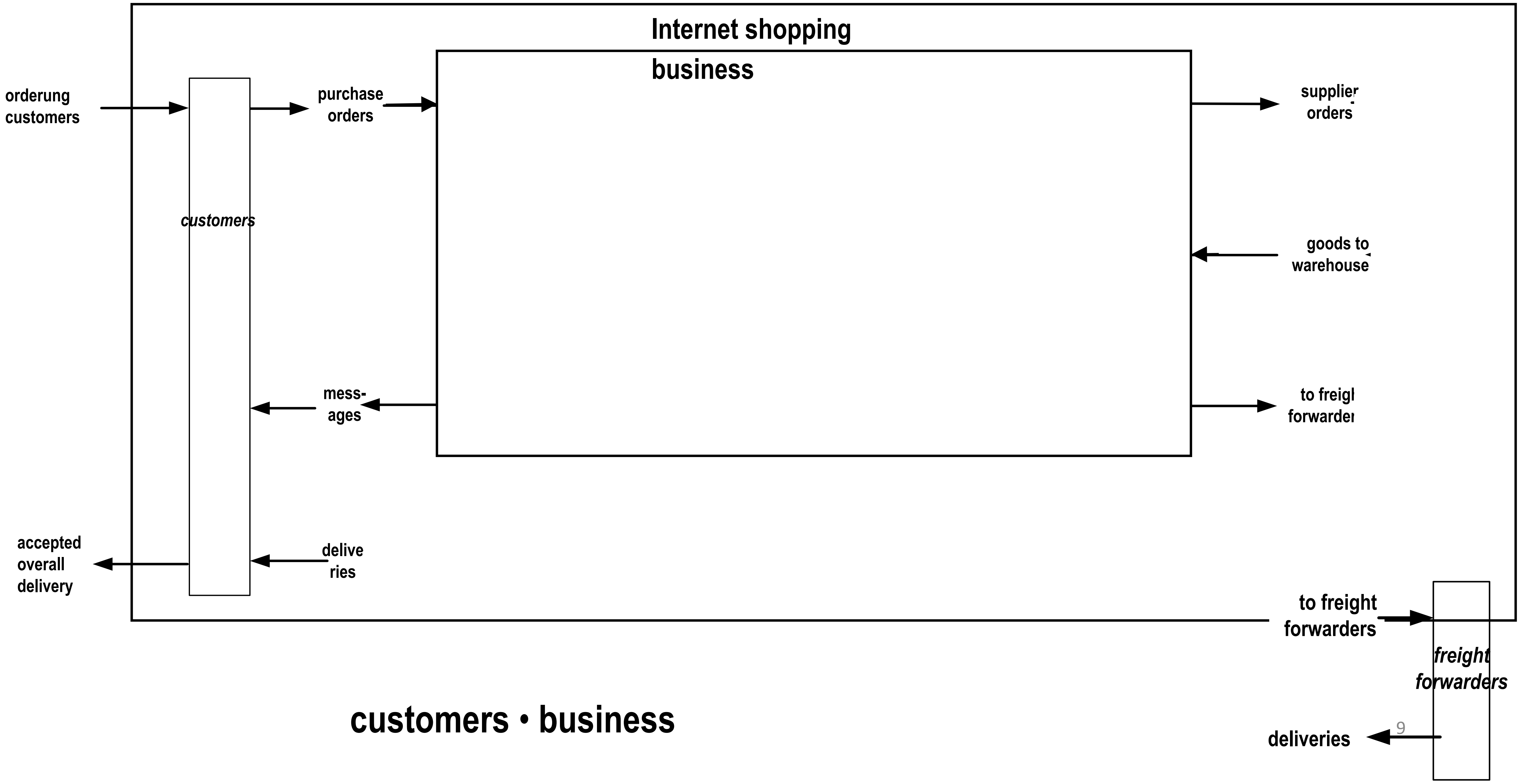

Internet shopping
orderung
customers
accepted
overall
delivery



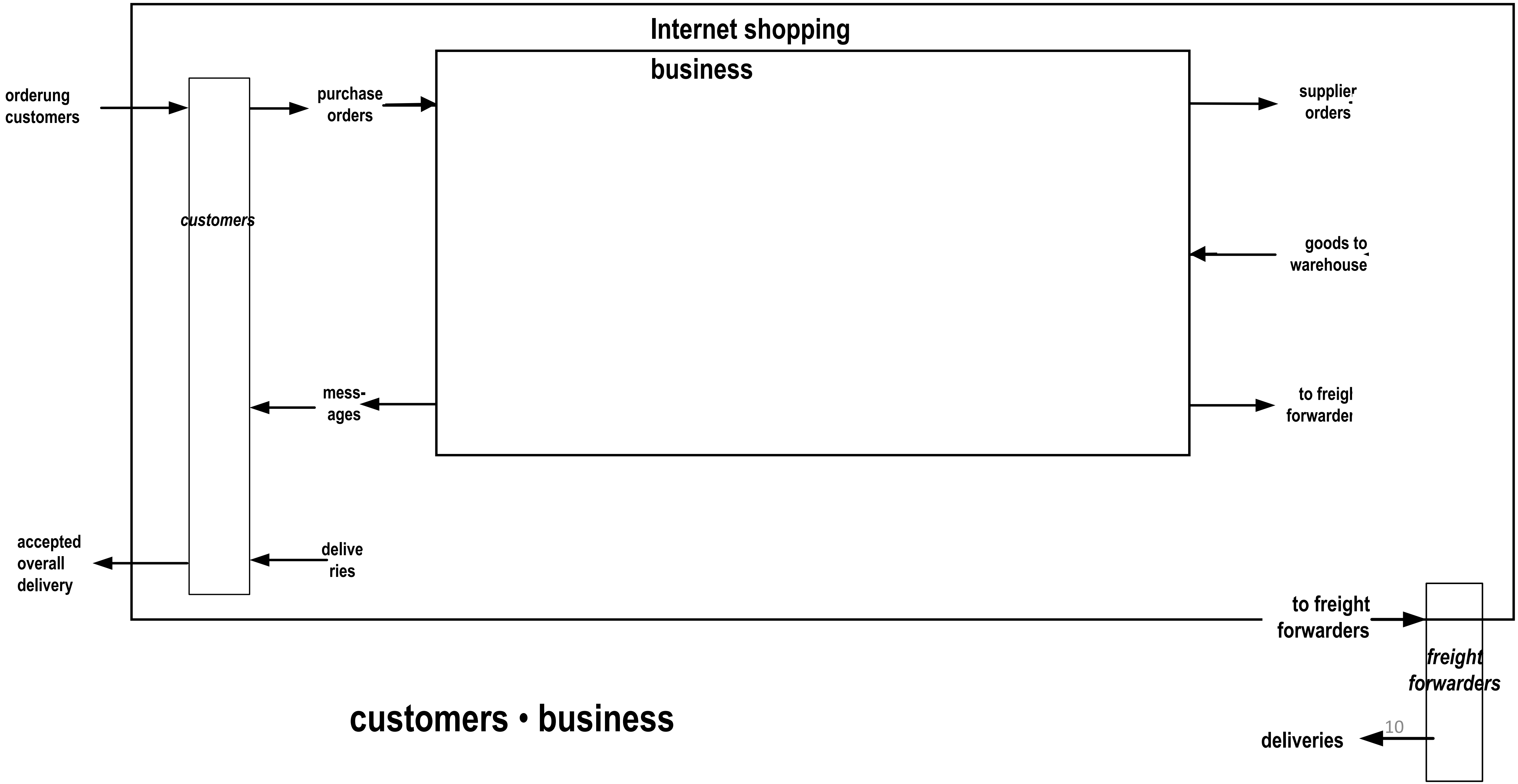
Internet shopping
business
orderung customers
customers
purchase orders
purchase orders
supplier orders
goods to warehouse
mess-ages
mess-ages
to freigl forwarder
accepted overall delivery
delive ries



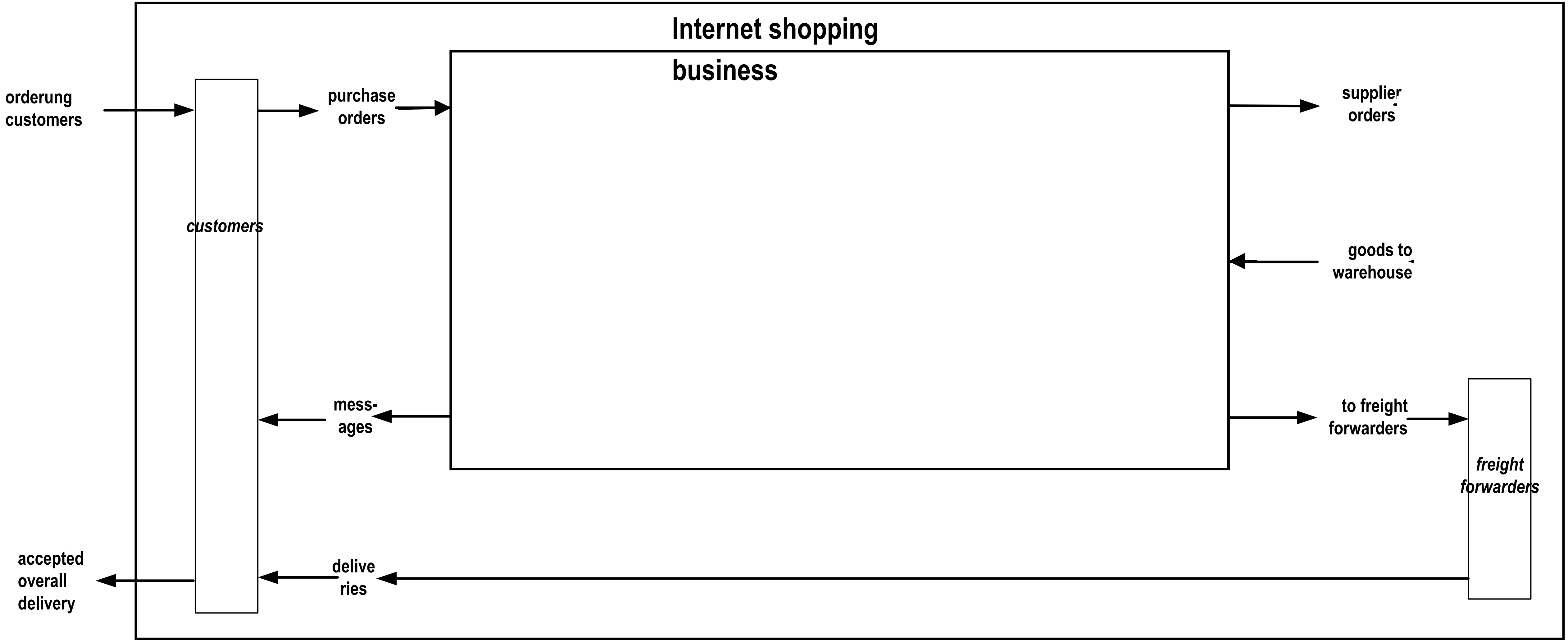
Internet shopping
business
orderung customers
customers
purchase orders
purchase orders
supplier orders
goods to warehouse
mess-ages
mess-ages
to freigl forwarder
accepted overall delivery
delive ries



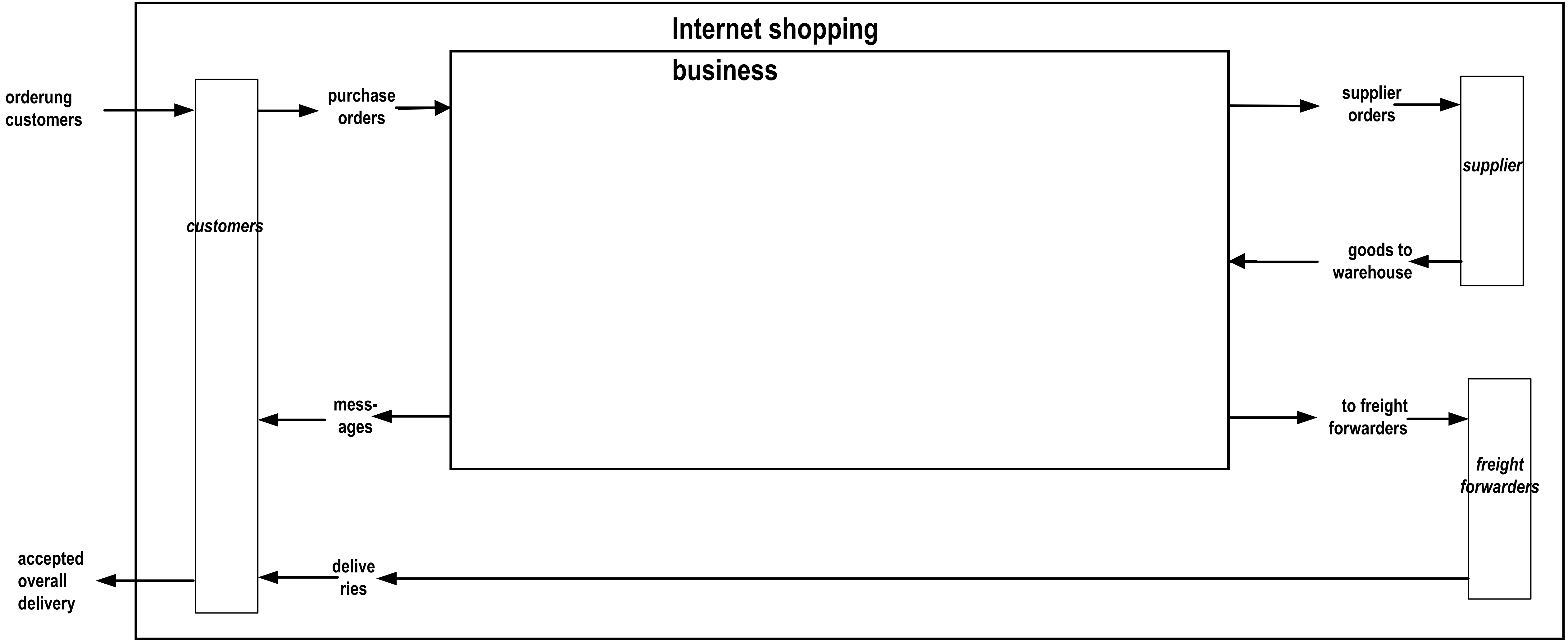
customers • business



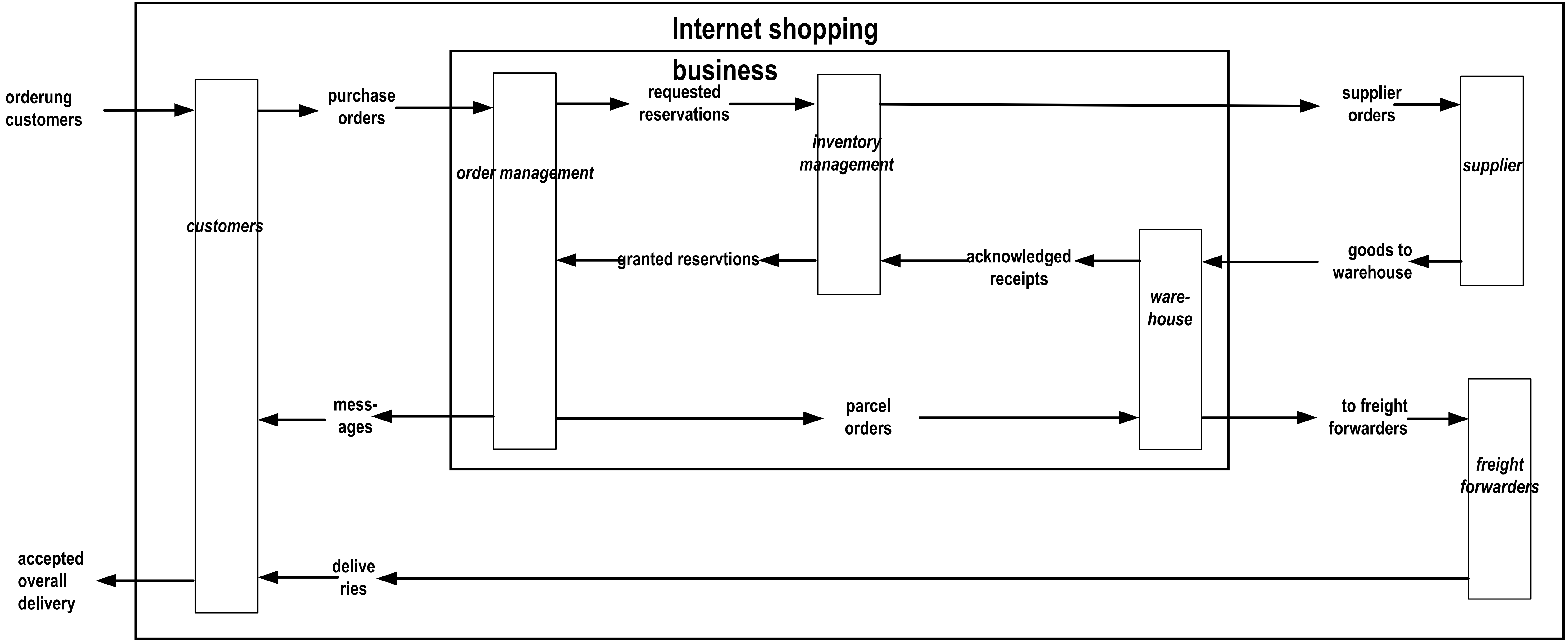
customers • business



customers • business • freight forwarders



customers • business • freight forwarders • suppliers



customers • business • freight forwarders • suppliers
where business = order management • inventory management • warehouse

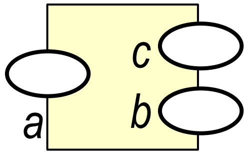
1 Architecture: modules and composition

Def. *Module:*

graph G with two distinguished

sets of labelled nodes, *G and G*

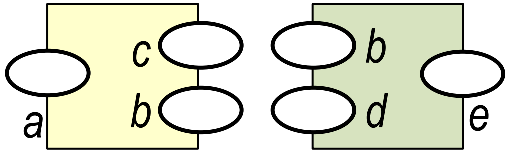
left and *right* interface



Theorem. (G • H) • K = G • (H • K).

Def.: *Composition* G • H of modules G and H:

merge equally labelled nodes of G* and *H.



1. Architecture: modules are intuitive!

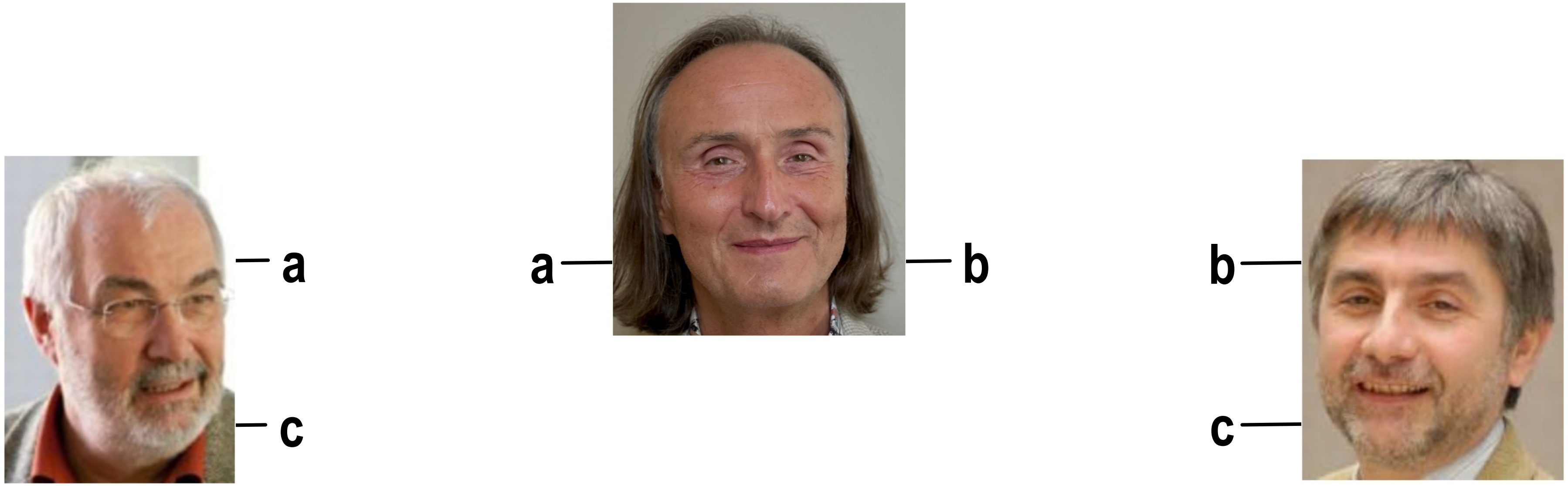
left and a ***right*** interface are quite natural:

G** and **G

- ***input*** and ***output*,**
- ***customer*** and ***supplier*,**
- ***requester*** and ***provider*,**
- ***consumer*** and ***producer*,**
- ***buy side*** and ***sell side*,**
- ***predecessor*** and ***successor*,**
- ***assumptions*** and ***guarantees*,**
- ***pull*** and ***push*,**
- ***left*** and ***right*,**

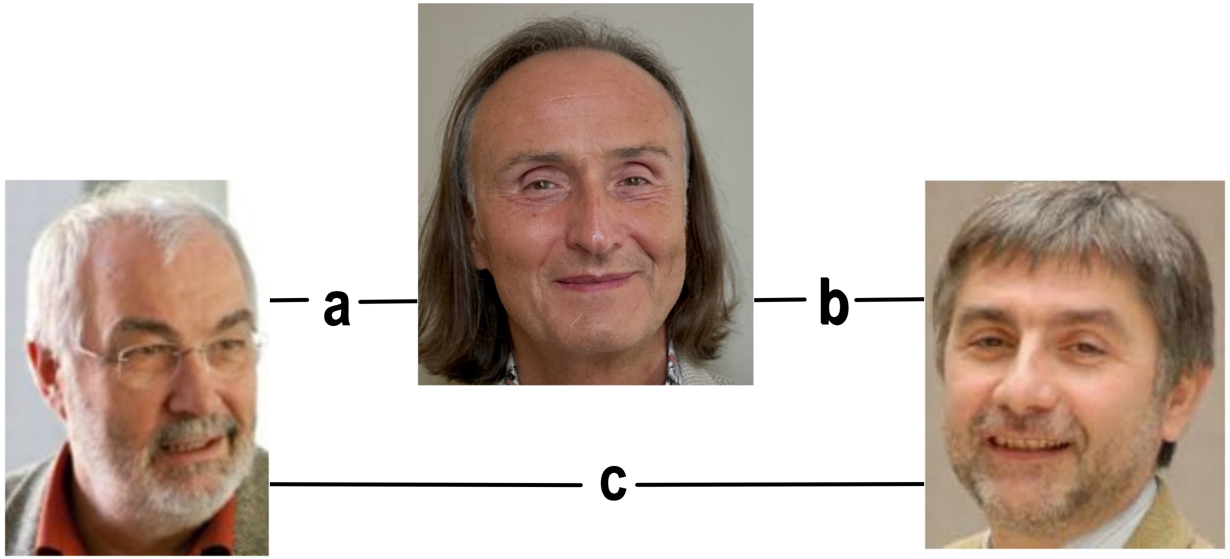
etc.

1. Architecture: Example



W • F • B

1. Architecture: Example

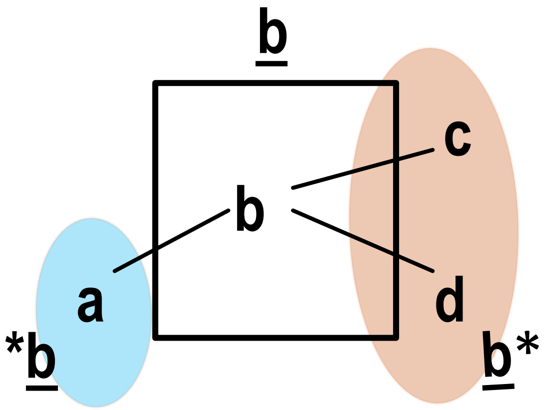


W • F • B

1. Architecture: composition is universal!

Each graph node yields node modules!

a node module $\underline{\mathbf{b}}$ *of node* **b**



Theorem. Each graph can be composed from node modules

$\underline{\mathbf{a}} \bullet \underline{\mathbf{b}} \bullet \underline{\mathbf{c}} \bullet \underline{\mathbf{d}}$

interfaces?

Composition with …

- *one interface: not associative* (G • H) • K $\neq$ G • (H • K).

- *three interfaces: - not necessary (above Theorem)*

  - technically intricate

- **two** interfaces: the basis

  for ***any*** kind of composition!

1. Architecture: how manage complex composition?

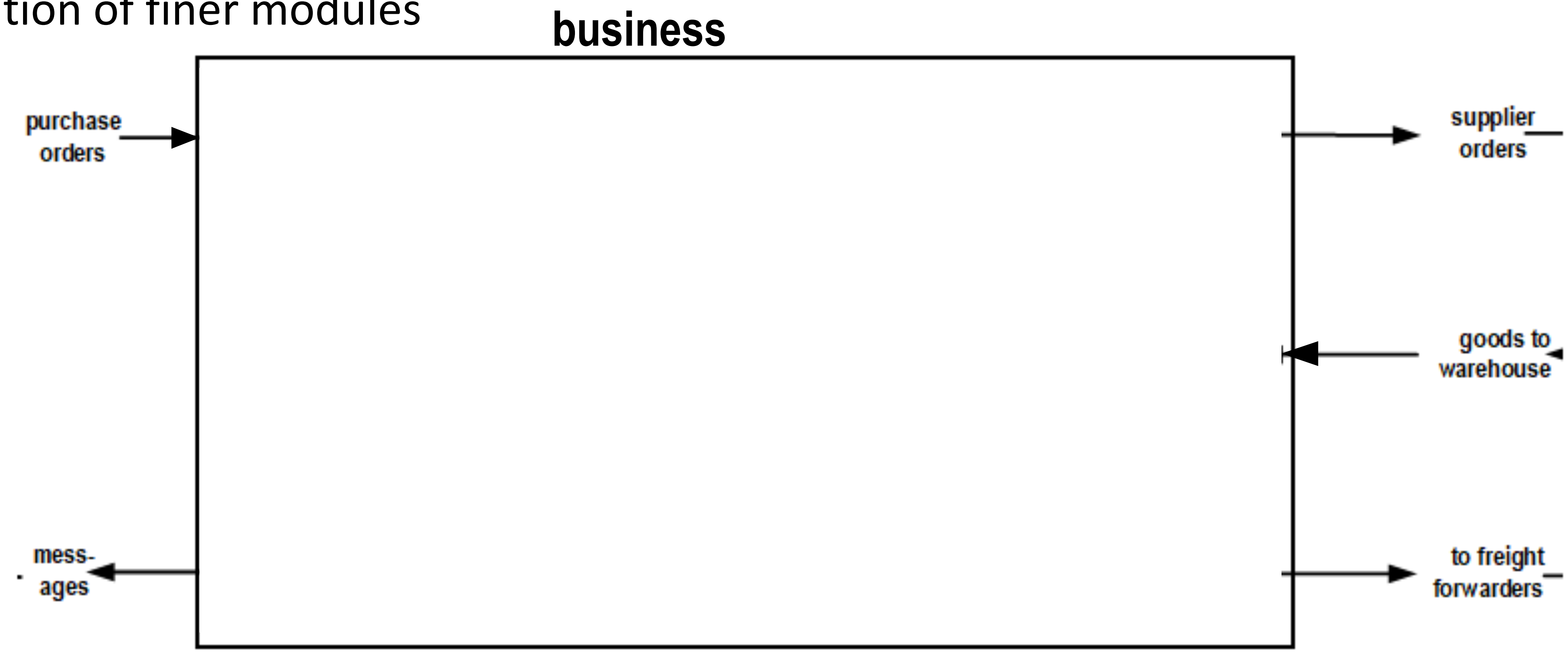
… by help of adapters:

replace A * B

by A • adapter • B,

1. Architecture: how manage refinement?

… by composition of finer modules



business = order management • inventory management • warehouse

1. Architecture: how manage “really big” systems?

by keeping every concept ***local, inside modules!***

- level of abstraction
- composition
- refinement
- adapter
- data (few global names)
- behavior (no global states)

2. Statics

statically given:

- data,
- real life items.
- operations on data and items.

How organize all this?

Learn from Logic!

Structures and Signatures

a structure

ground sets

derived sets

constants

functions

another structure

ground sets
KN = all id card holders *customers*
AR = {small car, medium, large} *articles*
WA = {VW1, VW2, … FORD1, FORD2, MERCEDES} *goods*
TE = {1.1. … 31.12.} *dates*
SP = {Maier, Müller, Schulz} *freight forwarders*

derived sets
AP = **AR x ℕ** *article items*
AL = ***M*(AP)** *article lists*
AM = ***M*(AR)** *sets of articles*
WM = ***M*(WA)** *sets of goods*

constants
p1: AP = (small car, 4),
p2: AP = (big, 2)
K: P(KN) = {Peter} *ordering customers*
G: AL = {(small car, 30), (medium, 2), (large, 1)} *initially listed articles*
H: WM = {VW, …. VW, BMW, BMW} *initially listed goods*
R: P(SP) = {Maier, Müller} *initally available freight forwarders*

functions
f(w) = „w“, for w ∈ **WA**

(a,n)‘ = n•[a] for (a,n) ∈ **AP**

a Σ–structure

ground sets

KN = all id card holders *customers*

AR = {small car, medium, large} *articles*

WA = {VW1, VW2, … FORD1, FORD2, MERCEDES} *goods*

TE = {1.1. … 31.12.} *dates*

SP = {Maier, Müller, Schulz} *freight forwarders*

derived sets

AP = AR x ℕ *article items*

AL = *M*(AP) *article lists*

AM = *M*(AR) *sets of articles*

WM = *M*(WA) *sets of goods*

constants

p1: AP = (small car, 4),

p2: AP = (big, 2)

K: P(KN) = {Peter} *ordering customers*

G: AL = {(small car, 30), (medium, 2), (large, 1)} *initially listed articles*

H: WM = {VW, …. VW, BMW, BMW} *initially listed goods*

R: P(SP) = {Maier, Müller} *initally available freight forwarders*

a signature, Σ

ground sorts

KN *customers*

AR *articles*

WA *goods*

TE *dates*

SP *freight forwarders*

derived sorts

AP = AR × ℕ *article items*

AL = *M*(AP) *article lists*

AM = *M*(AR) *sets of articles*

WM = *M*(WA) *sets of goods*

constant symbols

p1, p2: AP *ordered article positions*

B: P(KN) *ordering customers*

G : AL *initially listed articles*

H : WL *initially available goods*

R : P(SP) *available freight forwarders*

function symbols

f : WA ⟶ AR

f : WL ⟶ AM

(_‘): AP ⟶ AM

(_‘): AL ⟶ AM

variables

k: KN

x: AR

X,Y: AL

Z: M(WA)

t: TE

w: WA

s: SP

m, n, p: ℕ

properties

(a,n)‘ = n[a] für (a,n)∈ **AP**

[p1, … , pn]‘ = p1‘ + … + pn‘ für

[p1, … , pn] ∈ **AL**

G‘ = f(H)

Terms of a signature Σ

f(x)

f(a, g(y))

Infinitely many terms

In a Σ-structure,

each term has a meaning!

a signature, Σ

ground sorts
KN *customers*
AR *articles*
WA *goods*
TE *dates*
SP *freight forwarders*

derived sorts
AP = **AR** $\times \mathbb{N}$ *article items*
AL = ***M*(AP)** *article lists*
AM = ***M*(AR)** *sets of articles*
WM = ***M*(WA)** *sets of goods*

constant symbols
p1, p2: AP *ordered article positions*
B: P(KN) *ordering customers*
G : AL *initially listed articles*
H : WL *initially available goods*
R : P(SP) *available freight forwarders*

function symbols
f : WA $\longrightarrow$ AR
f : WL $\longrightarrow$ AM
(_‘): AP $\longrightarrow$ AM
(_‘): AL $\longrightarrow$ AM

variables
k: KN
x: AR
X,Y: AL
Z: M(WA)
t: TE
w: WA
s: SP
m, n, p: $\mathbb{N}$

properties
(a,n)‘ = n[a] für (a,n)$\in$ **AP**
[p1, … , pn]‘ = p1‘ + … + pn‘ für [p1, … , pn] $\in$ **AL**

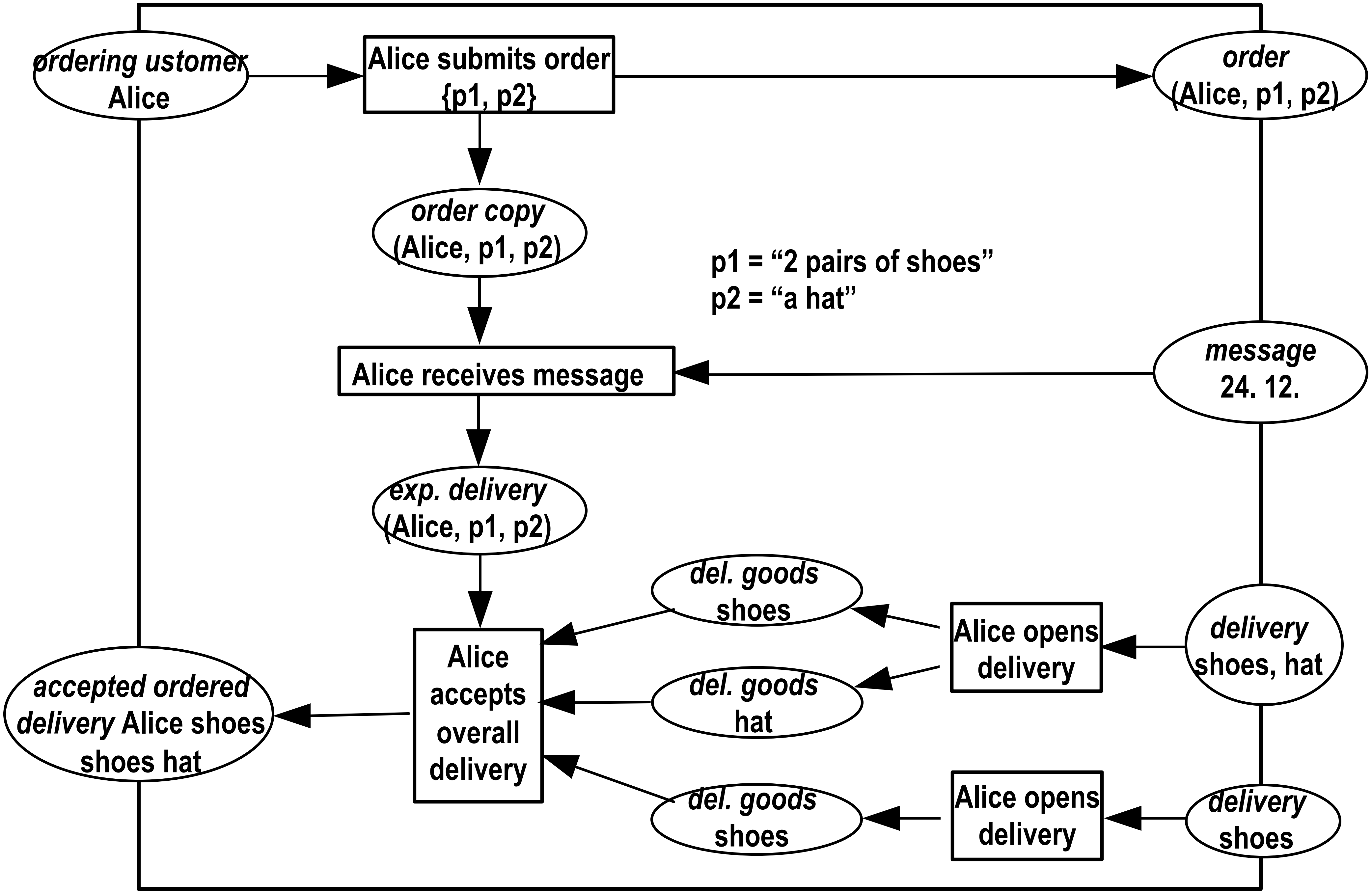
G‘ = **f(H)**

3. Dynamics

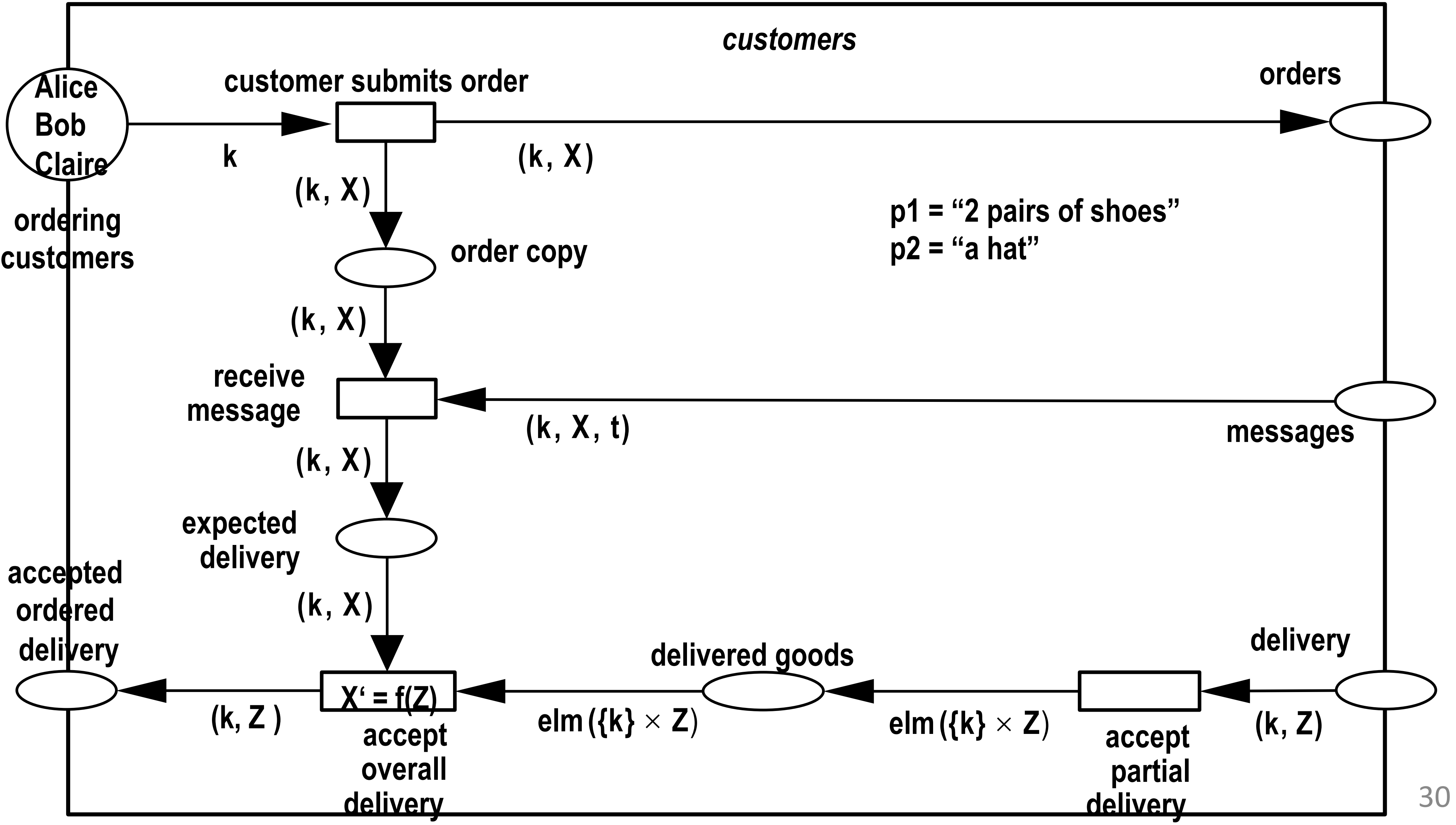
local states

an event has local cause and effect

run of custo-mer Alice

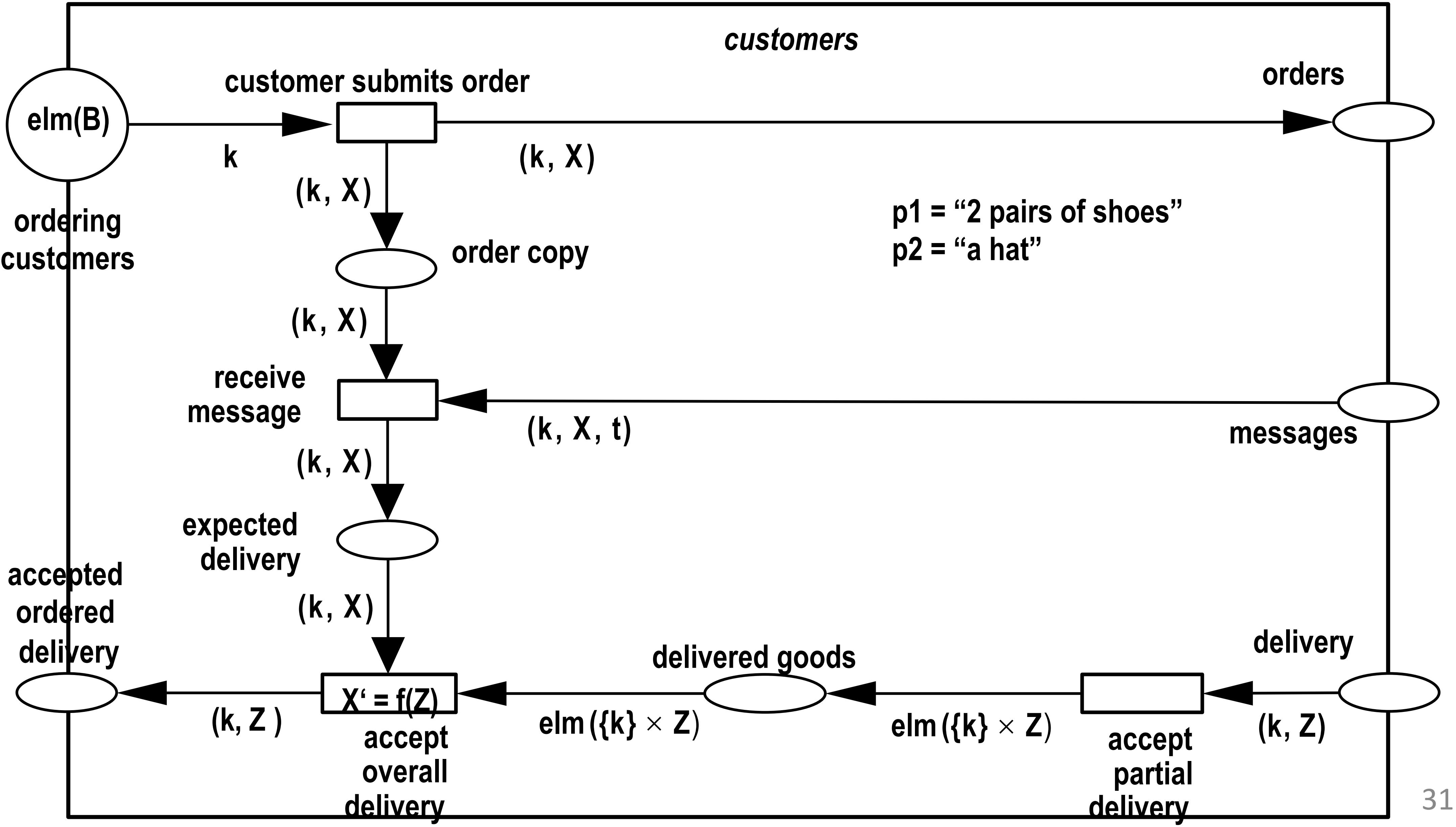


three custo-mers



k: *customer*
X: *article list*
Z: *set of goods*

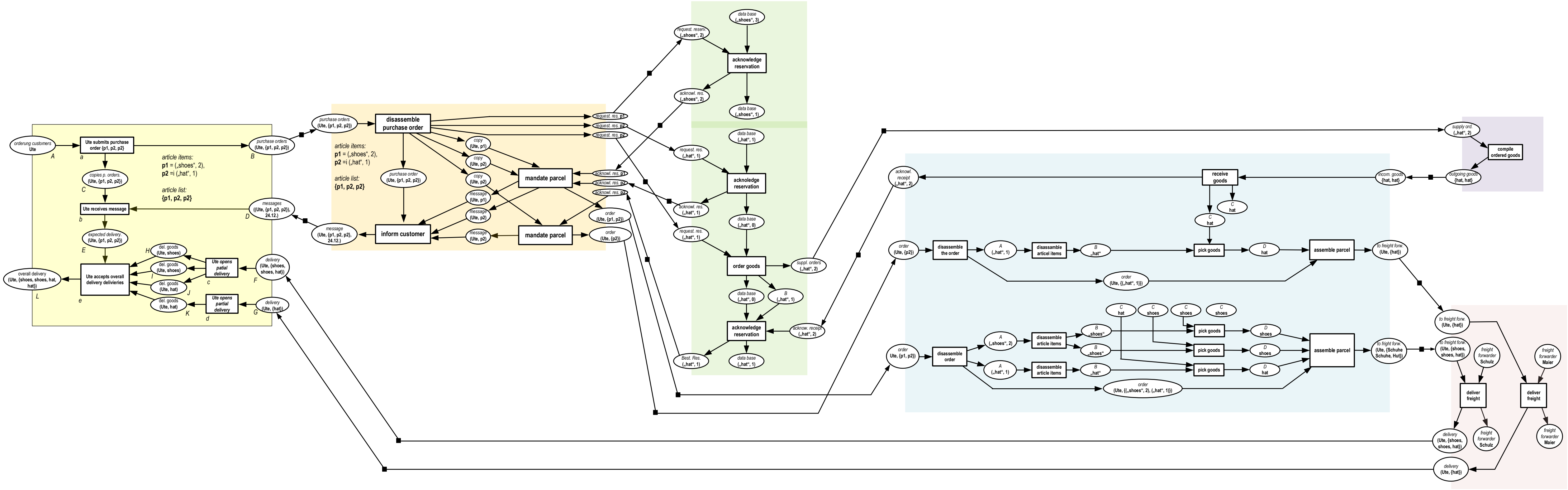
schema for custo-mers



k: *customer*
X: *article list*
Z: *set of goods*
B: *set of custo-mers*

4. Put it all together

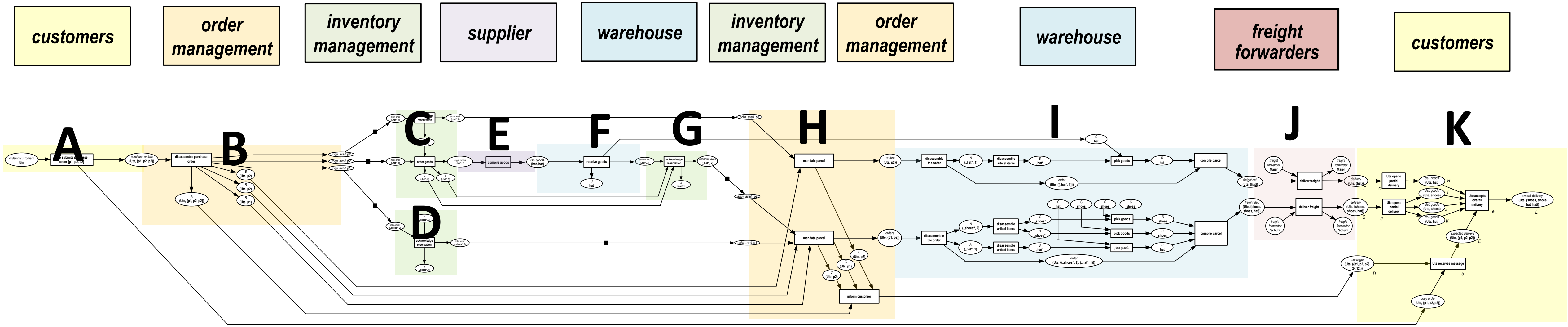
overall run of Alice‘s order



just write

<u>**customers**</u> **•** <u>**order management**</u> **•** <u>**inventory management**</u> **•** <u>**warehouse**</u> **•** <u>**supplier**</u> **•** <u>**freight forwarders**</u>

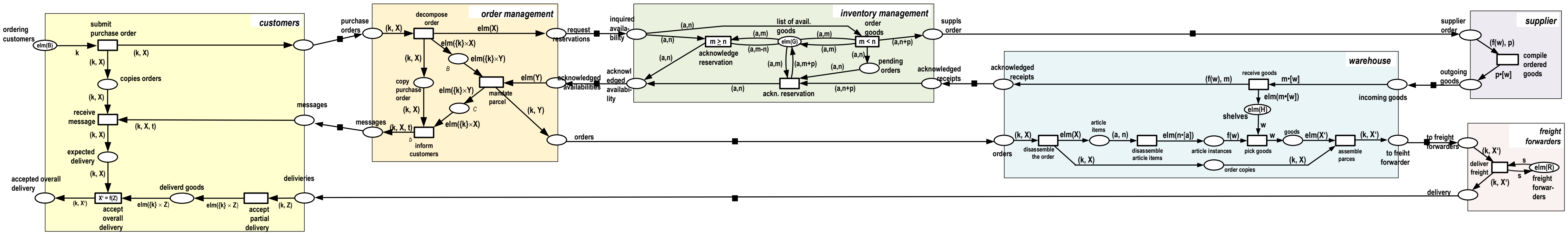
overall run of Alice‘s order represented from left to right



just write

A • B • C • D • E • F • G • H • I • J • K

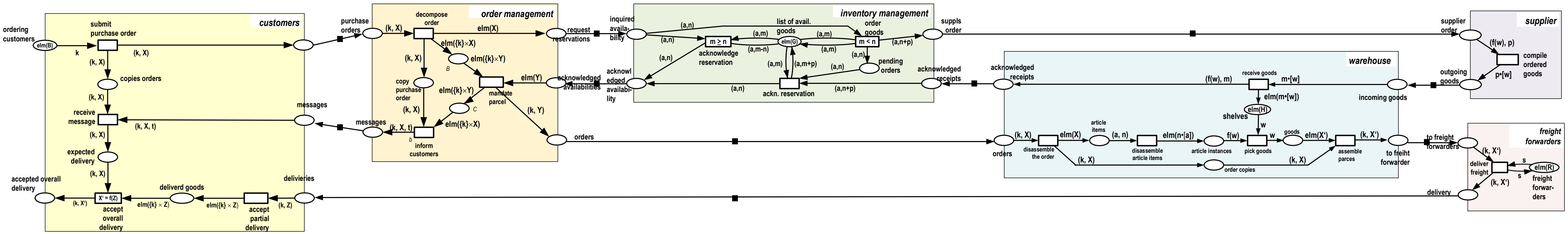
schema for interent shopping



just write

customers • order management • inventory management • warehouse • supplier • freight forwarders

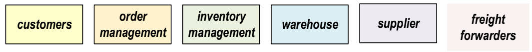
schema for interent shopping



This is a formal model!

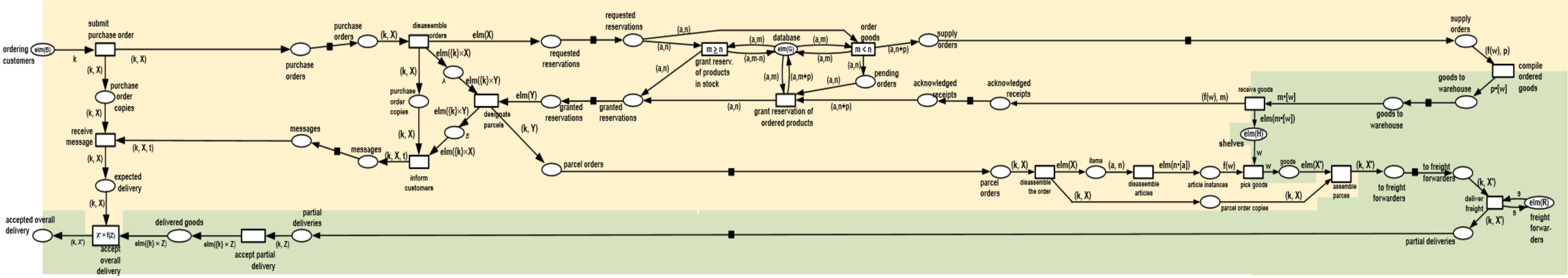
You can formally prove:

- Each order is eventually served.
- A good is delivered only if it has been ordered before.
- Each requested reservation is eventually acknowledged.
- Each mandated article has a good in the shelves.



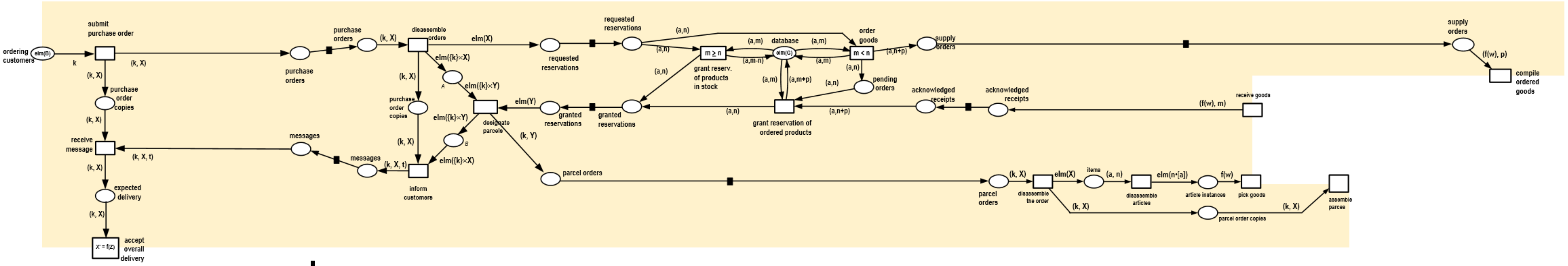
alternative refinement

business = paperwork • items



alternative refinement

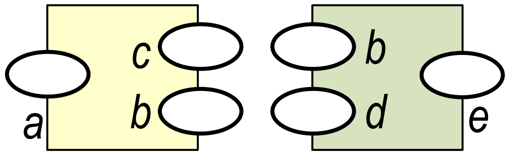
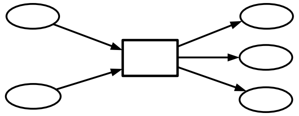
business = paperwork • items



paperwork

items

How to model the internet of everything

classical informatics	***… yes, but …***	***… adjusted***	***… such as***	***… technically***
modules and composition: merge „equal“ interface elements	yes, however not *one* interface	but two!		composition calculus
statics (data, items): symbolic representation	yes, however not with symbol *chains*	but with terms over a signature!	**f(x, g(a,y))**	predicate logic, algebraic spec.
dynamics: steps	yes, however, not *global* states and steps	but local ones!		Petri nets

classical computer science
- jumps in the right direction
- but falls short

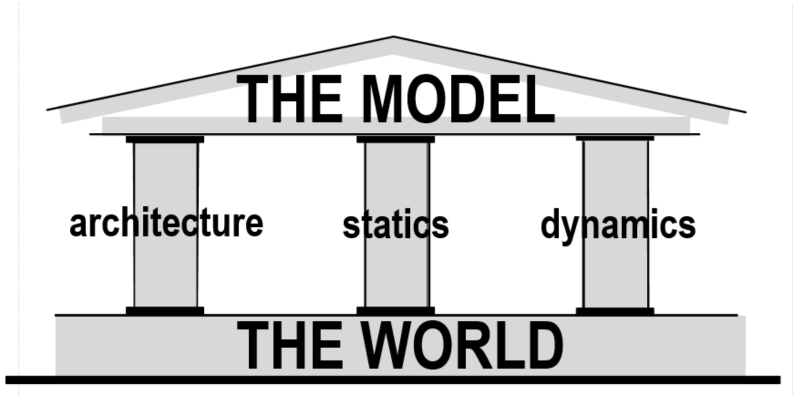
HERAKLIT adjusts this!

Don‘t hesitate!

Join the HERAKLIT.org research program and development project

How to model the internet of everything

SummerSoc 22 July 7, 2022



do it with

HERAKLIT

Peter Fettke

Wolfgang Reisig

German Research Center for Artificial Intelligence (DFKI) and Saarland University, Saarbrücken, Germany

Humboldt-Universität zu Berlin Germany

Institut für Informatik
Humboldt-Universität zu Berlin

UNIVERSITÄT DES SAARLANDES