PROGRAMMIER-PARADIGMEN IM BEZUG AUF MODELLIERUNG NEBENLÄUFIGER SYSTEME

Anwendungen I

von Sigurd Sippel

Betreuer: Kai v. Luck

NEBENLÄUFIGKEIT

Shared State

p: x = 0

t1: x = x + 1

t2: x = x + 1

t1:read(x) t1:write(x) t2:read(x) t2:write(x) x = 2

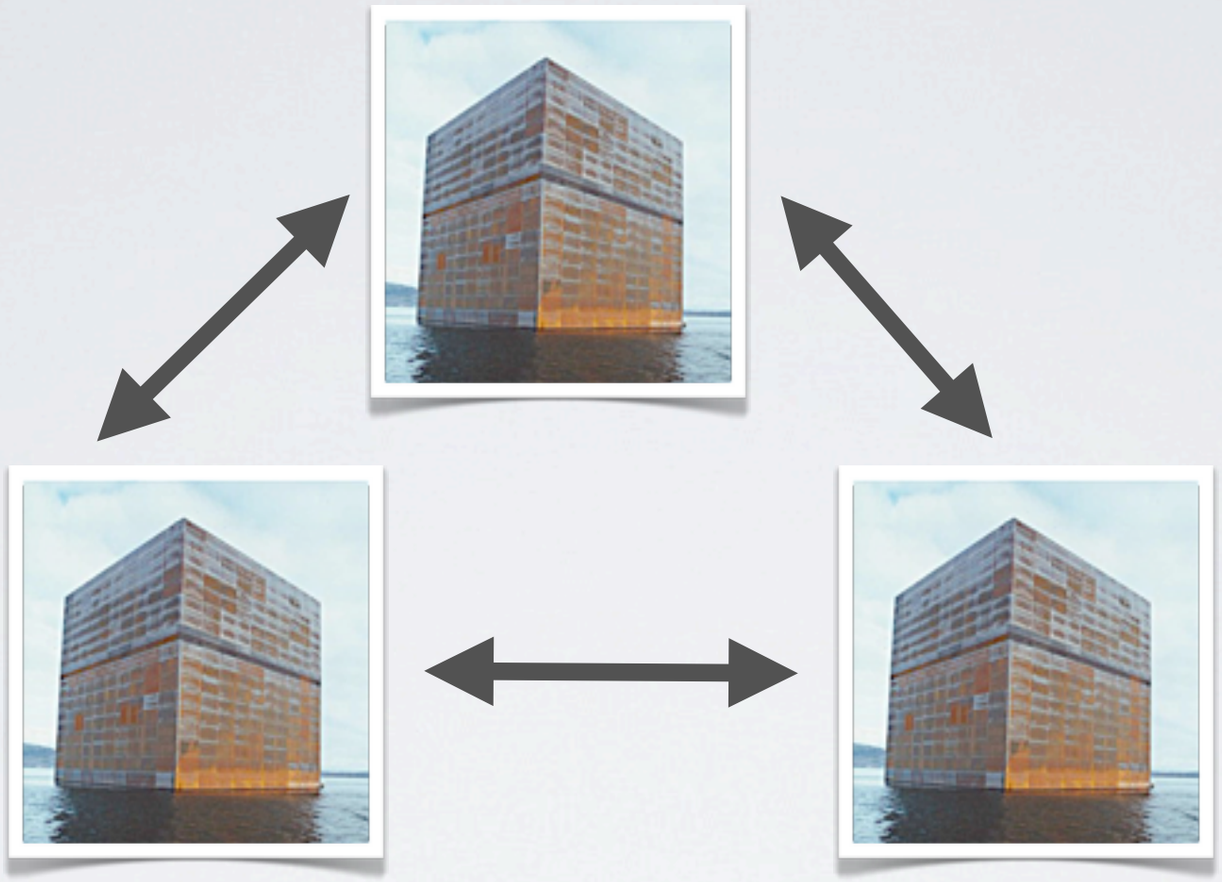
t1:read(x) t2:read(x) t1:write(x) t2:write(x) x = 1

MONOLITH

One Node

Single Core

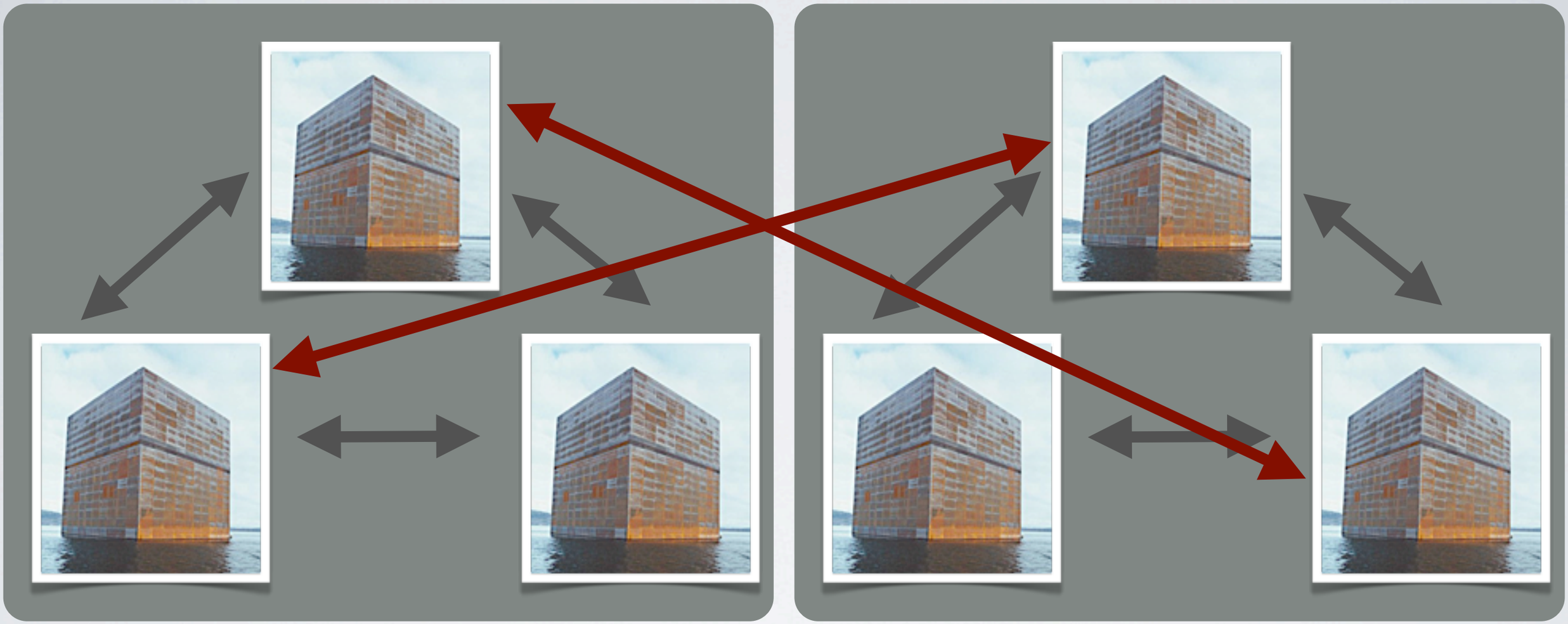
VERTEILUNG



One Node → Netzwerk

Single Core → Multicore

VERTEILUNG



One Node → Netzwerk → Cloud

Single Core → Multicore → Manycores

ZWEI WELTEN

objektorientiert

- Shared State
- Seperation
- Manipulation
- Freigabe

funktional

- Immutable values
- Message Passing
- Queue

AKTORENMODELL

Architekturmuster

lose Kopplung

kein shared state

Verhalten

autonom

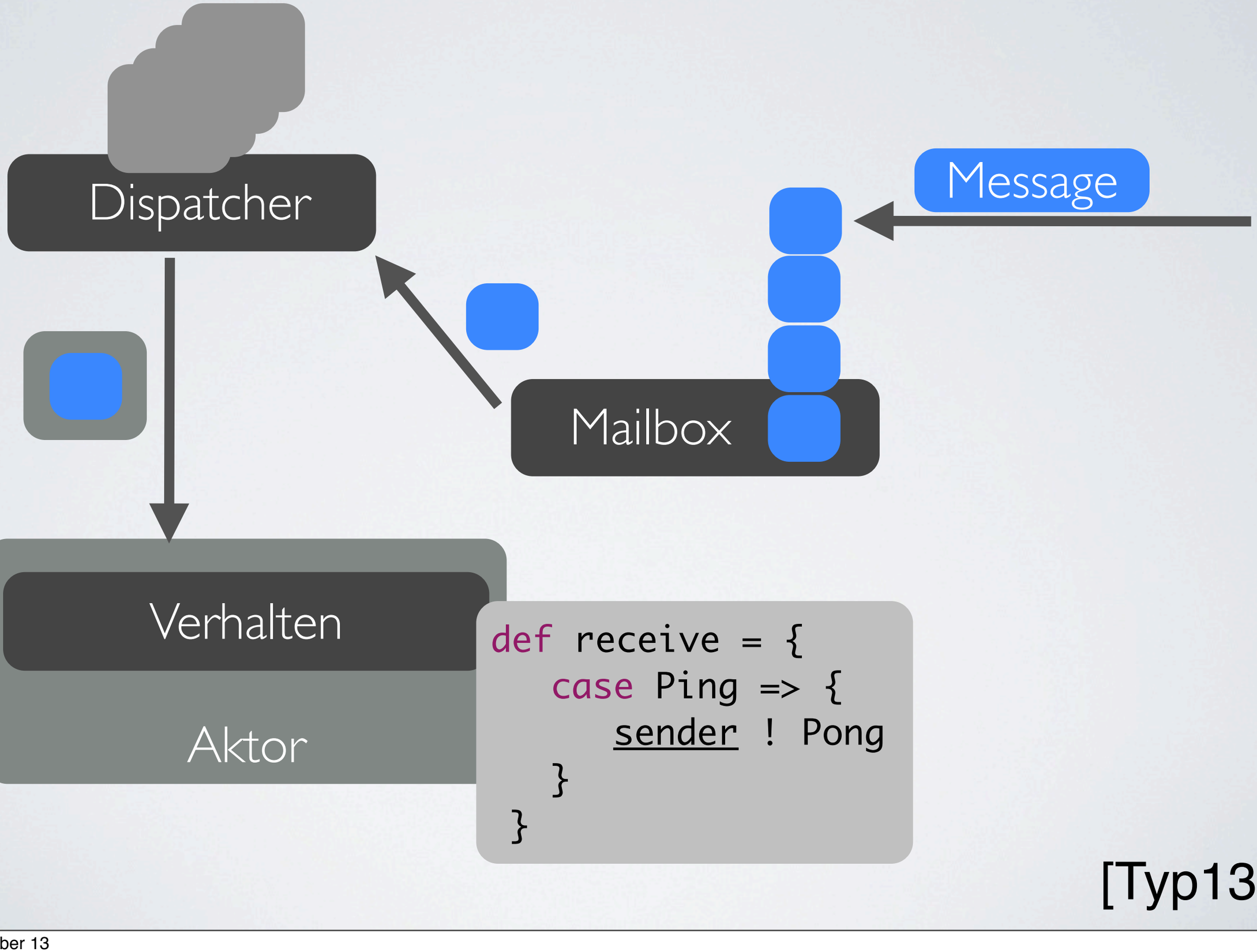
Nachricht als Vertrag

Distributed Artificial Intelligence

Planen

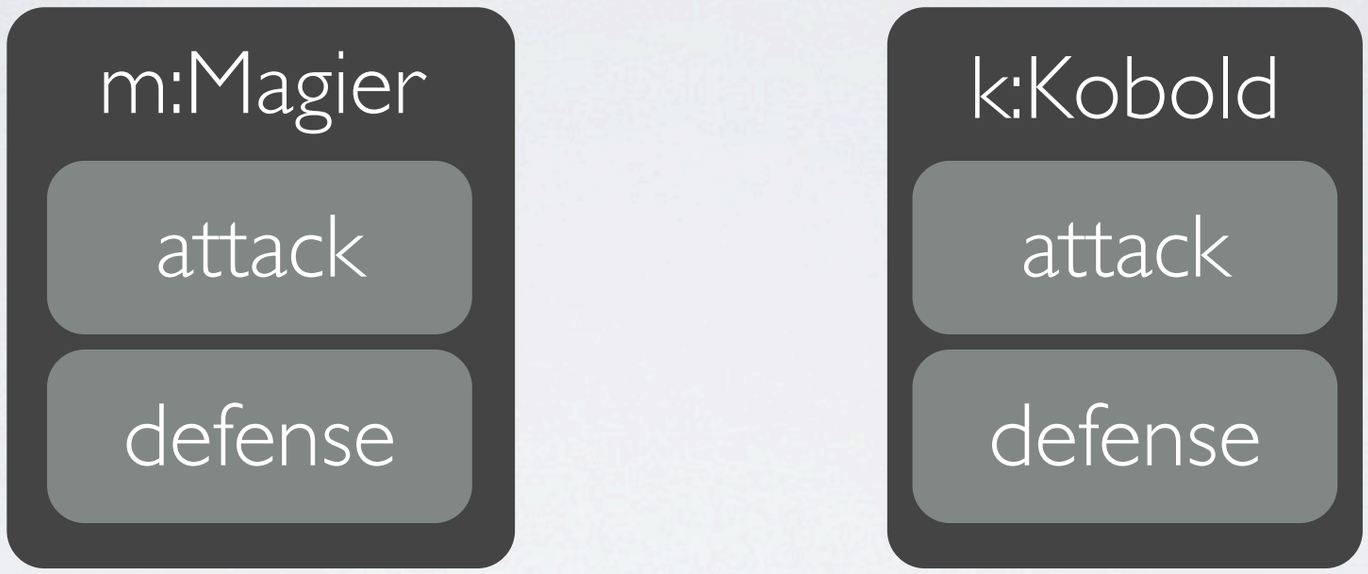
[HBS73]

AKTORENMODEL



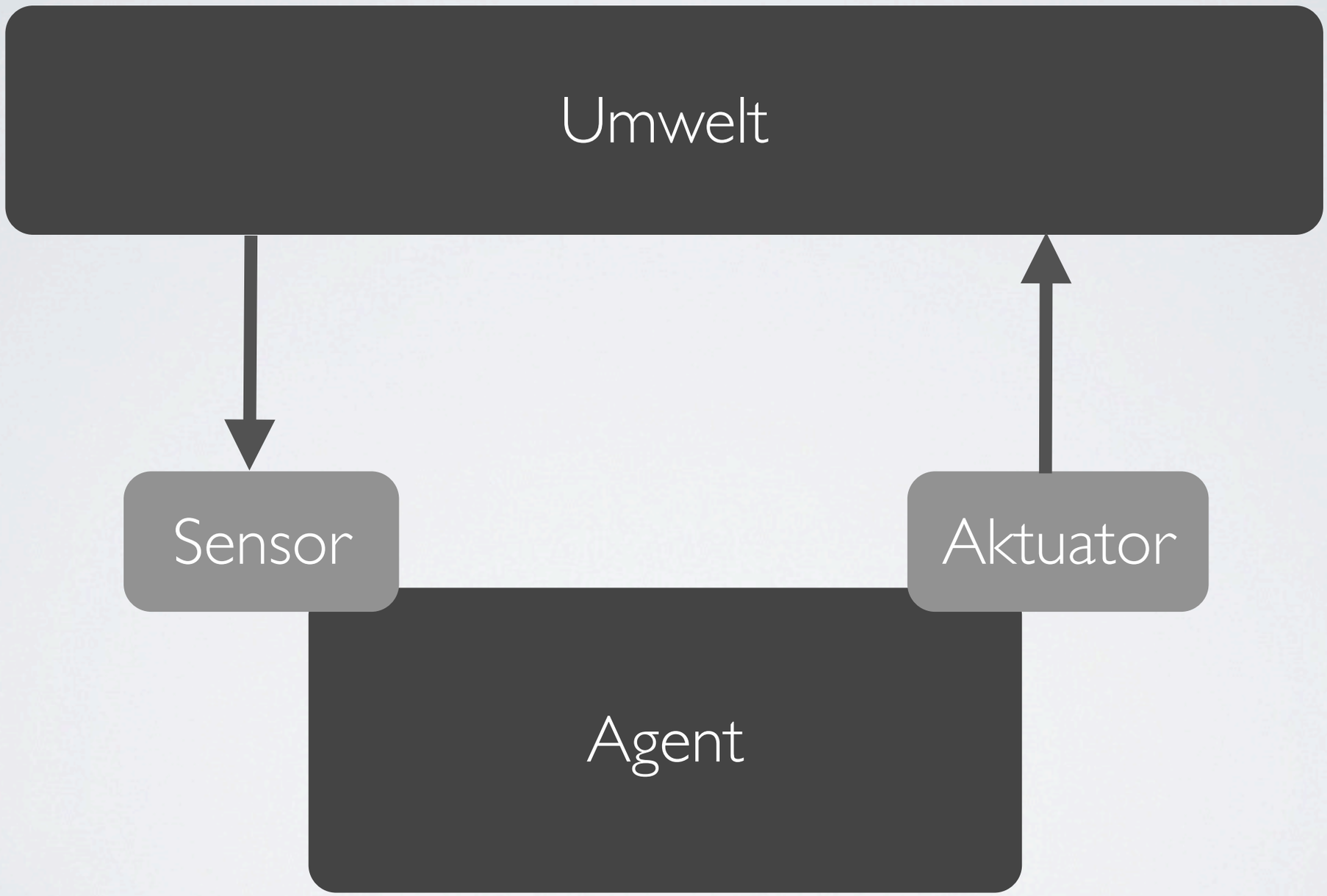
[Typ13]

AKTOR BEISPIEL



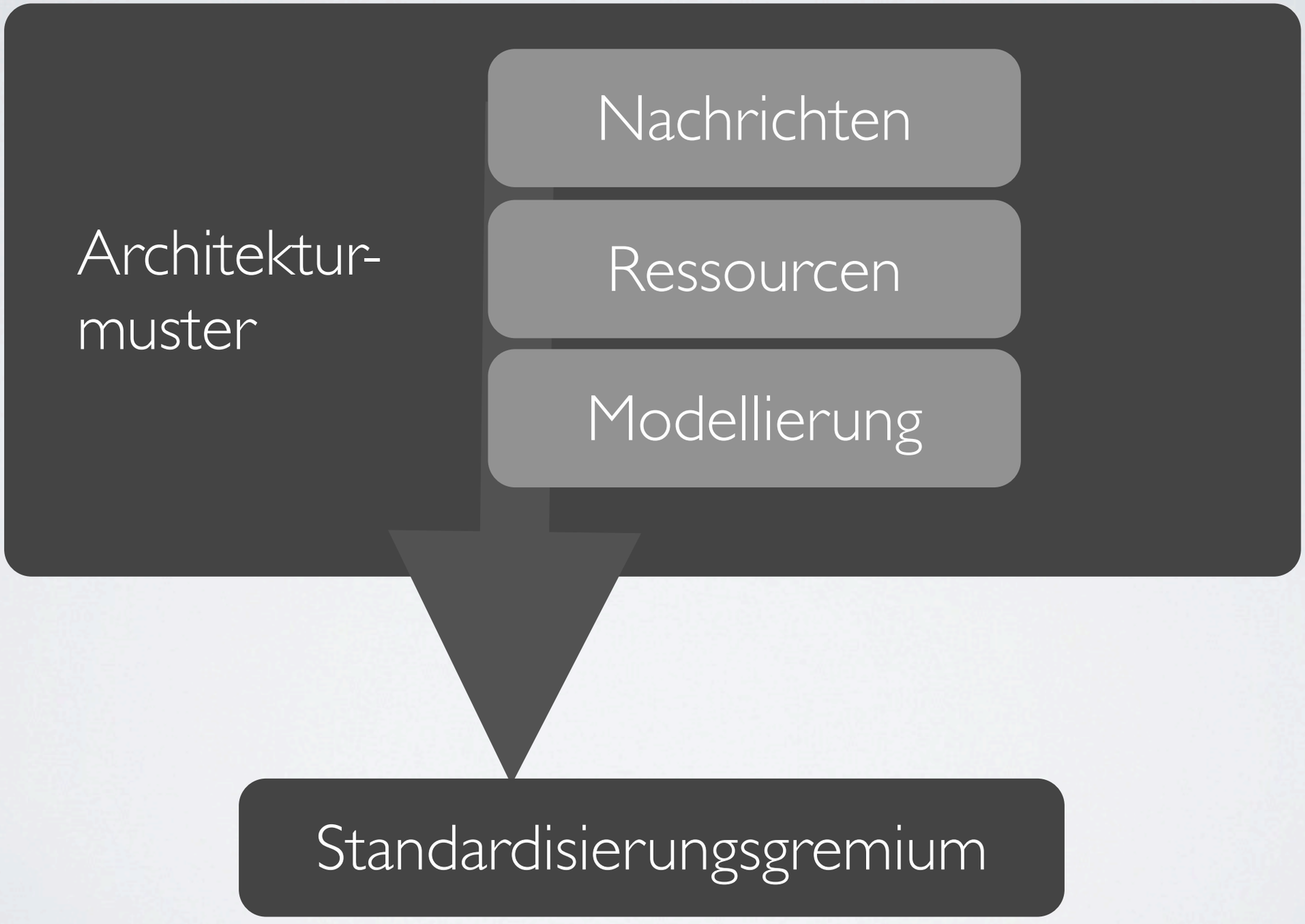
Fight(attack)

AGENT



FIPA

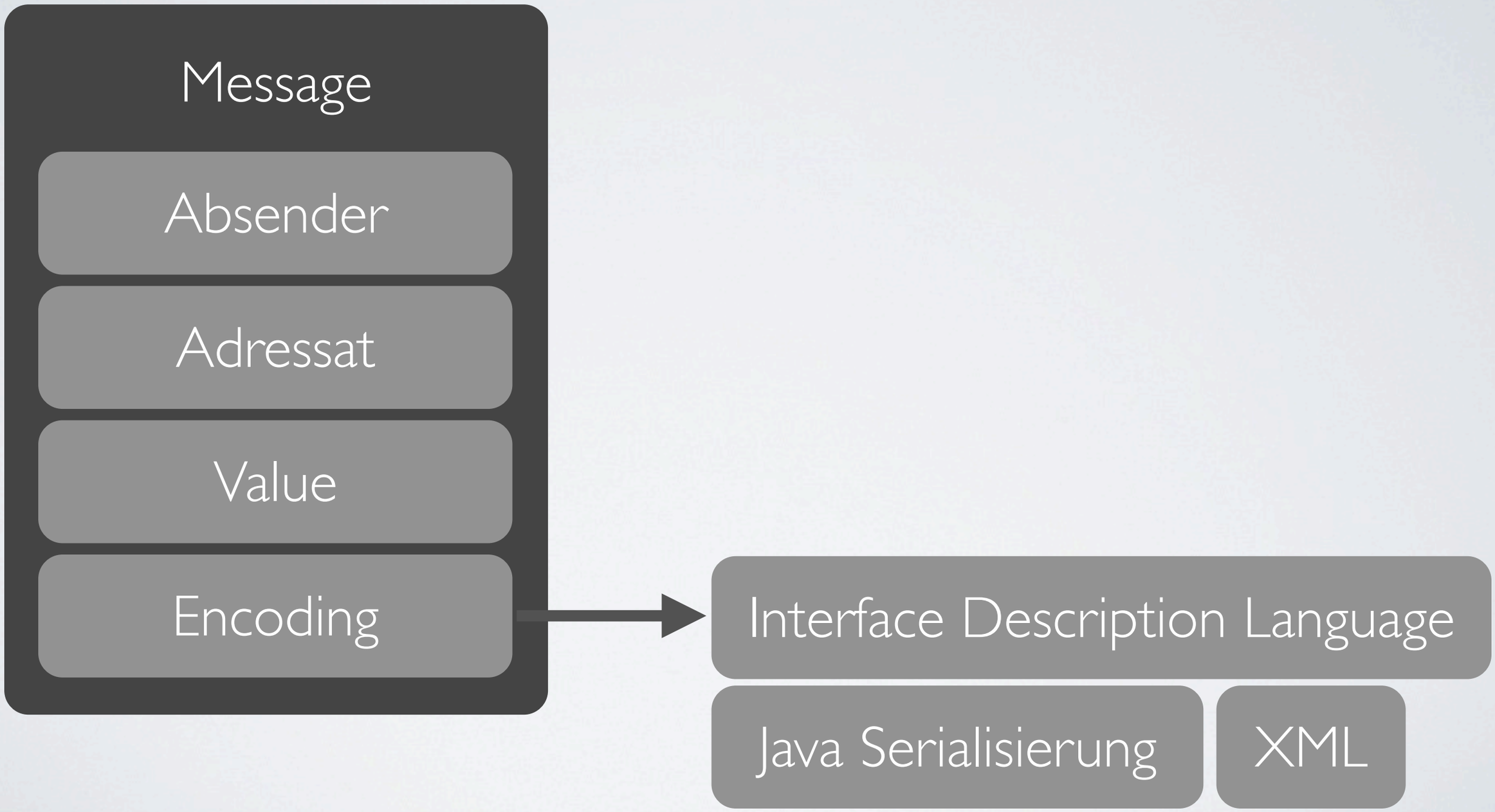
Foundation for Intelligent Physical Agents



[FIP02c]

ACL

Agent Communication Language



[FIP02a]

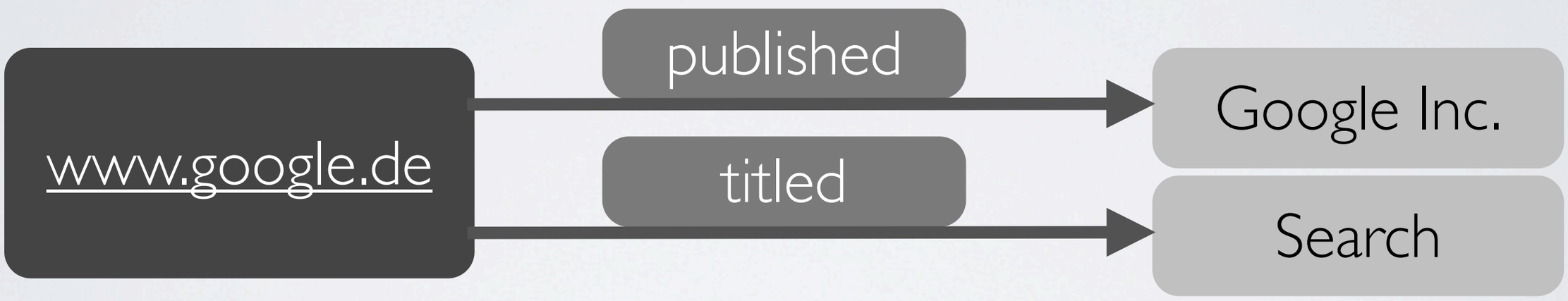
RDF

Resource Description Framework

XML
Turtle

Resource | Property | Property value

Statement



SPARQL

```
SELECT ?title
 WHERE{
 ?res published ?pub .
 ?res titled ?title }
```

title
Search

[CLS01]

Motivation Konzepte

FIPA DESIGN PROCESS

System Requirements

Analyse und Identifizierung

Agent Society

Rollenbeschreibung

Ontologie

Protokolle

Agent Implementation

Deployment

[FIP02b]

PETRINETZE

$N(P,T,W,M_0)$

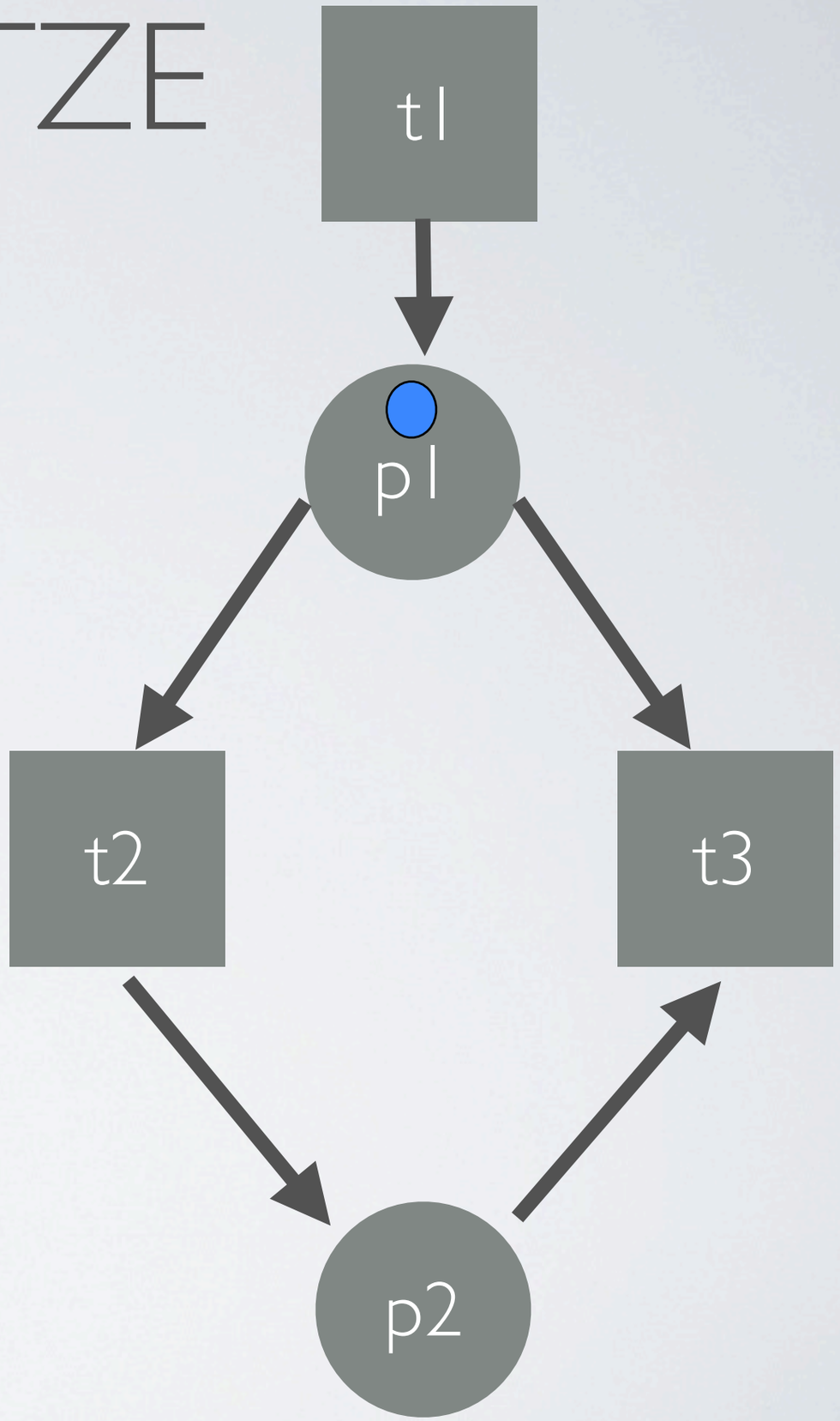
$P = \{p1\}$

$T = \{t1, t2, t3\}1$

$W = \{(t1, p1), (p1, t2)$
$(p1, t3), (t2, p2),$
$(p2, t3)\}$

$M_0(x) = \{1: x = p1;$
$0: sonst\}$

formal beschrieben



HISTORIE

Petri Netze

Adam Petri 1962

Actor Model

Carl Hewitt 1973

FIPA

FIPA 1996

Erlang

Joe Armstrong 1986

RDF

Frank Manola, Eric Miller 1997

KONFERENZEN

6th_ICAART 2014

Pieter Spronck
Tilburg University

REA Pattern

23th_ICAIJ 2013

Andreas Herzog
University of Toulouse

MAS Logics

20th_ECAIJ 2012

Jacques Ferber
Université de Montpellier

Symbolic Layer for Agents

GAMEDESIGN

Regeln

Kobold kann Magier im Wald besiegen

Akteure

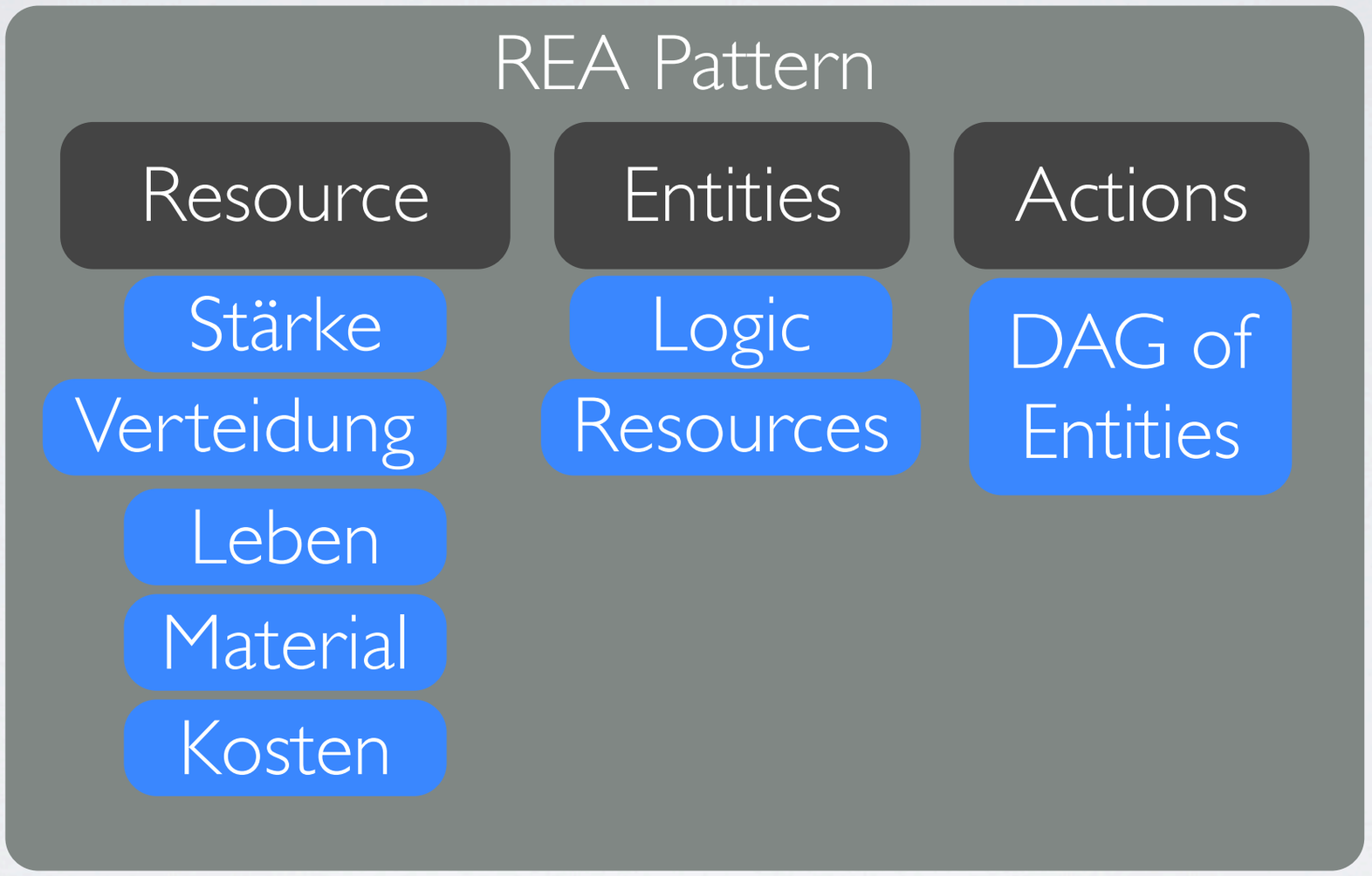
Magier, Kobold

Umwelt

Ein Magier und ein Kobold stehen im Wald

REA PATTERN

Resource Entity Action: Generalized Design Pattern for Real Time Strategy Games



[ADGO+13]

ACTIONS

rs = Waffe(20),Leben(500)

Resources

Kobold

Source Entity

Regel für den Angriff:
Leben‘ = Leben-Waffenstärke

rt‘ = rt + (0,-20) = (15,980)

rt = Waffe(15),Leben(1000)

Resources

Magier

Target Entity

ZIEL

Modellierung von nebenläufigen, skalierbaren Systemen

+

Ausführung auf Aktoren

||
v

Vereinigung theoretischer und praktischer Konzepte

Anwendung: Gamedesign

QUELLEN

[ADGO+13] Abbadi, Mohamed ; Di Giacomo, Francesco ; Orsini, Renzo ; Plaat, Aske ; Spronck, Pieter ; Maggiore, Giuseppe: Resource Entity Action: A Generalized Design Pattern for RTS games. (2013)

[CLS01] Candan, K. S. ; Liu, Huan ; Suvarna, Reshma: Resource Description Framework: Metadata and Its Applications. In: SIGKDD Explor. Newsl. 3 (2001), Juli, Nr. 1, 6–19. http://dx.doi.org/10.1145/507533.507536. – DOI 10.1145/507533.507536. – ISSN 1931–0145

[FIP02a] FIPA: ACL Message Structure Specification, 2002. http://www.fipa.org/specs/fipa00061/SC00061G.pdf. – Online; Abruf am 17.11.2013

[FIP02b] FIPA: Design Process Documentation Template, 2002. http://www.fipa.org/specs/fipa00097/SC00097B.pdf. – Online; Abruf am 17.11.2013

[FIP02c] FIPA: FIPA Abstract Architecture Specification, 2002. http://www.fipa.org/specs/fipa00001/SC00001L.pdf. – Online; Abruf am 17.11.2013

[HBS73] Hewitt, Carl ; Bishop, Peter ; Steiger, Richard: A universal modular ACTOR formalism for artificial intelligence. In: Proceedings of the 3rd international joint conference on Artificial intelligence. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1973 (IJCAI'73), 235–245

[Nie88] Nierstrasz, O. M.: Two Models of Concurrent Objects. In: Proceedings of the 1988 ACM SIGPLAN Workshop on Object-based Concurrent Pro- gramming. New York, NY, USA : ACM, 1988 (OOPSLA/ECOOP '88). – ISBN 0–89791–304–3, 174–176

[Typ13] TypeSafe: Akka Documentation 2.1.1. Online Abruf (04.03.2013) doc.akka.io/docs/akka/2.1.1/Akka.pdf, 2013

FIPA DESIGN PROCESS

Process for Agent Societies Specification and Implementation

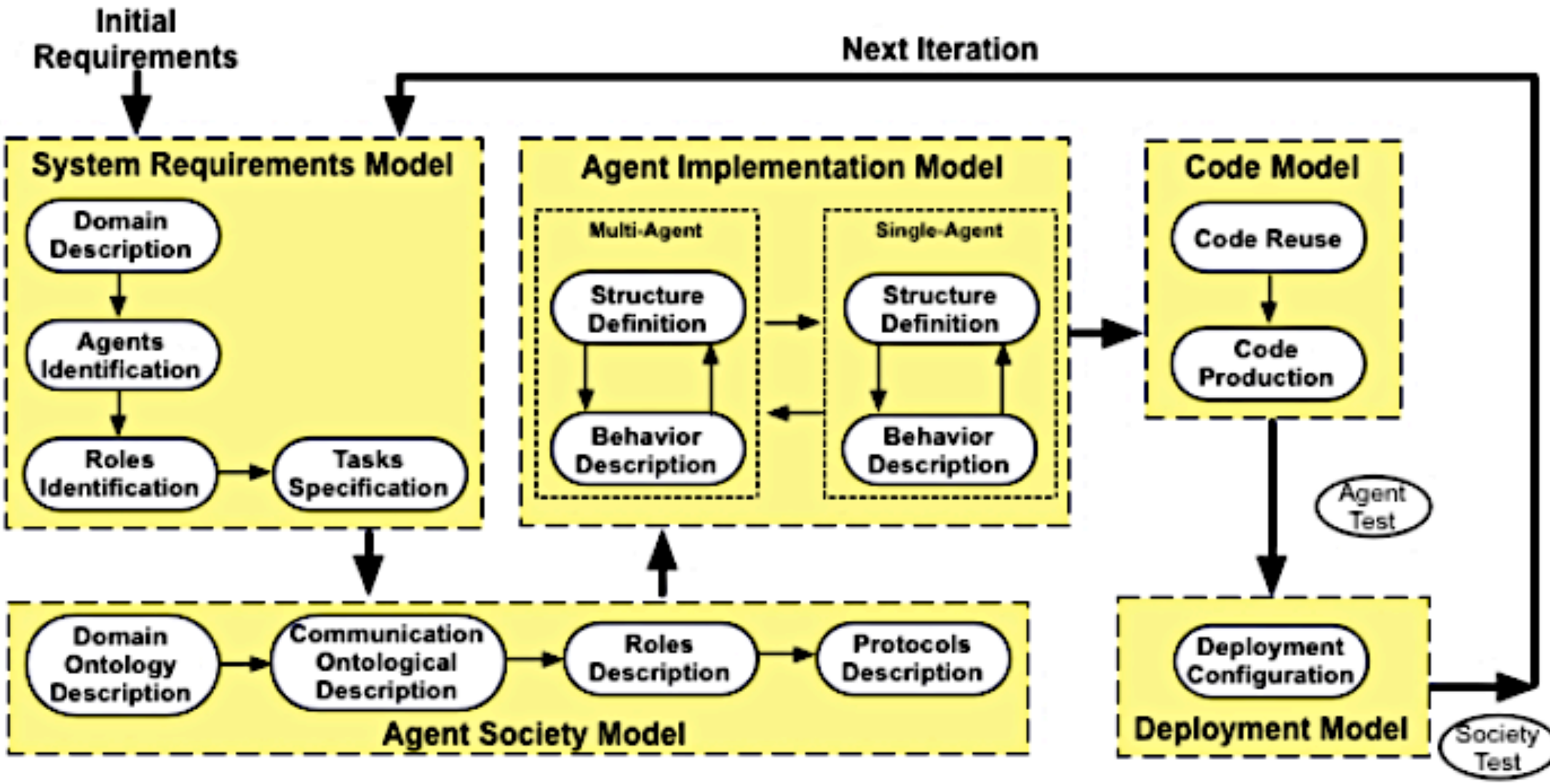
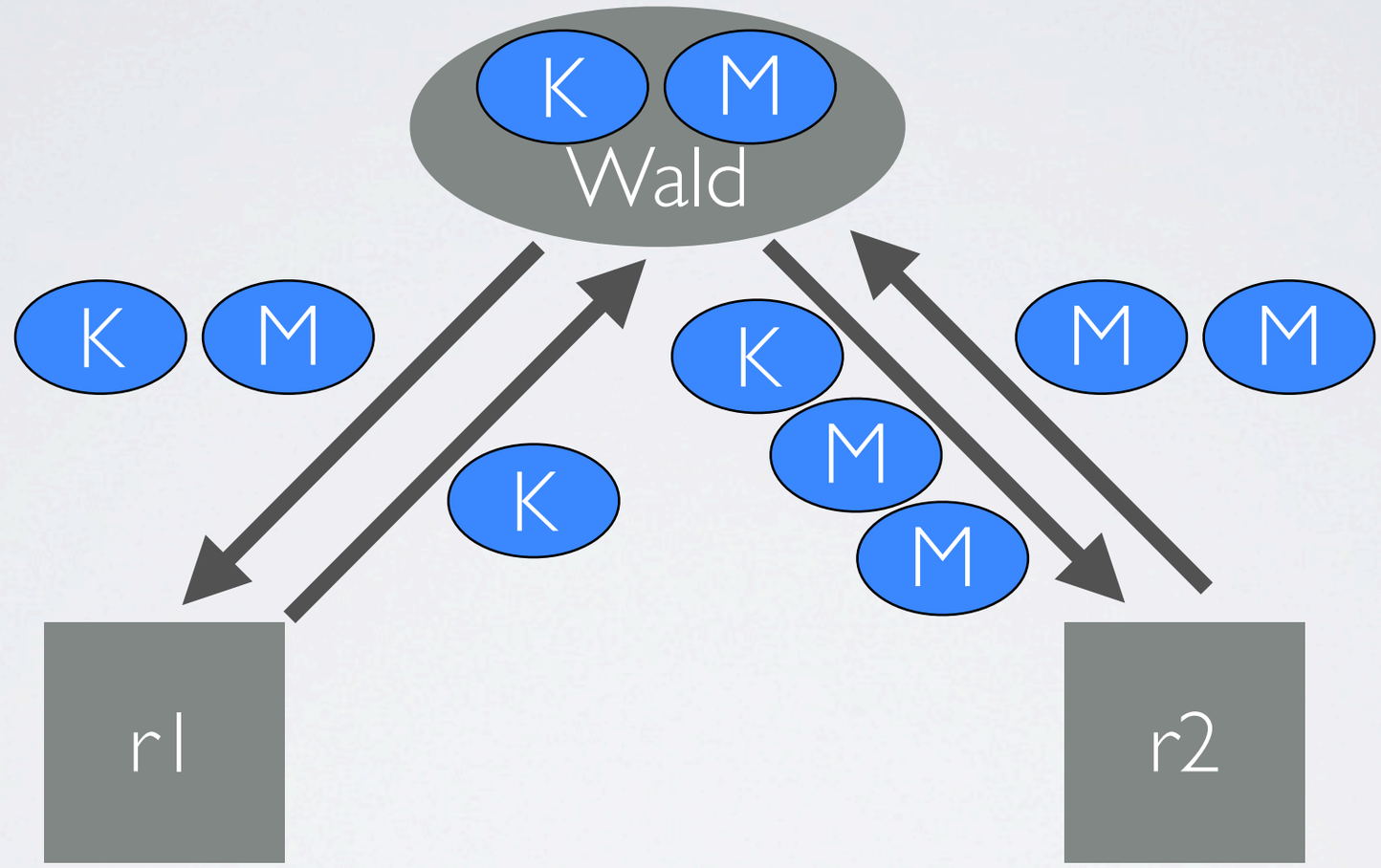


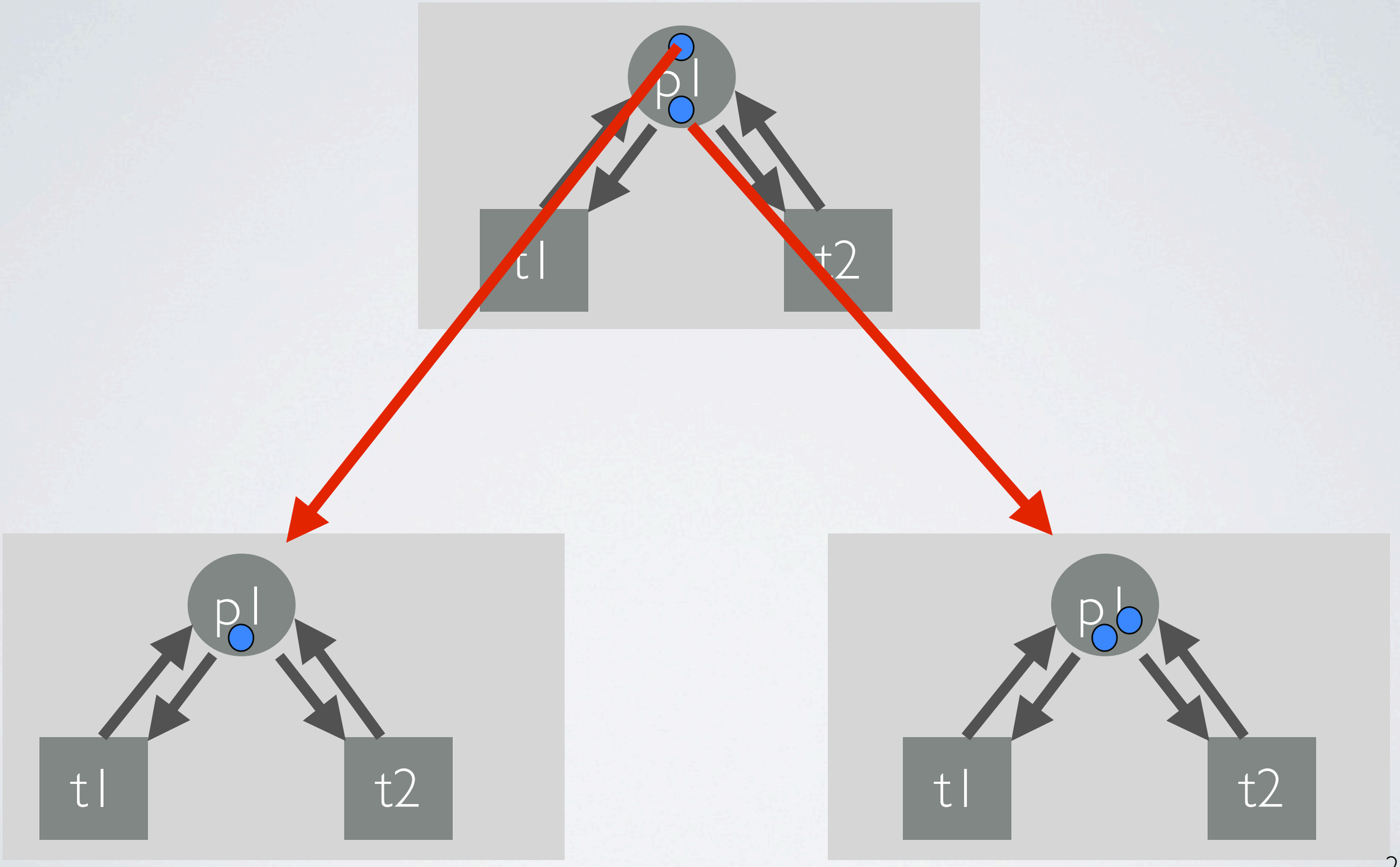
Figure 3. The PASSI design process

[FIP02a]

PROZESS



PARTITIONIERUNG



PARTITIONIERUNG

100 Magier

MagierActor für 50 Magier

Node

MagierActor für 50 Magier

Node

2: Nodes

THE SYMBOLS

A symbolic layer for autonomous component-based software agents

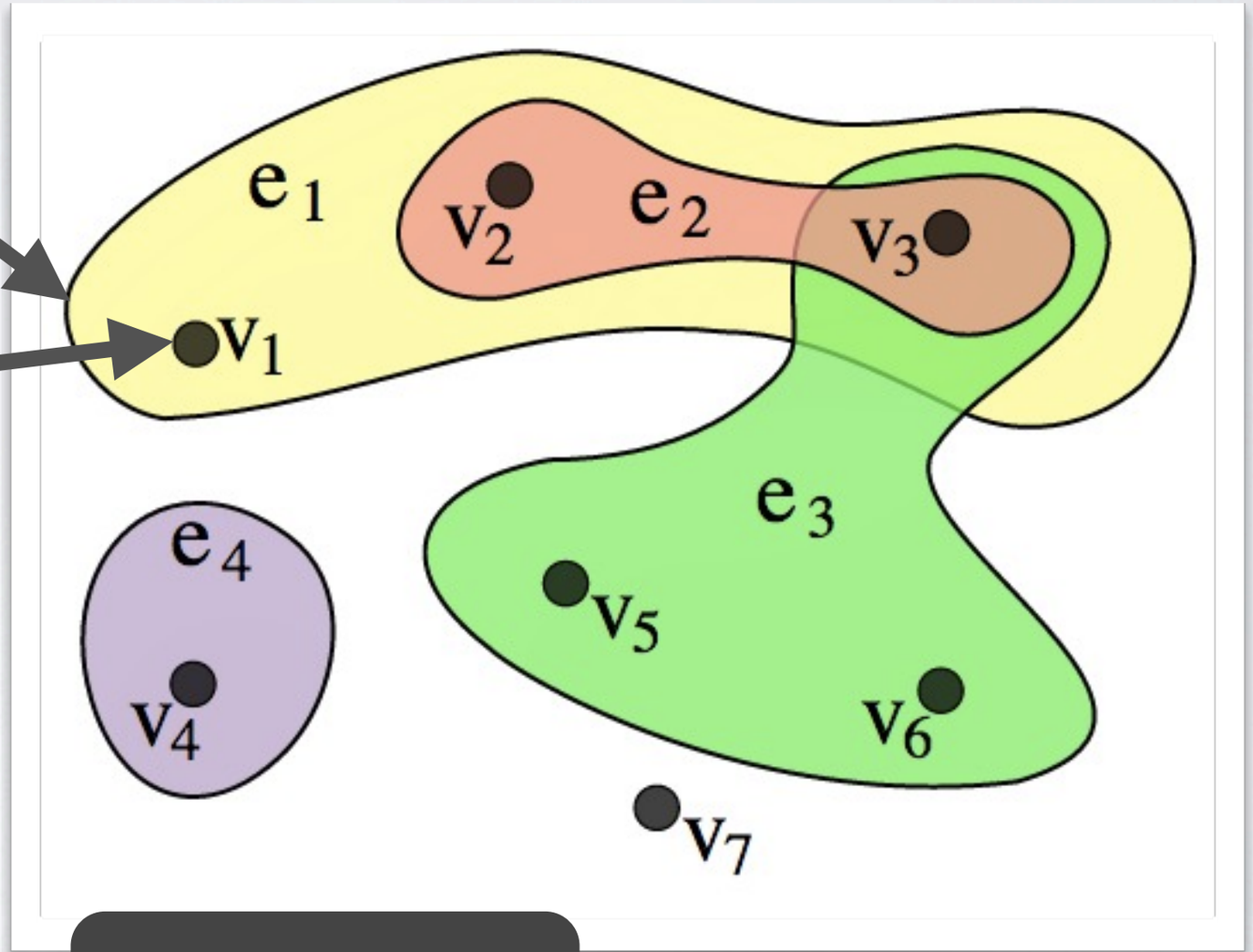
L = {Links}

S = {Symbols}

I : S => {(Sx,Sy)}

i : S => I

m: S‘ => S



< S, I, i, m >