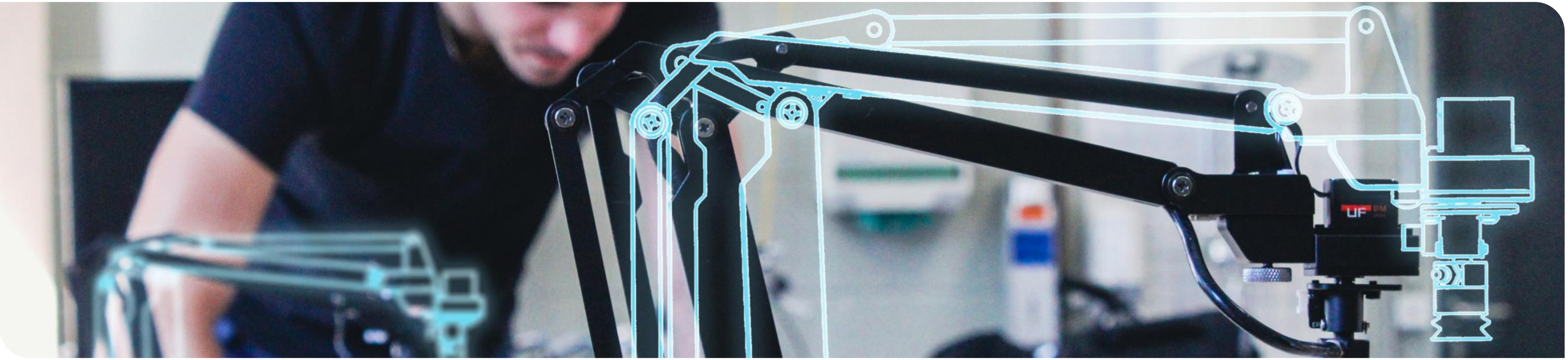
Karlsruher Institut für Technologie

Institut für Technik der Informationsverarbeitung

Modelling of Material Handling Systems

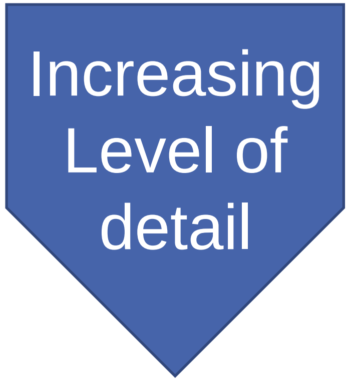
Seamless Engineering



Modelling of Material Handling Systems

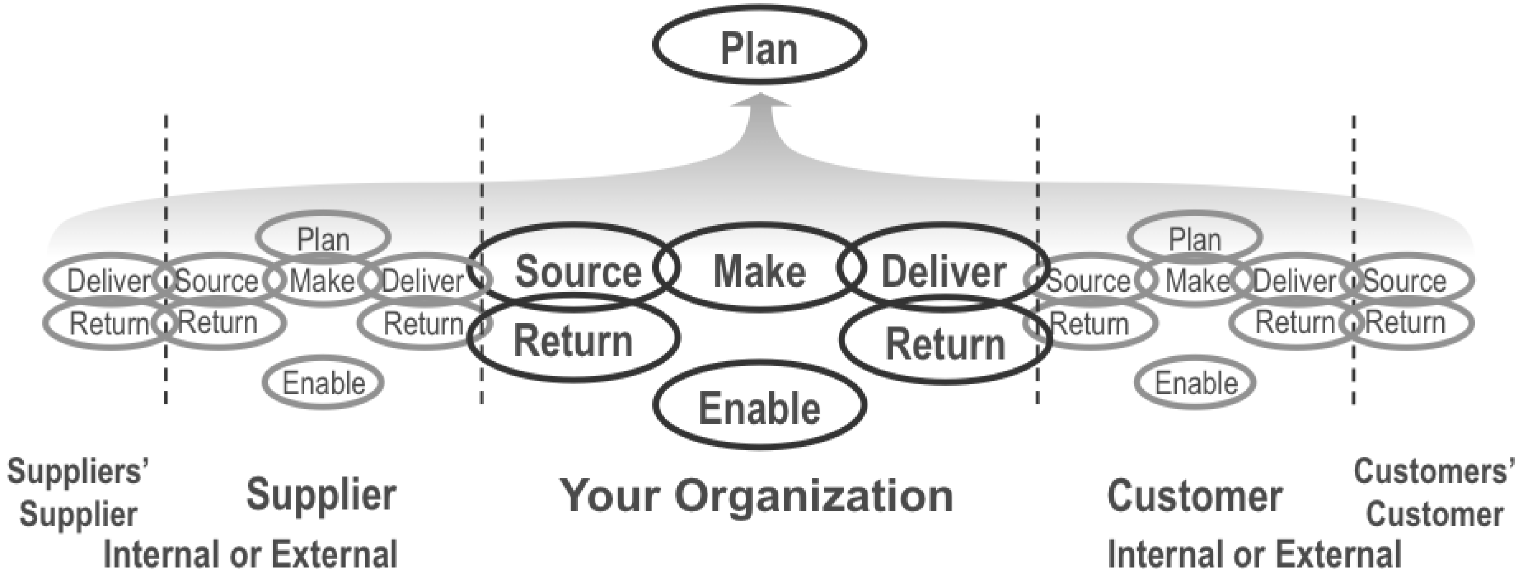
- There are no widely accepted formal methods yet.
- Usually, the requirements are documented verbally – producing pretty thick books
  - Flow charts, layouts and Event-driven Process charts are used additionally and embedded

- We propose a mixture of Supply Chain Reference (**SCOR**)
- Distribution Centre Reference Model (**DCRM**)
- Modular Material Handling (**MMH**)



Increasing Level of detail

SCOR – Supply Chain Operations Reference Model: Level 1



SCOR Modell

Building Block Approach

Processes

Metrics

Best Practice

Technology

Quelle: Supply Chain Councill,
www.apics.org

SCOR – process types: Level 1

PLAN

- Strategic Planing and Surveillance of the Supply Chain
- Controlling of Processes according to demand

SOURCE

- Processes and material handling for sourcing of raw material

MAKE

- Processes and material handling used for producing products

DELIVER

- Processes and material handling used for delivery of products

RETURN

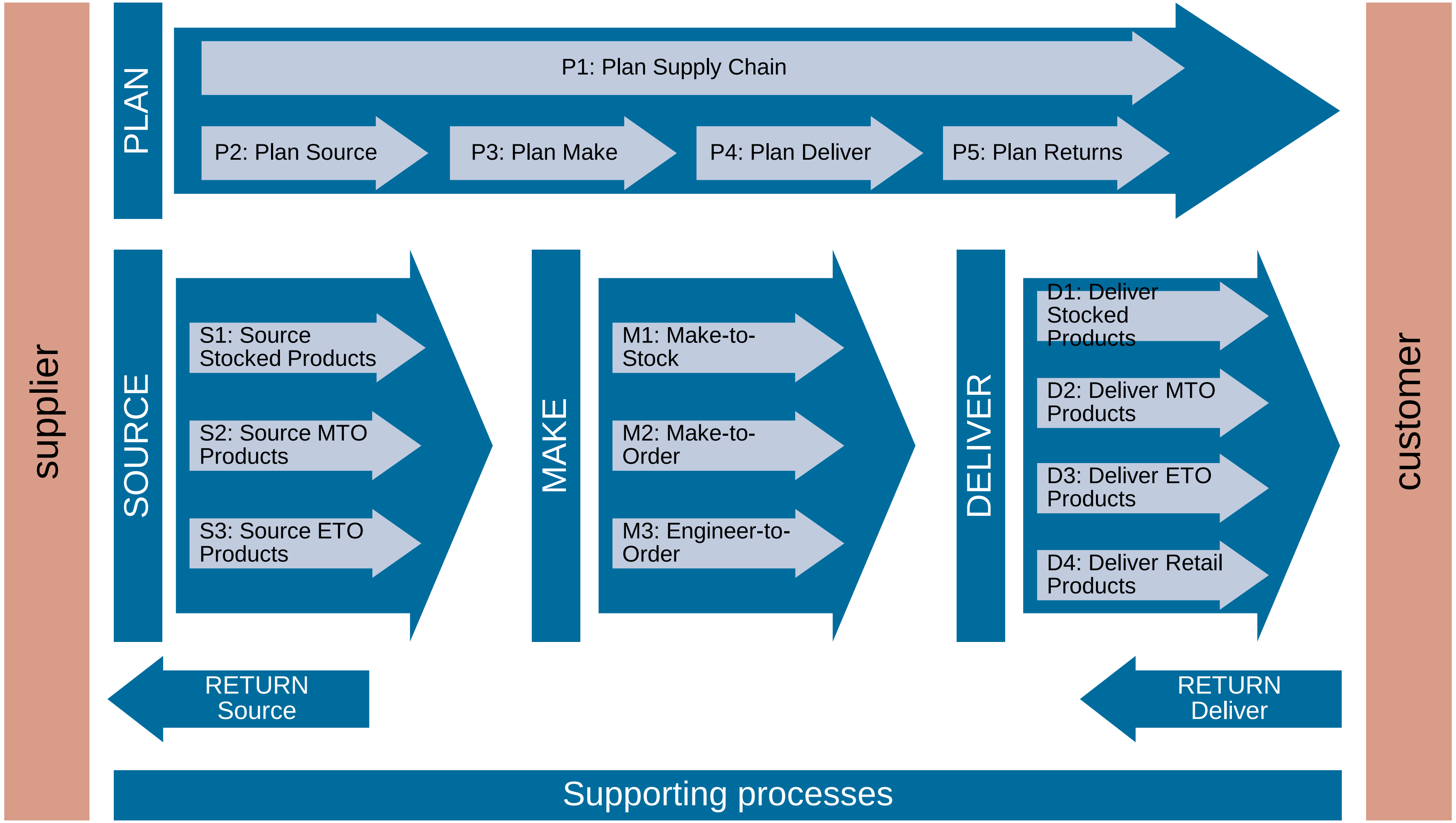
- Handling of returns (defect parts, wrong deliveries, recycling)

ENABLE

- Enabling and support of the above 5 main processes

Source: Supply Chain Councill,
www.apics.org

SCOR – Model 6.0 – Processes: Level 2 (process categories)



Quelle: Supply Chain Councill,
www.apics.org

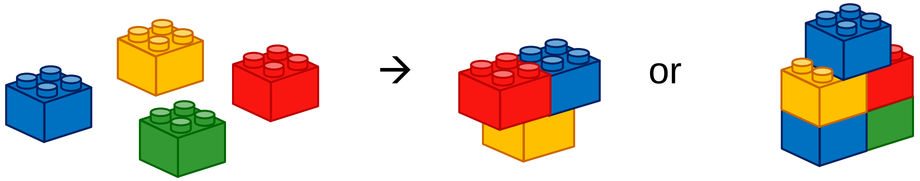
Distribution Centre Reference Model

The DCRM yields a methodology for a task-oriented benchmarking of distribution centers

- Comparability due to consideration of the same task

  e.g. storage and picking of pallets.

- Modular construction system consisting of 26 well-defined tasks enables to structure each individual distribution centre → Identification of the accomplished task



- Benchmarking of each identified task with all equivalent tasks → Different tasks may have different benchmarking partners

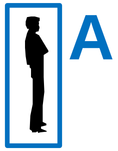
  Sportsman A:

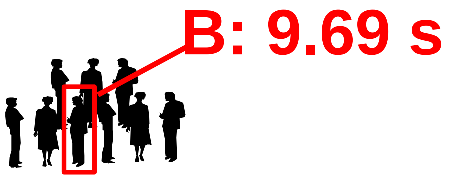
  Sprint performance over 100m: 10 s

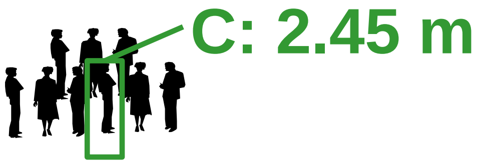
  High jump performance: 2.00 m

  → Evaluation of the sprint performance: Comparison with all sprinters

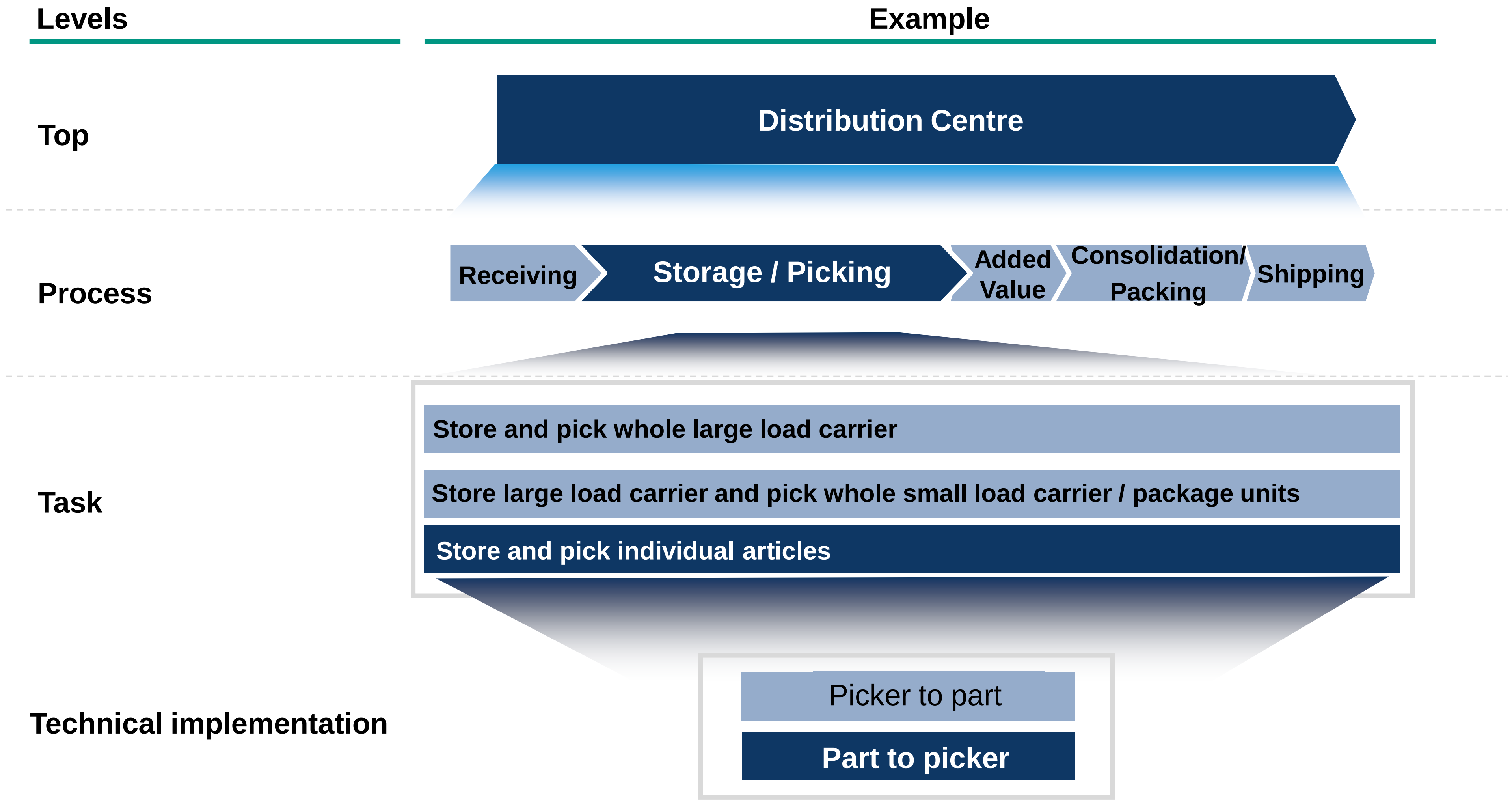
  → Evaluation of the high jump performance: Comparison with all high jumpers







The model is hierarchically structured into four levels of detail



Levels

Example

Top

Distribution Centre

Process

Receiving

Storage / Picking

Added Value

Consolidation/ Packing

Shipping

Task

Store and pick whole large load carrier

Store large load carrier and pick whole small load carrier / package units

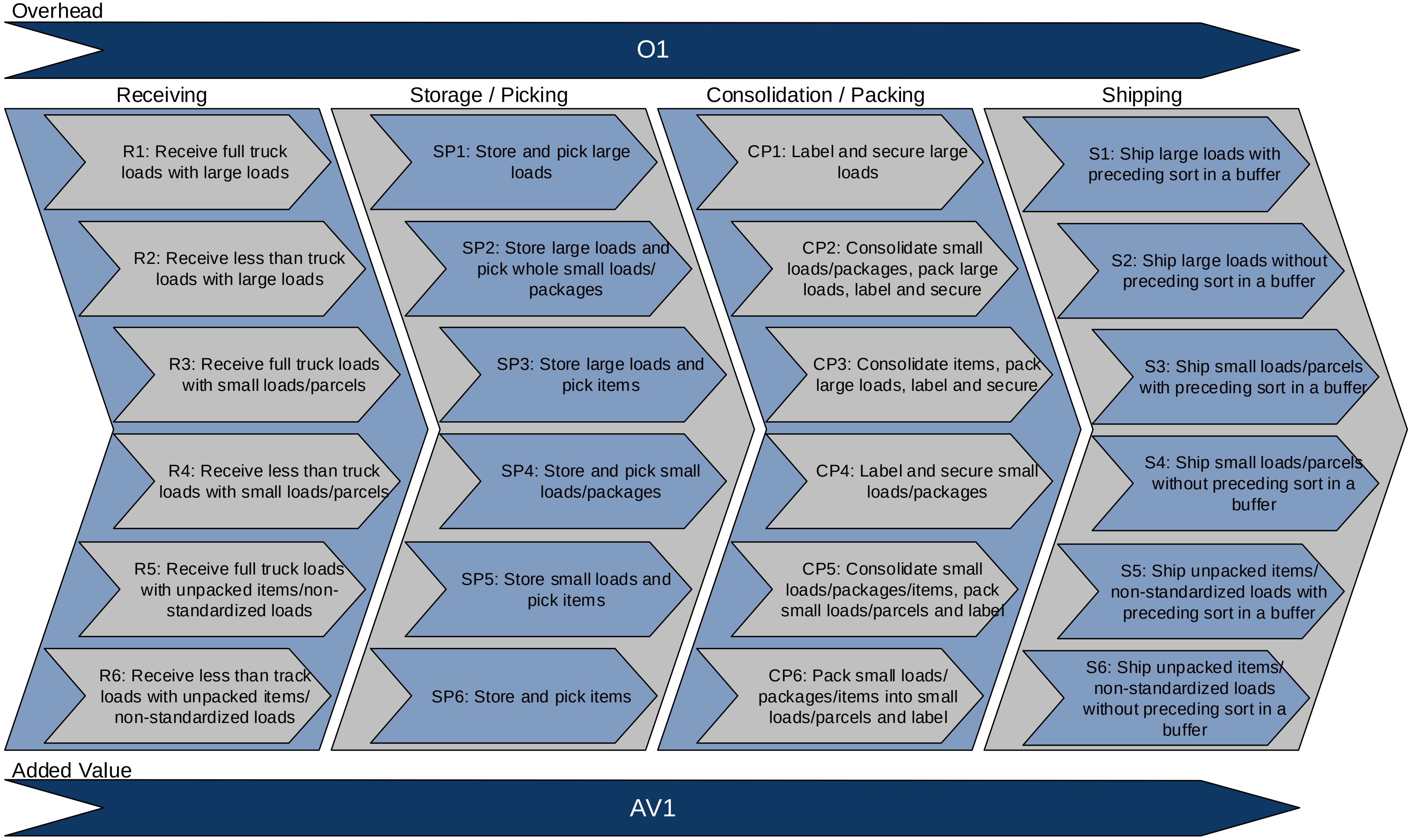
Store and pick individual articles

Technical implementation

Picker to part

Part to picker

We distinguish 24 different warehouse tasks plus overhead and added value



Overhead

O1

Receiving

- R1: Receive full truck loads with large loads
- R2: Receive less than truck loads with large loads
- R3: Receive full truck loads with small loads/parcels
- R4: Receive less than truck loads with small loads/parcels
- R5: Receive full truck loads with unpacked items/non-standardized loads
- R6: Receive less than track loads with unpacked items/ non-standardized loads

Storage / Picking

- SP1: Store and pick large loads
- SP2: Store large loads and pick whole small loads/ packages
- SP3: Store large loads and pick items
- SP4: Store and pick small loads/packages
- SP5: Store small loads and pick items
- SP6: Store and pick items

Consolidation / Packing

- CP1: Label and secure large loads
- CP2: Consolidate small loads/packages, pack large loads, label and secure
- CP3: Consolidate items, pack large loads, label and secure
- CP4: Label and secure small loads/packages
- CP5: Consolidate small loads/packages/items, pack small loads/parcels and label
- CP6: Pack small loads/ packages/items into small loads/parcels and label

Shipping

- S1: Ship large loads with preceding sort in a buffer
- S2: Ship large loads without preceding sort in a buffer
- S3: Ship small loads/parcels with preceding sort in a buffer
- S4: Ship small loads/parcels without preceding sort in a buffer
- S5: Ship unpacked items/ non-standardized loads with preceding sort in a buffer
- S6: Ship unpacked items/ non-standardized loads without preceding sort in a buffer

Added Value

AV1

The Logic behind the DCRM

CLASSIFACTION OF TASKS

Classification of Tasks by the Example of „Storage and Picking“

Goal:	**Structuring** of processes, and **performance evaluation of tasks** independent of their **technical implementation**
Definition:	A task is a specificity of a process which transfers the system from a defined initial state into a defined final state

General Task	Storage and picking of loading units of a certain size

Parameters
Orders and order positions per time unit
Range of products (Number of different items)
Size of Warehouse
Specific requirements for storing and picking
…

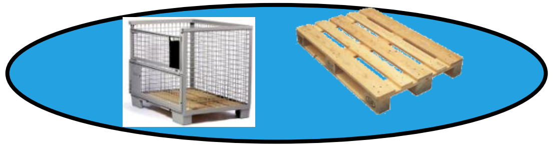
Loading units are classified regarding their need for handling equipment

General Task

Storing and Picking of
Loading Units of a Certain Size

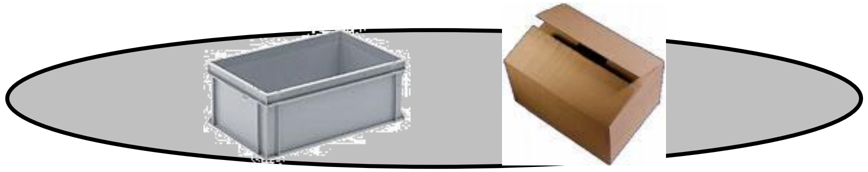
Large Loads

= Need handling equipment like a forklift truck due to volume or weight

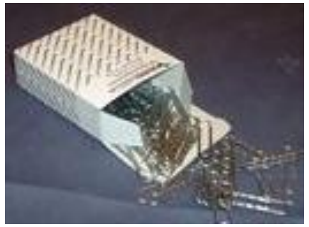


Small loads/Packages/Parcels

= Can be handled without equipment



Item
= smallest selling unit



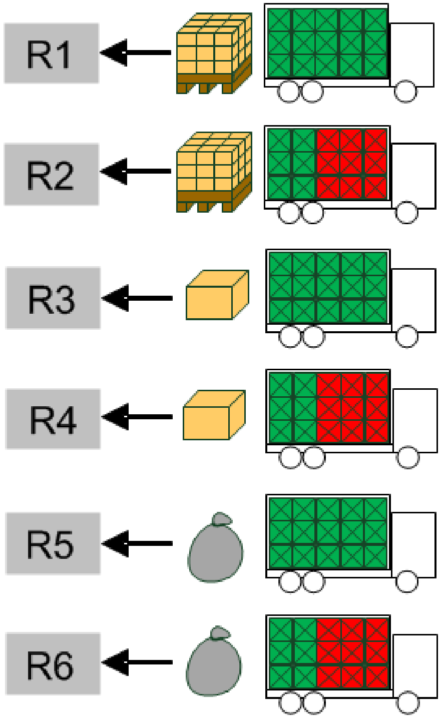
Classification of Tasks by the Example of „Storage and Picking“

General Task	Storing and Picking of Loading Units of a Certain Size

Kind of loading unit that is being **stored**	Kind of loading unit that is being **picked**		
	Large Loads	Small loads/Packages	Items
Large Loads	SP1	SP2	SP3
Small loads/Packages	Not possible	SP4	SP5
Items	Not possible	Not possible	SP6

Tasks of the process „Storage and Picking“	
SP1:	Store and pick large loads
SP2:	Store large loads and pick small loads/packages
SP3:	Store large loads and pick items
SP4:	Store and pick small loads/packages
SP5:	Store small loads and pick items
SP6:	Store and pick items

In receiving, tasks are differentiated by the kind of loading unit and the exclusivity of the transport means



Receive full truck loads with large loads

Receive less than truck loads with large loads

Receive full truck loads with small loads/parcels

Receive less than truck loads with small loads/parcels

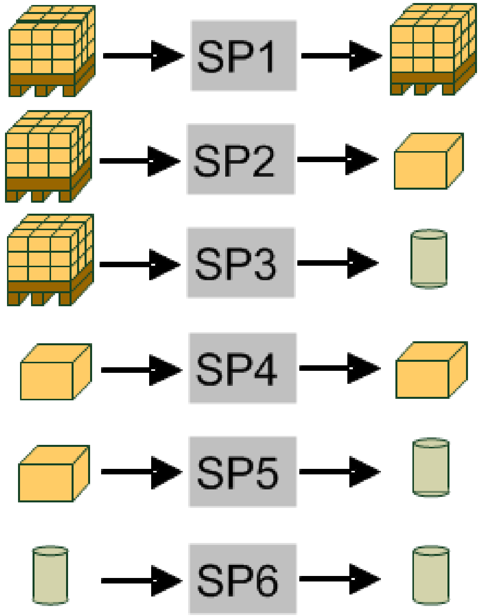
Receive full truck loads with unpacked items/non-standardized loads

Receive less than truck loads with unpacked items/non-standardized loads

<table>
<tr><td>Kind of loading unit</td><td>Large load</td><td colspan="2">Small load</td><td>Unstandardized</td></tr>
<tr><td>Exclusivity of transport means</td><td colspan="2">Full truck load</td><td colspan="2">Less than truck load</td></tr>
</table>

In storage and picking, tasks are differentiated by the kind of loading unit that is being stored or picked

Receiving
Storage & Picking
Consolidation & Packing
Shipping



Store and pick large loads

Store large loads and pick small loads or packages

Store large loads and pick items

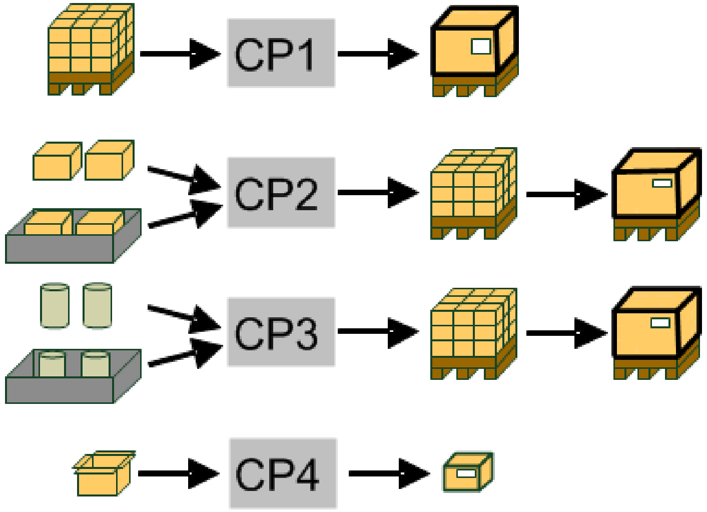
Store and pick small loads/packages

Store small loads and pick items

Store and pick items

Kind of loading unit **(storage)**	Large loads	Small loads	Items
Kind of loading unit **(picking)**	Large loads	Small loads	Items

In consolidation and packing, tasks are differentiated by the kind of loading unit and the necessity to sort or pack (1/2)



Label and secure large loads

Consolidate small loads/packages, pack large loads, label and secure

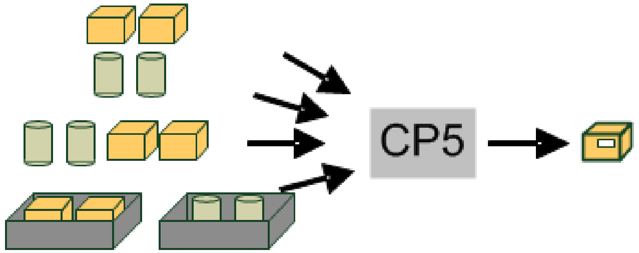
Consolidate items, pack large loads, label and secure

Label and secure small loads/packages/packages

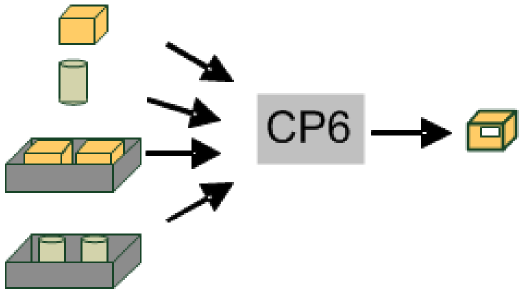
Task	Incoming	Outgoing	Securing?	Consolidation/Sorting?
CP1	Large Load	Large Load	Yes	No
CP2	Small load	Large Load	Yes	Yes
CP3	Item	Large Load	Yes	Yes
CP4	Small load	Small load	Yes	No

In consolidation and packing, tasks are differentiated by the kind of loading unit and the necessity to sort or pack (2/2)

Receiving → Storage & Picking → Consolidation & Packing → Shipping



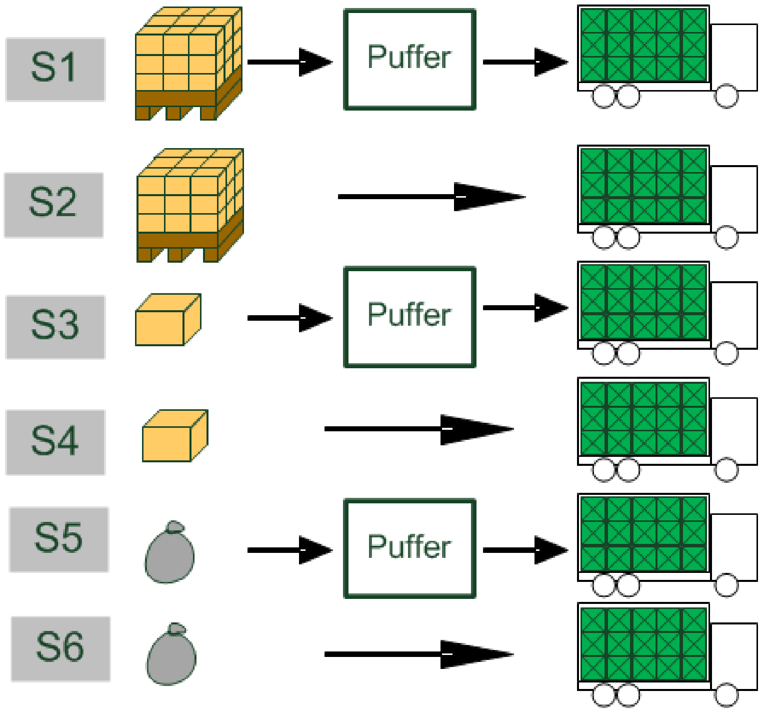
Consolidate small loads/packages/items, pack small loads/parcels and label



Pack small loads/packages/items into small loads/parcels and label

Task	Incoming	Outgoing	Securing?	Consolidation/Sorting?
CP5	Small loads or items	Small load	No	Yes
CP6	Small loads or items	Small load	No	No

In shipping, we differentiate the kind of loading unit to be shipped and whether sorting in a buffer is required



Ship large loads with preceding sort in a buffer

Ship large loads without preceding sort in a buffer

Ship small loads/parcels with preceding sort in a buffer

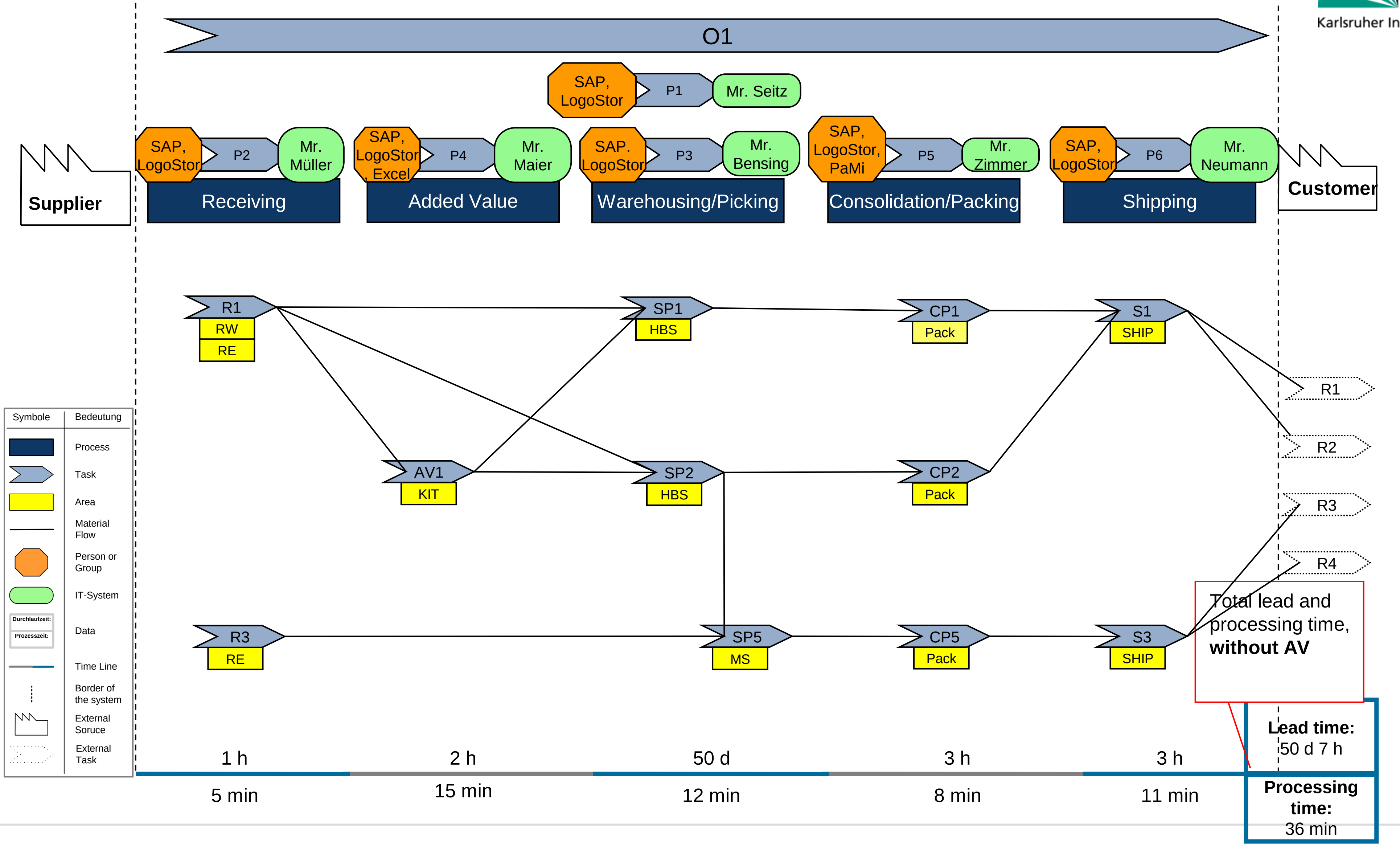
Ship small loads/parcels without preceding sort in a buffer

Ship unpacked items/non-standardized loads with preceding sort in a buffer

Ship unpacked items/non-standardized loads without preceding sort in a buffer

<table>
<tr><th>Kind of loading unit being shipped</th><th colspan="2">Large loads</th><th colspan="2">Small loads</th><th colspan="2">Non-standardized</th></tr>
<tr><td>Preceding sort in a buffer?</td><td colspan="3">Yes</td><td colspan="3">No</td></tr>
</table>

DCRM – Map: Kelm Inc.



Modular Material Handling

Idea

Agenda

1 Vision

2 Framework Overview

3 Cyber Functions

4 Further Plans

Agenda

1 Idea

2 Framework Overview

3 Cyber Functions

4 Further Plans

Vision

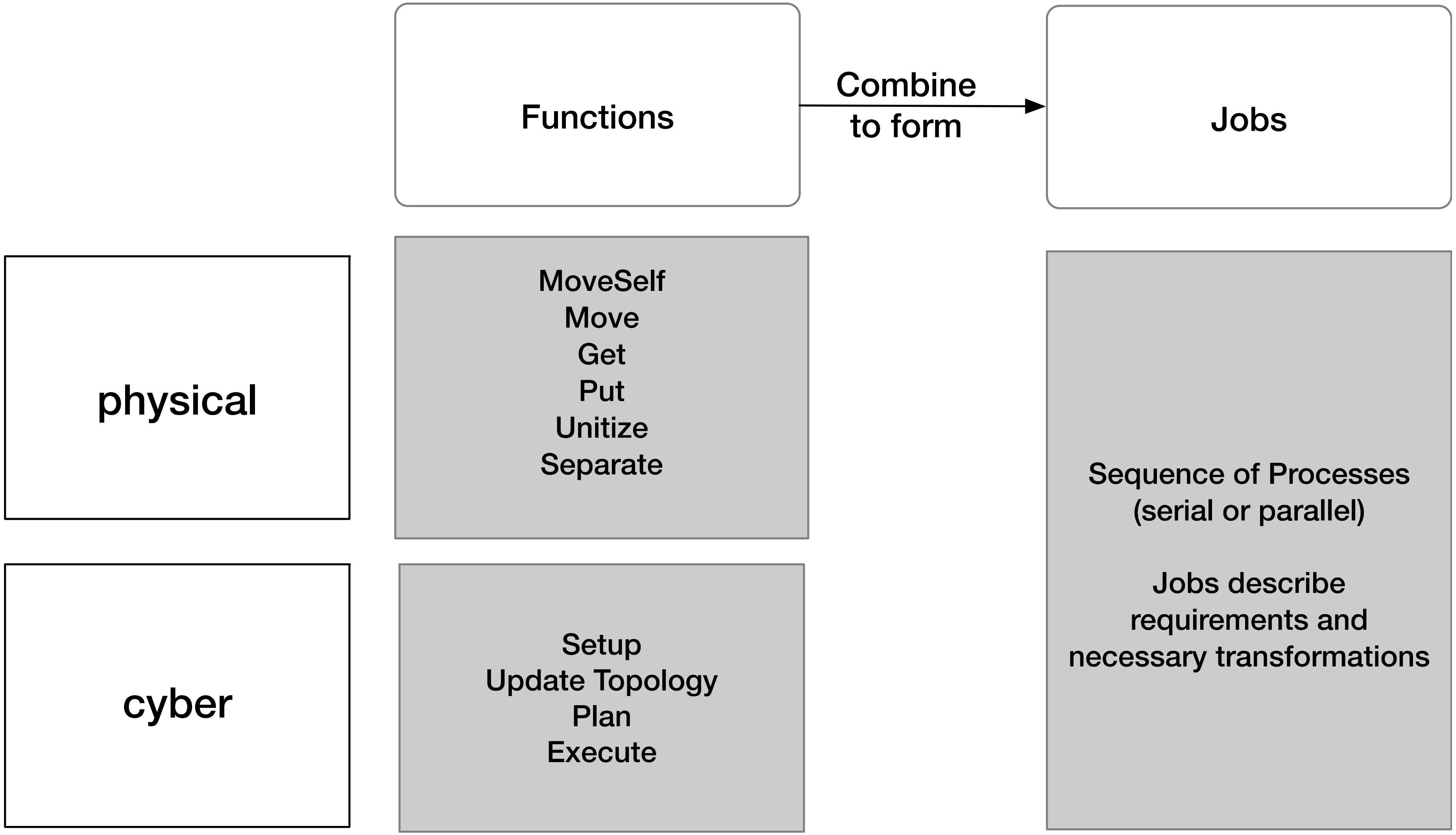
- Imagine installing a new material handling system in 1 day.
- Imagine changing a material handling system over a lunch break—done by the people who use it.
- Imagine leasing and returning some or all of a material handling system as requirements change during the year.

next
intralogistics

Three Hypotheses

- We can reduce all material handling tasks to a limited set of simple functions distributed among multiple material handling modules.
- There exists a methodology to distribute tasks among modules without an external designer or controller.
- We can meet all material handling requirements with a small set of connectable, reconfigurable modules.

Modeling Framework



Functions

Combine to form

Jobs

physical

MoveSelf
Move
Get
Put
Unitize
Separate

cyber

Setup
Update Topology
Plan
Execute

Sequence of Processes (serial or parallel)

Jobs describe requirements and necessary transformations

Agenda

1 Idea

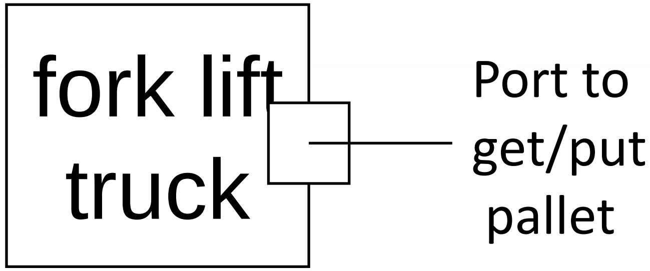
2 Framework Overview

3 Cyber Functions

4 Further Plans

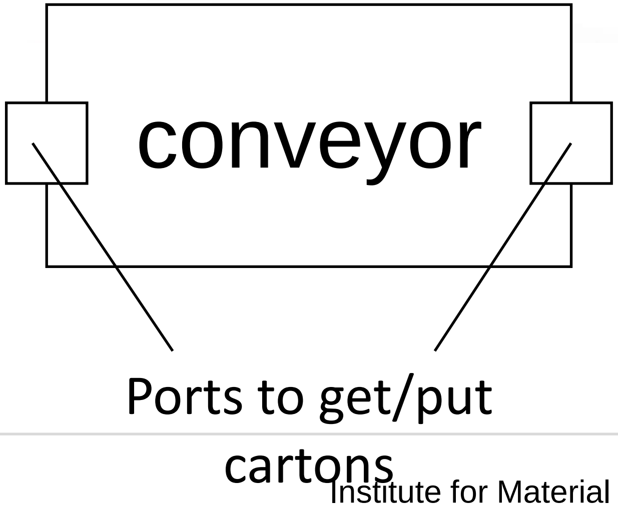
Store

Get and Put



fork lift truck

Port to get/put pallet



conveyor

Ports to get/put cartons

Move

MoveSelf

Move

Group and Ungroup

Unitize

Separate

Execution Type

active

passive

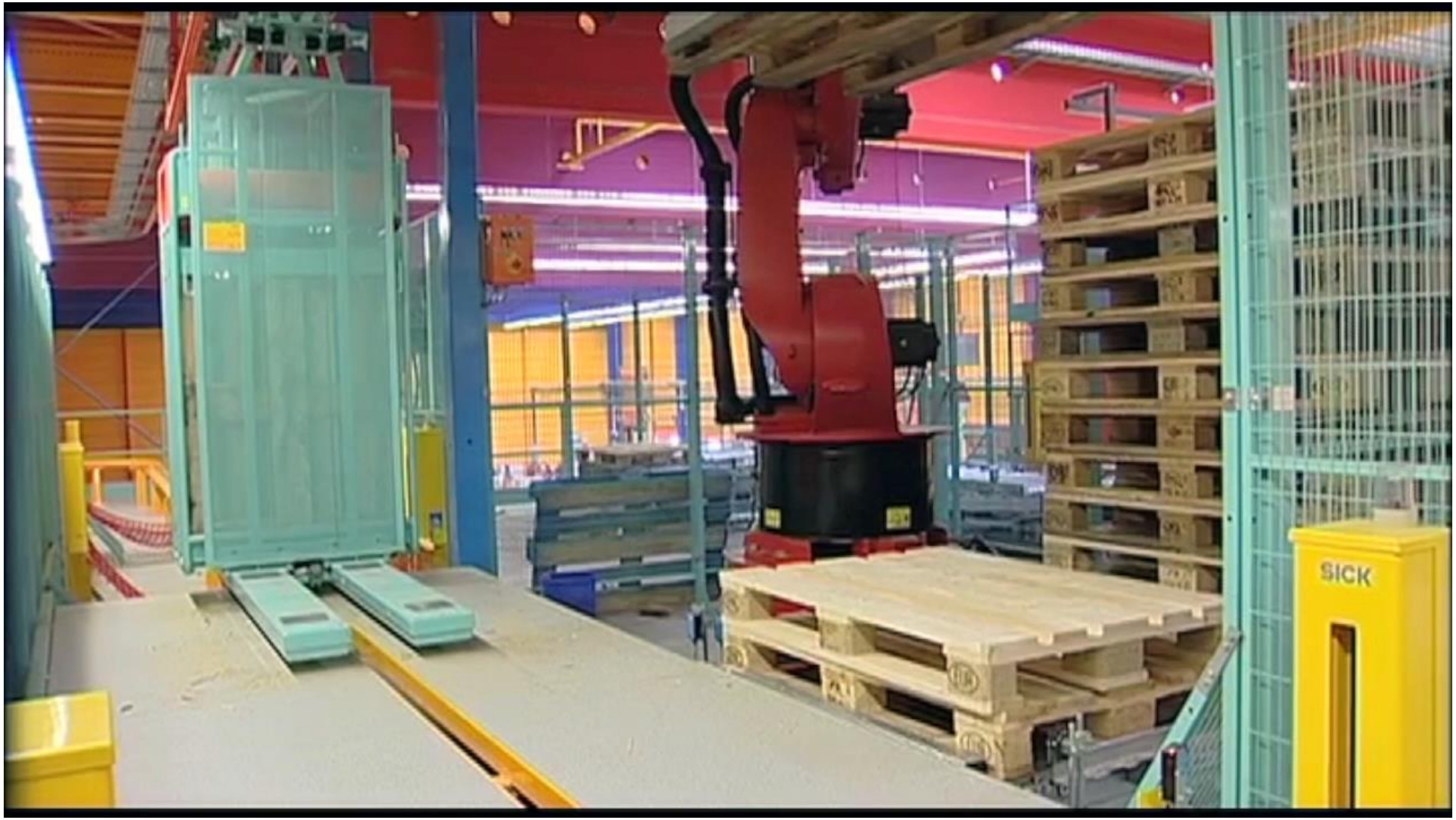
joint

Execution types must match!

Exercise

Exercise

Exercise



Agenda

1 Idea

2 Framework

3 Cyber Functions

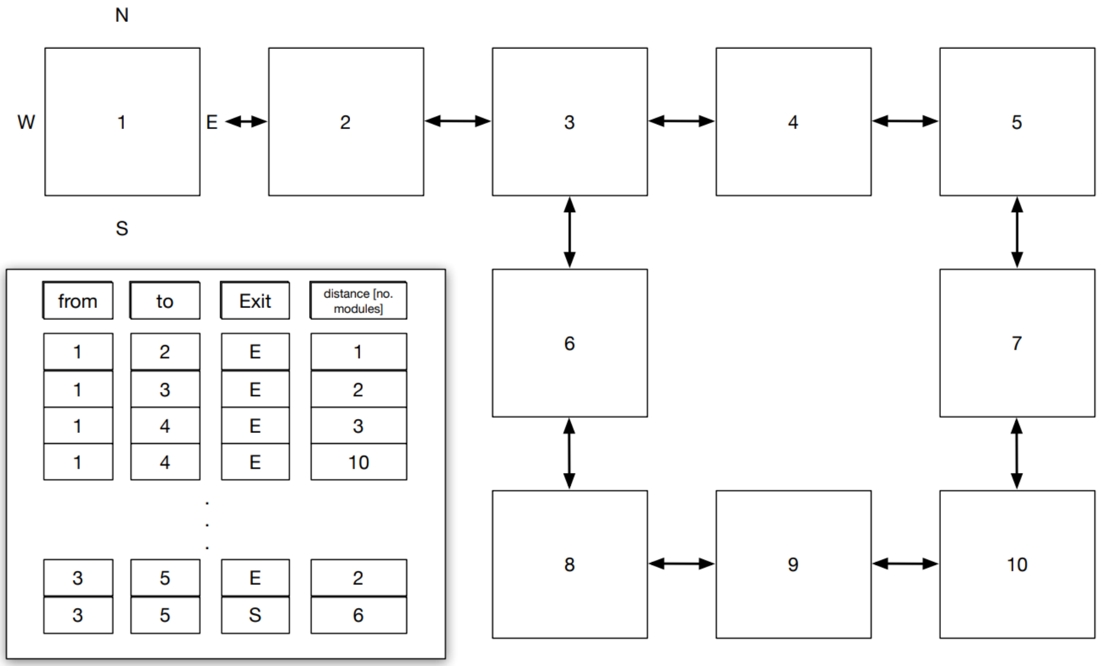
4 Further Plans

Overview Cyber Functions

Setup Functions

Planning Functions

Execution Functions



from	to	Exit	distance [no. modules]
1	2	E	1
1	3	E	2
1	4	E	3
1	4	E	10
	⋮		
3	5	E	2
3	5	S	6

„distance vector“

Example of a network with portions of the associated distance vector for Modules 1 and 3.

Topology

- Network
  - All Modules, I can communicate with everybody else
- Neighborhood
  - Everyone in the network with whom I could interact (but maybe not right now)
  - I must be compatible

✓

✗

- Connectedness
  - Everyone in the network with whom I could interact, and I’m *physically* connected
  - I must be connected to my neighbor before starting a transaction

Planning Functions (1/4)

- Job level (job = get pair of shoes and T-Shirt and pack parcel for Kai Furmans)
  - Source known, destination known
    - Find chain of modules able to transport from source to destination
  - Source known, destination unknown
    - Handling unit and location known, destination to be found and then see above
  - Source unknown, destination known
    - The item type (e.g., part number), required quantity, and destination (e.g., packing area) are known; the current storage location(s) of the corresponding items must be found and then see above

Planning Functions (2/4)

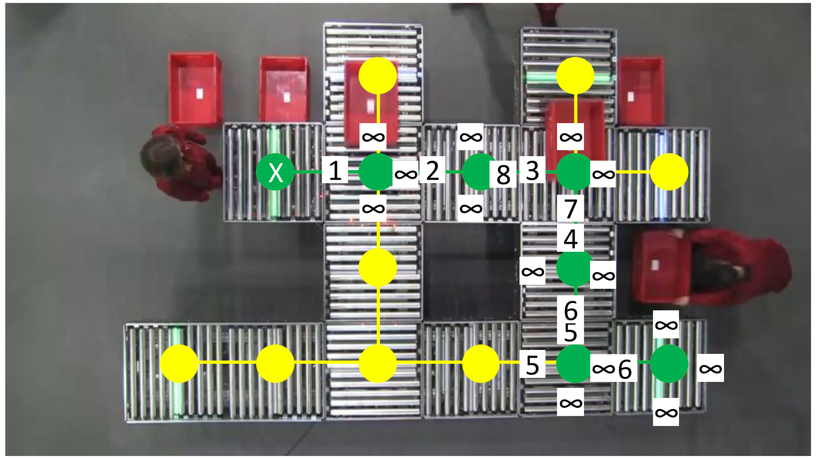
- Find functions
- Find route:
  - find a suitable route from the source to the destination and make reservations for a handling unit to be moved along this route.
- • Find object:
  - identify a set of modules where objects satisfying these characteristics are held (or stored). In the current implementation, we assume a list of available locations for each object is known.
- • Find storage location:
  - identify a set of modules that satisfy the characteristics and are not currently holding anything. In the current implementation, we assume a list of available empty locations for each job type is known.

Planning Functions: Routing (3/4)

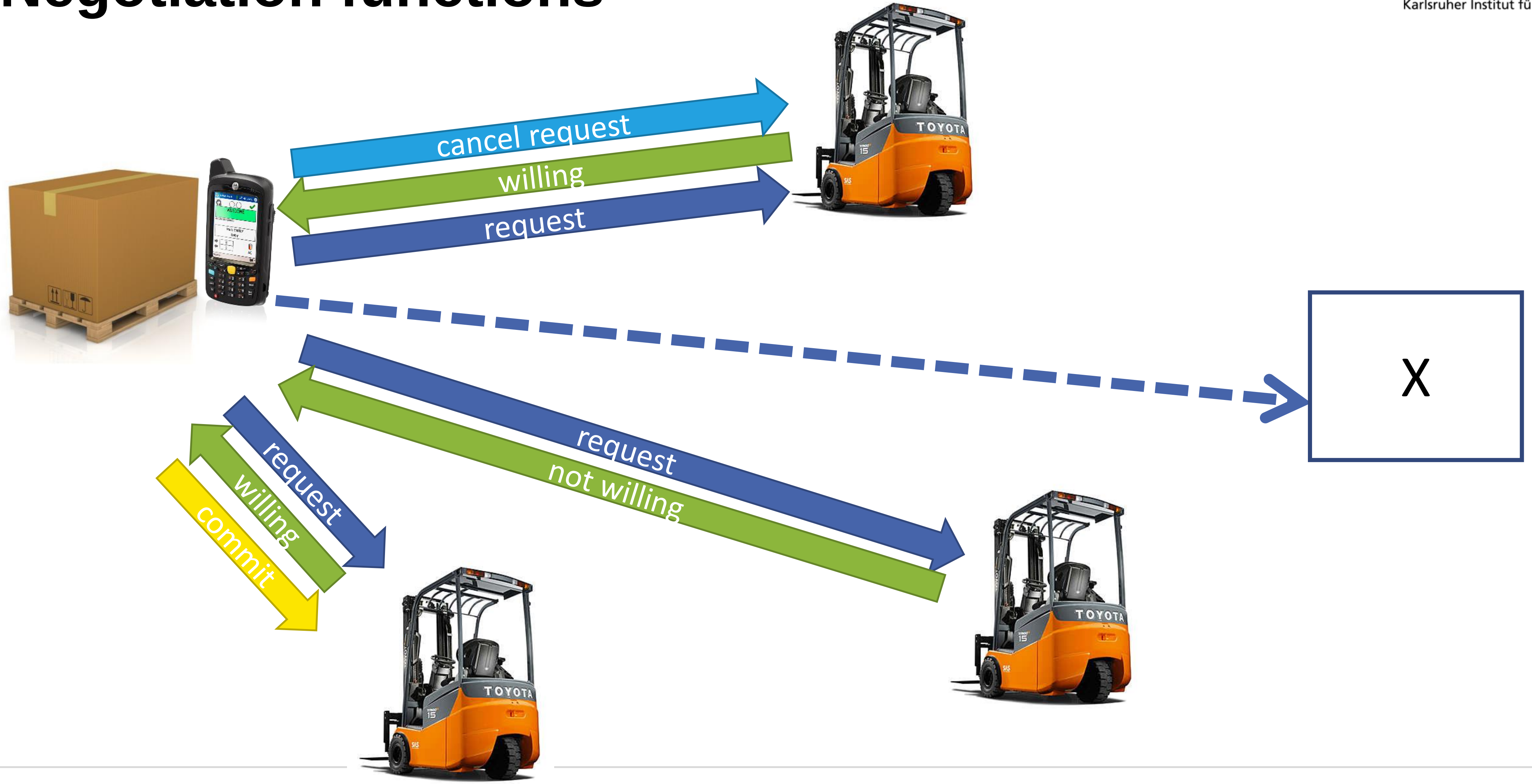
- Based on the topology
- Decentral route calculation
- Deadlock free
- For any kind of module
- Questions answered by Routing
  - Which is theoretically the fastest path to a destination?
  - Which is actually the fastest way? (under consideration of existing reservations)
  - Where is the next separator to break up a pallet into smaller handling units?
  - …

Planning Functions (4/4)

- Find next module:
  - This internal function identifies based on its distances vector the next best, but not yet requested module on the route.
- Request:
  - send request to the selected module to see if it is available to participate in the process at a future point in logical time. Check request: an internal function, that determines the earliest (logical) time that a request can be fulfilled. It might be necessary to send requests further downstream before the earliest time can be determined.
- Reply to request:
  - Returns *willing* if the request can be fulfilled, possibly with a logical time at which the request can be fulfilled. Returns *not willing* if the request cannot be fulfilled.
- Commit:
  - Sends a *commit* message to a selected module if it answered with *willing*. Cancel request: Sends a cancel request message to all modules requested but not committed to.



Negotiation functions

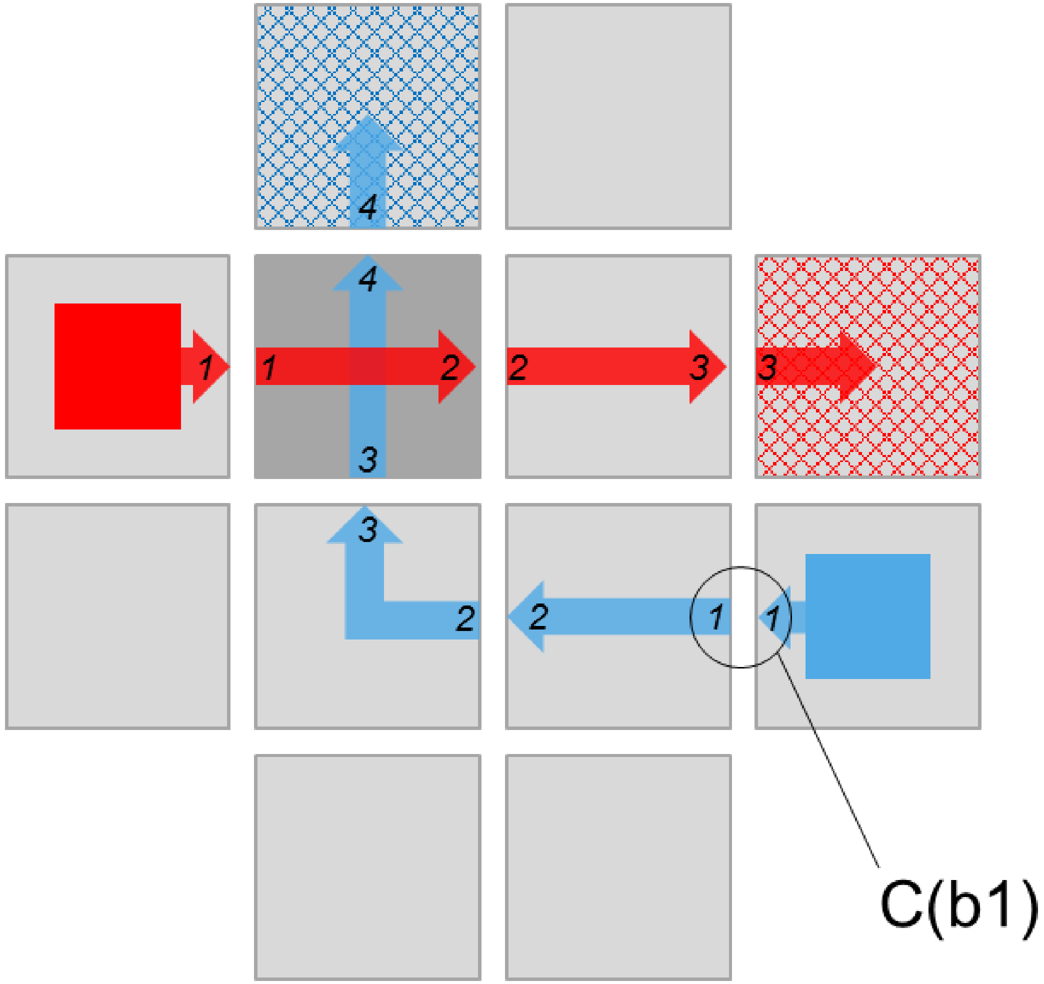


Management of Storage Locations

- Find storage location:
  - an empty storage location for an object with specified characteristics is requested.
- • Empty storage location available:
  - responds to a request with yes or no, and possibly with an available time. Yes implies that the holder is empty and that no previous reservation before that time has been accepted.
- • Commit to storage location:
  - notifies a storage location (a holder) that it should book the reservation at the appointed time.
- • Cancel request for storage location:
  - informs the storage location that it will *not* be used.
- • Process requests:
  - an internal function that keeps track of requests and responses.

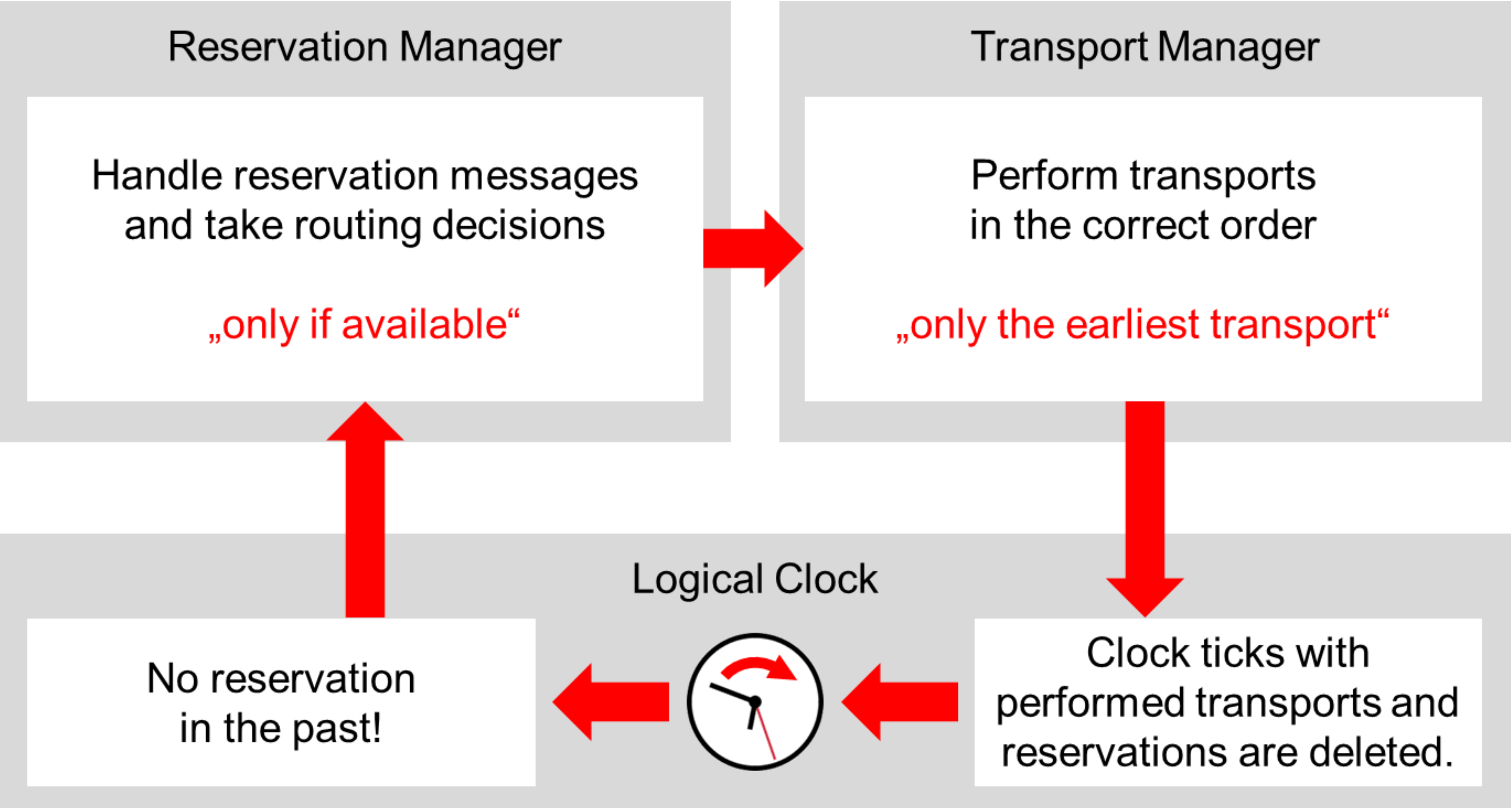
- There are caretakers!

Reservations



Source: Seibold (2016), Seibold, Furmans, Gue (2020)

Control Components in each module (1)



Reservation Manager

Handle reservation messages and take routing decisions

„only if available“

Transport Manager

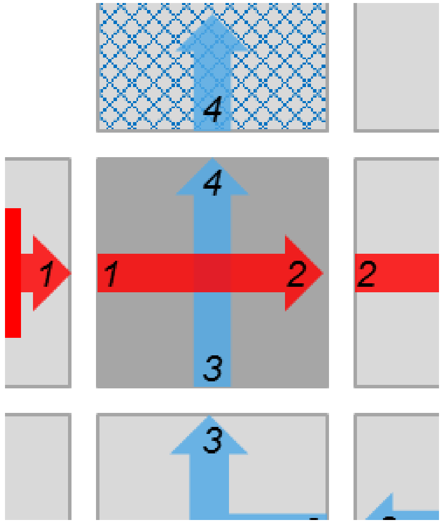
Perform transports in the correct order

„only the earliest transport“

Logical Clock

Clock ticks with performed transports and reservations are deleted.

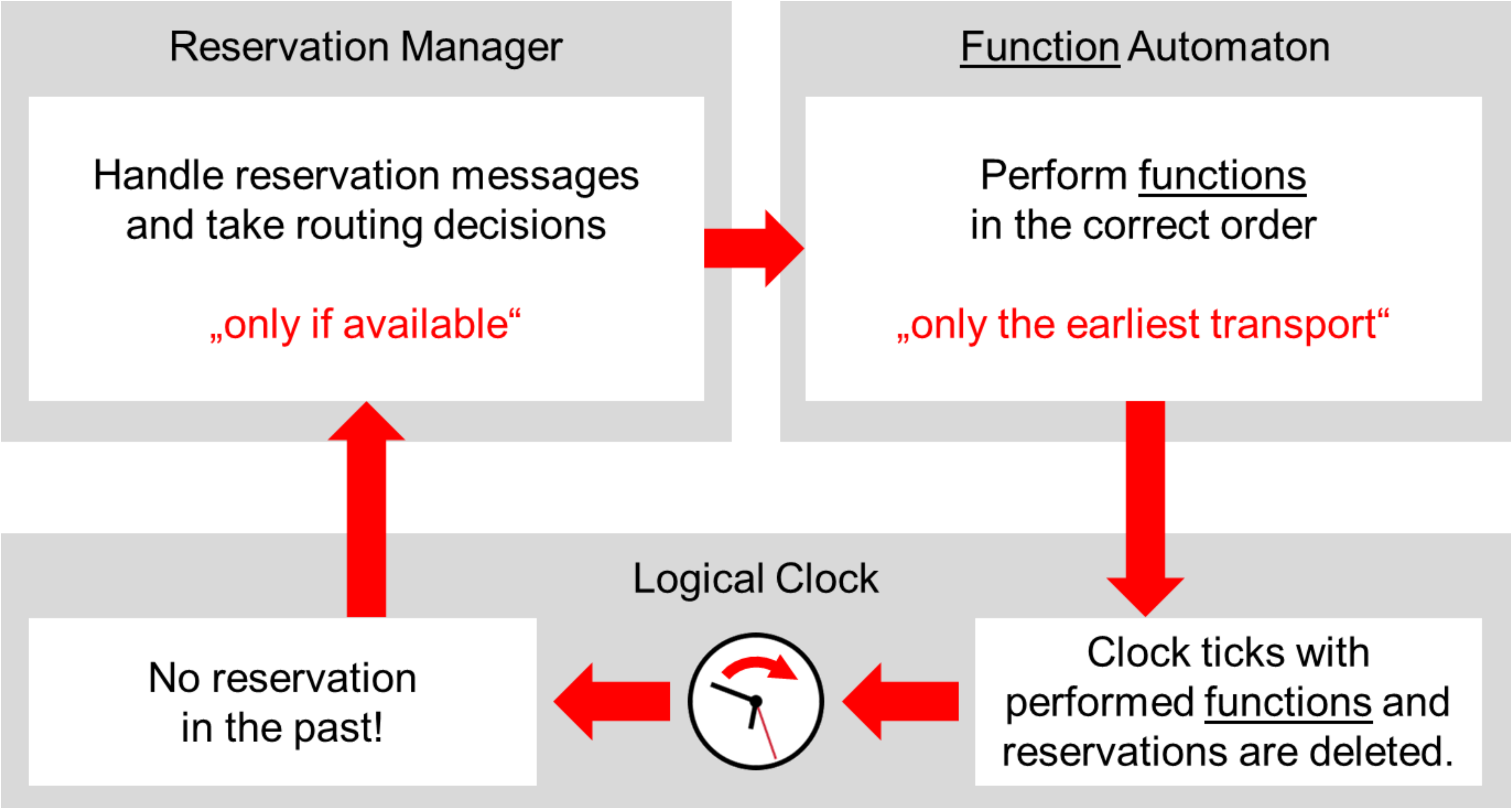
No reservation in the past!



example

Source: Seibold (2016)

Control Components in each module (2)



Reservation Manager

Handle reservation messages and take routing decisions

„only if available“

Function Automaton

Perform functions in the correct order

„only the earliest transport“

Logical Clock

Clock ticks with performed functions and reservations are deleted.

No reservation in the past!

Agenda

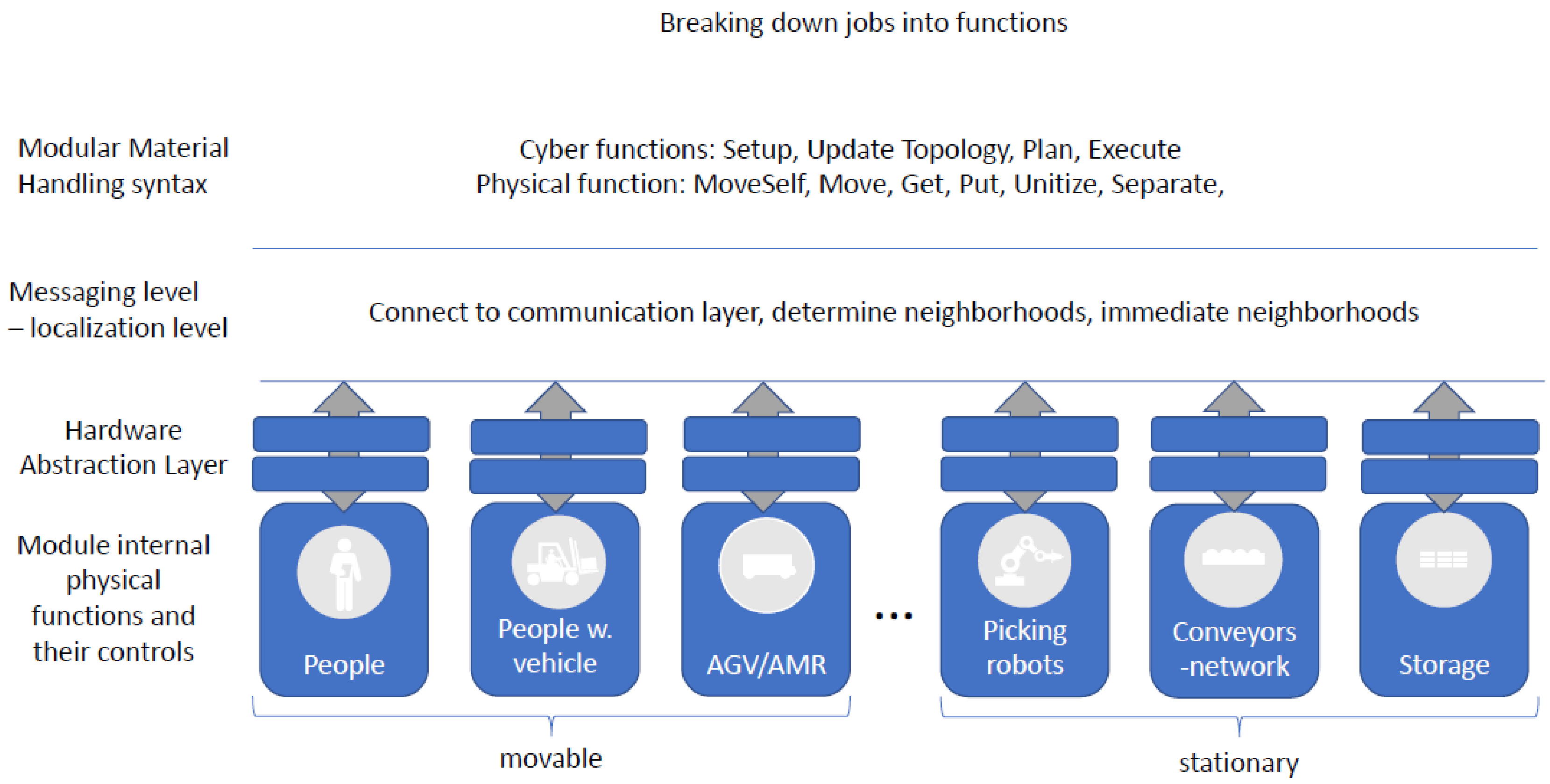
1 Idea

2 Framework

3 Cyber Functions

4 Further Plans

Architecture Modular Material Handling System



Breaking down jobs into functions

Modular Material Handling syntax

Cyber functions: Setup, Update Topology, Plan, Execute
Physical function: MoveSelf, Move, Get, Put, Unitize, Separate,

Messaging level – localization level

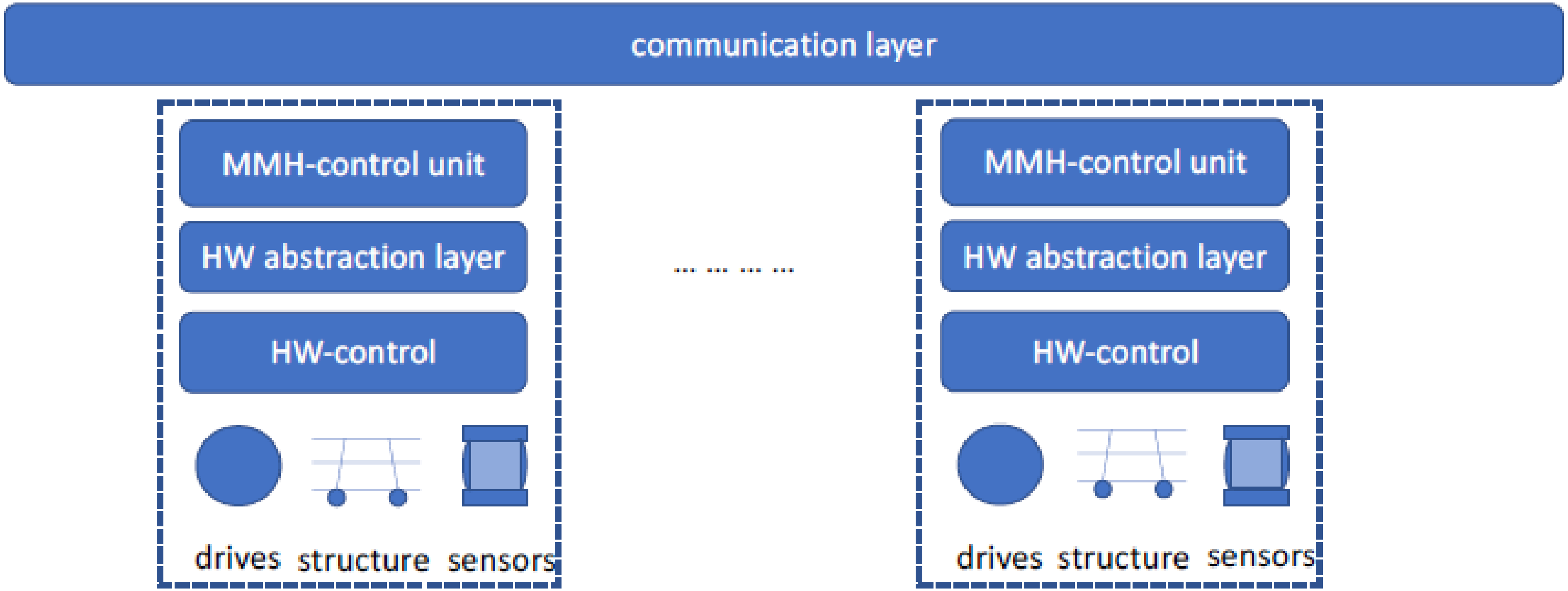
Connect to communication layer, determine neighborhoods, immediate neighborhoods

Hardware Abstraction Layer

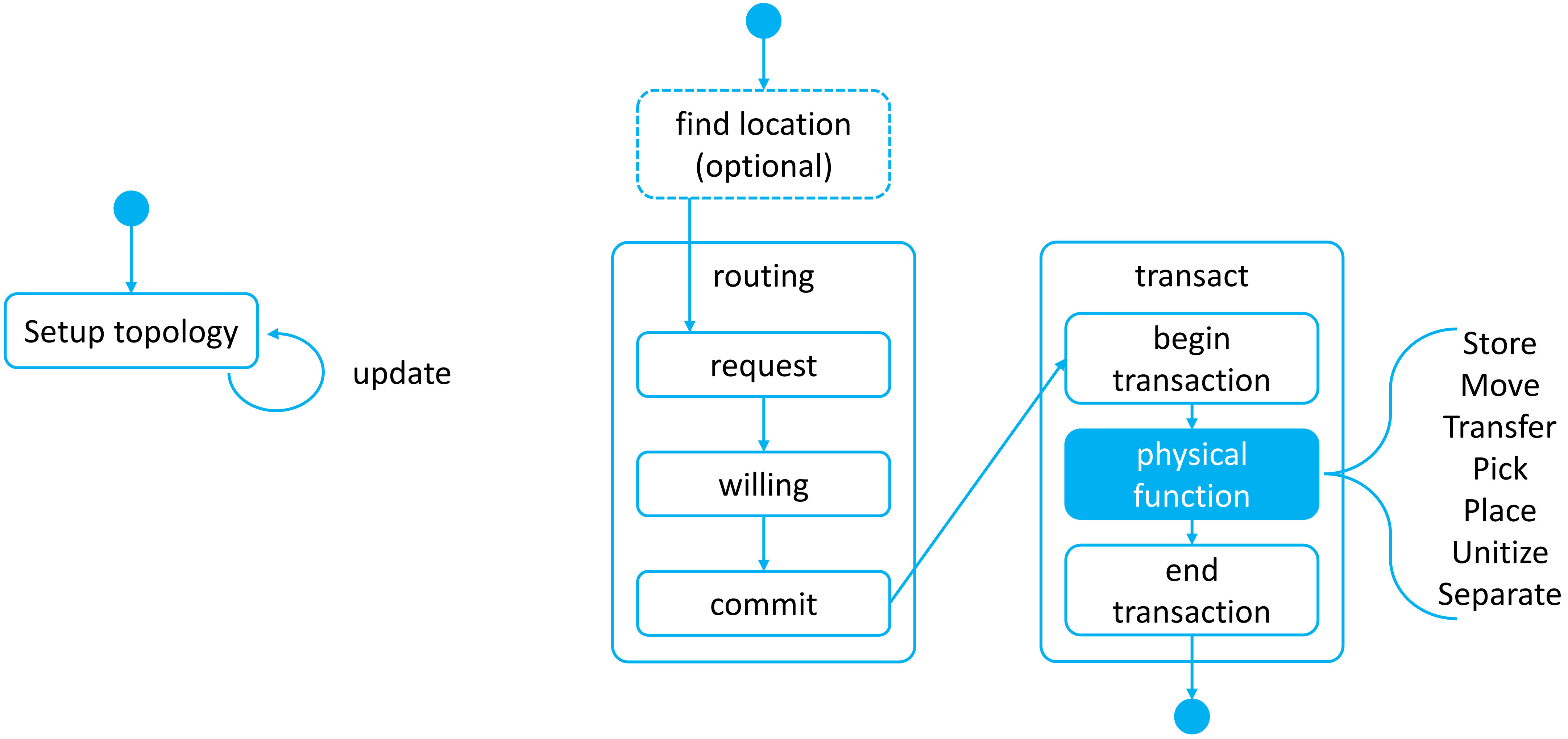
Module internal physical functions and their controls

Overview of the architecture of a modular material handling system.

Structure of a Module



Cyber Fuctions



Setup topology

update

find location (optional)

routing

request

willing

commit

transact

begin transaction

physical function

end transaction

Store
Move
Transfer
Pick
Place
Unitize
Separate