UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

Requirements Engineering & Management

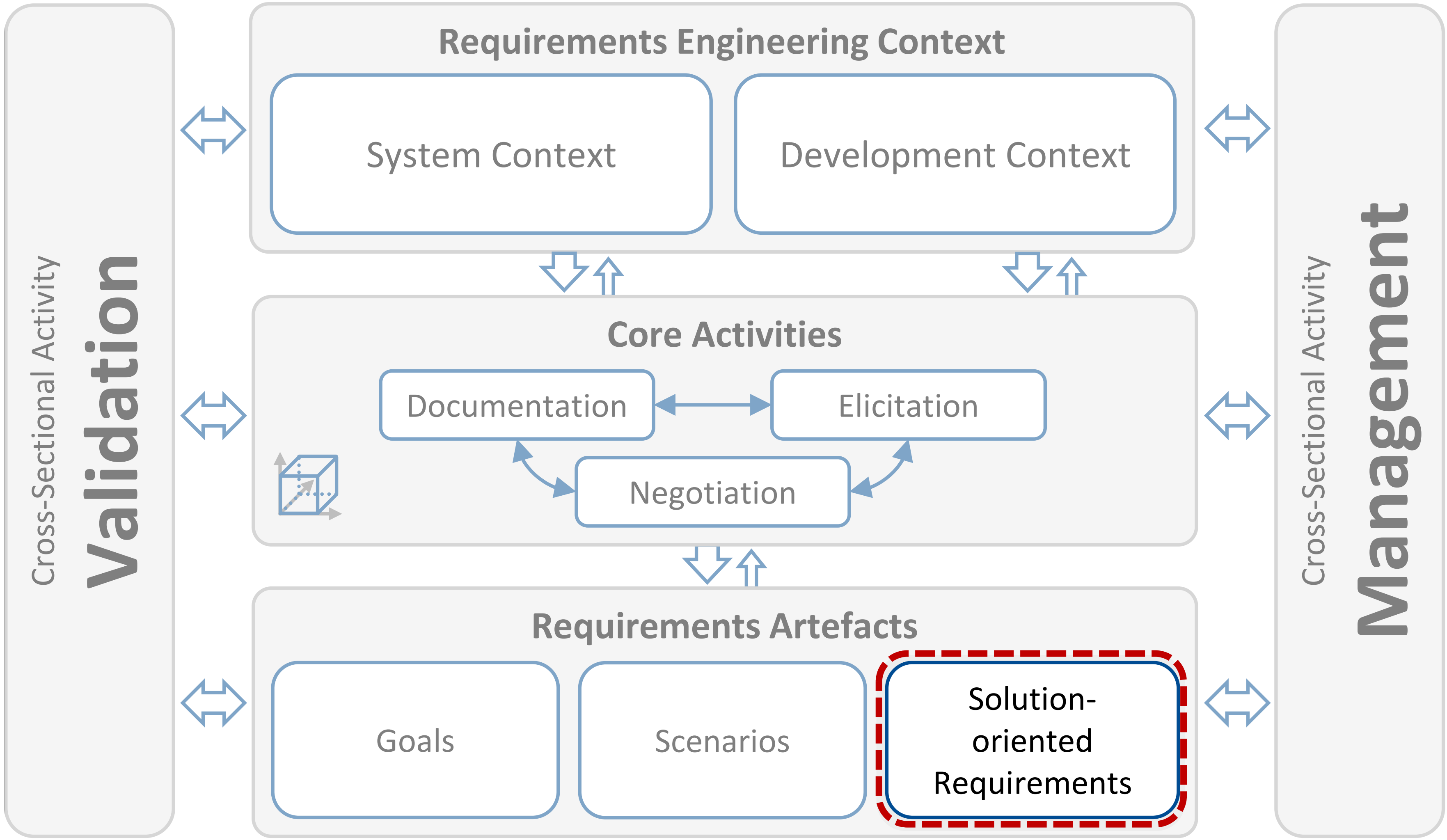
Solution-Oriented Requirements – Functional Modelling II

Prof. Dr. Klaus Pohl

PALUNO The Ruhr Institute for Software Technology

Framework for Requirements Engineering

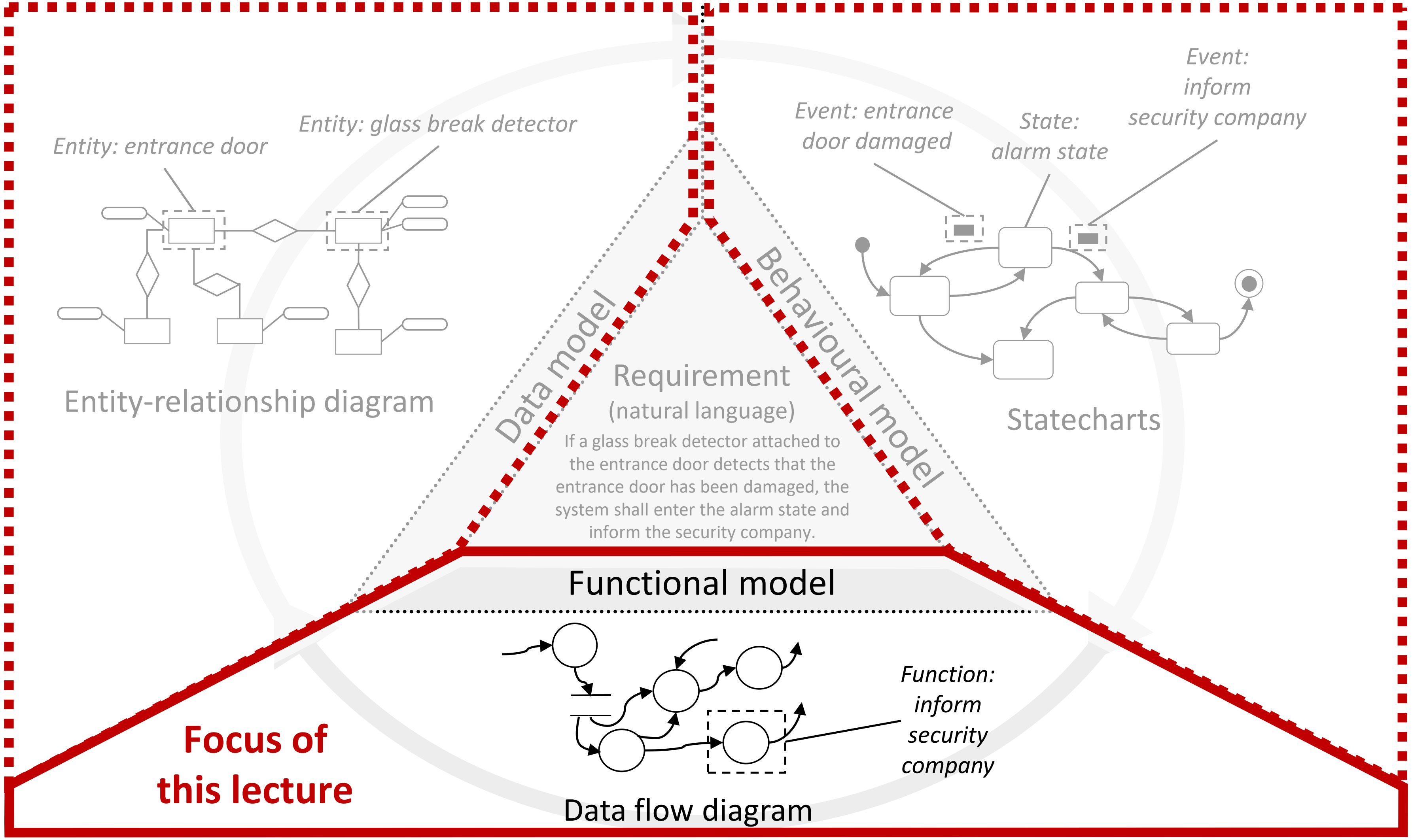
UNIVERSITÄT DUISBURG ESSEN
Offen im Denken



Agenda

1. Guidelines for Modelling Data Flow Diagrams
2. Typical Errors in Data Flow Diagrams
3. Methods for Modelling Data Flow Diagrams

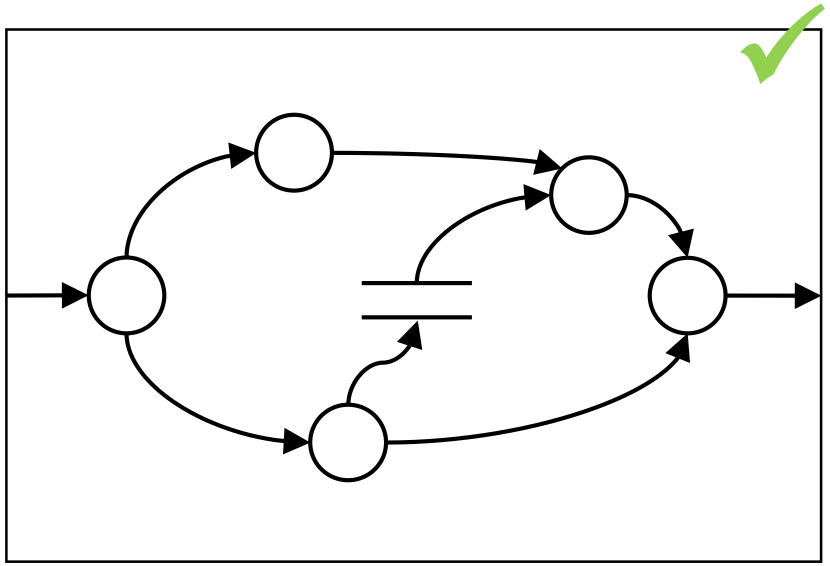
Model-based Documentation in the three Perspectives

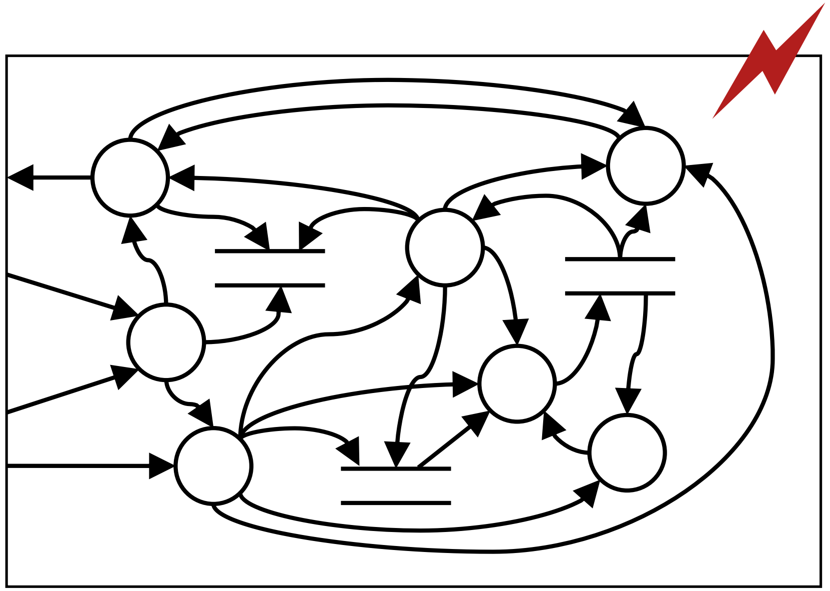


1. Guidelines for Modelling Data Flow Diagrams

Appropriate Diagram Size

- Pay attention to **keep each Data Flow Diagram readable**! The **aim** of DFDs is to structure a problem and **support** the **communication** about the problem!
- A single diagram should not contain more than **7 ± 2** processes/data stores.

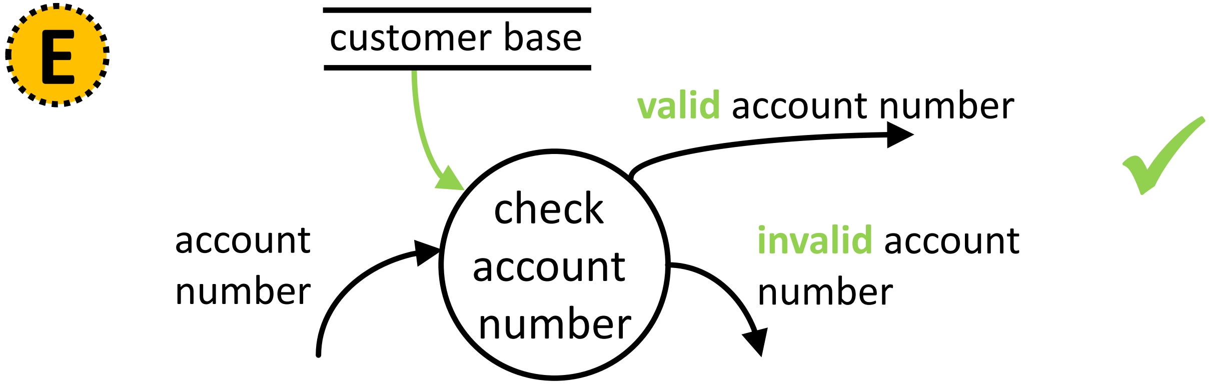




based on [DeMarco 1979], p. 82

Naming of Data Flows (1)

- Use **unique and meaningful names** indicating the kind of data carried by the flow.
  - **Exception**: Names of data flows for read and write access to a data store can be omitted if the store already describes the flows sufficiently.
- The name of a data flow should be **comprehensible**, and should **characterize the information the data flow carries**, as well as key properties of the information.

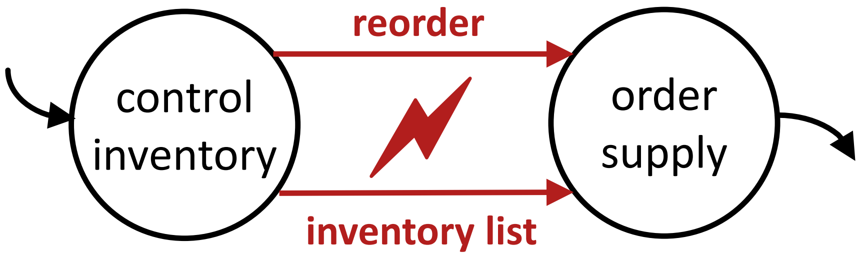


based on [DeMarco 1979], p. 66, 96f

Naming of Data Flows (2)

- **If it is difficult to find a good name** for a data flow, the object considered **might not actually be a data flow**. In that case, consider **restructuring** the model so that correct names can be identified more easily.
- If two distinct **data flows** from process P_1 to another process P_2 carry data packages that can be regarded **as composites**, consider modelling them as a single data flow.

E



Modelling the two data flows “recorder” and “inventory list” separately **increased diagrammatic complexity** and may hinder communication.

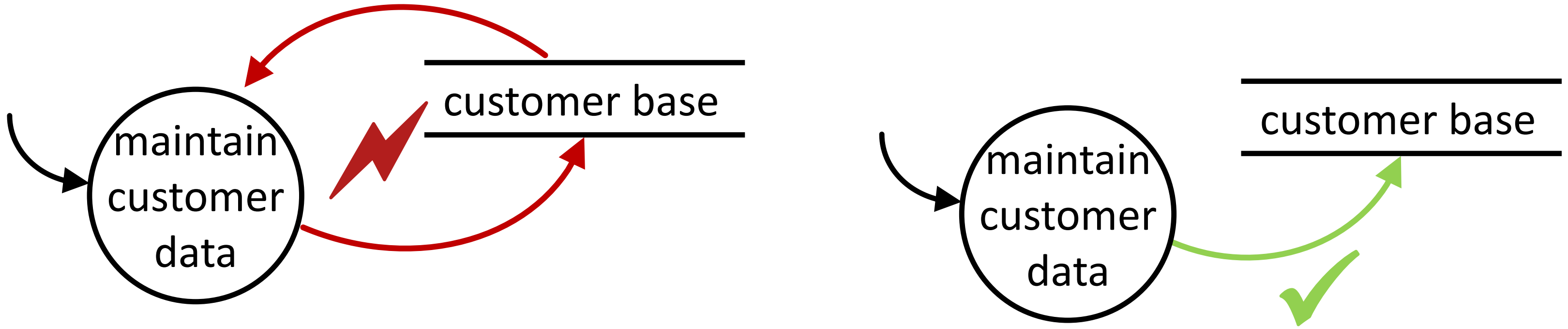
[DeMarco 1979], p. 66

Model Only the Main Data Flows

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

- If a process changes existing data in a data store, it must **read** these data first.
- However, in order **to reduce complexity** only a data flow from the process to the data store should be drawn if the **main task of the process is to change the data**.

E

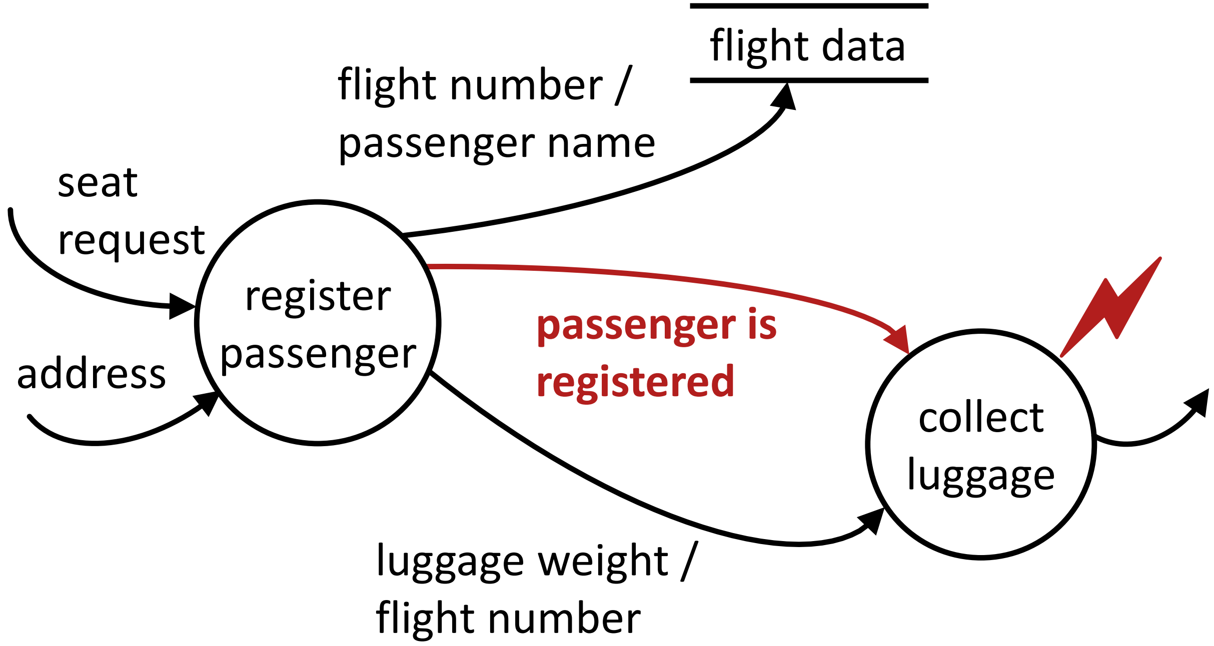


[DeMarco 1979], p. 58

Model Data Flows and Not Control Flows

- Data flows are **not used to model**:
  - **Control flow**: Data flows do not provide information about the time when processes are executed.
  - **Sequences** or orders of process execution
  - Triggering **events** of processes

E

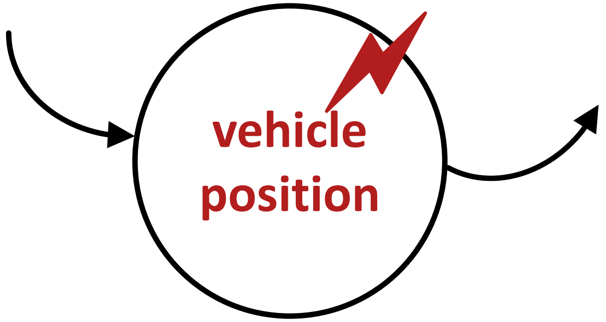


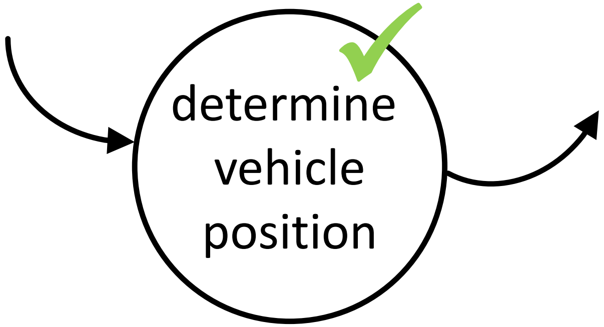
[DeMarco 1979], p. 68

Naming of Processes

- The name of a process should consist of a **verb and a noun**, and should be **comprehensible** and **meaningful**. The name should fully characterize what the process does.
- If it is **difficult to find the right name** for a process, **consider restructuring** the model.

E





[DeMarco 1979], p. 66ff

Guidelines – Summary

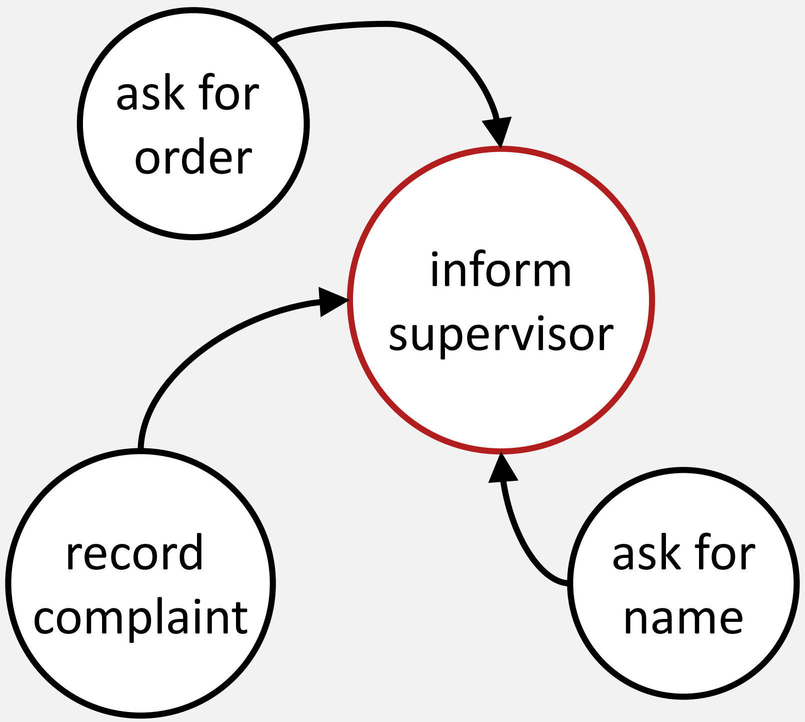
- Appropriate Diagram Size
- Naming of Data Flows
- Model only the Main Data Flows
- Model Data Flows and Not Control Flows
- Naming of Processes

2. Typical Errors in Data Flow Diagrams

Indicators for Common Errors (1)

E

Process as information sink within the system



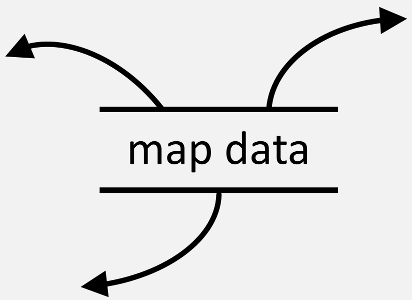
Miraculous creation of data

[Yourdon 1989], Ch. 9.2.5

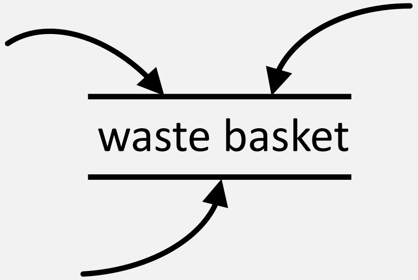
Indicators for Common Errors (2)

E

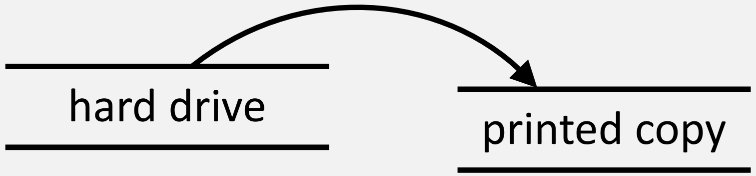
Read-only data stores



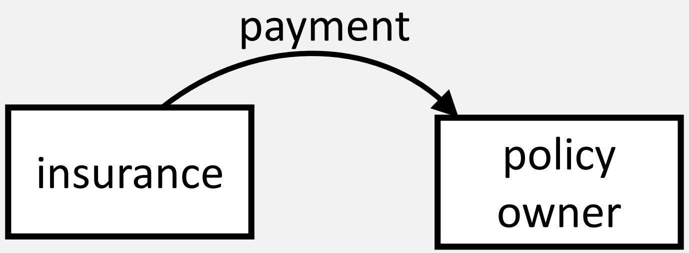
Write-only data stores



“jumping“ data



Data flows in the context

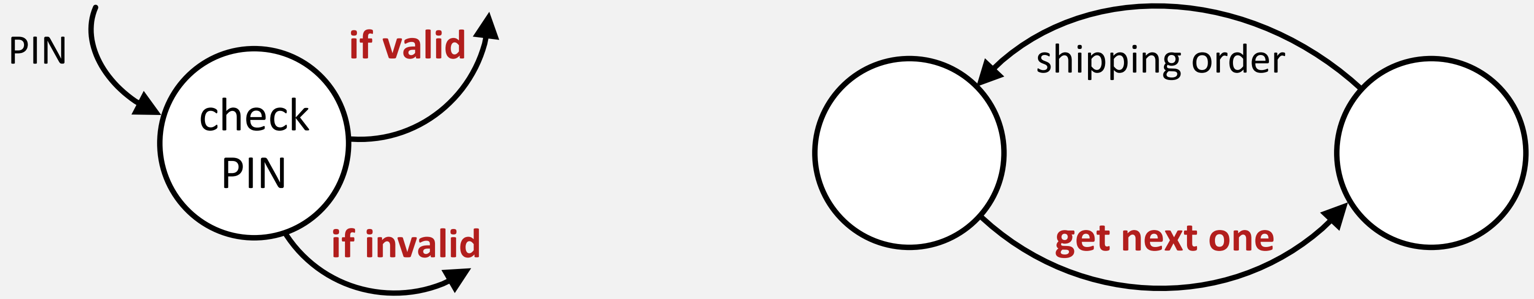


[DeMarco 1979], Ch. 6.1 and Ch. 9.1.4

Indicators for Common Errors (3)

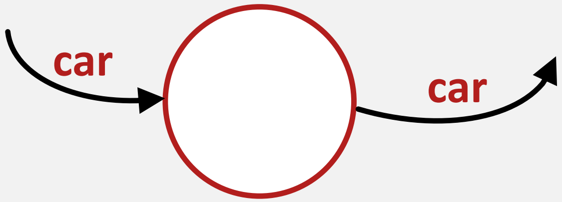
E

Control flows



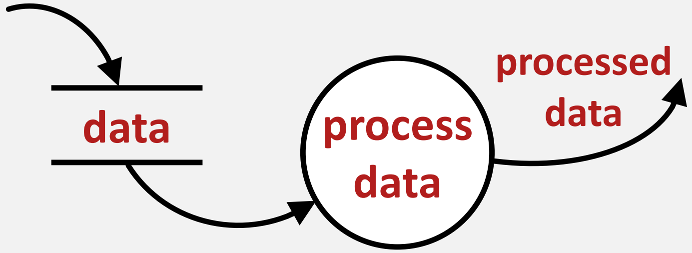
[DeMarco 1979], Ch. 6.6

Processes without functionality



[DeMarco 1979], Ch. 9.1.1

Meaningless names

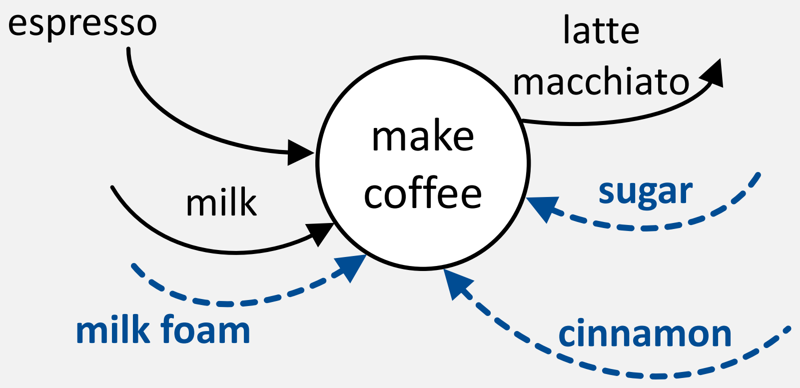


[DeMarco 1979], Ch. 9.2.2

Indicators for Common Errors (4)

E

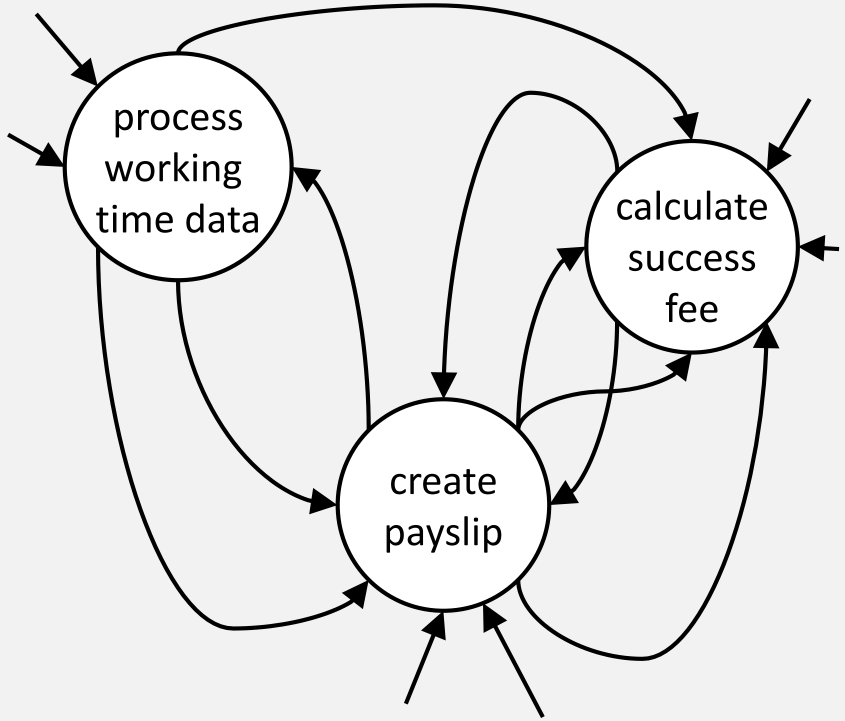
Law of information preservation



Missing information:
Milk foam, sugar, cinnamon

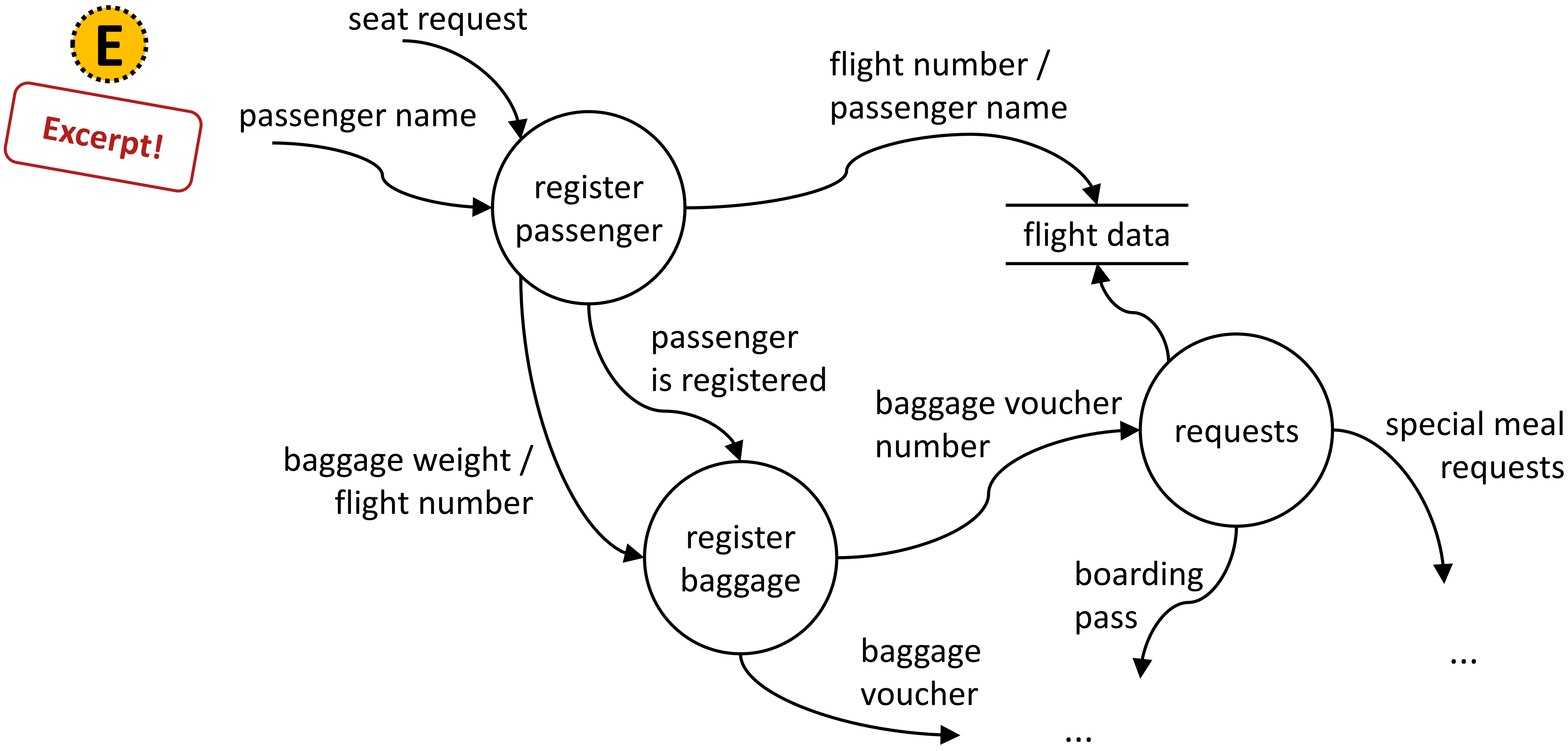
[DeMarco 1979], Ch. 9.1.3

Too complex interfaces

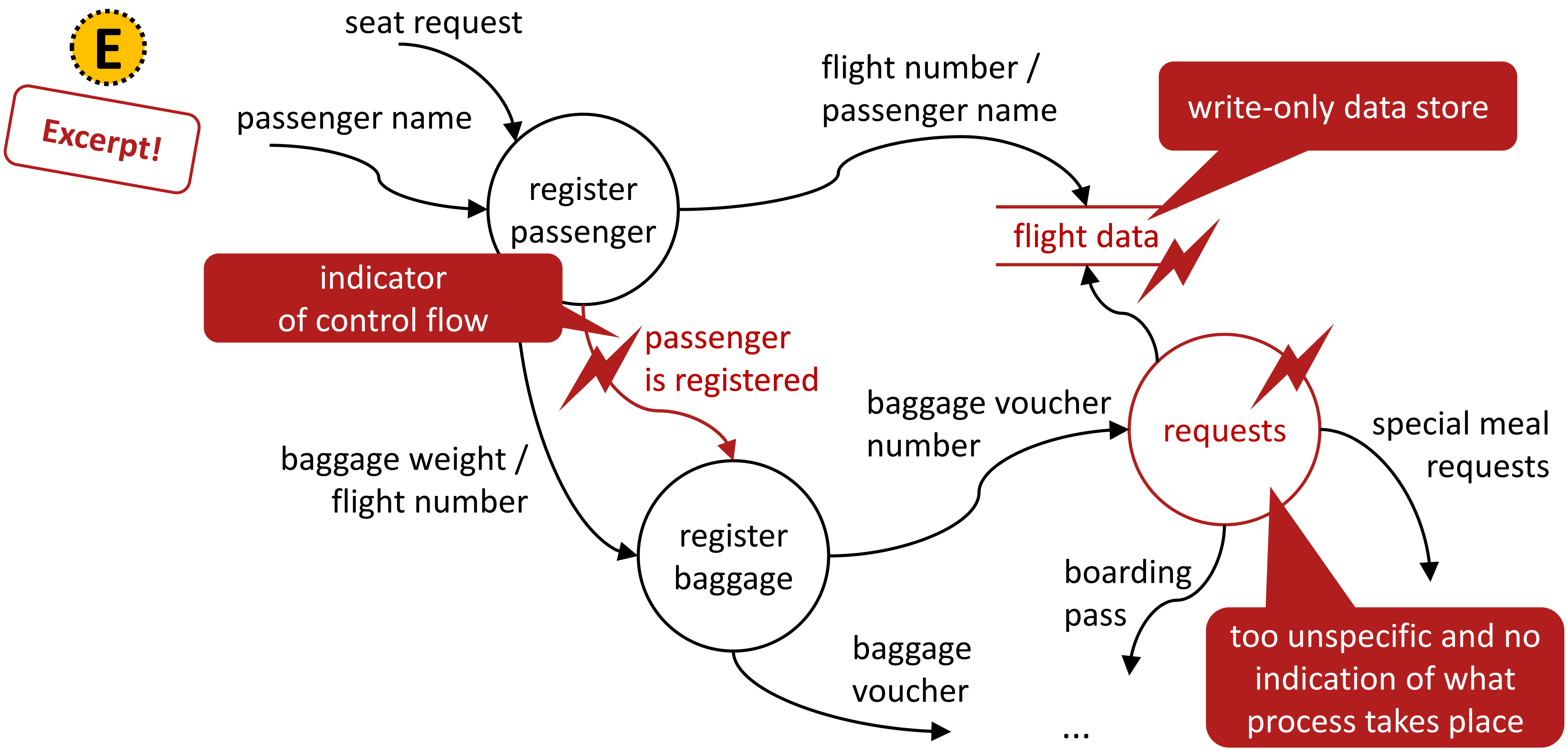


[DeMarco 1979], Ch. 9.2.1

DFD with Indicators for Errors (1)

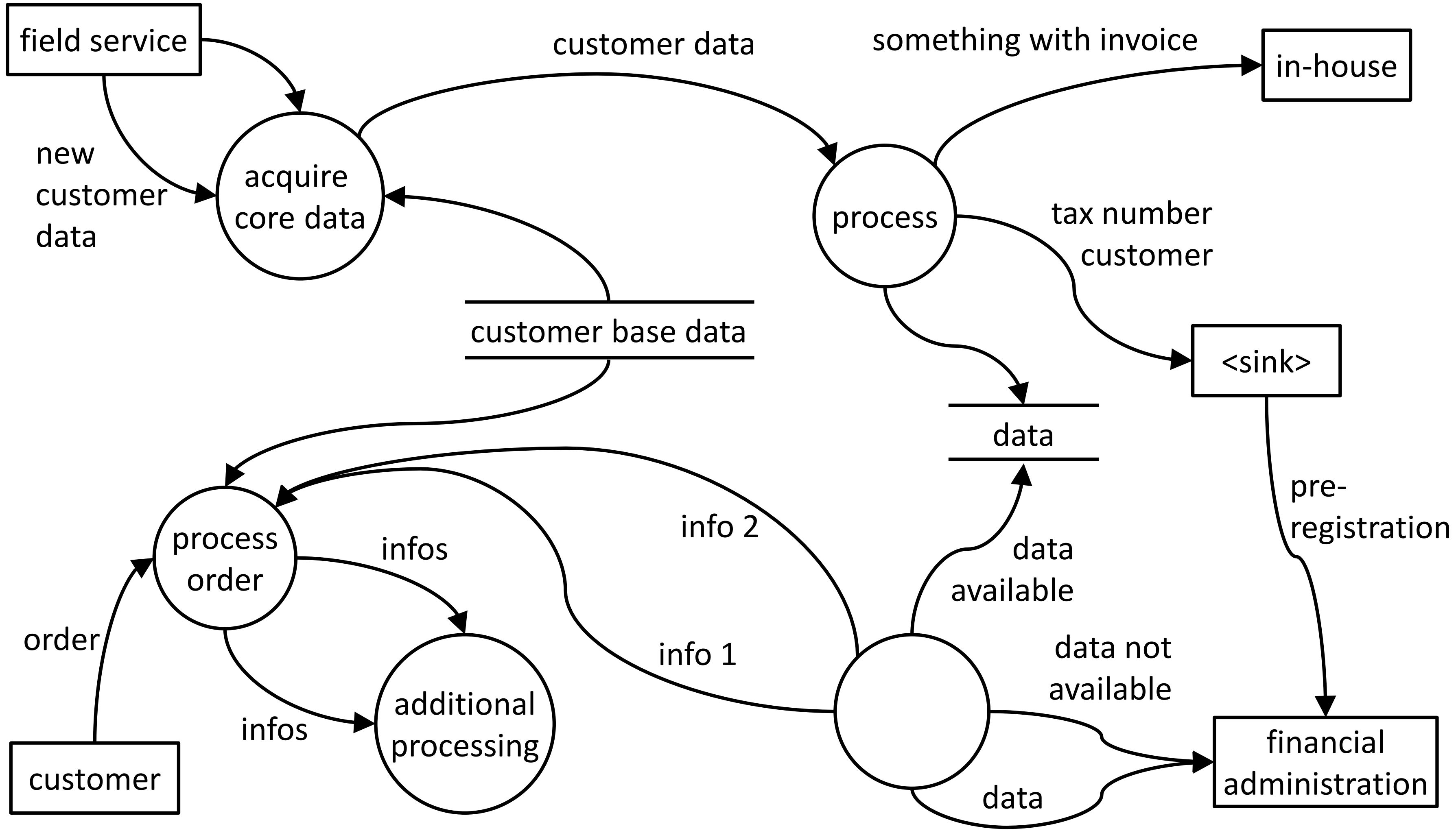


DFD with Indicators for Errors (2)

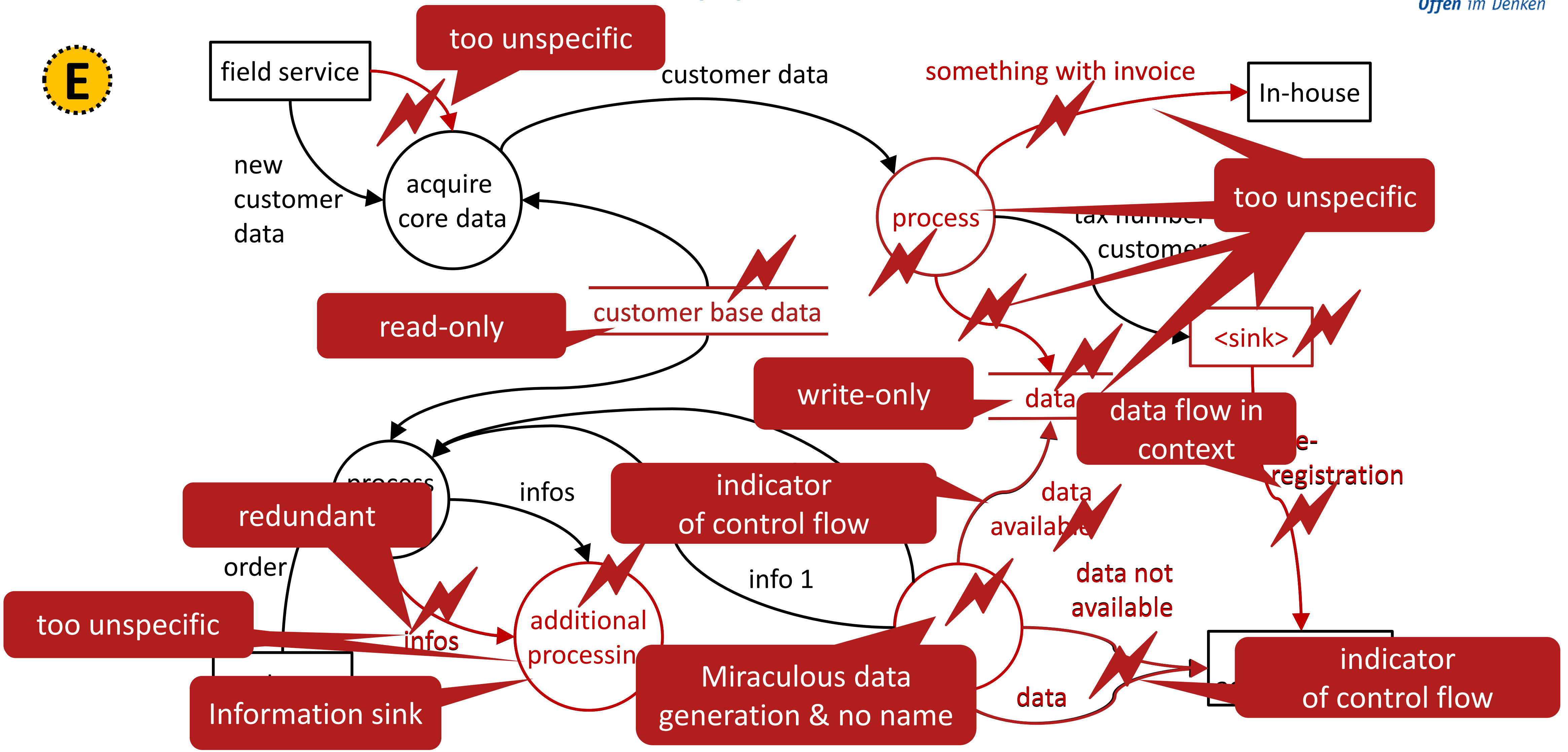


DFD with Indicators for Errors (3)

E



DFD with Indicators for Errors (4)



3. Methods for Creating Data Flow Diagrams

Motivation for Modelling Techniques

- **Understanding** Data Flow Diagrams is **relatively easy**.
- **Creating** Data Flow Diagrams **is significantly harder**:
  - What are criteria for **defining the scope of the system**?
  - What are criteria for **decomposition**?
  - What are criteria for **modelling**?
- **How to avoid making errors** in identifying information to be modelled and errors in modelling this information?
- **Techniques of Structured Analysis** guide the identification of information and its definition in Structured Analysis models.

Modelling Techniques

1. Context Delineation

based on [Yourdon 1989]

Basic idea: Delineate top-level system objectives and model the DFD.

Context delineation is the foundation for many other approaches, such as Event-based Partitioning and Jigsaw Puzzle.

2. Event-Based Partitioning

based on [McMenamin and Palmer 1984] and [Yourdon 2006]

Basic idea: Identify triggers from the context and model the processes producing responses to them.

3. Jigsaw Puzzle

based on [McMenamin and Palmer 1984]

Basic idea: Determine principle functionality, create data flow fragments and stitch them together.

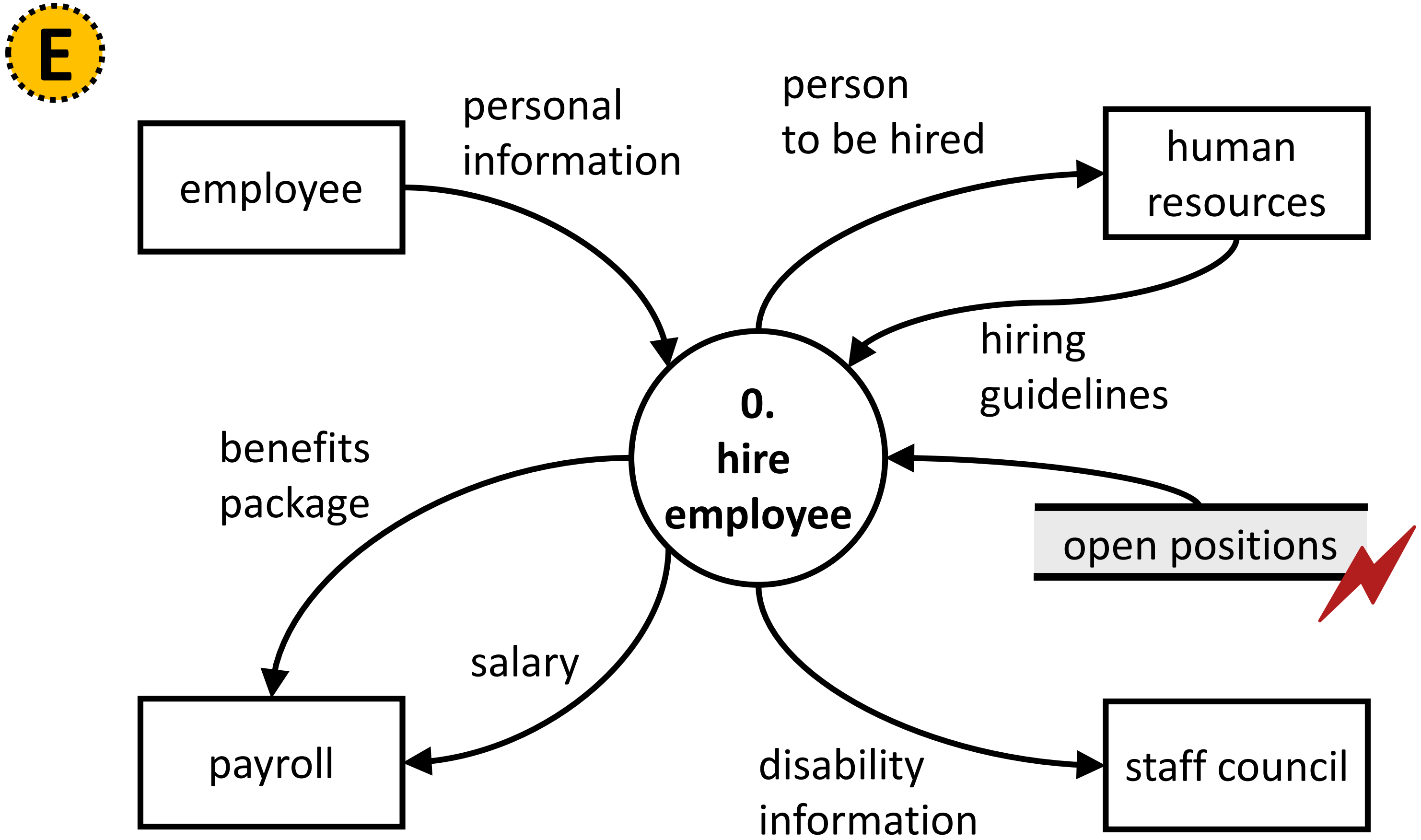
4. Pursuit of Data

Basic idea: Trace input data to output data honouring all transformations in between.

1. Context Delineation: Method Outline

1. Determine the purpose of the system
   - Describe the important **goals** (at most one paragraph), to be refined and more precisely specified in the course of the project.
   - Describe **quantifiable properties**.
2. Create a context diagram
   - Model the system as **1 – 4 processes**.
   - Model the system context in terms of **sources and sinks** (persons or organizations) and **data stores** (created or used by other systems).
   - Model **data flows** between system and context (**stimuli** and **responses**).
3. Define list of events
   - Describe **external stimuli** that are input for processes, as well as **responses** the context expects from the system.

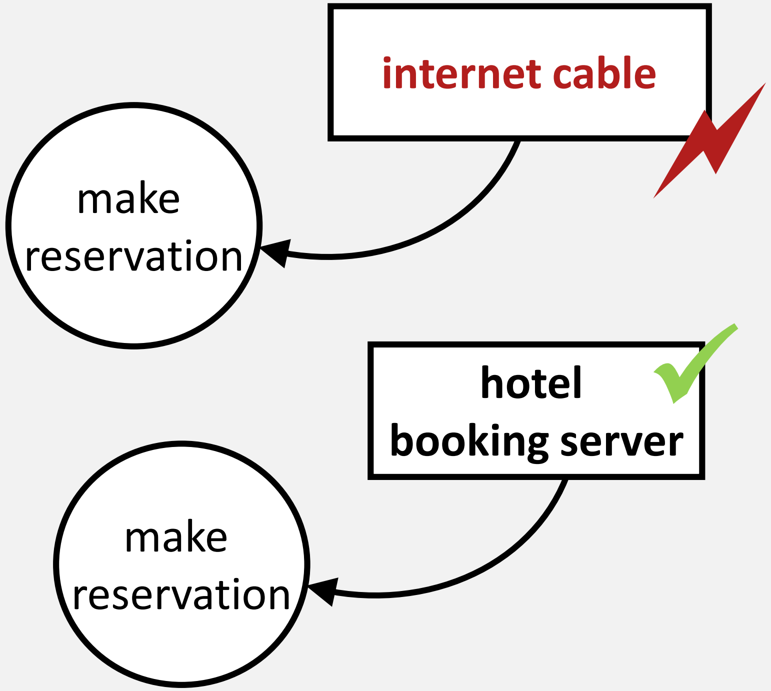
1. Context Delineation: Context Diagram



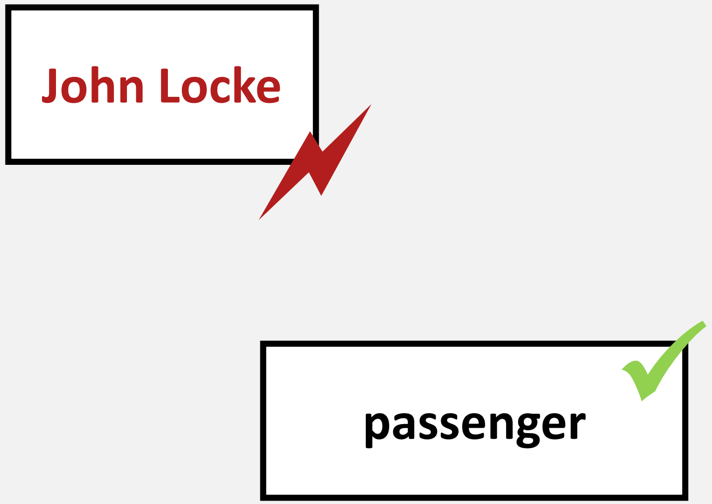
Modelling **external data stores** is **not allowed** in (original) data flow modelling.

1. Context Delineation: Hints and Heuristics (1)

“Real” sources and sinks instead of transport media

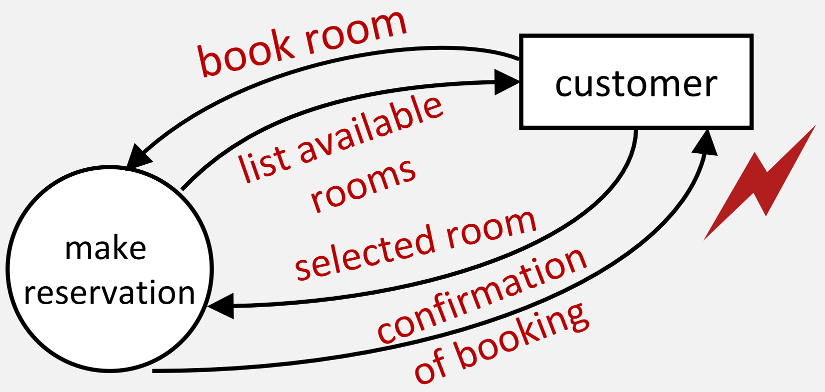


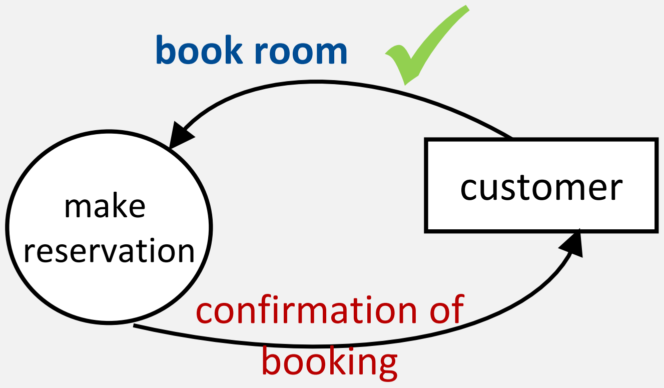
Role names instead of person names



1. Context Delineation: Hints and Heuristics (2)

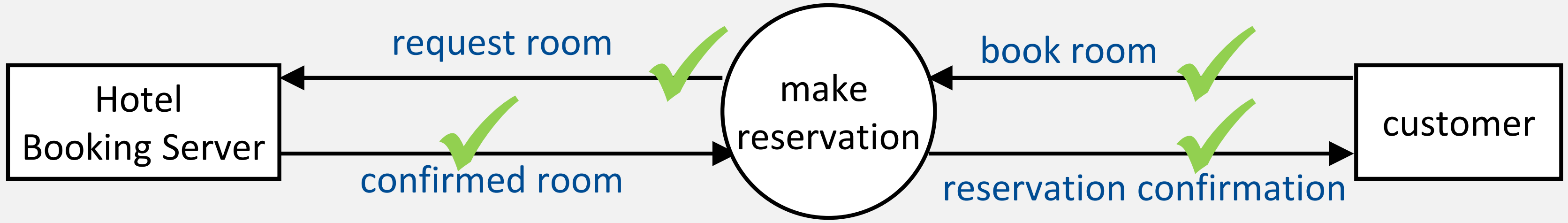
Net data instead of dialogues





Exception: necessary stimuli

(if otherwise the system or the source/sink does not know that data is required)



2. Event-based Partitioning (1)

- Focus on **events** occurring in the real world (i.e., in the system context):
  - **External events**, which are visible to the system by incoming data flows.
  - **Temporal events**, which describe relevant points in time that are determined by observing clocks (calendars,…) and internal data stores.
- Events can only be **perceived** but **not affected** by the system.
- Model the intended system **reactions** on these events in form of scenarios.
- The intended system reactions need to be **planned**.
  - **Stimulus-response** system

2. Event-based Partitioning (2)

Method Outline

1. Determine the goals (vision) of the system (see Context Delineation).
2. Delineate the system from its context (see Context Delineation).
3. Identify relevant events.
4. Model event scenarios.
5. Integrate and hierarchize the models.
6. Complete the model.

2. Event-based Partitioning (3)

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Step 3: Identify relevant events

- Create an **event list** by phrasing the events:
  - „WHO (= which source/sink) does WHAT?".
  - Or „IS It NOW TIME FOR …".
- Identify the respective **data flows** informing the system about the **occurrence of the events**.

E

Examples of events from a early-warning system for mining:
- Sensors provide measurements
- User initializes or updates thresholds
- User requests detailed information on a sensor
- User acknowledges an alert
- Time for periodical report of compressed data
- Time for periodical daily or shift protocol

2. Event-based Partitioning (4)

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

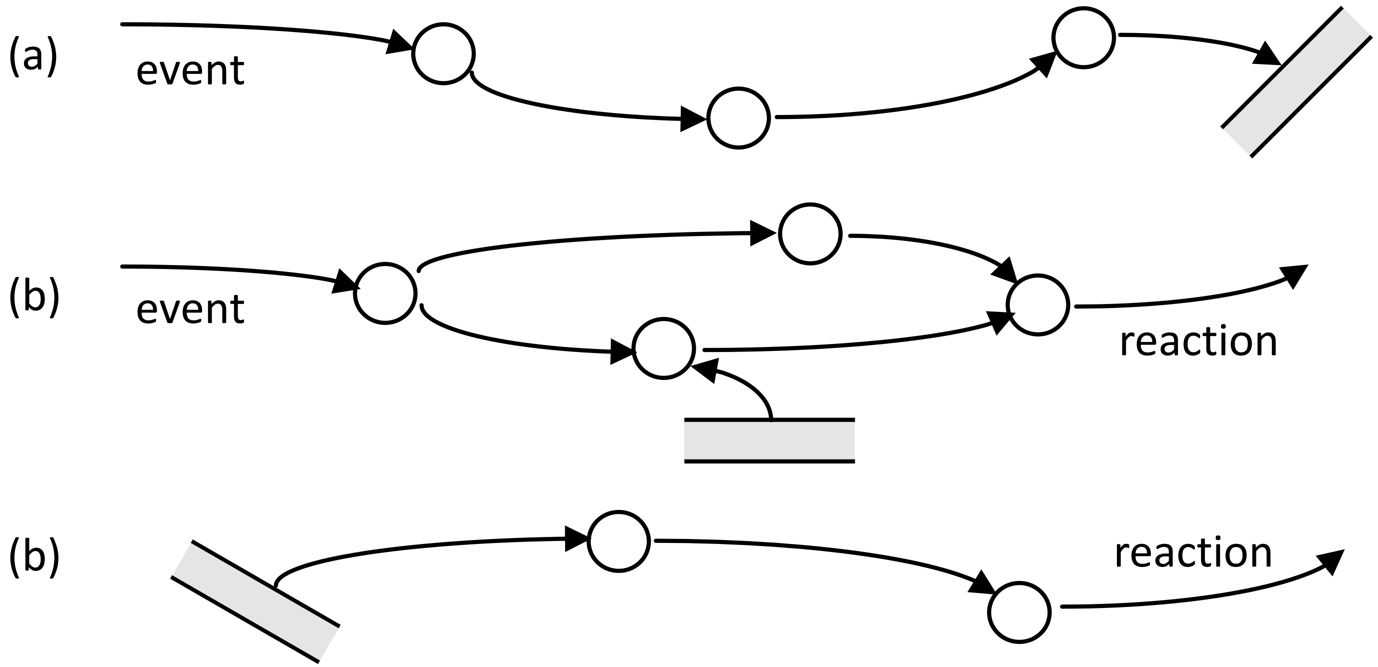
Step 4: Model event scenarios

Trace the data flows connected to an event, until

(a) a **data store** is reached, or

(b) an **output (system response)** is delivered to the context.

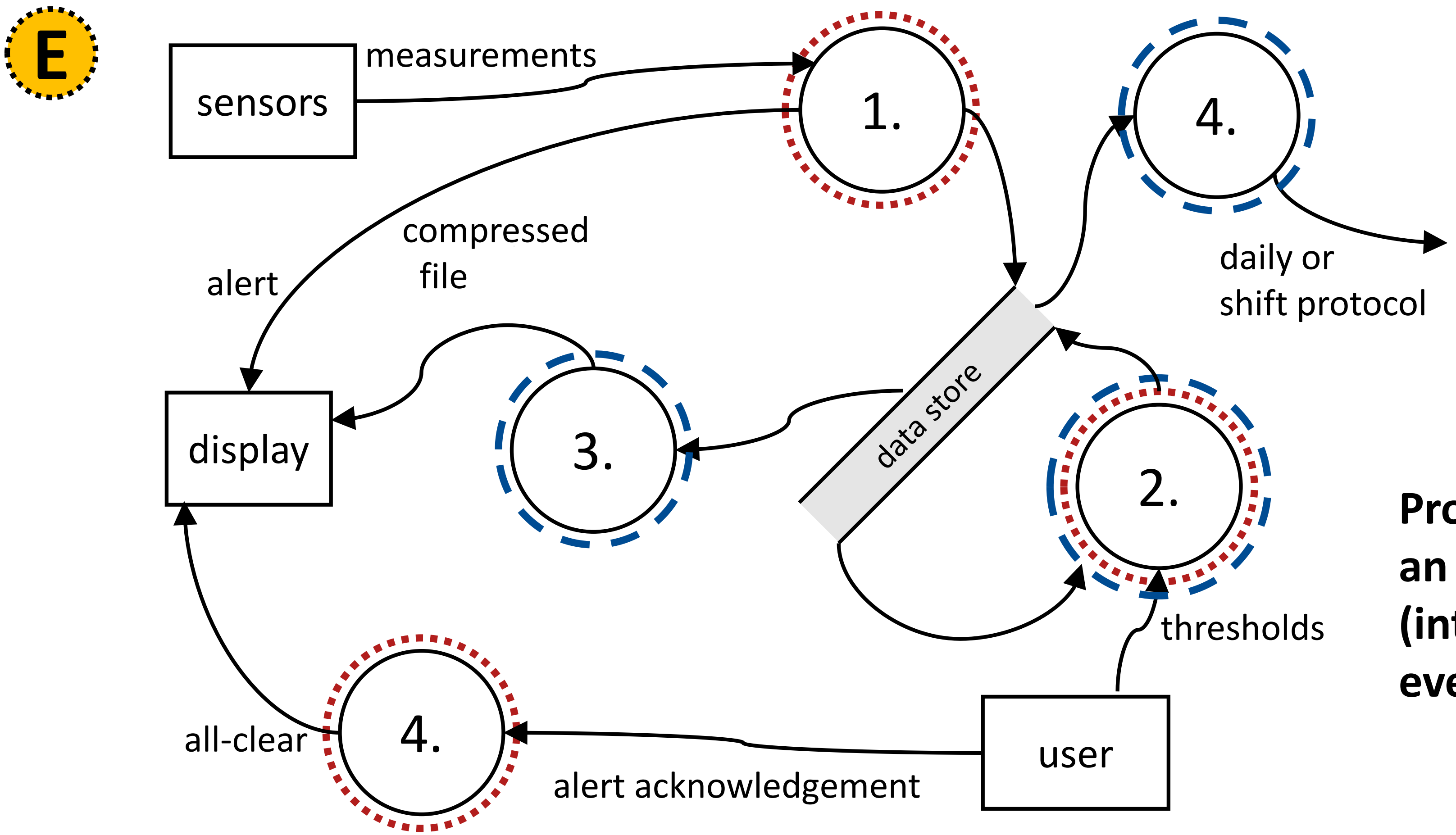
E



2. Event-based Partitioning (5)

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Step 5: Integration and Hierarchization



Reactions to temporal events

Reactions to external events

Process 2 reacts to an external AND (internal) a temporal event.

2. Event-based Partitioning (6)

Step 6: Completion of the Model

- Create a **data dictionary** describing relevant data stores and data flows.
- Write **mini specifications** for the functional primitives (processes not refined).

Advantages of the Approach:

- Widely used and well **established approach** (independent of an application domain).
- Based on events, **scenarios can easily be created**.
  Examples for what happens after the occurrence of an event.
- Supports **discussions** with users and **thinking in terms of relations** between functions.

Disadvantages of the Approach:

- Does not necessarily result in appropriate **data store structures**.
- In case of large-scale systems, the problem of **diagram partitioning** is shifted to **event bundling**.

3. Jigsaw Puzzle (1)

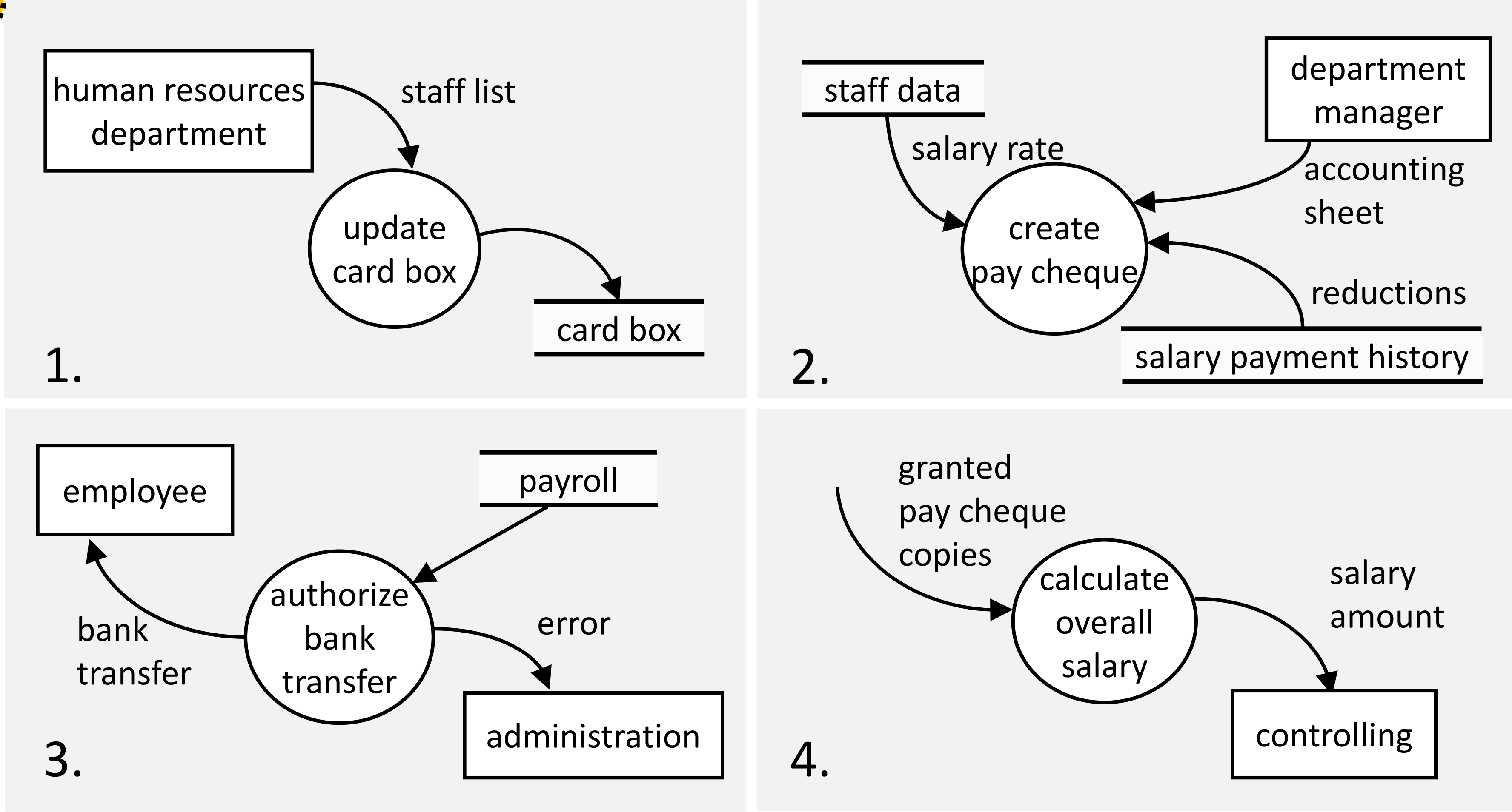
- **Function-centered** approach:
  Focus on separate functions, create respective diagram fragments (**puzzle pieces**).

- Method outline:
  - **Identify** functions and function sequences.
  - **Model** respective **processes** with **inputs**, **outputs** and **data stores**.
  - **Identify** connections from the **interfaces** of the functions.
  - **Consolidate** the **diagrams**.

3. Jigsaw Puzzle (2)

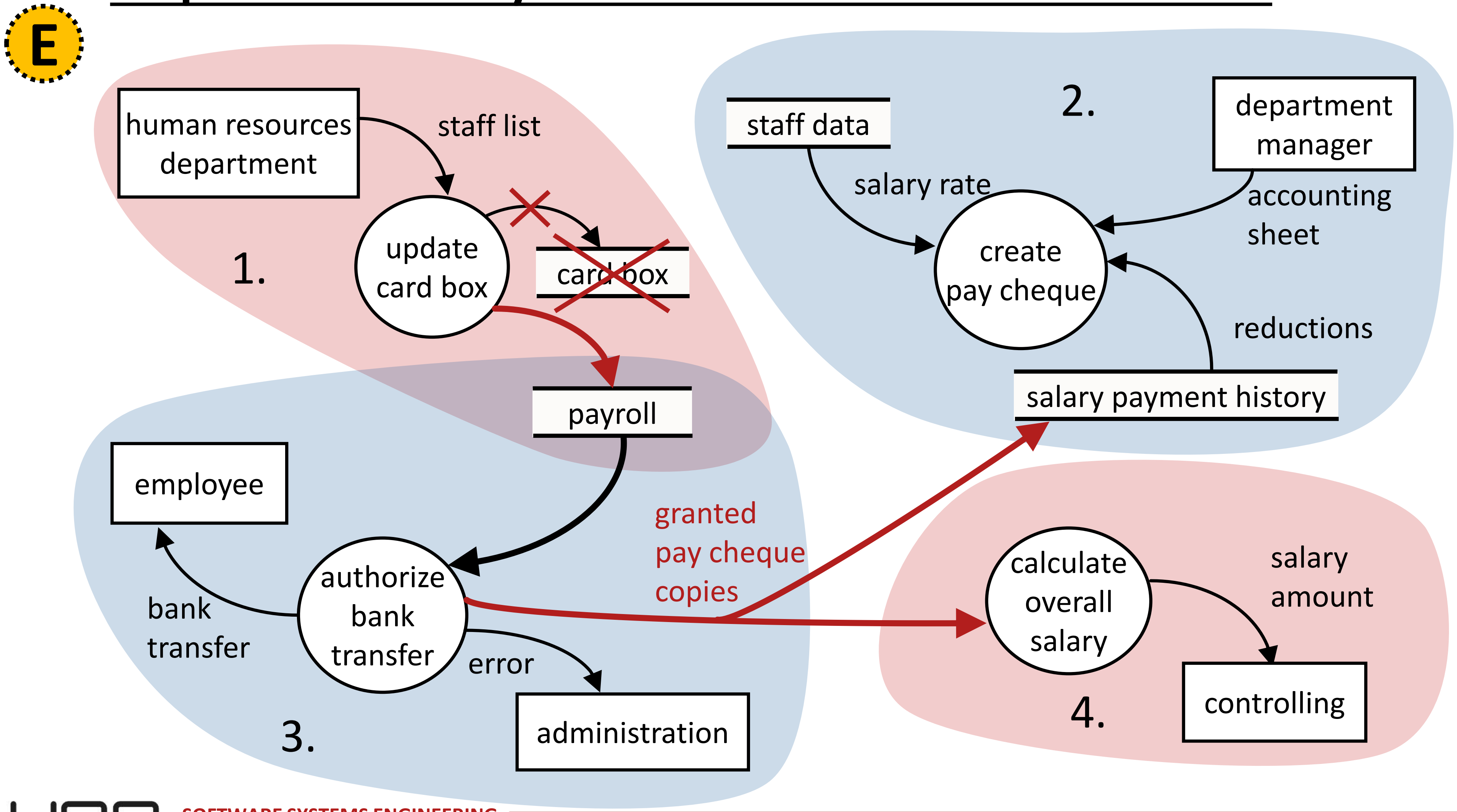
E

Step 1 & 2: Identify functions and model respective DFDs



3. Jigsaw Puzzle (3)

Step 3 & 4: Identify interfaces and consolidate model



3. Jigsaw Puzzle (4)

Advantages of the Approach:

- **Intuitive** and easily understandable.
- Quick **partial success**.

Disadvantages of the Approach:

- **Consolidation** of the different parts is often difficult.
- Only successful when applied to **small problems**.
- In **conflict** with the **paradigm of data flows** (by focusing on functions without considering the relations between them).

4. Pursuit of Data (1)

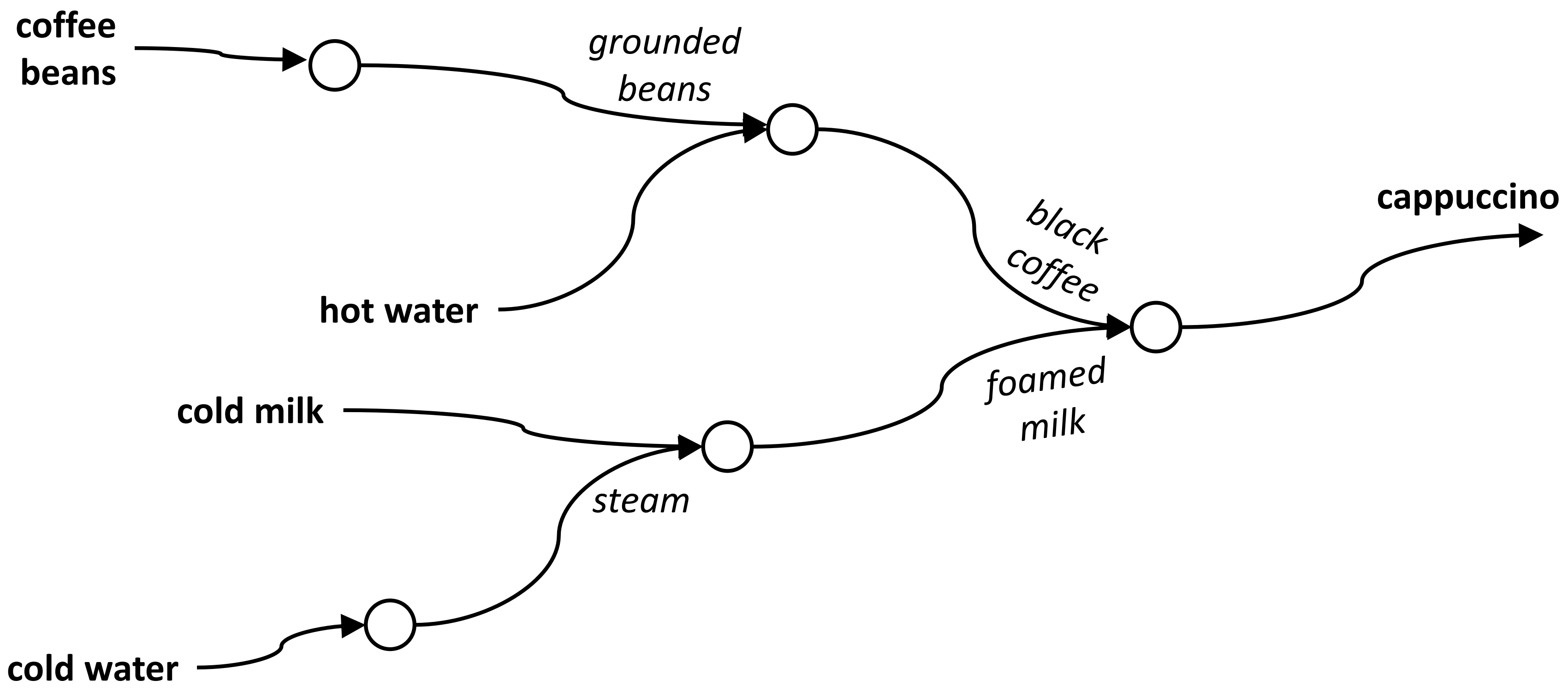
- **Focus on input and output data** and trace their transformation. Thereby functions are identified automatically.

- Method outline:
  1. **Identify** system **input data** and system **output data**.
  2. Identify **intermediate data**.
  3. **Model** the flow between the data identified (e.g. between input and intermediate data or output data and intermediate data)
  4. Introduce “connectors” between the flows (initial processes)
  5. **Assign** reasonable, non-abstract names to processes.

4. Pursuit of Data (2)

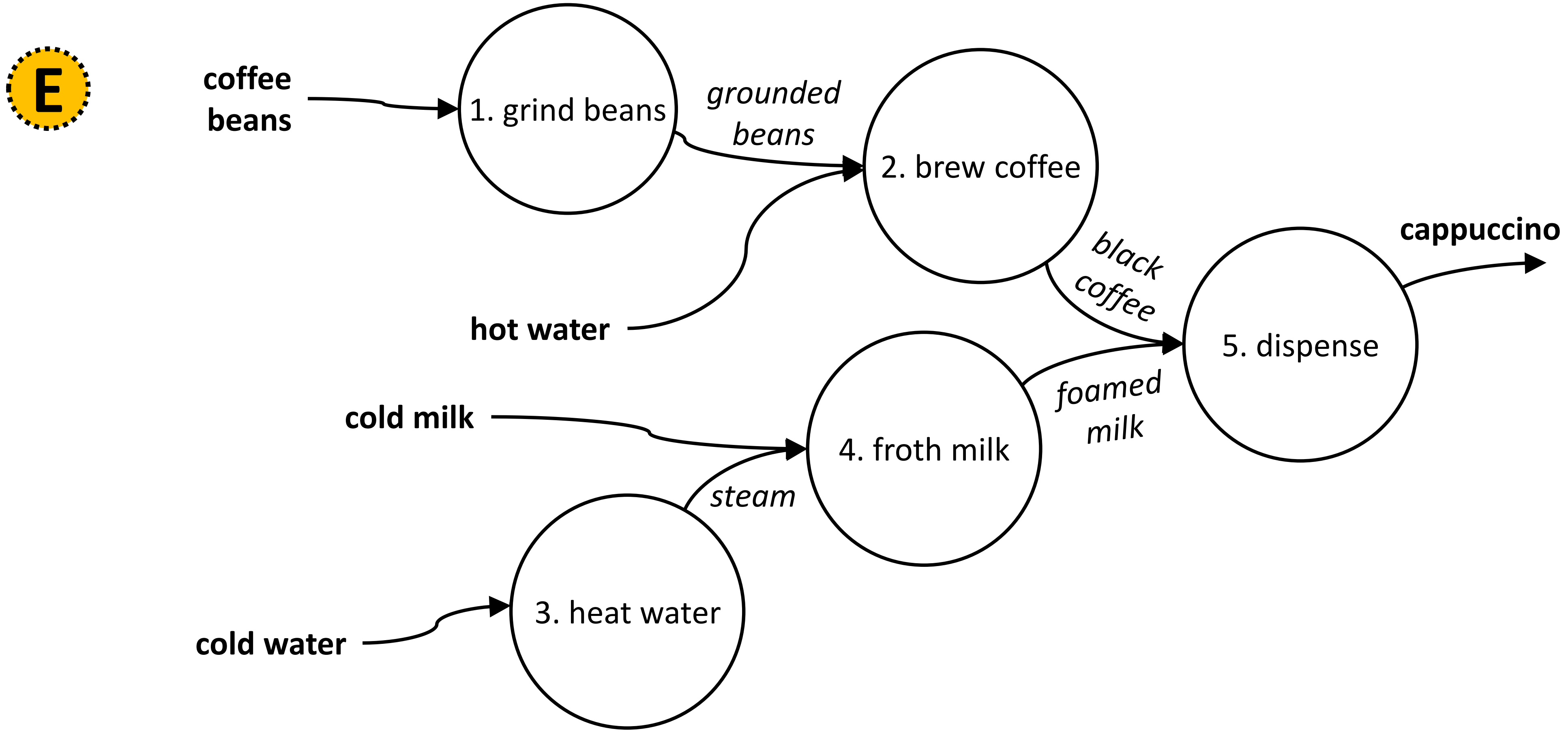
Step 1 - 3: Identify input, output and intermediate data and model the flow between the data

E



4. Pursuit of Data (3)

Step 4 - 5: Introduce connectors and assign names to them



4. Pursuit of Data (4)

Advantages of the Approach:

- Successful on systems that are mainly characterized by **transformations**.
- Supports thinking in terms of **data flows**.
- Supports distinguishing **essential functions** from non-essential ones.

Disadvantages of the Approach:

- **Intermediate data structures** are often not explicitly modelled and visible.
- Difficult to apply on **large-scale systems**.
- Results in large and **complex diagrams**.

Summary

- Reading data flow diagrams is easy; creating them is much more complicated!
- Data flow diagrams should be easy to comprehend.
  - Use appropriated names for data flows, stores and processes.
  - Keep the data flow diagram size appropriate! Avoid too complicated interfaces for processes and too many model elements for diagrams.
- Pay attention to identify typical indications of modelling errors!
  - Among others, avoid modelling control flows and “jumping” data, read-only and write-only memories etc.
- Use suitable technique/method to support the creation of data flow diagrams:
  - Context delineation
  - Event-based partitioning
  - Jigsaw puzzle approach
  - Pursuit of data
- Event-based partitioning has proved to yield the best results in practice.

Literature

[DeMarco 1979]	T. DeMarco: Structured Analysis and System Specification. Yourdon Press, Eaglewood Cliffs, 1979.
[McMenamin and Palmer 1984]	S. M. McMenamin, J. F. Palmer: Essential Systems Analysis. Prentice Hall, London, 1984.
[Yourdon 1989]	E. Yourdon: Modern Structured Analysis. Prentice Hall, Englewood Cliffs, 1989.
[Robertson and Robertson 1998]	J. Robertson, S. Robertson: Complete Systems Analysis. Dorset House Publishing, 1998.
[Hatley et al. 2000]	D. Hatley, P. Hruschka, I. Pirbhai: Process for System Architecture and Requirements Engineering. Dorset House, New York, 2000.
[Yourdon 2006]	E. Yourdon: Just Enough Structured Analysis. 2006.

Literature for Further Reading

[Ross and Schoman 1977] D. T. Ross, K. E. Schoman: Structured Analysis for Requirements Definition. IEEE Transactions on Software Engineering, Vol. 3, No. 1, 1977, pp. 6-15.

[Raasch 1992] J. Raasch: Systementwicklung mit strukturierten Methoden. Hanser Verlag, 1992.

[Robertson and Robertson 1996] J. Robertson, S. Robertson: Vollständige Systemanalyse. Hanser Verlag, 1996 (dt. Übersetzung).

Image References

[1] Licensed by http://www.iconshock.com/

[2] Provided by Microsoft Office

Legend

D Definition

E Example

Requirements Engineering & Management

Vielen Dank für Ihre Aufmerksamkeit