Requirements Engineering & Management

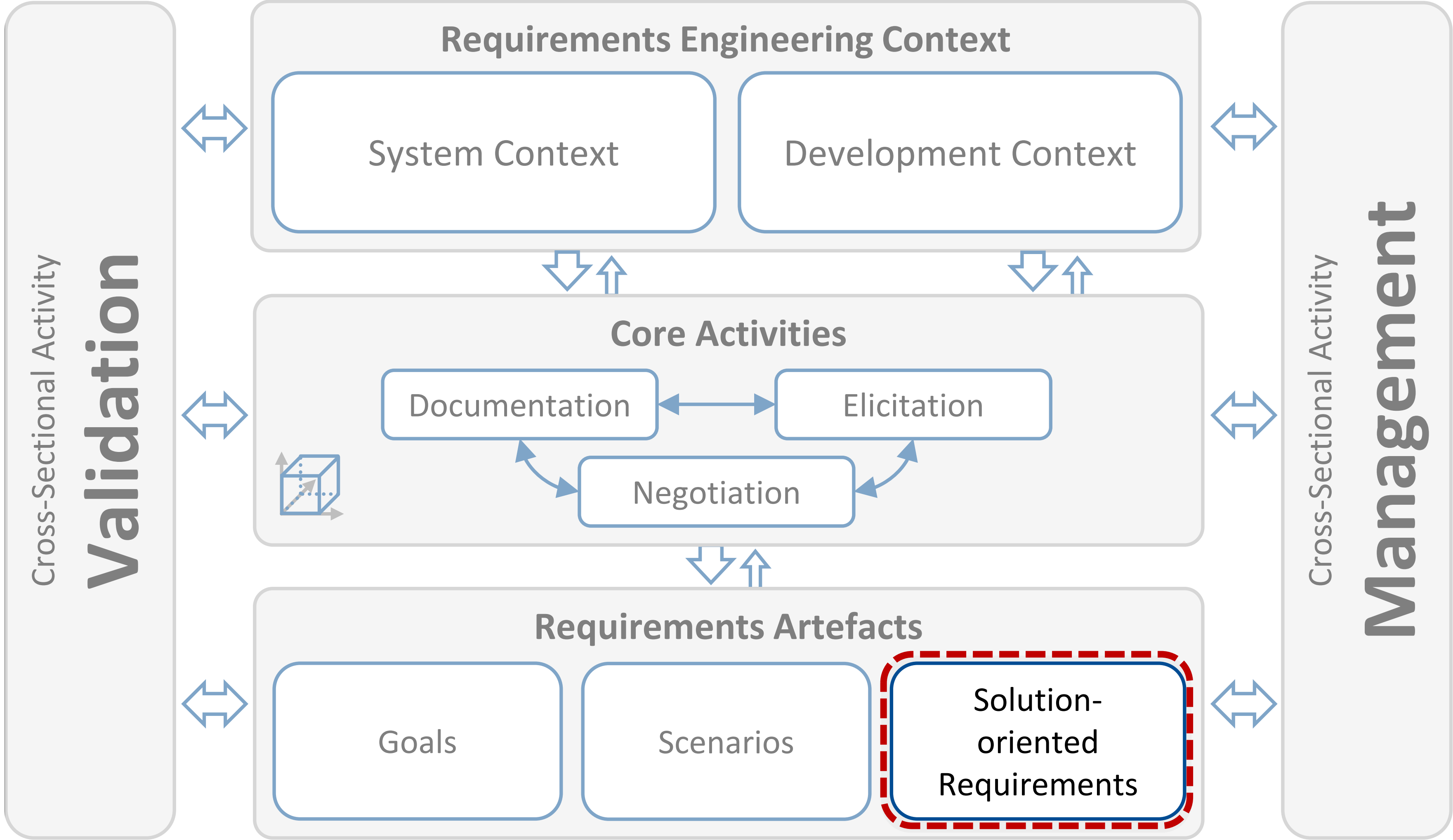
Solution-Oriented Requirements – Functional Modelling I

Prof. Dr. Klaus Pohl

PALUNO
The Ruhr Institute for Software Technology

Framework for Requirements Engineering

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken



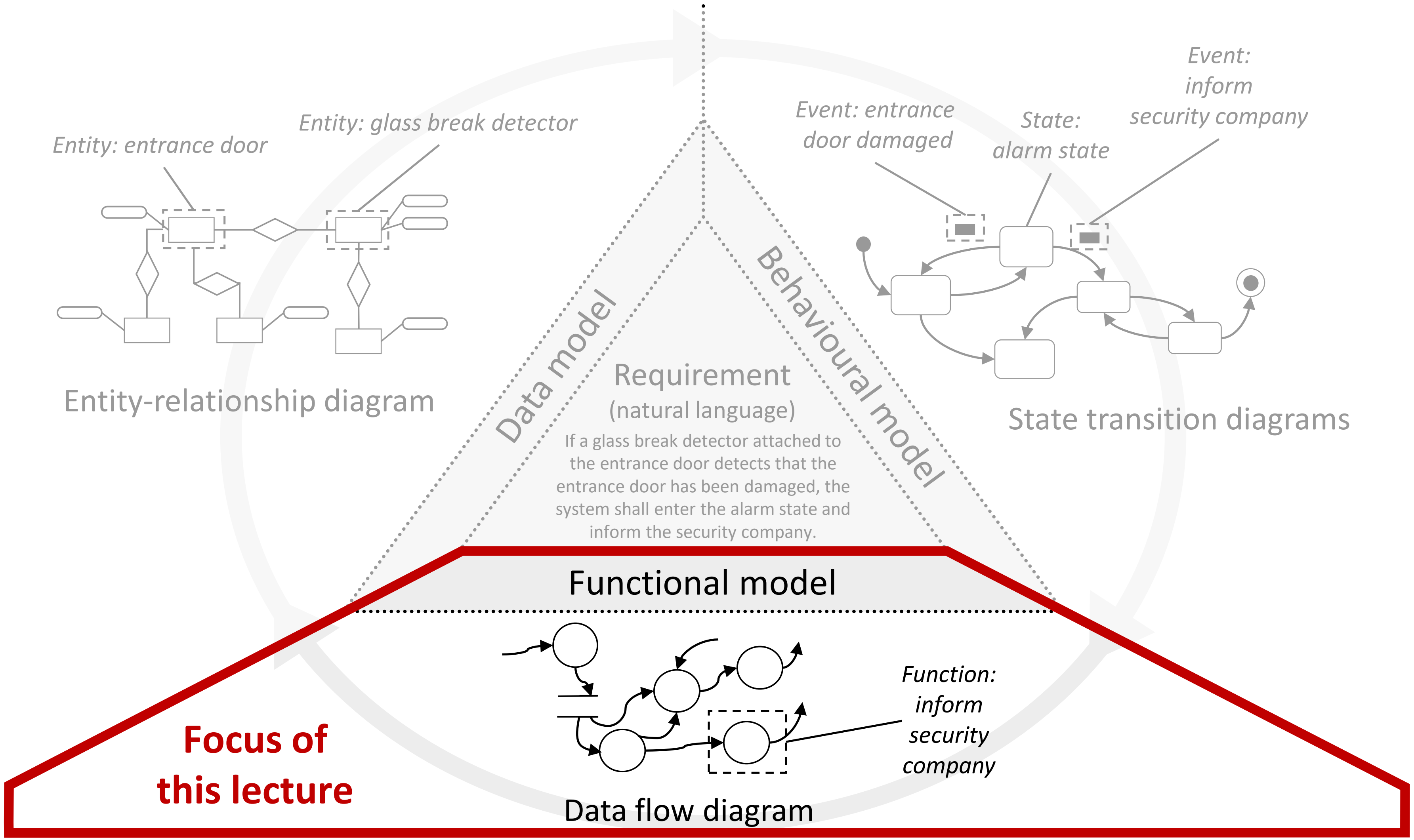
Agenda

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

1. Fundamentals of Functional Modelling
2. Data Flow vs. Control Flow
3. Data Flow Diagrams
4. Structured Analysis Overview
5. Data Dictionaries
6. Hierarchization of Data Flow Diagrams
7. Mini Specifications

1. Fundamentals of Functional Modelling

Model-based Documentation in the Three Perspectives



Functional Modelling Concepts and Abstractions (1)

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

- **Functional modelling languages**:

  Definition of **modelling constructs and rules** for documenting processes (functions), the manipulation of data by processes and the input-output relationships (data flows) among the processes.

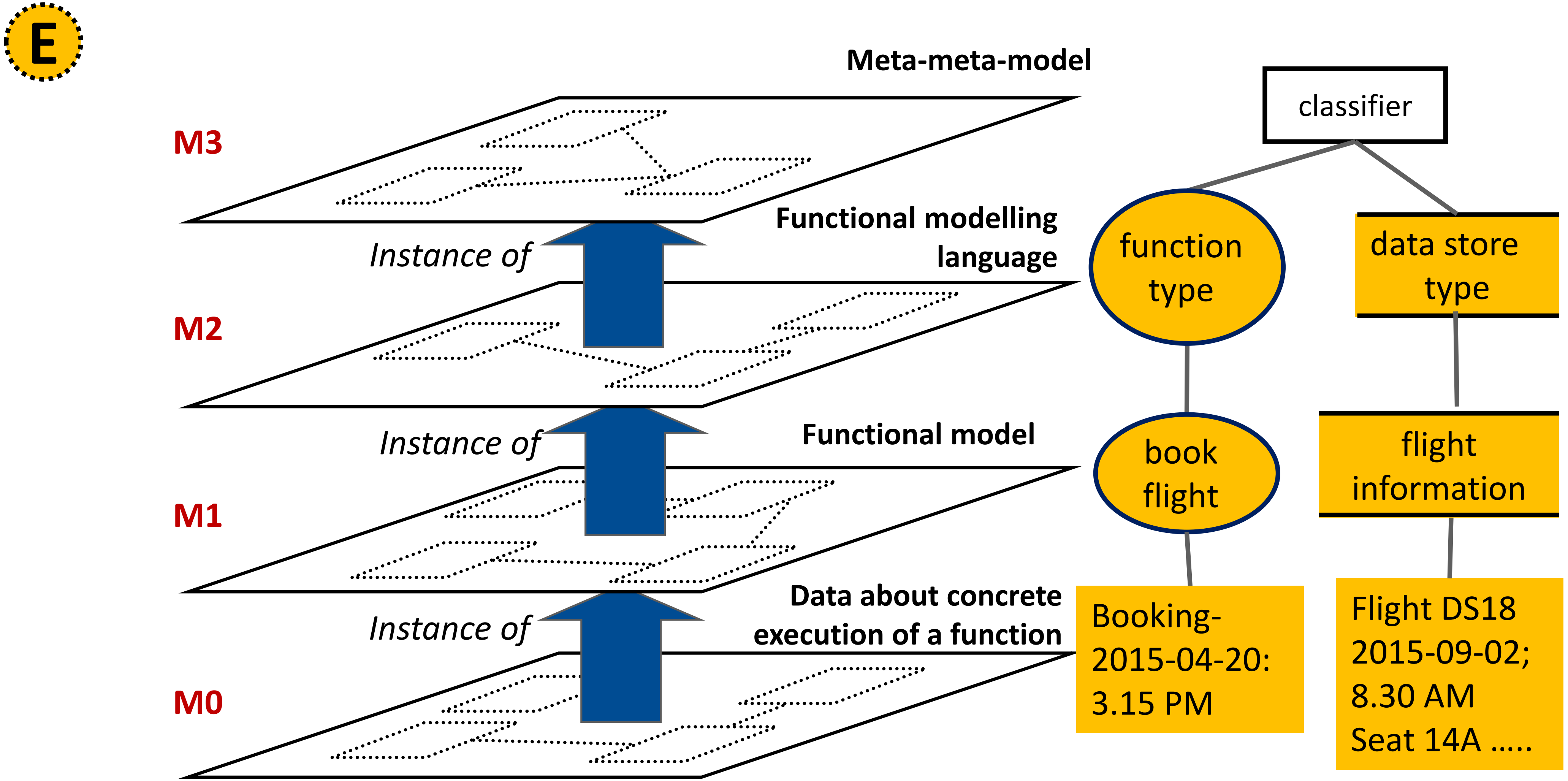
- **Functional model**:

  Definition of **functions** (types), **data flows** (types) between the functions and **data stores** (types) of **a system**.

- **Functional model instance**:

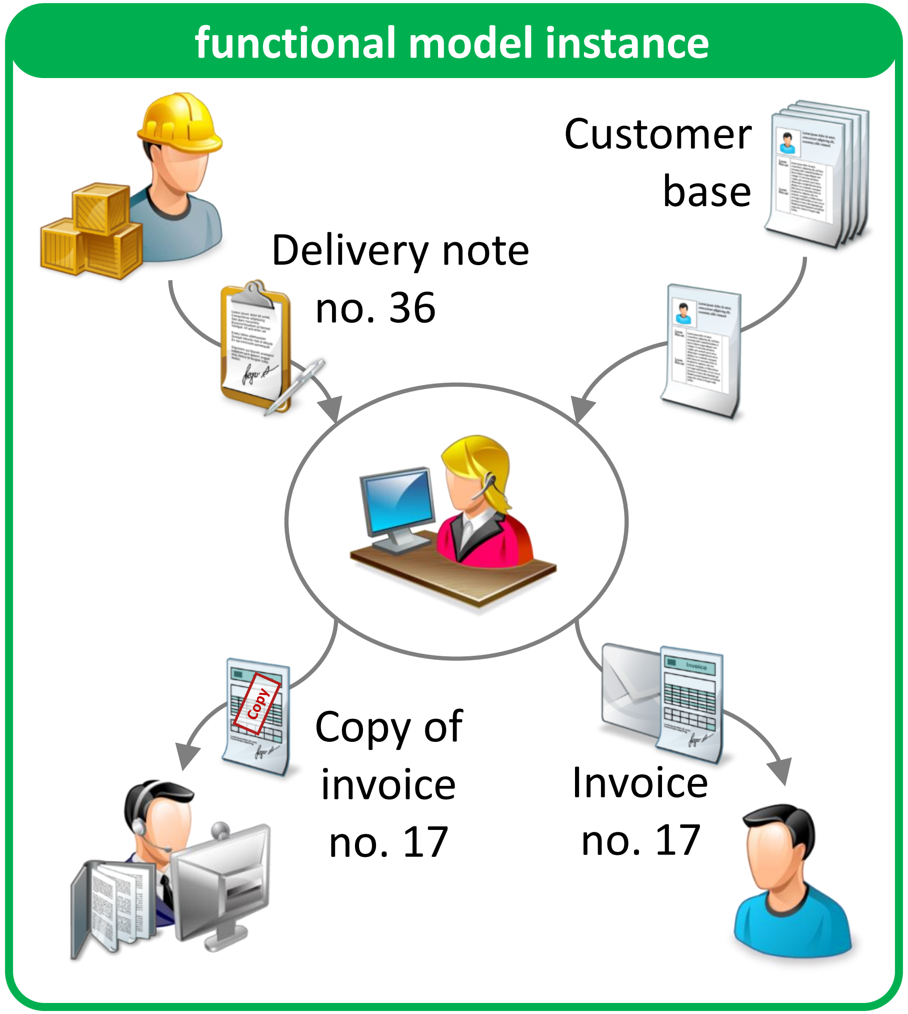
  Data about a **concrete execution** of a **function**, **concrete interactions** executed and **concrete data produced/consumed** during the execution and stored in a data store.

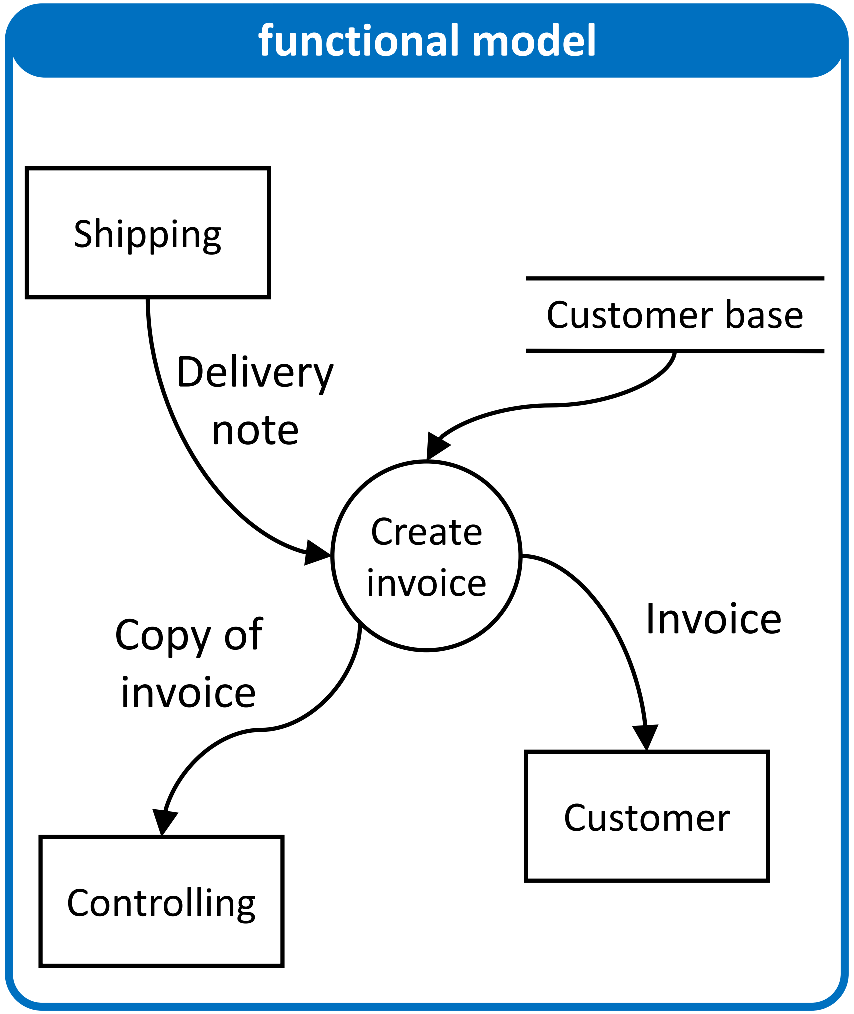
Four Modelling Layers: Functional Modelling



Modeling Layer hierarchy according to [OMG 2004]

Functional Modelling Concepts and Abstractions (2)





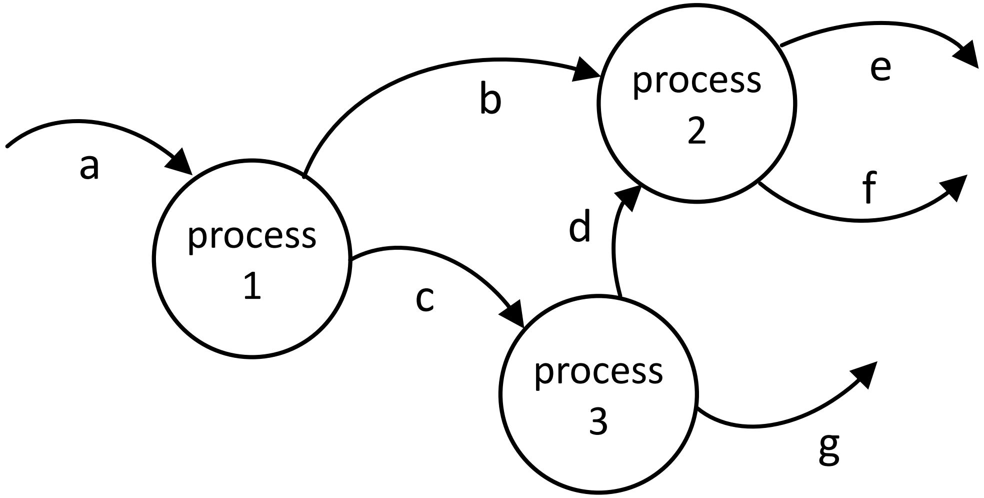
All icons from [1]

2. Data Flow vs. Control Flow

Properties of Data Flows

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

- Data flows describe **pipelines between processes**. (see [DeMarco 1979], p. 63)
- Pipelines transmit **packages of information of known composition**. (see [DeMarco 1979], p. 342)
- Packages of information contain **material or immaterial objects** (e.g. sheets of paper, payment information).



No explicit information about **process sequences**; all process can, in principle, be **active at the same time**.

Properties of Control Flows

- Control flows define **process execution sequences**, as well as **events and conditions** under which processes are executed.
- A control flow symbolizes the **passing of a trigger** from one activity to the next.
- **Note**: A control flow typically **includes a flow of information**, e.g., an event includes its source and time of occurrence.

Only one process can be active at one point in time, e.g., process 2 can execute only if process 1 is completed.

3. Data Flow Diagrams

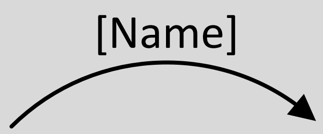
Modelling Construct: Data Flow

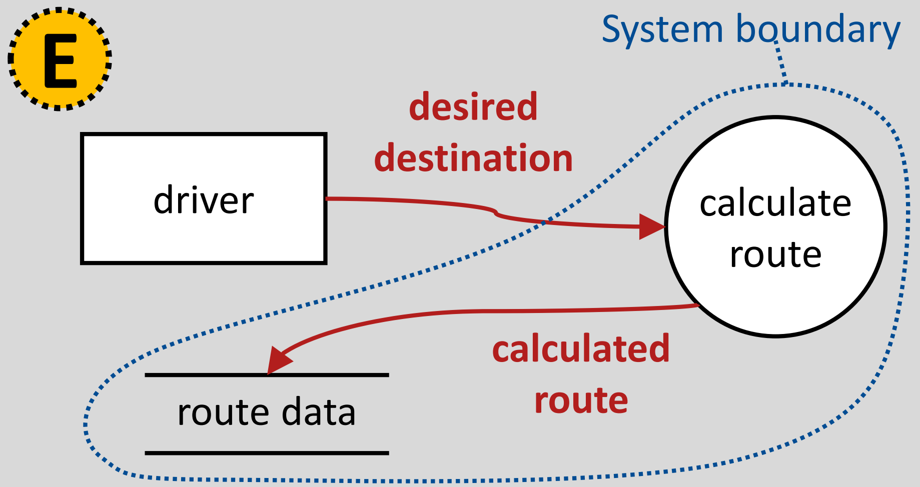
UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Data Flows:

- Describe the **transportation of information packages** of known composition.
  Information packages may contain material or immaterial objects (e.g., information, products, energy, etc.).
- Can be defined between:
  - Two processes.
  - A process and a data store.
  - A source/sink and a process.

Notation





Modelling Construct: Data Store

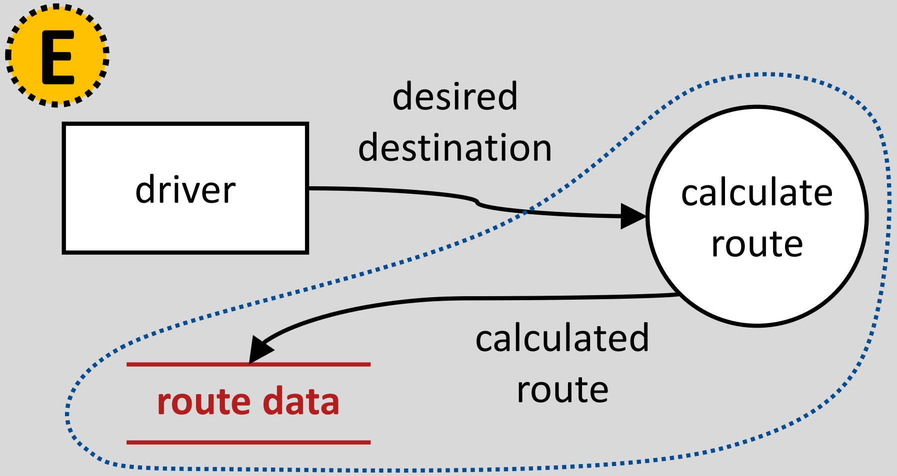
UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

A Data Store:

- Defines a (physical or technical) **repository of data** (e.g., files, folder, etc.).
- Contains "data at rest".
- **Persistently** records data produced by the system itself or received from the system context.

A process can **access** or **retrieve** the data in a data store.

Notation



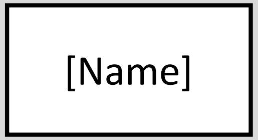
Modelling Construct: Source, Sink

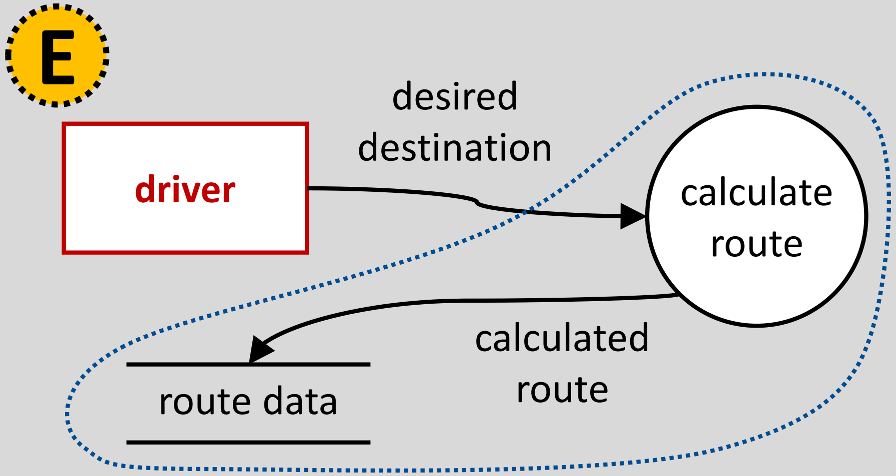
UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Sources and Sinks:

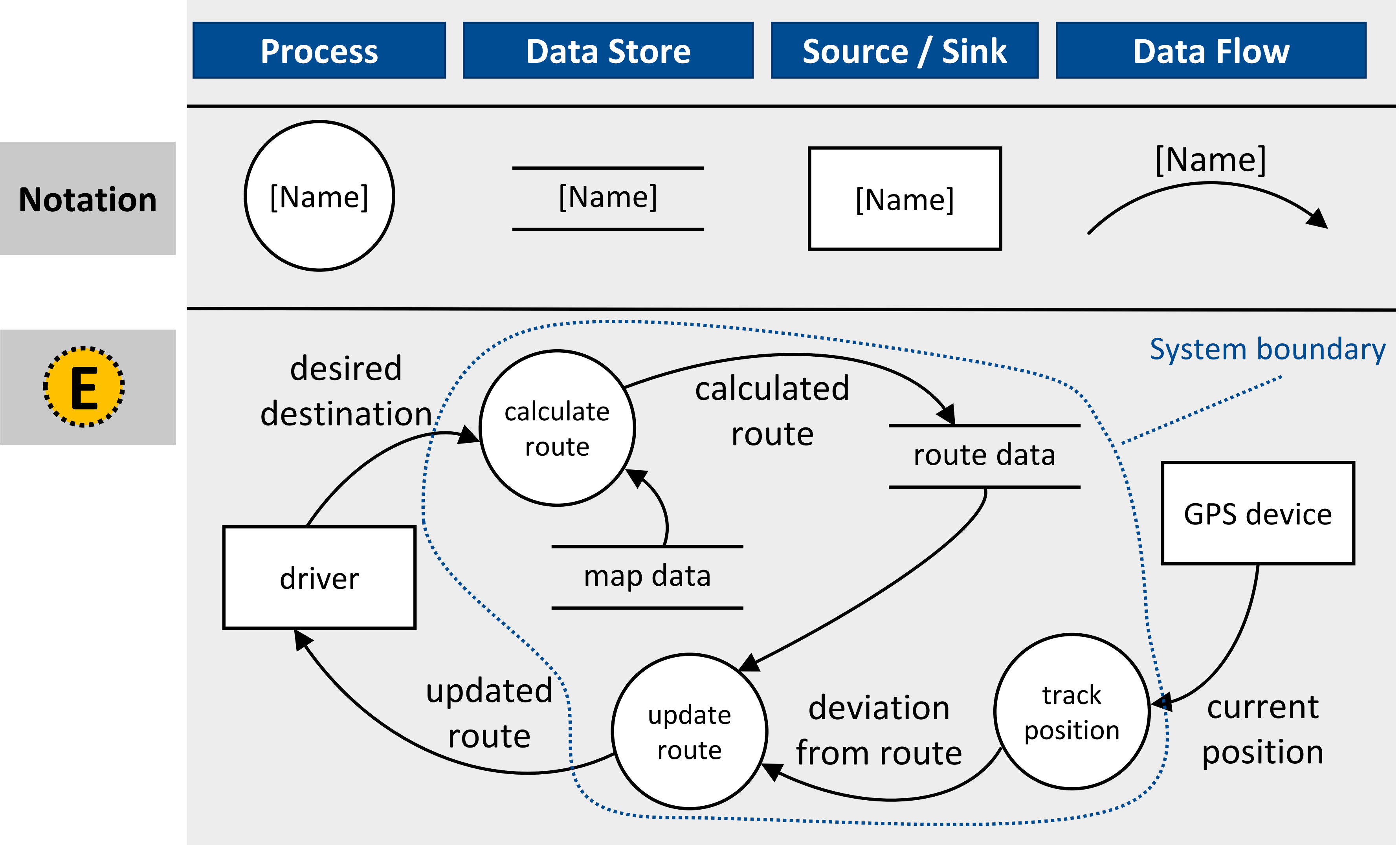
- Are **external objects** outside of the system boundary (e.g., persons, organizations, other systems), i.e. sources and sinks reside **in the context** of the system.
- Are **sender** of information packages to the system and/or **receiver** of information packages from the system.

Notation





Summary of Modelling Constructs



4. Structured Analysis Overview

Motivation for Structured Analysis

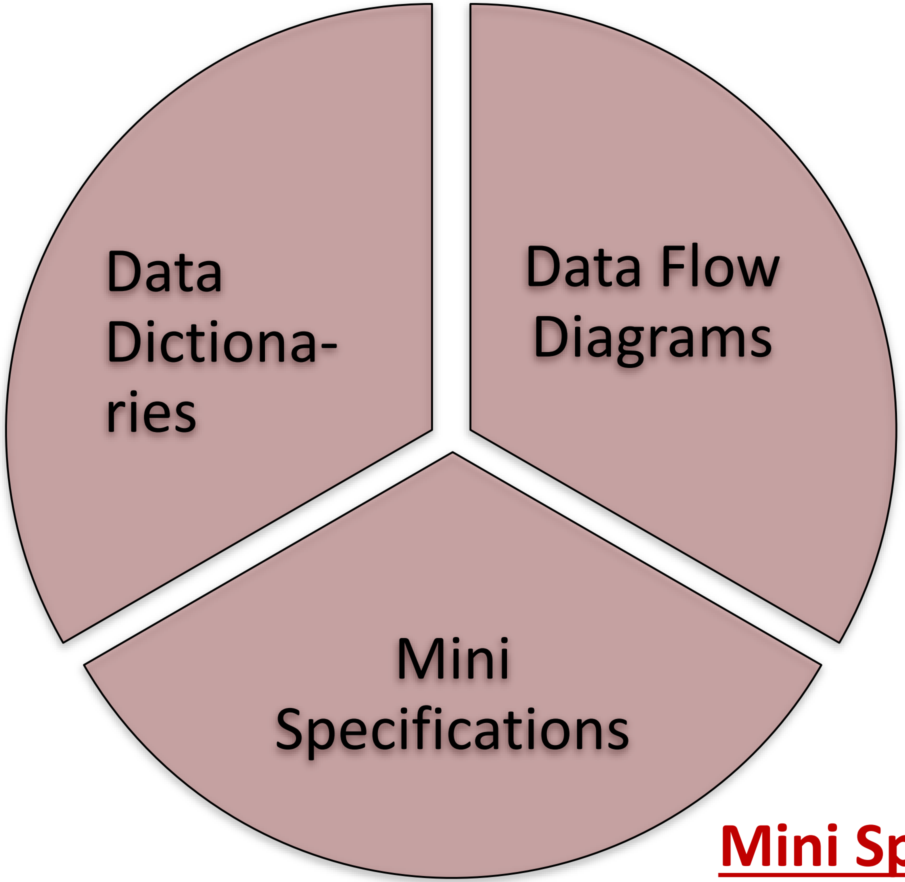
"Analysis is the study of a problem prior to taking some action." ([DeMarco 1979], p. 4)

- Primary outcome of a Structured Analysis is a structured specification document which: ([DeMarco 1979], p. 32)
  - is highly maintainable,
  - reduces complexity by means of effective partitioning techniques,
  - uses graphical representations rather than narrative text.
- Structured Analysis aims at supporting communication about a problem by structuring the problem (models of the problem) from abstract to detailed.

Three Components of Structured Analysis

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Data Dictionaries define the composition of the data in data stores and flows.



Data Flow Diagrams (DFD) define processes and data flows between processes and sources/sinks.

Mini Specifications define primitive functions.

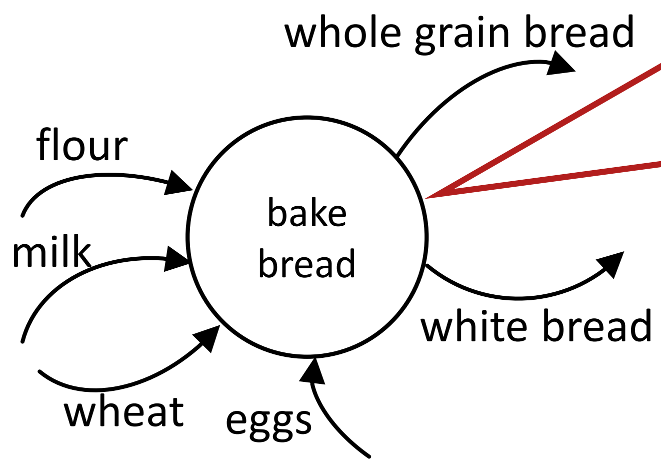
5. Data Dictonaries

Motivation

Data flow diagrams can be **ambiguous**, i.e., they can be interpreted differently

- by **different stakeholders** and/or
- at **different points in time**.

E



Ambiguous process “bake bread“:
- What is the relationship between the inputs and the outputs?
- When is which output produced?
- Are all inputs needed for both outputs?
  - If so, what is the difference between the outputs?
  - If not, which output requires which input?

Definition

D A data dictionary **defines** the **structure** of each **data flow** and **data store** in a Data Flow Diagram.

- The entries of a data dictionary are typically, defined in an EBNF (extended Backus-Naur Form) language.
- In the following we outline a language for defining entries of a data dictionary.
- Note: It is not the aim of a data dictionary to define the data structures used to realise the system.

Equivalence Operator, AND-Operator

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

- “+” means “and”.
  Composition of a data elements from other data elements.
- “=“ means “is equivalent to”.
  Definition of a data element in terms of other data elements.

Notation	E
+	x **=** a **+** b
=	x **is equivalent to the composition of** a **and** b
	name = given name + family name

Optional Operator

- "(..)" means "optional".
- Zero or one of the elements or expressions in brackets.

Notation

(…)

E

x = a + b **(+ c)**

x is equivalent to the composition of a and b, **and in some cases to the composition of a and b and c**

address = street + city **(+ country)**

Selection Operator

- „[…|…|…]“ means “either – or”.
- Choice of exactly one of several possible data elements.
- Exclusive choice options are separated by “|”.

Notation

[…|…|…]

E

x = a + **[b | c]**

x is equivalent to the composition of a and **(either b or c)**

sandwich = bread + **[cheese | meat]**

Iteration Operator (1)

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

- „{…|…|…}" means "selection of".
- Repetitions of the data element in {…}.

Notation

{…|…|…}

E

x = **{ a | b | c }**

x is equivalent to **selection** (0 to N with N=no. of elements) of the data elements **a, b and c**

pet = **{dog | cat | bird}**

Iteration Operator (2)

- Number of repetition of data elements can be constrained with lower (x) and upper boundaries (y).
- Lower boundary can be defined without an upper boundary and vice versa.

Notation

x{…|…|…}y

E

x = **1{ a | b | c }2**

x is equivalent to **at least one** but **at most two** selections of the data elements **a, b and c**

pet = **1{dog | cat | bird}2**

Data Primitive

- Primitive data, which is not further decomposed, is defined by using `”´.
- Typically defines well understood or atomic information packages.

Notation

“…”

E

x = “0”

The terms **“0“** is **well-known** and **not further defined or refined**

Drinks = {“water“|”mango juice”| …}

Comment

- Textual description or explanation.
- Can be used in case the data is well understood, but not necessarily an atomic information packages.
- Can be used to reference where the data is defined (e.g., reference to another supplementary document).

Notation

...

E

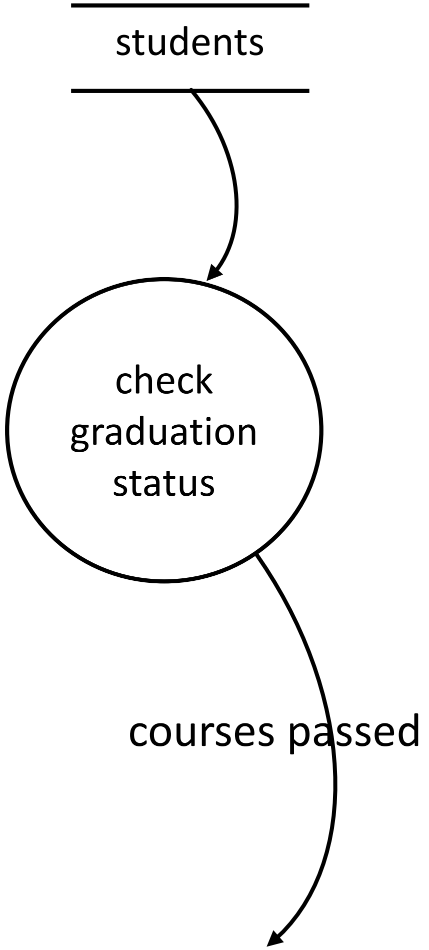
x = a + b *description*

description is a **comment**

name = given name + family name *family name always written last*

Data Dictionaries

Example



...
students = {student}
student = studentID
+ degree program
+ degree progress
+ student profile
degree program = degree name
+ {course to take + course description}
degree progress = {course passed} + {course failed}
student profile = name
+ {address}
+ date of registration
student since ...
address = street + zip + city + country + address type
address type = [“home“ | “parents“ | “emergency contact“]
...

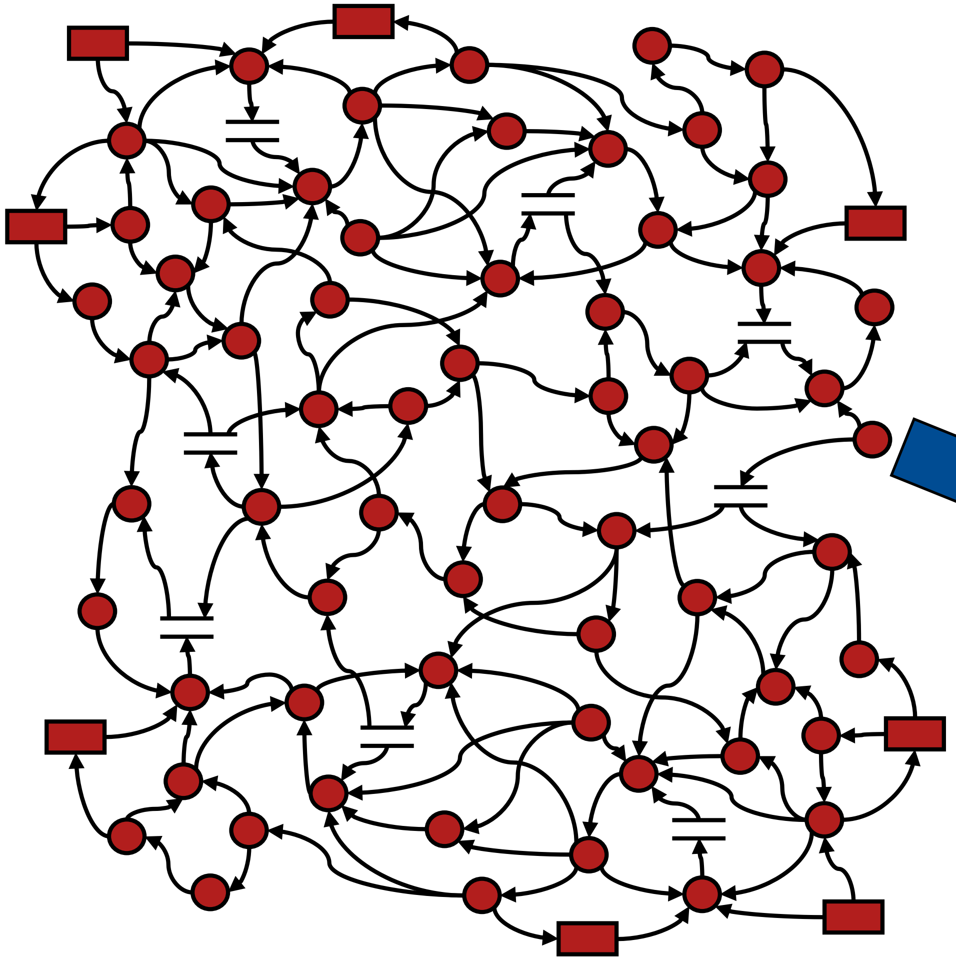
Hints

- Avoid **redundancies**.
- **Reuse** already **defined data elements**, where possible.
- Aim to **adopt terms known** to the stakeholders.
- Refine data element definition only if it is **necessary to achieve a better understanding**.
- Do not include **circular definitions** of data elements.
- Define **synonyms** to solve naming conflicts.
- Stop **defining terms further** when your user **clearly understands** their meaning.

based on [DeMarco 1979], Chapters 12.2 12.3, 12.4, 12.5 and 12.6

6. Hierarchization of Data Flow Diagrams

Reduction of Complexity: Levelling



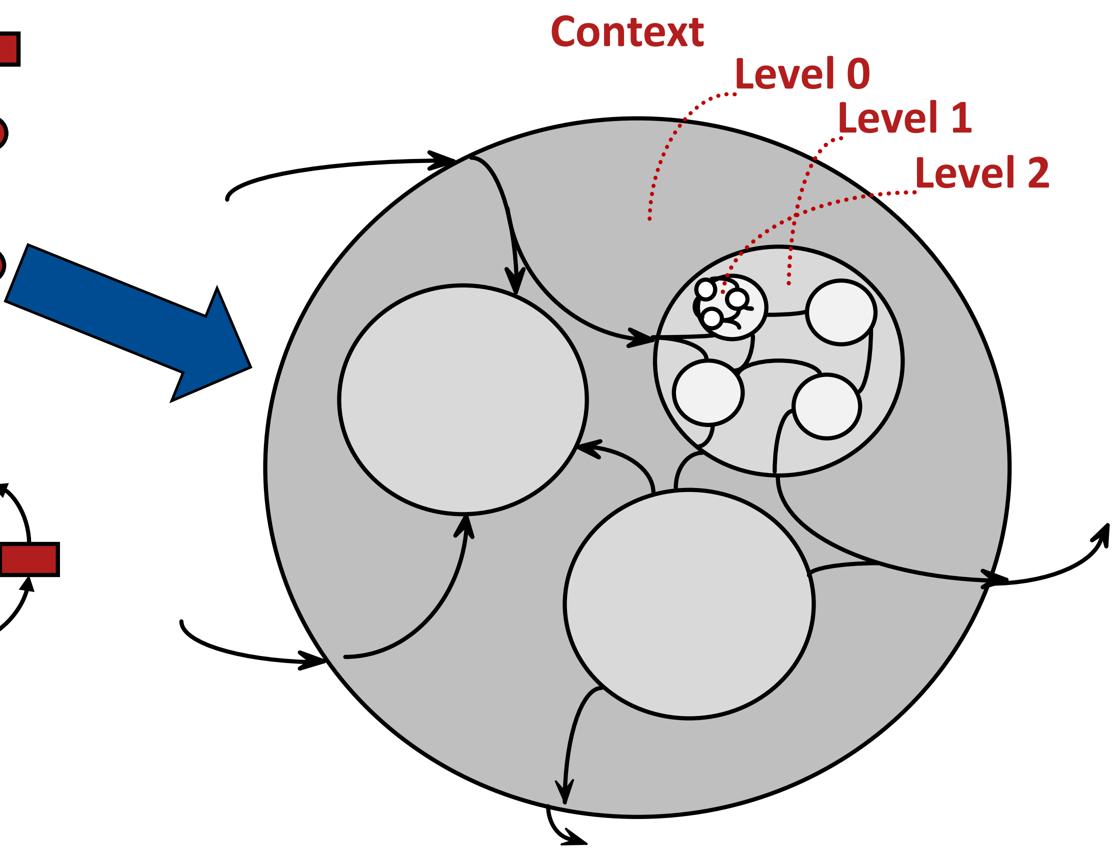
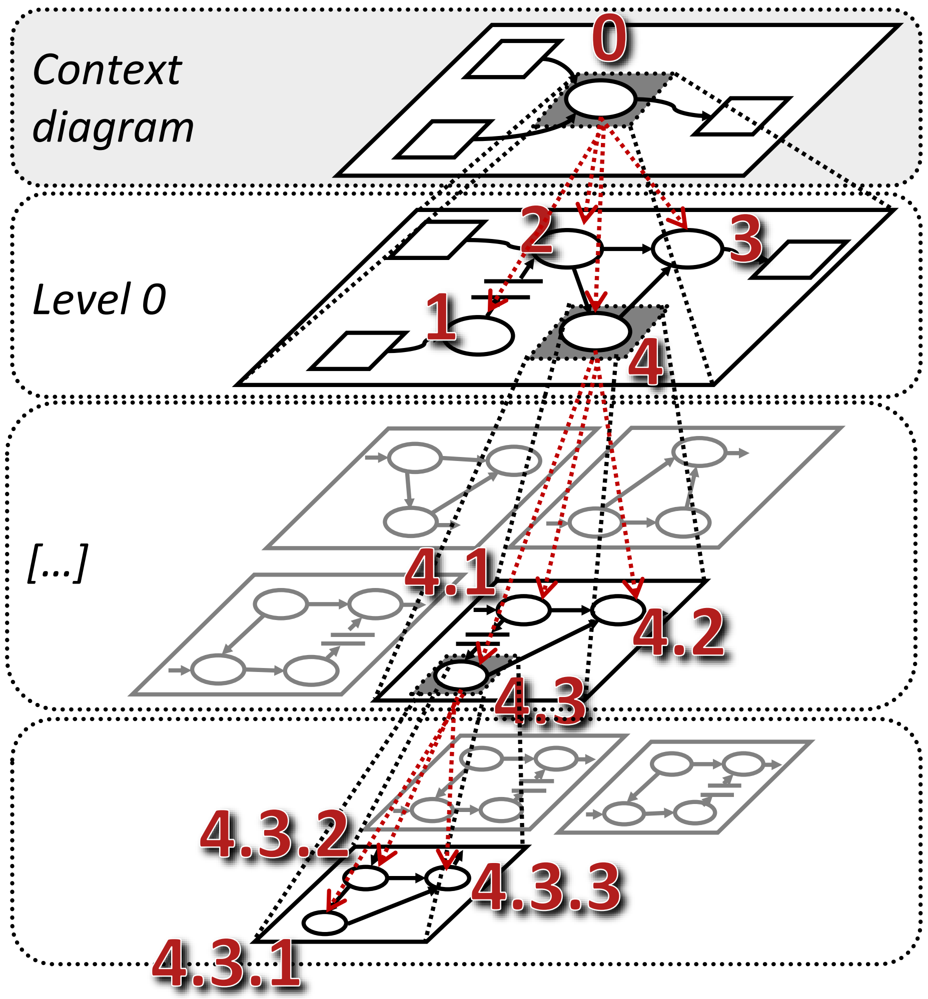


Illustration based on [McMenamin and Palmer 1984], p. 26

Data Flow Diagrams in Structured Analysis



- **Reduction** of a complex problem into a set of smaller ones that can be **analysed separately**.
- Data flow diagrams serve as the main tool for **partitioning the system** on different layers of abstraction.
- **Level of detail** increases on lower levels of the DFD hierarchy.
- Supports **top-down analysis**, i.e., the successive partitioning until **functional primitives** are identified.
- Supports **bottom-up analysis**, i.e., the successive **composition** of functional primitives to a **common superstructure**.

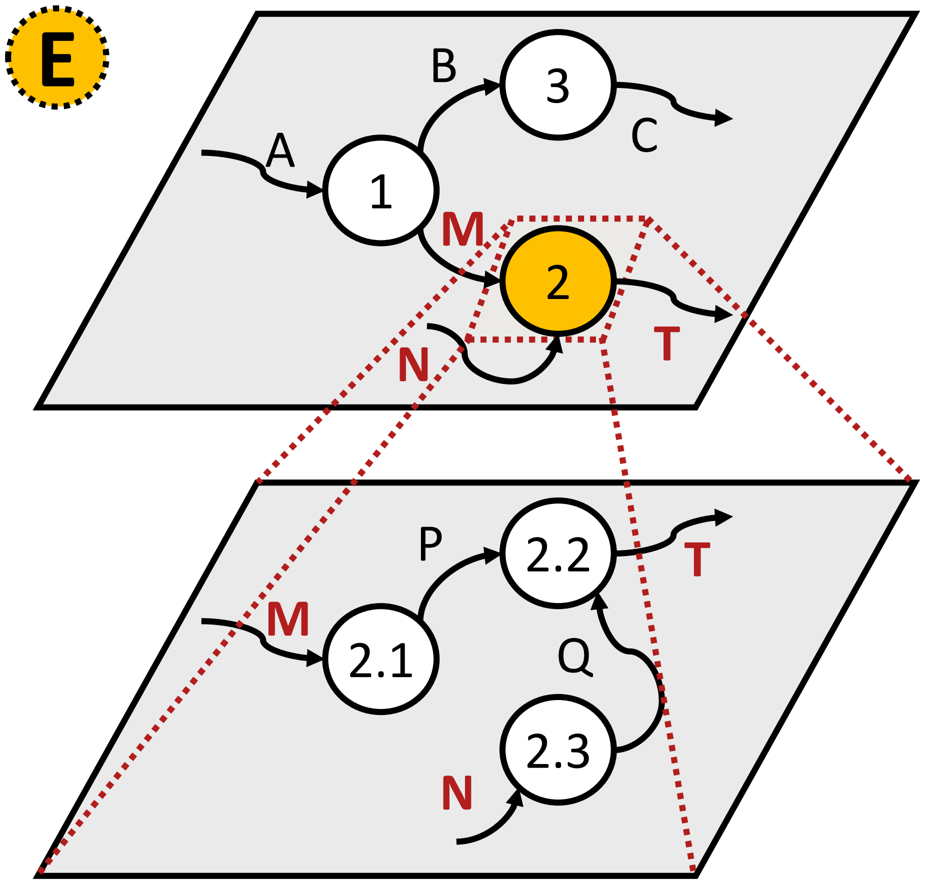
Balancing of Data Flow Diagrams

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

- Strict balancing rules for **assuring consistency between different levels** of DFDs in the DFD hierarchy.
- Three types of balancing:
  - **Visible balancing**
  - **Data dictionary balancing**
  - **Data store balancing**

Balancing of DFDs: Visible Balancing

- Every **input and output data** flow of the parent process node **must also be directly visible in the child data flow diagram**.
- **Consistency** is immediately visible and can be checked easily.



Balancing of DFDs: Data Dictionary Balancing

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

- The balance must **not necessarily be visible** in the data flow diagram.
- **Data flows** can be **split up**, if the data dictionary defines a **respective composition**.
- May involve **several levels of detail** in the data dictionary.

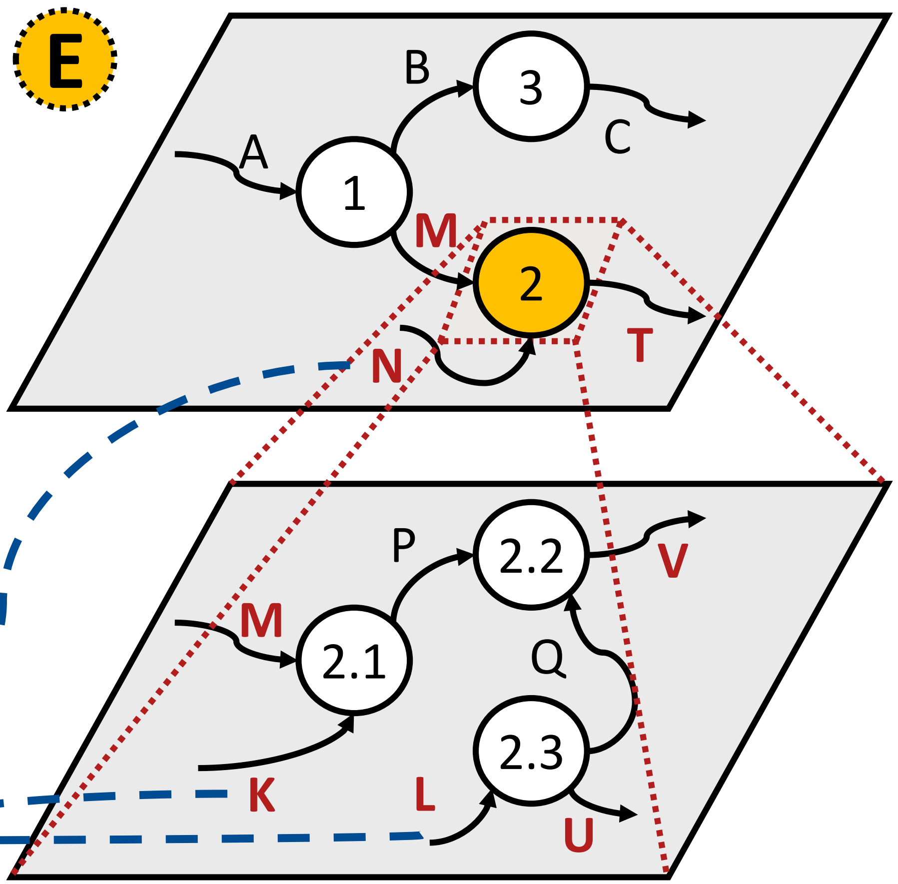
Data dictionary:

…

N = K + L

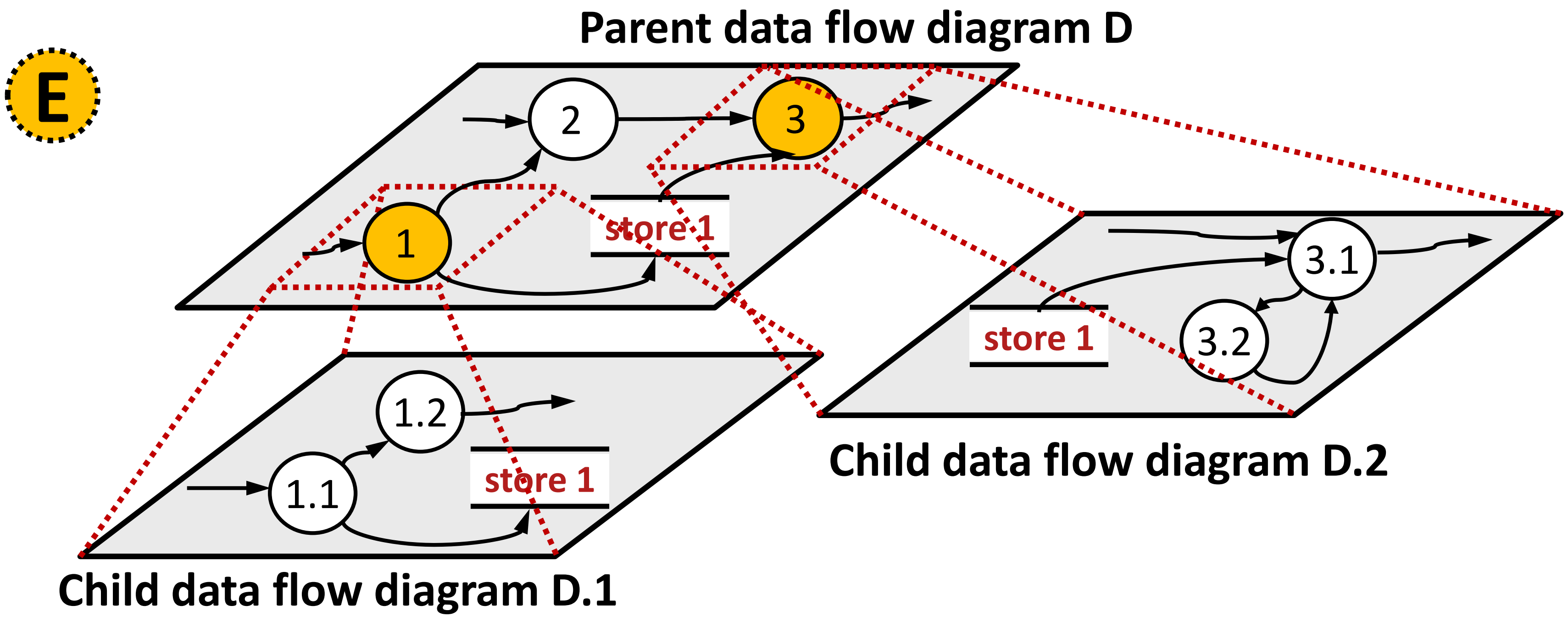
T = [U | V]

…



Balancing of DFDs: Visual Data Store Balancing

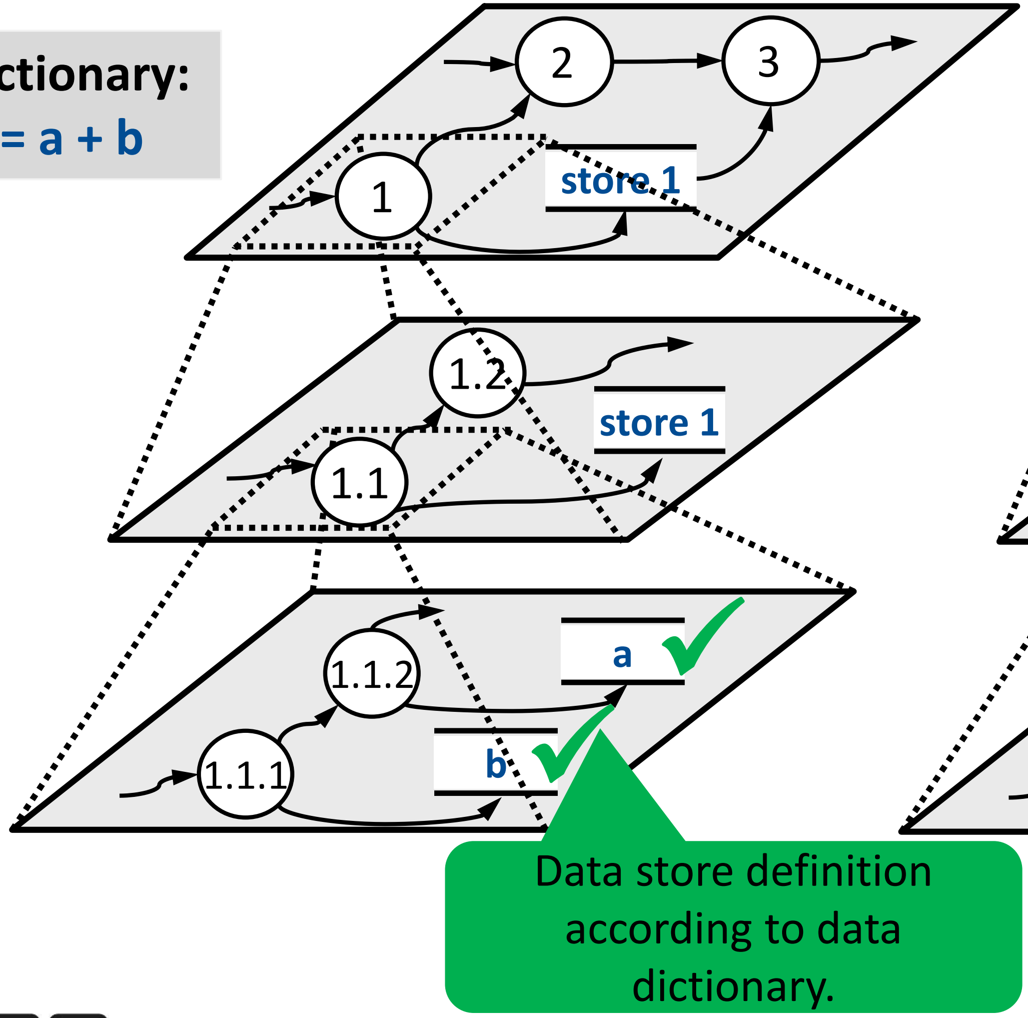
<u>All child data flow diagrams</u> have to define **<u>all read and write accesses</u>** for each data store defined at their parent data flow diagram.

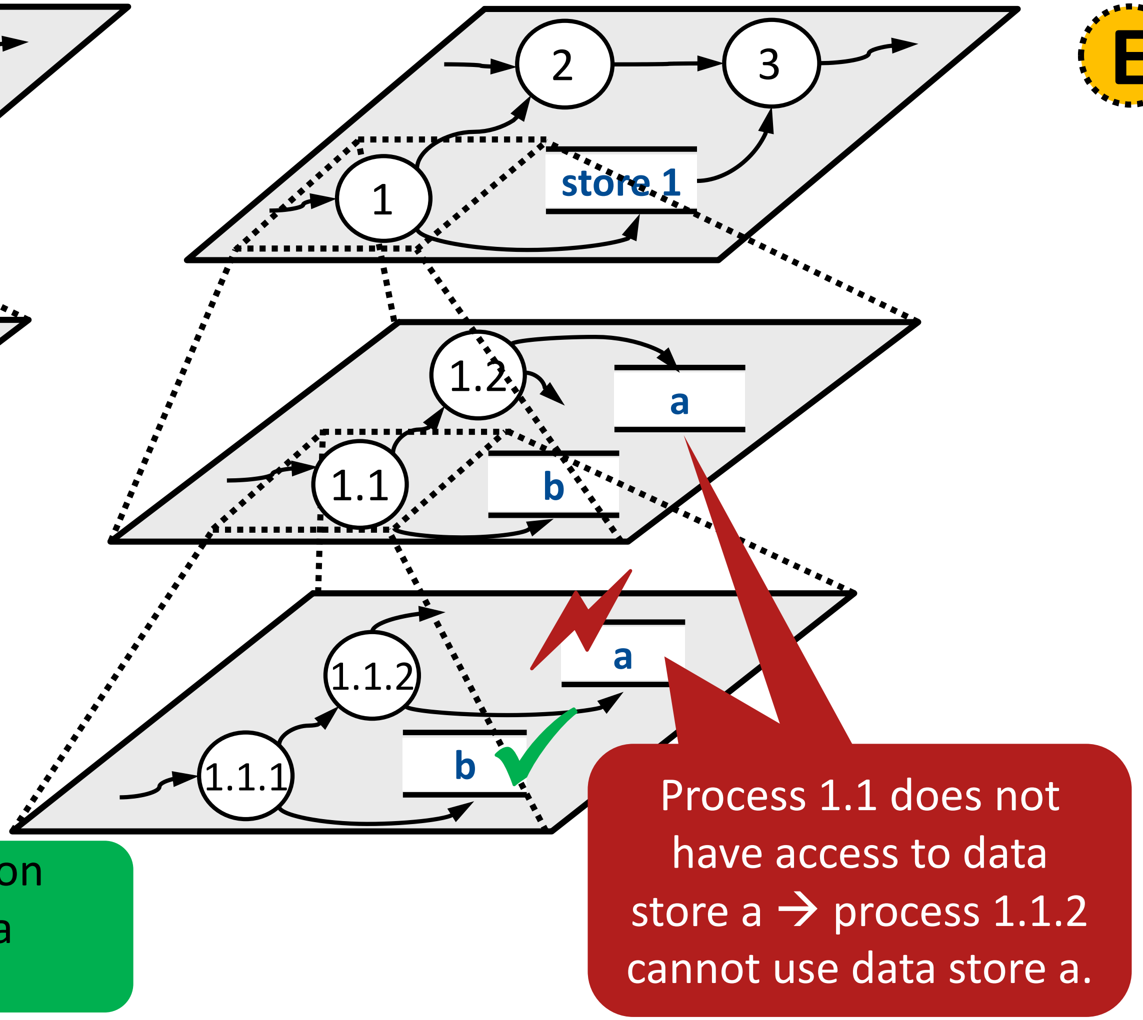


Balancing of DFDs: Data Dictionary
Data Store Balancing (1)

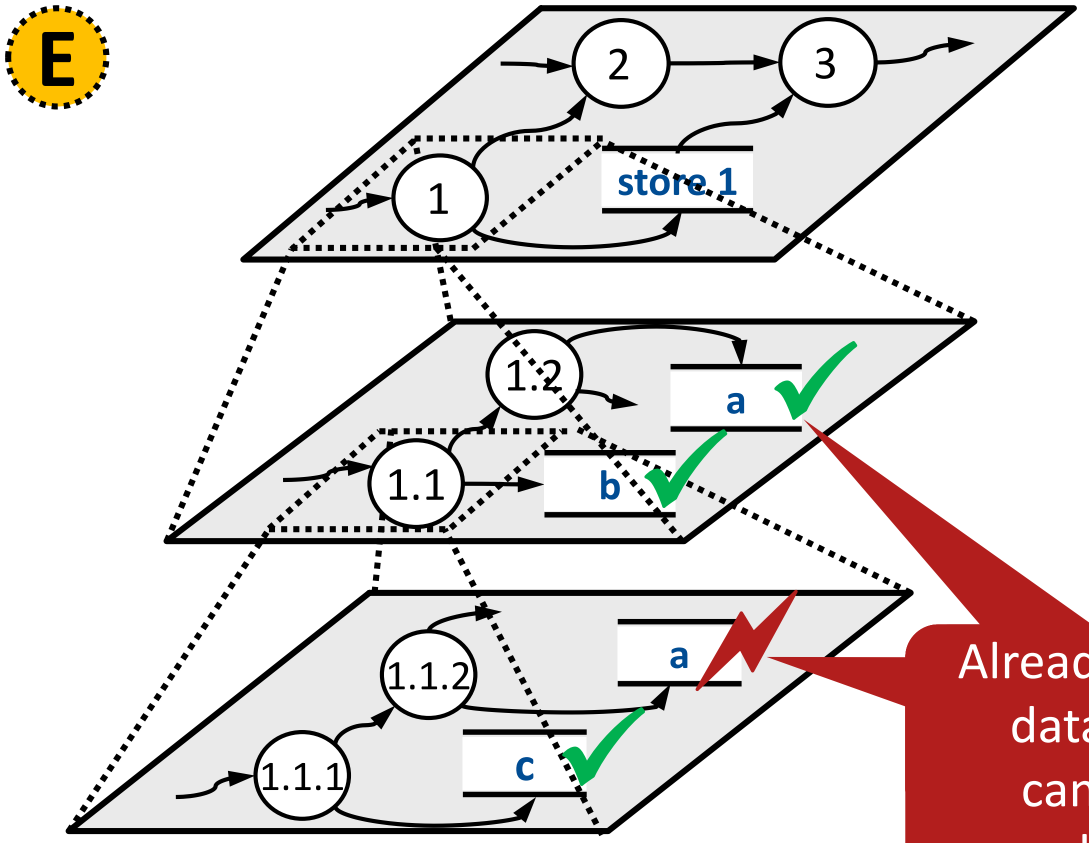
E

Data dictionary:
store 1 = a + b





Balancing of DFDs: Data Dictionary
Data Store Balancing (2)



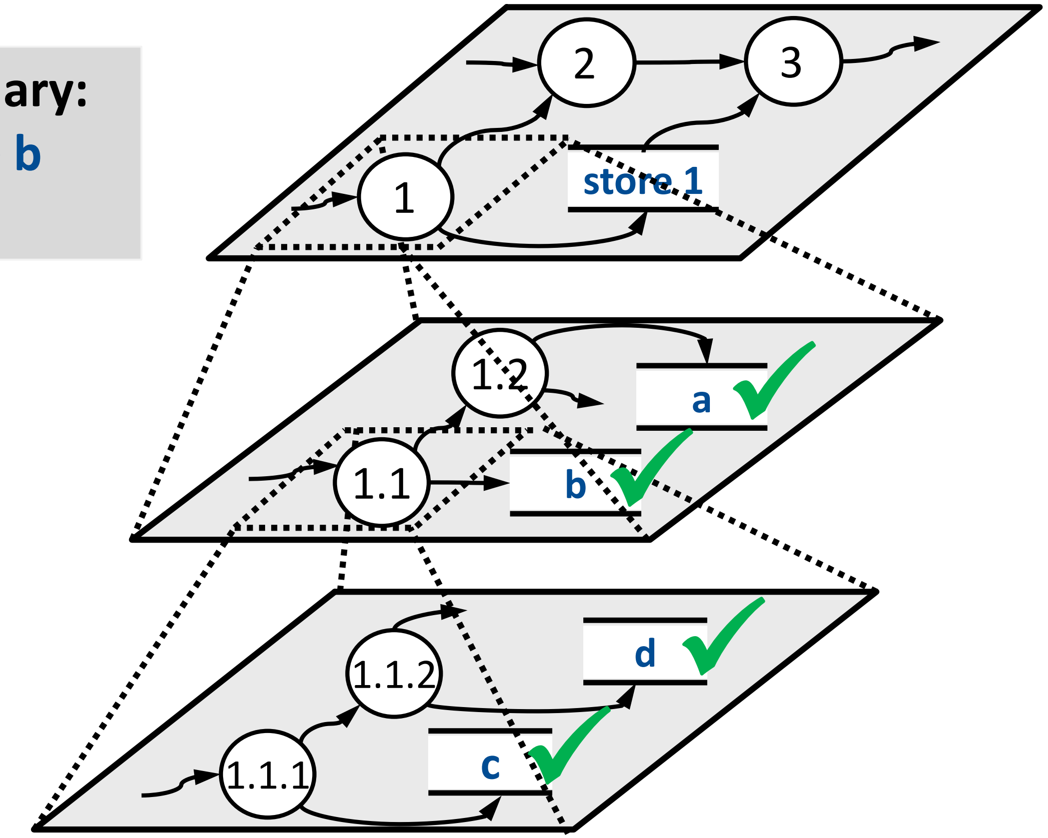
Data dictionary:
store 1 = a + b
b = c + a

Already defined data stores cannot be redefined.

Balancing of DFDs: Data Dictionary Data Store Balancing (3)

E

Data dictionary:
store 1 = a + b
b = c + d



Balancing of DFDs in Practice

- In practice, visible balancing is not frequently used.
  - Limited use: No decomposition across different DFD levels
    → Many applications not supported
  - Can become visually complex in the DFDs
- In practice, Data Dictionary balancing and Data Dictionary Data Store balancing are often combined.

7. Mini Specifications

Motivation

- **Functional primitives** are processes/function which **are not further refined** in a data flow diagram.
- The **functional primitive** are only vaguely and ambiguously defined in the DFDs.
- There is a **lack** of detailed **information** on how a **primitive process produces its outputs** based on its inputs.
- **Solution**:
  Definition of functional primitives in mini specifications.

see [DeMarco 1979], p. 76f

Properties

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

A Mini Specification

- describes **how a primitive process (function) produces its outputs** based on its **inputs**
  - in terms of a coarse **strategy** without defining a concrete algorithm.
- is typically defined using **natural language**.
- typically defines the functionality of the functional primitive as **a sequence of steps**, which use inputs to produce the outputs.

see [DeMarco 1979], p. 85

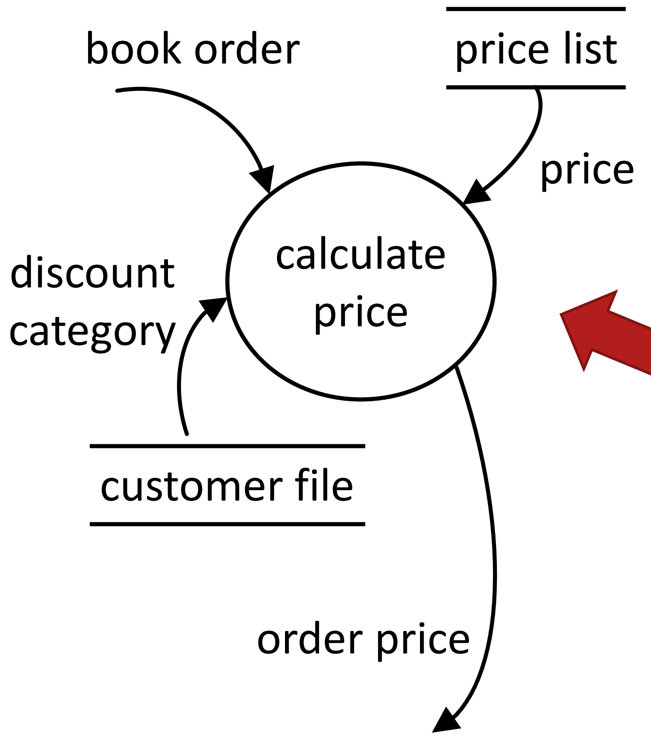
Example

UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

E

Excerpt of a Data Flow Diagram



Data Dictionary (excerpt)

book order	= customer number + { book number }
customer file	= customer number + discount category
price list	= book number + discount category + price

Mini Specification of the process "calculate price"

For each book order, do the following things:

1. Look up the discount category in the customer file for the customer number from the book order
2. For each book number in the book order, do the following things:

   Look up the price in the price list for the combination of book number and discount category
3. Add all prices to determine the sum.
4. If the sum is higher than $100, subtract 10% to calculate order price.

Hints

UNIVERSITÄT DUISBURG ESSEN

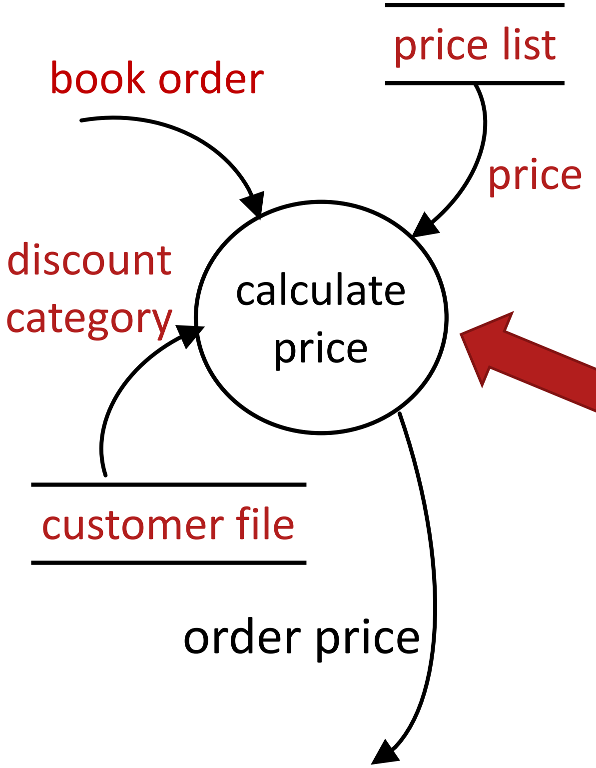
Offen im Denken

- Recommended **length** of a mini spec: **½ to 1 page**.
  - Longer mini specs are usually **hard to understand**.
  - Shorter mini specs indicate **too fine-grained partitioning**, which may require more effort in finding relevant information about the system.
- Avoid **redundancies**.
- Use **unambiguous style** of writing.
  - Reference terms defined in the **data dictionary**.
  - Keep sentences short.
  - Use **positive** active/passive formulations.
  - Specify conditions **before** the successive actions.

see [McMenamim and Palmer 1984], p. 282ff

Example for Interrelation of SA Components

E Excerpt of a Data Flow Diagram



Data Dictionary (excerpt)

book order	= customer number + { book number }
customer file	= customer number + discount category
price list	= book number + discount category + price
price	= …..

Mini Specification of the process "calculate price"

For each **book order**, do the following things:

1. Look up the **discount category** in the **customer file** for the **customer number** from the book order.
2. For each **book number** in **the book order**, do the following things:
   Look up the **price** in the **price list** for the combination of **book number** and **discount category**.
3. Add all **prices** to determine the sum.
4. If the sum is higher than $100, subtract 10% to calculate order price.

Summary

Structured Analysis consists of three components:

- Data Flow Diagrams
  - document processes (functions), the manipulation of data by processes and the data flows between the processes and between the sources/sinks, which are located in the context.
- Data Dictionaries
  - contain definitions of all data flows and data stores, as well as their composition.
- Mini Specifications
  - describe how primitive processes (functions) produce their outputs based on their inputs.

Hierarchized data flow diagrams are used to decompose the system and thereby reduce complexity. Balancing rules are used to ensure correct hierarchization.

Literature

[DeMarco 1979] T. DeMarco: Structured Analysis and System Specification. Yourdon Press, Englewood Cliffs, New Jersey, 1979.

[McMenamin and Palmer 1984] S. M. McMenamin, J. F. Palmer: Essential Systems Analysis. Prentice Hall, London, 1984.

[Robertson and Robertson 1998] J. Robertson, Suzanne Robertson: Complete Systems Analysis. Dorset House, New York, 1998.

[Yourdon 1989] E. Yourdon: Modern Structured Analysis. Prentice Hall, Englewood Cliffs, 1989.

[Hatley et al. 2000] D. Hatley, P. Hruschka, I. Pirbhai: Process for System Architecture and Requirements Engineering. Dorset House, New York, 2000.

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.

[Hatley and Pirbhai 1988] D. J. Hatley, I. A. Pirbhai: Strategies for Real Time System Specification. Dorset House, New York, 1988..

[Davis 1993] A. M. Davis: Software Requirements – Objects, Functions, and States. 2nd edition, Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[OMG 2004] Object Management Group: Meta Object Facility (MOF) 2.0 Core Specification. OMG ptc/03-10-04.

Image References

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

[2] Provided by Microsoft Office

Legend

D Definition

E Example

UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

Requirements Engineering & Management

Vielen Dank für Ihre Aufmerksamkeit

PALUNO
The Ruhr Institute for Software Technology