A Survey of Software Engineering with an Aspect on Database Design

Gregor Büchel
Fachhochschule Köln
Institut für Nachrichtentechnik
gregor.buechel@fh-koeln.de

Content (2)

1. The Phase Model of Software Engineering
2. Methods and Concepts of Requirement Analysis and System Definition
3. Design of Database Systems

1. The Phase Model of Software Engineering (3)

Software Engineering (SE) is the part of computer science that deals with concepts, methods, and tools to develop software systems.

Software Engineering is programming on a “large scale” while coding programs is programming on a “small scale”.

The benefits from SE are received in large software systems like

- ticket reservation systems (airlines)
- sales and distribution systems (industry)
- personal information systems (companies)

1. The Phase Model of Software Engineering (4)

Why is SE useful?

- SE helps to identify reusable components of software.
- SE supports decisions like: Which software should be bought (standard software)? Which software should be developed by own means?
- SE helps to test, to migrate, and to extend software systems.
- SE provides necessary documents for users, maintainers and developers of a software system.

1. The Phase Model of Software Engineering (5)

The **phase model** of SE is related to the phases in the lifecycle of a software system. The main phases are:

(1) Requirement analysis for a new software system.

(2) System definition.

(3) Design of a software system.

(4) Implementation of a system and testing.

(5) Maintenance of a system.

Each phase has defined documents at its end and has documents as inner milestones. E.g. relevant milestone of phase (4) is the system's documented source code.

2. Methods and Concepts of Requirement Analysis and System Definition (6)

2.1 Requirement Analysis [BAL96]

The phases (1) and (2) are dialogues between the developer (d) and the customer (c) of the new system.

Start of phase (1): A **catalogue of requirements** (c).

Activities:

(a11) Identification of the workflows in the customer's company, which should be supported by the system.

(a12) Description of the system's external interfaces (the sources and the drains of external data, which should be processed by the system).

(a13) Identification of the main components of the system.

(a14) A general description of the information flows between the components.

(a15) Development of a controlled vocabulary of concepts for the objects in the scope of the system.

2.1 Requirement Analysis (7)

Results:

(r11) Requirement specification (“Lastenheft”) (d).

(r12) A general analysis on costs and benefits of the new systems (d) / (c).

(r13) A decision, which components should be bought and which should be developed (c).

(r14) A decision to start phase (2).

Ex.0: Catalogue of requirements for the system W3BUCHH (8)

1. Die Waren einer WWW-Versandbuchhandlung können für alle WWW-User (PUBLIC) über HTML-Seiten angezeigt werden.
2. WWW-User, die Versandkaufhauskunden werden möchten, können sich per HTML-Formular unter Angabe einer gültigen EMAIL- und Versandadresse registrieren lassen.
3. Bestellungen werden über HTML-Formular ausgeführt. Hierbei gibt der Kunde eine gültige Kreditkartennummer ein. Die Kreditkartennummer wird verschlüsselt mit einem zertifizierten Verfahren übertragen.
4. Der Kunde erhält für eine Bestellung eine Auftragsbestätigung per EMAIL.
5. Auf dem Postweg erhält der Kunde seine bestellten Waren. Für die Versandabwicklung ist das Lager zuständig. Damit das Lager den Versand ausführt, bekommt es den für die Bestellung gültigen Lieferschein. Der Lieferschein wird der Ware beigepackt.
6. Ist der Lieferschein erstellt, bekommt die Kundenbuchhaltung (externes System) eine Kopie des Lieferscheines (= erstellte Rechnung). Sie veranlaßt dann die Rechnungszahlung via CC-Lastschrift und leistet die Zahlungsverfolgung.

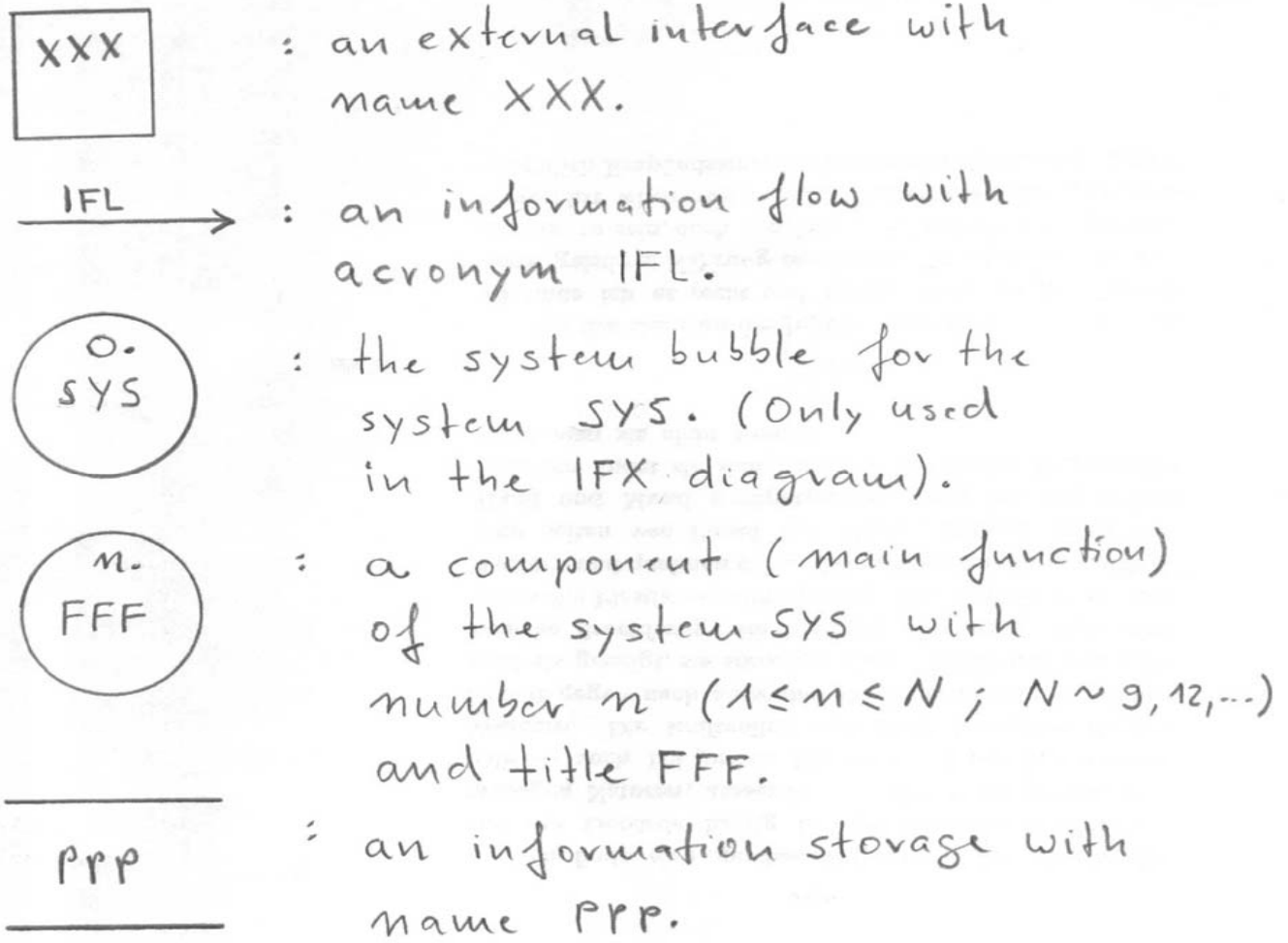
Concepts of SE are influenced by paradigms of programming. We have two main families of concepts:

1) Structured analysis and design technology (=:SADT) ([DEM79],[BAL96]) (← structured programming).

2) Object-oriented analysis and design (=:OOA/OOD), ([BOO91], [RUM93],[BAL96],[UML07]) (← object-oriented programming).

Both families of concept have a graphically based modelling language (=: ML). In OOA/OOD there exist a language standard: The Unified Modelling Language 2.0 (=: UML). These modelling languages consist of well-defined diagram types (icons). With these icons, acronyms in the icons, and explanations in diagrams' legends a controlled vocabulary could be built.

2.2.1 Elements of SADT modelling language (10)



Examples of Information Flow Diagrams (11)

In SADT two types of diagrams are defined:

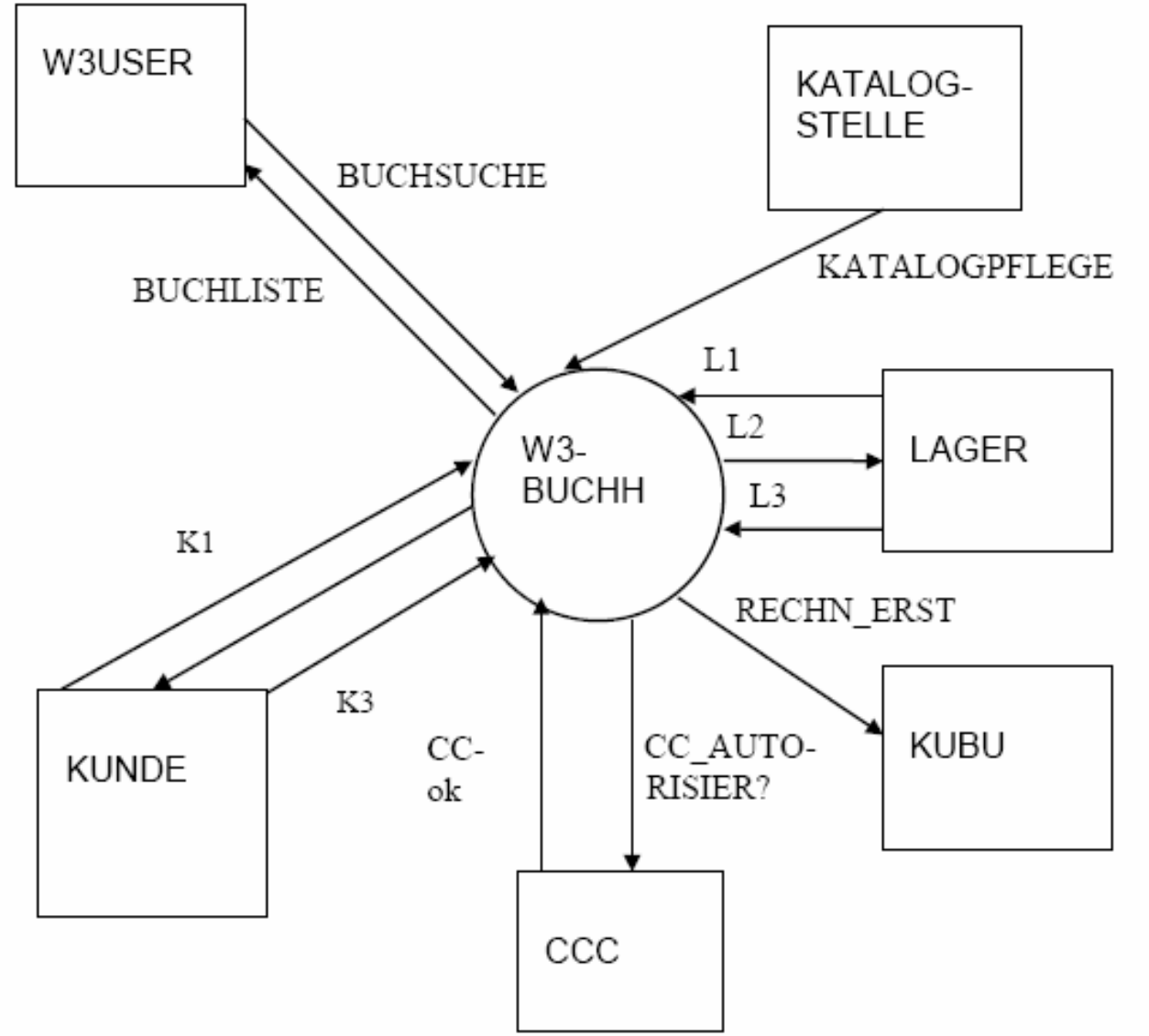
A) IFX := an information flow diagram for external interfaces.

B) IF0 := an information flow diagram for the top level decomposition of the system.

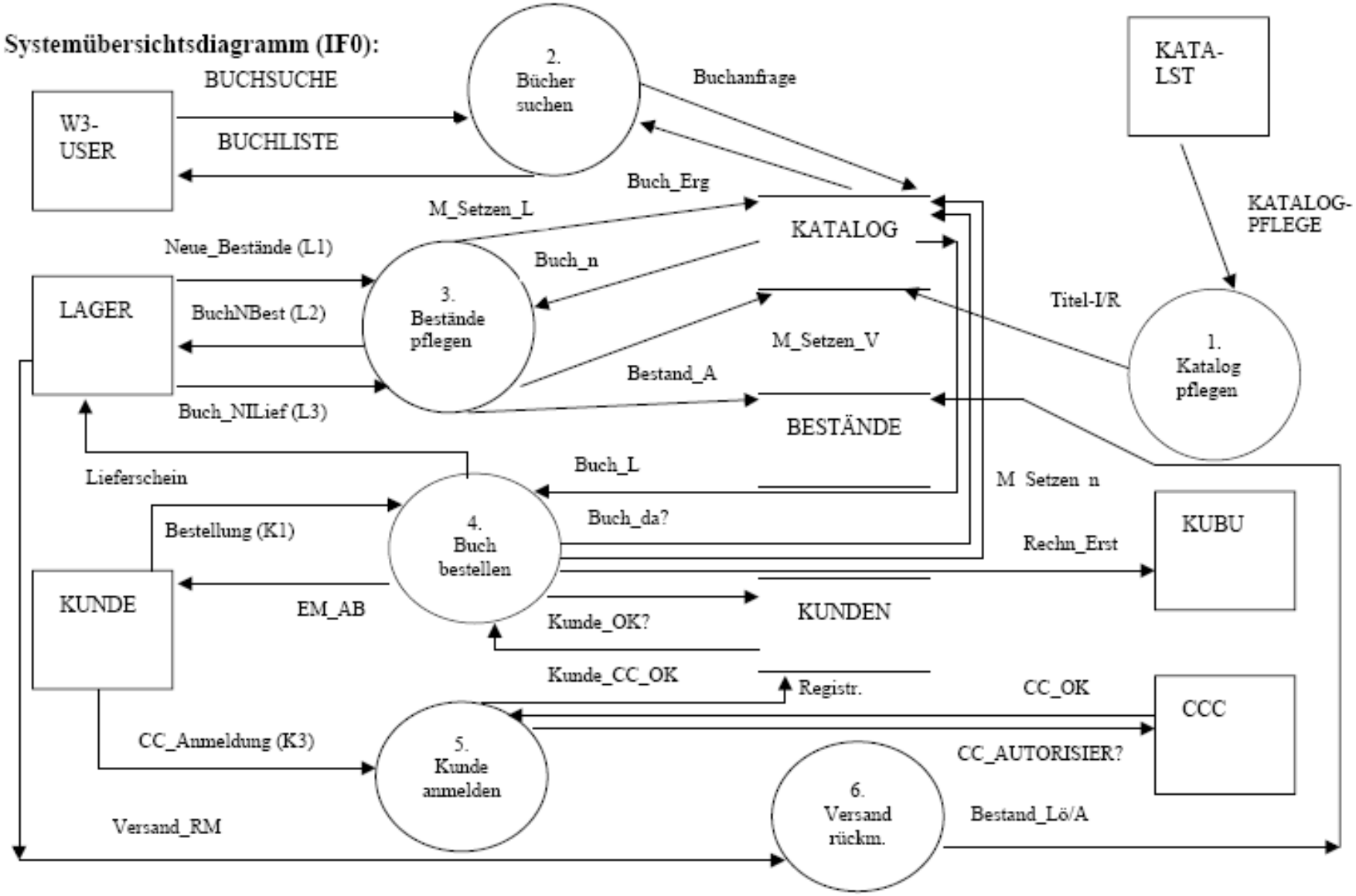
Rem.1: IFD support the top-down analysis principle of SE. E.g. we have IFn diagrams for the decomposition of component n on level 1, IFn.m diagrams for the decomposition of component n.m on level 2 etc.

Rem.2: IFX, IF0 are milestone results for the activities (a12), (a13) and (a14).

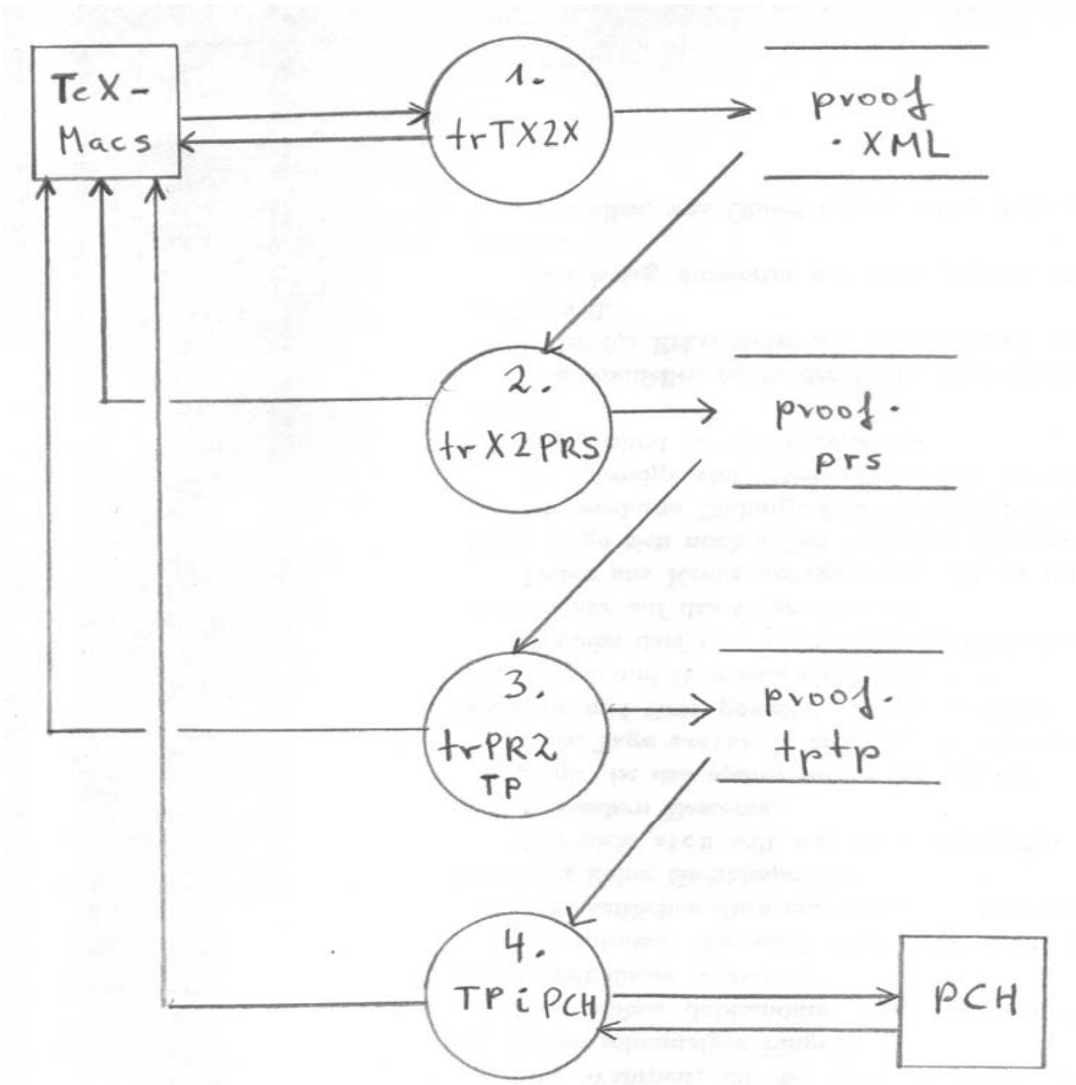
Ex.1: IFX for W3BUCHH



Ex.2: IF0 for W3BUCHH



Ex.3: IF0 (light): NAPROCHE



2.2.2 Elements of UML 2.0 (15)

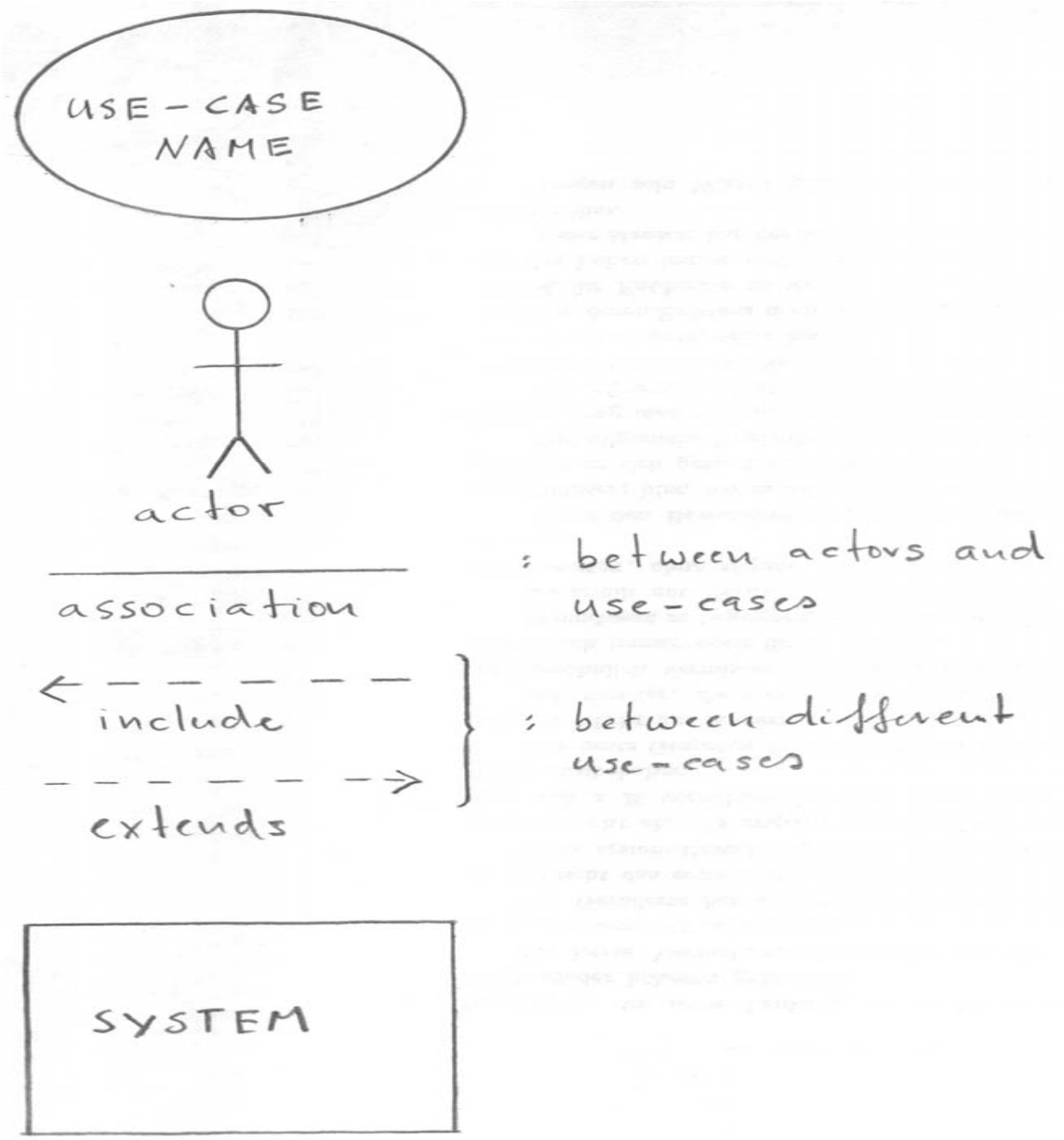
UML 2.0 contains more than 12 types of diagrams. They could be divided into two groups:

A) diagrams for structural modelling.

B) diagrams for behavioural modelling.

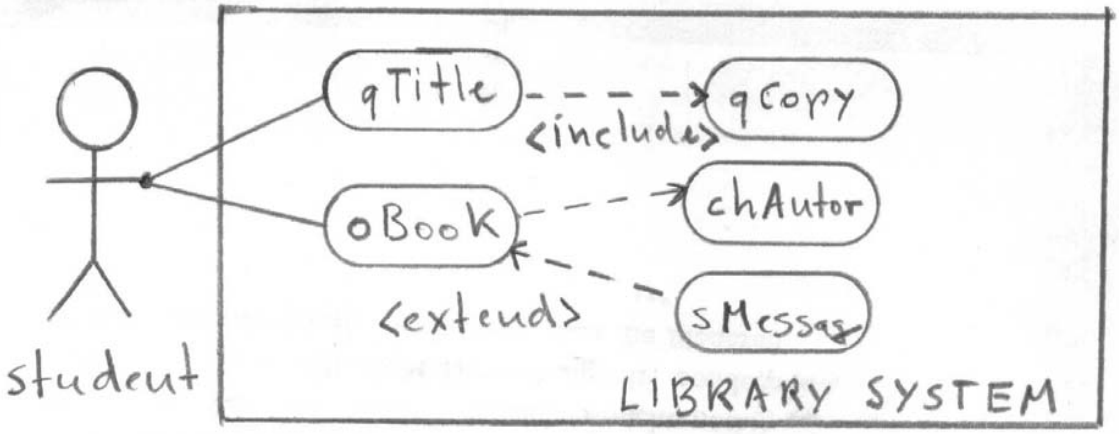
As an example for B) we will discuss **use-case diagrams**, and as an example for A) **class diagrams.**

2.2.2 Elements of UML2.0: Use-Case Diagrams (16)



Elements of use-case diagrams (icons)

Ex.4: Use-Case Diagram: Ordering Books (17)



legend: USE-CASES[1]:

qTitle: query title

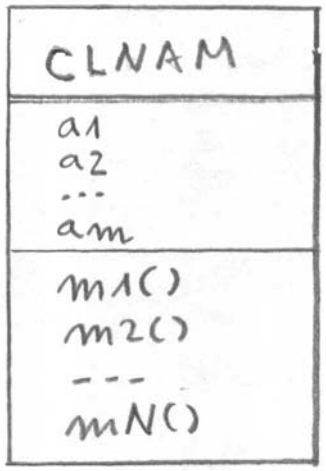
qCopy: query copy of a book

oBook: order the copy

chAutor: check authorization

sMessag: send message

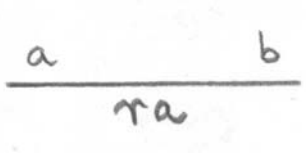
Class Diagrams: Icons (18)



CLNAM : class name

a1, …, am : attributes

m1(), …, mN(): methods



ra : an association between two classes A and B.

a, b: cardinalities: a objects of A are related by ra to b objects of class B.

Class Diagrams: Icons

$AE = \{x \mid x \text{ object of type } A\}$

$BE = \{y \mid y \text{ object of type } B\}$

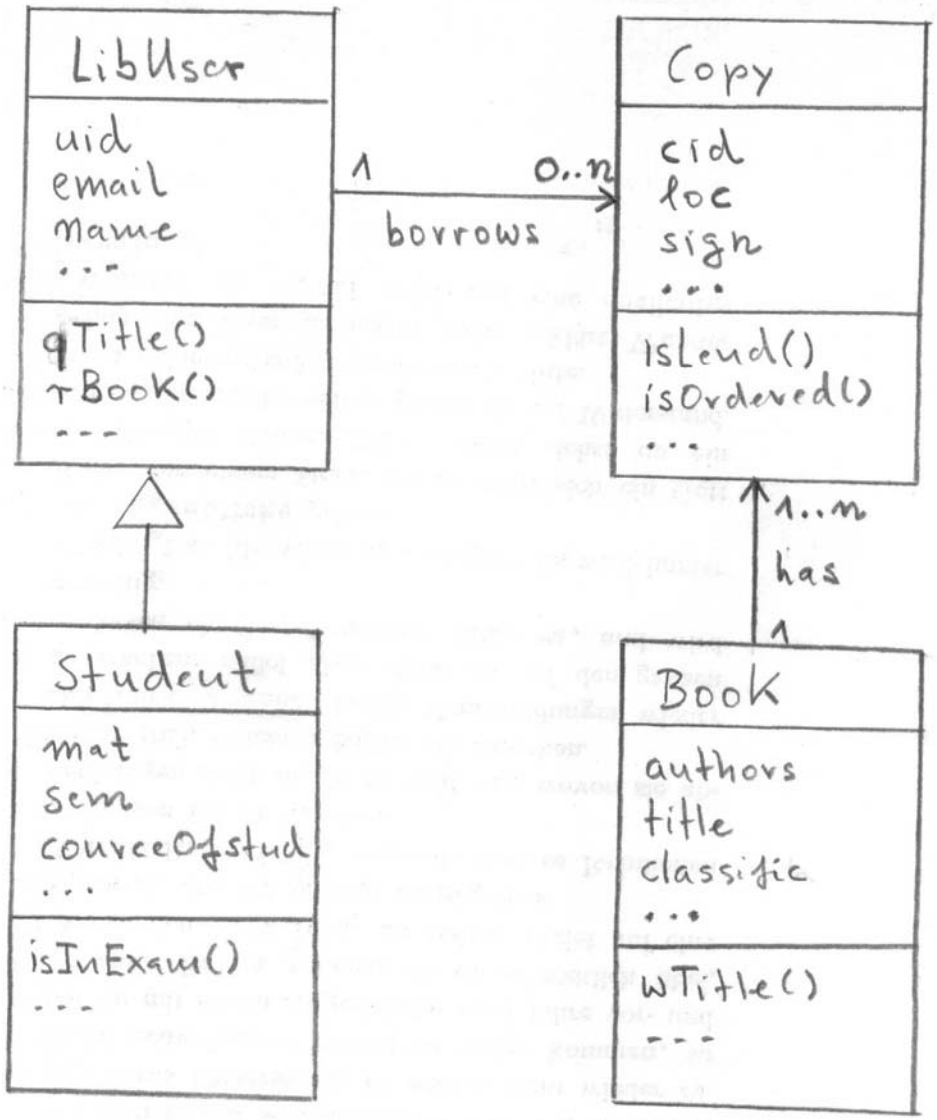
$ra \subseteq AE \times BE$

$ra = \{(x,y) \in AE \times BE \mid ra(x,y)\}$

Associations could be interpreted as relations between the extends of the classes.

Inheritance: class A (subclass) inherits properties and behaviour from class B.

Ex.5: Class Diagram: Library User (20)

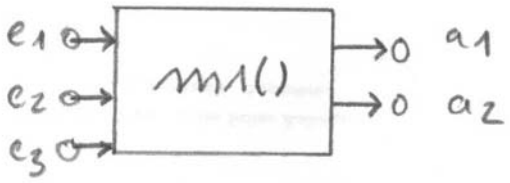


2.3 System Definition (21)

Start: requirement specification

Activities:

(a21) A detailed description of the inner structure of the components. A component is decomposed in modules. A module is an independently programmable unit (like a function or a class) with defined entries and exists:



(a22) A precise definition of communication between the components: a) synchronous communication by direct information flows.

b) asynchronous communication by an information storage, which is connected with the participating components with information flows.

(a23) A detailed description of information flows on the level of **information elements**.

(a24) A precise analysis of external interfaces. For instance GUI (:= graphical user interfaces) should be exactly analysed under the aspects of user's behaviour and of information elements, which will be deliverd to the system.

(a25) A detailed definition of the inner structure of information storages on the level of information elements.

2.3 System Definition (23)

Result:

(r21) A system definition from the user's viewpoint with structural and behavioural descriptions. (d)

(r22) A decision to start phase (3).

System Definition: Remark to (a24): (24)

Guidelines for the user interfaces: TeXmacs:

(i) one formula in one texmacs field of type "equation" ("Gleichung"): (best practise): $A \wedge \forall_x P(x)$

(ii) one formula distributed in two texmacs fields of type "equation": $A \wedge \forall_x P(x)$

Should an use of TeXmacs like in (i) become a guideline standard?

2.4.1 Data Dictionary Notation (DDN)

The DDN is used for the description of information flows (IFL) and information storages (ISP). In difference to the Backus Naur Form (BNF), which is a bottom-up notation for syntax rules, DDN is a top-down notation. DDN is a data type independent notation.

For an IFL or ISP we have a sequence of production rules, which is terminated at the level of information elements (IE).

(S1) IE_NAME := IE_EXPLANTION (in natural language)

Ex.6: DDN for the information element YEAR in W3BUCCH: YEAR := year of publication, written with 4 digits.

2.4.1 Data Dictionary Notation (DDN) (26)

Definition of an IFL or ISP by a production rules, which consists of defined IE_NAMES, IFL_PARTS and operators:

(S2) IFL_PART:=rsPROD(IP1, IP2, …, IPn)

In the right side of the production rule the following operators are used:

(1) sequence: + (no symbol in BNF)

(2) alternative: […|…] (in BNF: |)

(3) repetition: {…}, {…}a:b , min=a, max=b

(4) option: (…) (in BNF: […])

Ex.7:

BOOK:=AUTHOR+TITLE+(YEAR)+{CATCHWORD}

Ex.8 Landau: German-English Vocabulary (27)

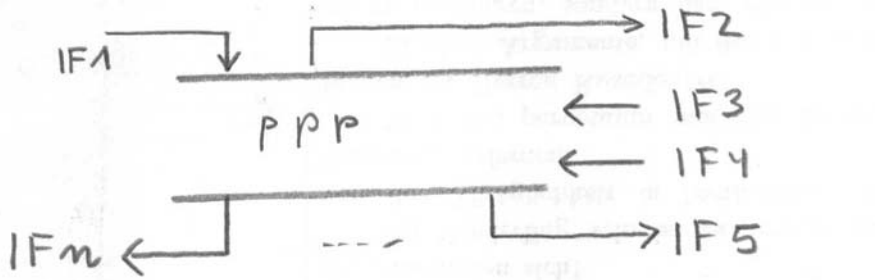
Some entries in Landau's vocabulary:
aufzählen, (auf+zählen) to enumerate, to count, to list.
dann, then; _**und nur**_, if and only if.
heben, to lift (cog., heave).
lernend, (present participle of: lernen) learning.
oder, or.
Satz, theorem, sentence, proposition.

DDN:
LGEVART:=SW+(SYN_ANG)+(MORPH_ANG)+TR_LIST
TR_LIST:=TR_LIST1|TR_LIST2|…
TR_LIST1:={ENGLW+(ETYM_ANG)}
TR_LIST2:=TR_LIST1+{SUB_SW+TR_LIST1}

SW:=catchword; SYN_ANG:=syntactic information;
MORPH_ANG:=morphological information; TR_LIST: list of translation;
ENGLW:=English word; ETYM_ANG:=etymological information;
SUB_SW:="sub catchword"

2.4.2 Entity-Relationship Diagram (28)

Given: An information storage:

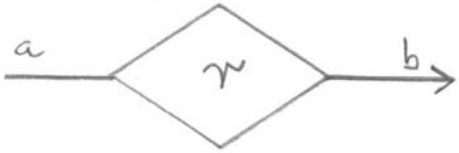


The next step: Transformation of the information storages of a system into an entity-relationship model.

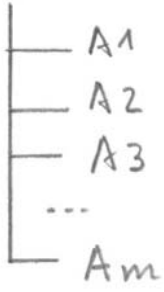
Entity types are defined by a list of attributes. The **attributes** are IE. Relations can exist between sets of elements of given entity types. A **relationship** describes the type of a relation, that should be stored.

Elements of an ERD

: entity type E.

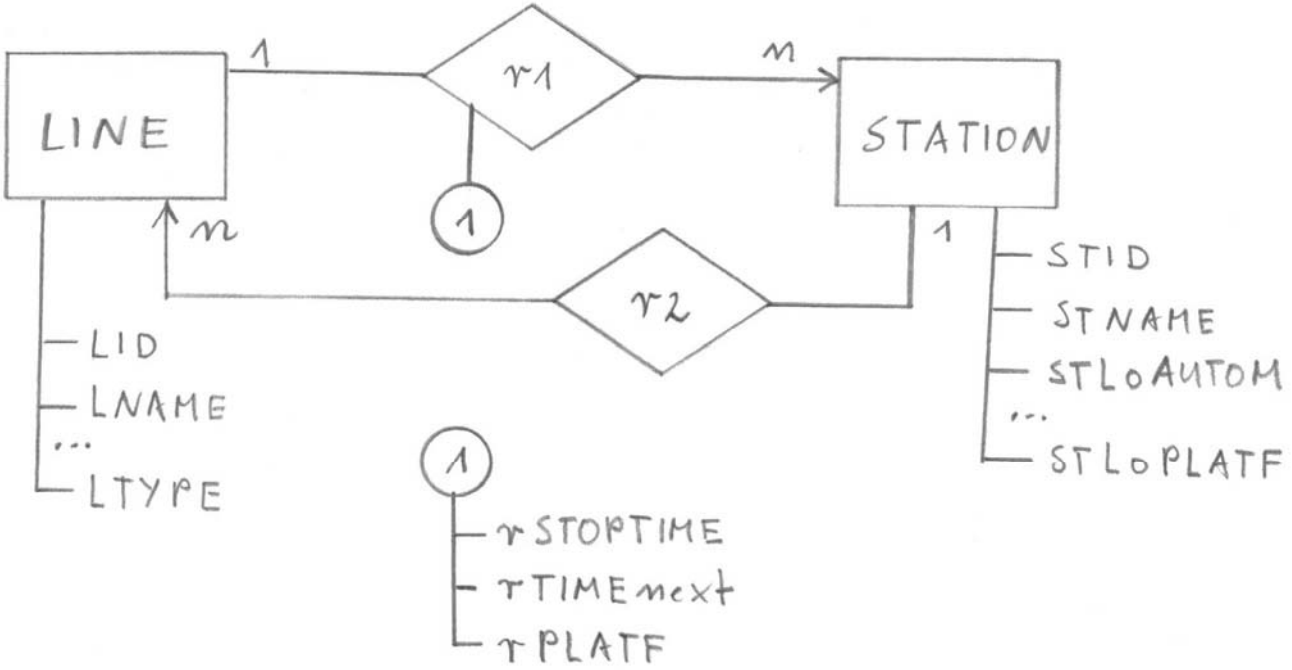


: a (directed) relationship r with cardinalities $a, b \in \{0, 1, c, n, m\}$ $n, m \in \mathbb{N}$; $c \in \{0, 1\}$.



: list of attributes A1, A2, ..., Am of one entity type E.

Ex.9: An ERD for a public transport system (30)



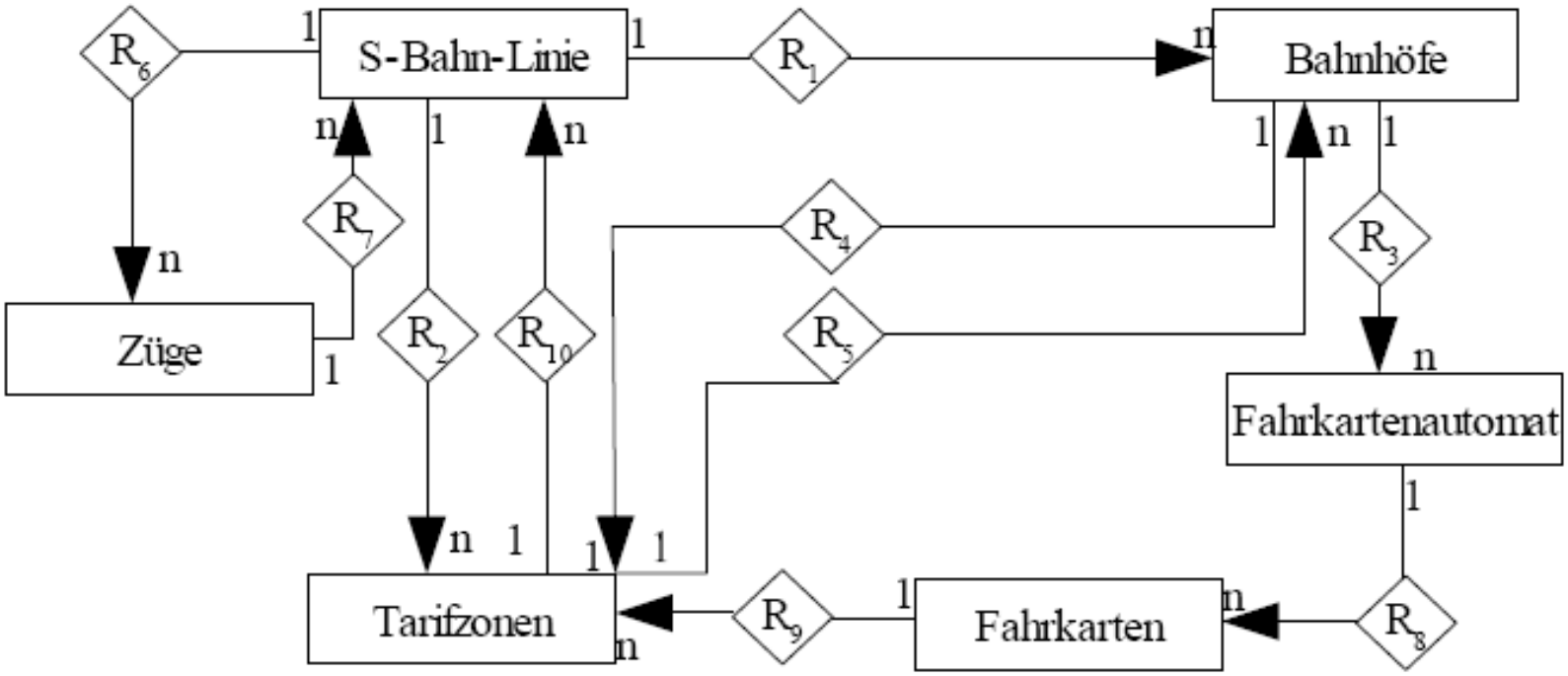
Legend:

r1 := stops at ...

r2 := is stop of ...

Ex.10: ERD for PTS (31)

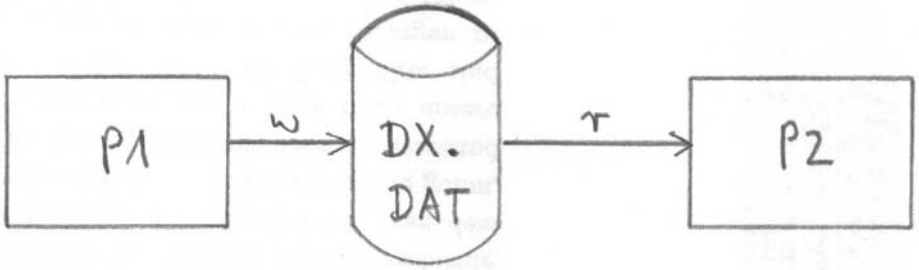
ERD für Verkehrsverbund:



3. Design of Database Systems (32)

3.1 General Aspects of Database Systems:

If there is no database system, data are persistently stored in data sets, which are managed by the computer's operating system:



Disadvantages (file system):

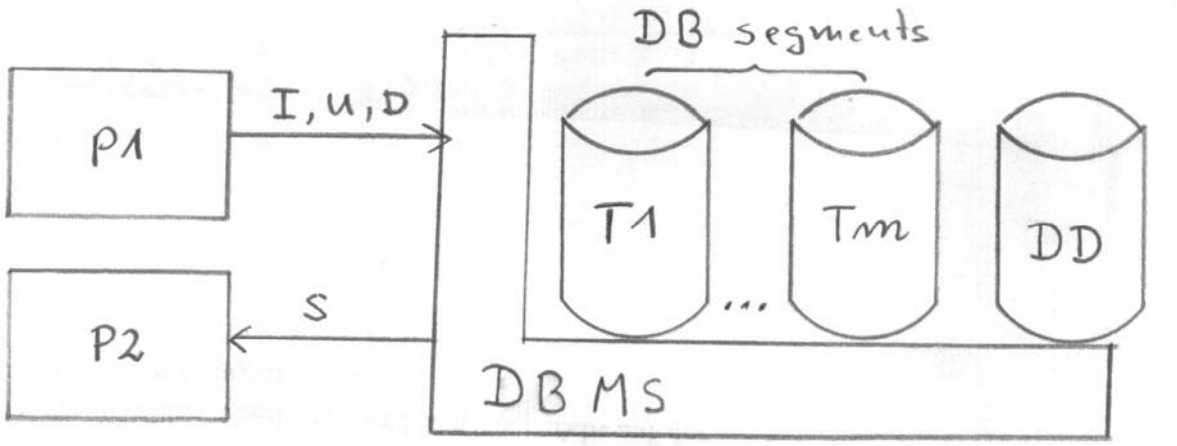
(d1) no direct access support.

(d2) no mapping support: RAM-data (P1) -> data set (DX.DAT).

(d3) no metadata support for the records in the data set.

3.1 General Aspects of Database Systems: (33)

Database system:



Advantages of a database system:

(v1) Persistence: Data with complex structure are mapped under the condition of structural integrity in a secondary storage.

(v2) Mechanisms for direct access.

(v3) Independence of data structures: All metadata of user data are stored in a data dictionary. The structures are independent from application programs. The definition of metadata is done by a DDL.

3.1 General Aspects of Database Systems: (34)

DDL:=data definition language.

The DDL is related to database's data model.

(v4) Control mechanisms for data integrity. (integrity of data types, of primary and foreign keys, of attribute values, …).

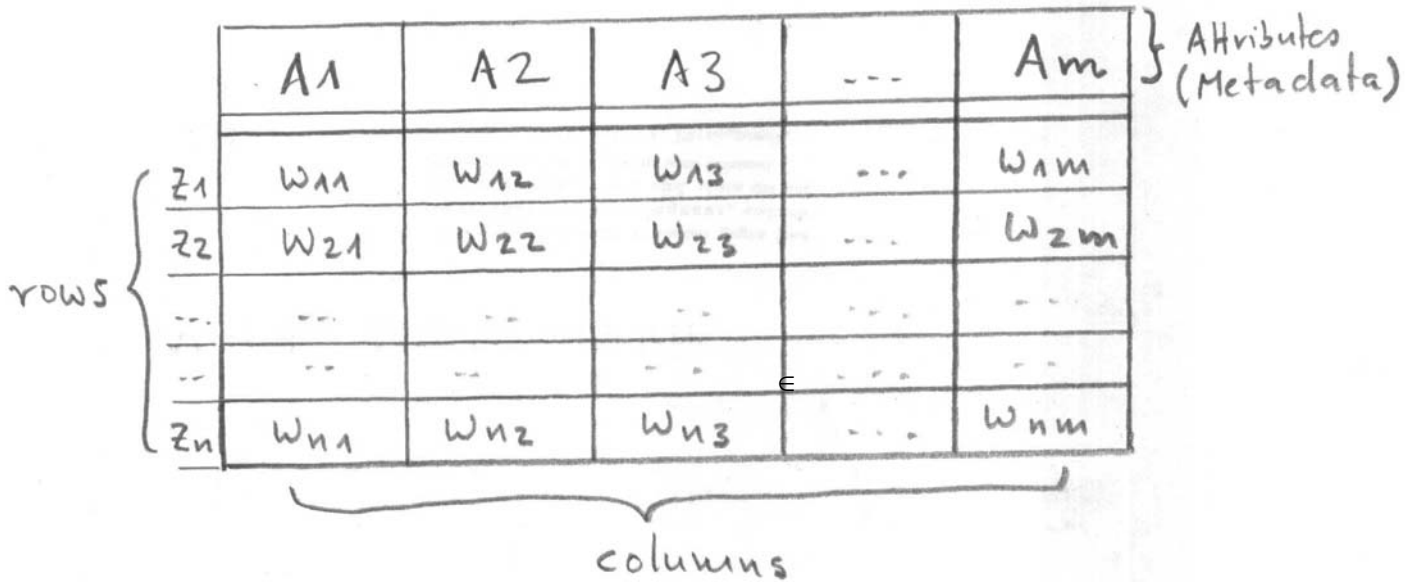
(v5) Interpretation of a query language (QL). (integrated command interpreter).

...

Rem1.: DBS = DB + DBMS (a database system is the set of stored data together with an database management system).

3.2 Relational Database Systems (RDBS) (35)

DB segments are tables:



A) wij : value of data type dtj = dt(Aj)

B) Wj:=dom(Aj) , domain of the attribute Aj. Wj is determined by dt(Aj) and integrity conditions, which are defined for Aj.

C) For every row zi=(wi1,wi2, ..., wim) we have:

zi $\in$ dom(A1)xdom(A2)x … x dom(Am)

3.2 Relational Database Systems (RDBS) (36)

The scheme RS(T) of a table T in a RDB is given by:

$RS(T) = \{ (Aj, dtj, Wj) | \ 1 \leq j \leq m\}$

The standardized QL and DDL for RDBS is SQL.

SQL-Norms: SQL92 (ISO); ORDB: SQL99 (ISO).

Ex.11: The relational scheme RS(Line) (37)

RS(T1) = RS(Line) for the table T1 in a RDB for a public transport system:

A_j	LID	LNAME	LTYPE	...
dt_j	int	char(30)	char(5)	...
CI_j	PRIK		CHECK(B)	...

B = LTYPE IN ('RB','RE','S',...)

SQL-DDL command for T1:

```
create table
( LID integer primary key,
  LNAME char(30),
  LTYPE char(5) NOT NULL CHECK (LTYPE in ('RB','RE','S'))
)
```

3.3 Design of RDBS (38)

The relational data model is a **flat** data model. The highest data structure, one can build, is:

TABLE:=SET OF(z : TUPLE OF(A1:dt1; …; Am:dtm))

If the data structure is of higher complexity (e.g. Ex.10), you have to normalize the data model:

Ex.12: Normalization of the data model (39)

$RS(PTS) = RS(T1) \cup RS(T2) \cup RS(T3) \cup RS(T4) \cup \ldots$

with:

RS(T3):

Aj	HID	LID	STID	STOPTIME	...
dtj	int	int	int	int	...
CIj	PRIK	FKEY(T1)	FKEY(T2)	NOT NULL	...

RS(T4):

Aj	AID	STID	AUTO_DESCR.	IP_A	...
dtj	int	int	varchar(150)	int	...
CIj	PRIK	FKEY(T2)	NOT NULL	NOT NULL	...

3.3 Object-oriented and object-relational Database Systems (OODBS/ORDBS) (40)

Object-oriented programming languages have advantages:

(1) High order complexity of classes

(2) Collection types for data collections (LIST, SET, …)

(3) Capsulation of data and methods

These advantages led to proposals to define OODBS and later ORDBS.

Ex.13: An OODB-DDL definition for the entity type STATION:

```
Class Station
{int stid;
 String stname;
 ListOfObject l_automates;
 SetOfObject m_stopLines;
 …
}
```

SQL:1999/SQL:2003 defines the object-relational standard:

(a) Entity types with complex class structures can be mapped on user defined types.

(b) Collection types like SET, LIST can mapped on **nested tables**.

Ex.14: CREATE TYPE automate AS OBJECT

```
(autonr integer,
 auto_desc varchar(50),
 ip_a integer);
CREATE TYPE aut_nt AS TABLE OF automate;
CREATE TABLE station (
        stid integer primary key,
        stname varchar(50),
        l_automate aut_nt )
NESTED TABLE l_automate STORE AS lAut_nt_tab;
```

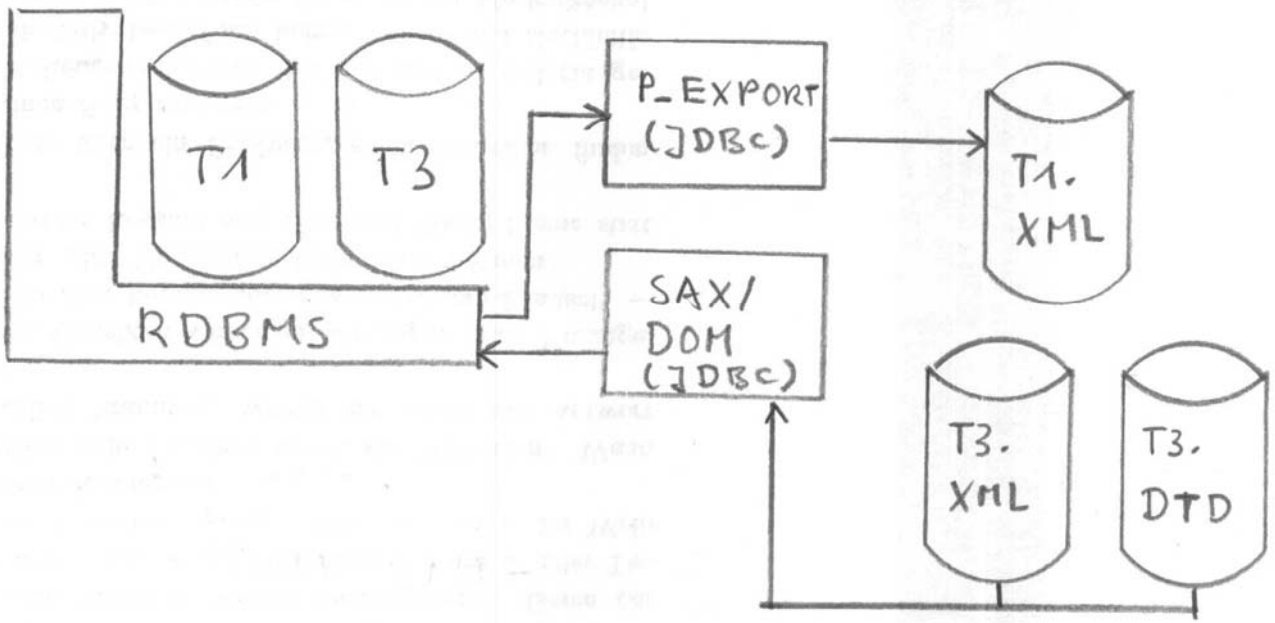
3.5 DBMS and XML: A short Outlook (42)

Persistence mechanisms for data sets:

structured	semistructured	unstructured
ENTITIES ATTRIBUTES DATA TYPES	ENTITIES ATTRIBUTES ---	--- --- ---
DB Scheme XML/XML-S OWL	XML/DTD	TXT HTML CSV

3.5 DBMS and XML: A short Outlook (43)

a) XML and RDB: Export / Import:



The complexity of an import program (e.g. Java/SAX) depends on the complexity of the XML-hierarchy of the XML input file.

3.5 DBMS and XML: A short Outlook (44)

b) Native XML Databases (XDBS):

Products like **eXist** have a simple utility for the import of XML files.

QL for a XDBS is XQuery.

Ex.15:

```
for $b in doc("Literatur.xml")
where $b/ejahr > 2000
return $b/title
```

c) Data type **XML** in SQL:2003 [TÜR03]: A complete XML document can be inserted as a column value in a RDB table. (XQuery terms in SELECT clauses).

Thank you for your interest!

References (46)

[BAL96] Balzert, Helmut: “Lehrbuch der Software-Technik”, Heidelberg [etc.] (Spektrum) 1996.

[BOO91] Booch, Grady: “Objektorientierte Analyse und Design”, Bonn; Reading, Mass. (Addison-Wesley) 1994.

[DEM79] DeMarco, Tom: “Structured Analysis and System Specification”, New York (Prentice-Hall) 1979.

[FOW97] Fowler, Martin; Scott, Kendall: “UML Distilled – Applying the Standard Object Modeling Language”, Reading, Mass. (Addison-Wesley) 1997.

[HEU97] Heuer, Andreas; Saake, Gunter: “Datenbanken – Konzepte und Sprachen”, Bonn, Albany (Thomson Publishing) 1997.

[LAN65] Landau, Edmund: “Grundlagen der Analysis – With a Complete German-English Vocabulary”, New York (Chelsea) 1965.

[RUM93] Rumbaugh, James; Blaha, Michael; Premerlani et. al.: “Objektorientiertes Modellieren und Entwerfen”, München; Wien (Hanser) 1993.

References (47)

[TÜR03] Türker, Can: “SQL:1999 & SQL:2003”, Heidelberg (dpunkt) 2003.

[UML07] Rupp, Chris; Queins, Stefan; Zengler, Barbara: “UML 2 - glasklar”, München, Wien (Hanser) 2007.

[VOS08] Vossen, Gottfried: “Datenmodelle, Datenbanksprachen und Datenbankmanagementsysteme”, München (Oldenbourg) 2008.